

Proceedings of the International Web Audio Conference 2019



4-6 December 2019

Proceedings of the International Web Audio Conference 2019

Credits:

Proceedings edited and produced by
Anna Xambó, Sara R. Martín, Gerard Roma

Book compiled and edited by
Gerard Roma

ISSN: 2663-5844

<https://www.ntnu.edu/wac2019>
<http://webaudioconf.com/proceedings>

sponsored by



Department of Electronic Systems



ROLI

moz://a



in collaboration with



WoNoMute



TRONDHEIM KOMMUNE



cinemateket
TRONDHEIM

Conference Commitee

Anna Xambó (NTNU) - Conference Chair, Paper Chair
Sara Martin (NTNU) - Conference Chair, Paper Chair
Gerard Roma (University of Huddersfield) - Paper Chair, Online Publications Chair
Øyvind Brandtsegg (NTNU) - Music/Artwork Chair
Ariane Stolfi (Brazil) - Music/Artwork Chair
Andreas Bergsland (NTNU) - Poster/Demo Chair
Charles Martin (University of Oslo) - Poster/Demo Chair
Miranda Moen (NTNU) - Diversity Chair
Alessia Milo (Queen Mary University of London) - Social Chair
Eigil Aandahl (NTNU) - Social Chair, Workshop Chair
Jørgen Nygård Varpe (NTNU) - Workshop Chair
Robin Mientjes (Oslo) - Logo Designer
Åshild Berg-Tesdal (NTNU) - Web Editor
Rune Åge Frantzen (NTNU) - Financial Officer
Cecilie Heimdal (NTNU) - Administration Executive
Inna Aalmo (NTNU) - Administration Executive
Ellen Karlsen Holmås (NTNU) - Administration Executive
Hanne Formo (NTNU) - Administration Executive
Sigurd Saue (NTNU) - Consultant
Letizia Jaccheri (NTNU) - Consultant
Norbert Schnell (Hochschule Furtwangen University) - Consultant
Christoph Guttandin (Media Codings) - Consultant
György Fazekas (Queen Mary University of London) - Consultant
Trond Engum (NTNU) - Consultant
Thomas Henriksen (NTNU) - Consultant
Robin Støckert (NTNU) - Consultant
Frederic Marx (Ableton) - Consultant
Jan Monschke (SoundCloud) - Consultant
Jack Armitage (Queen Mary University of London) - Consultant

Program Commitee

Eigil Aandahl (Norwegian University of Science and Technology)
Paul Adenot (Mozilla)
Jesse Allison (Louisiana State University)
Jack Armitage (Queen Mary University of London)
Brian Belet (SoundProof)
Cristiano Belloni (hya.io)
Andreas Bergsland (Norwegian University of Science and Technology)
Oeyvind Brandtsegg (Norwegian University of Science and Technology)
Angela Brennecke (Film University Babelsberg KONRAD WOLF)
Paul Brossier (aubio)
Michel Buffa (I3S/University of Nice/University Côte d'Azur)
Andrei Bundin (Herzen University)
Thibaut Carpentier (IRCAM)
Antonio Deusany de Carvalho Junior (Universidade de São Paulo)
Vincent Cellucci (Louisiana State University)
Catherine Faron-Zucker (Université Côte d'Azur, CNRS, INRIA)
Xavier Favory (Music Technology Group - Universitat Pompeu Fabra)
Frederic Font (Music Technology Group - Universitat Pompeu Fabra)
Jason Freeman (Georgia Institute of Technology)
Samuel Goldszmidt
Siddharth Gururani (Georgia Institute of Technology)

Christoph Guttandin (Media Codings)
Timothy Hsu (Indiana University-Purdue University Indianapolis)
Nicholas Jillings (Birmingham City University)
Luis Joglar Ongay (SonoSuite)
Ruth John (Google)
Chris Kiefer (University of Sussex)
David Kim-Boyle (The University of Sydney)
Alexander Lerch (Georgia Institute of Technology)
Chris Lowis (Go Free Range Ltd.)
Thanassis Lykartsis (TU Berlin)
Anand Mahadevan (Dolby)
Yotam Mann (New York University)
Charles Martin (University of Oslo)
Benjamin Matuszewski (IRCAM)
Blai Melendez-Catalan (Music Technology Group - Universitat Pompeu Fabra)
Stuart Memo
Alessia Milo (Queen Mary University of London)
Chase Mitchusson (Louisiana State University)
David Moffat (Queen Mary University of London)
Jan Monschke
Manuel Moussalam (DEEZER)
Derick Ostrenko (Louisiana State University)
Matthew Paradis (British Broadcasting Corporation (BBC))
Adam Parkinson (Goldsmiths, University of London)
Tero Parviainen (Conjoin Oy)
Ashis Pati (Georgia Institute of Technology)
Johan Pauwels (Queen Mary University of London)
Guillaume Pelerin (IRCAM)
David Poirier-Quinot (IRCAM)
Hugh Rawlinson (Spotify)
Charlie Roberts (Rochester Institute of Technology)
Gerard Roma (University of Huddersfield)
Jordan Santell (Mozilla)
Diemo Schwarz (IRCAM)
Boris Smus
Jeffrey Snyder (Princeton University)
Fabian-Robert Stöter (International Audio Laboratories Erlangen)
Ryan Stables (Birmingham City University)
Ariane Stolfi (Federal University of South Bahia, Brazil)
David Su (MIT Media Lab)
Jeff Switzer
Hiroyuki Takakura (CodeNinth)
Anna Terzaroli (Nicola Sala Conservatory)
Lucas Thompson
Pierre Alexandre Tremblay (University of Huddersfield)
Raimund Vogtenhuber (University of Arts Zurich / ICST)
Tony Wallace (Irritant Creative Inc.)
Lin Wang (Queen Mary University of London)
Yonghao Wang (Birmingham City University)
Olivier Warusfel (IRCAM)
Nils Werner (International Audio Laboratories Erlangen)
Chris Wilson (Google)
Anna Xambó (Norwegian University of Science and Technology)
Matthew Yee-King (Goldsmiths, University of London)
Ehsan Ziya



Welcome

Foreword from the Paper Chairs

We are very excited to welcome you to the fifth edition of the Web Audio Conference 2019 in Trondheim, Norway on 4-6 December, 2019. The conference features papers, posters, talks, demos, artworks, performances, keynotes and workshops.

WAC is dedicated to web audio technologies and applications. The conference brings together academic and artistic research, technical development, design and evaluation of cutting-edge audio-related web technologies. This year we have accepted 83% of the received papers under a single-blind peer review process (54 out of 65) and finally we have published a total of 47 submissions: 19 papers, 3 posters, 8 talks, 6 demos, 4 artworks and 7 performances. We have also curated 5 workshops about relevant topics for the community, which are led by experts in the field. The topics of the sessions of papers and talks range from music composition, live performance and live coding; diversity and inclusion; HCI and frameworks; audio processing and standards; collaboration and education; applications; and sustainability.

We have put an effort to improve the reviewing process by introducing the concept of conditional accept to improve the final submissions. We have also introduced the figure of the meta-reviewer. Each paper was reviewed by at least 3 reviewers including a meta-reviewer. We hope that this approach can continue in future editions. We have also offered the pre-submission draft feedback process for those less familiar with academic writing. This year we have also obtained an ISSN number for the series of proceedings of the conference and launched a sustainable website that compiles all the proceedings of WAC (2015-present).

It has also been important to include a diversity chair in the committee, a role that we hope will continue in future editions to make sure that we hear and welcome all the voices of the community. In alignment to welcoming more diversity to the community, we have also programmed 3 panels of discussion about relevant topics for the community such as W3C and the WAC community; industry & academia; and accessibility.

We hope that the conference will provide you with an inspiring forum of discussion and good atmosphere to meet with current and new colleagues from all around the world. Welcome to Trondheim and enjoy the programme and the conference!

Anna Xambó (Norwegian University of Science and Technology)

Sara R. Martín (Norwegian University of Science and Technology)

Gerard Roma (University of Huddersfield)

WAC 2019 Paper Chairs

Foreword from the Diversity Chair

Dear attendees,

This year we have made an extra effort in order to provide a welcoming, inclusive and diverse conference. One of them has been to offer a diversity grant for a total of 17 individuals, to include attendees and authors from underrepresented communities in web audio. To understand our philosophy on diversity, you can read it at <https://www.ntnu.edu/wac2019/diversity-theme>. We have also updated our Code of Conduct at <https://www.ntnu.edu/wac2019/coc>, please read it to make sure you are updated as well. We have made sure that the various venues are accessible and have provided info about it on the website. In addition we have encouraged the speakers to make their presentations accessible, and we will provide a separate room/quiet space for people who need a little time out from the conference hassle. We also encourage the attendees to be trans gender and non binary inclusive - such as not taking pronouns for granted. Check out these pictures for inspiration: <http://www.transfeminism.net/everyday-solidarity/>. Please get in touch if you have any questions or want to know more.

Miranda Moen (NTNU alumni)
WAC 2019 Diversity Chair

Foreword from the Poster/Demo Chairs

Welcome to the WAC poster/demo track! The demos and posters are located together in two different rooms (R3 and R90) over two sessions Wednesday and Thursday. You can look forward to a great variety of topics within the field of web audio this year, including tools for synthesis, recording, editing, algorithmic music generation (several of them collaborative), audio feature extraction, low level development tools, soundscape and ecoacoustics as well as tools for navigating song data databases. We hope you will take the time to meet the authors for a tryout or for discussion many of these highly intriguing contributions to the world of web audio!

Andreas Bergsland (Music Technology, NTNU)

Charles Martin (Computer Science, University of Oslo)

WAC 2019 Poster/Demo Chairs

Foreword from the Music/Artwork Chairs

We would like to take this opportunity to wish you a special welcome to the Music and Artworks tracks of the Web Audio Conference 2019. With the great number of exciting submissions, it was a pleasure to curate this interesting program showing the diverse ways Web Audio technologies can be used for artistic expression. The application of these technologies for live performance requires stable solutions with low latency and expressive responsiveness. The accepted submissions show a great variety, from live coding to participatory pieces, from cyber-hacked devices and sound maps to ensemble processing of web audio repositories. The artworks track represents a unique way of showcasing Web Audio technologies, where users can experience the works in a nonlinear fashion. Here we will find artworks based on generative music, algorithmic permutations, networked sound sculpture, natural and cultural soundscapes. We look forward to celebrate and experience these creations together with you.

Øyvind Brandtsegg (Music Technology, NTNU)
Ariane Stolfi (Federal University of South Bahia, Brazil)
WAC 2019 Music/Artwork Chairs

Contents

1	Papers	1
	Facilitating Team-Based Programming Learning with Web Audio <i>Anna Xambó, Robin Støckert, Alexander Refsum Jensenius and Sigurd Saue</i>	2
	The application of Networked Music Performance technology to access ensemble activity for socially isolated musicians <i>Miriam Iorwerth and Don Knox</i>	8
	Sounds Aware: A Mobile App for Raising Awareness of Environmental Sound <i>Tate Carson</i>	14
	Composing Spatial Music with Web Audio and WebVR <i>Cem Çakmak and Rob Hamilton</i>	19
	From the museum to the browser: Translating a music-driven exhibit from physical space to a web app <i>Astrid Bin, Christina Bui, Benjamin Genchel, Kaushal Sali, Brian Magerko and Jason Freeman</i>	24
	Join my party! How can we enhance social interactions in music streaming? <i>Alo Allik, Florian Thalmann and Cornelia Metzger</i>	30

iMuSciCA: A Web Platform for Science Education Through Music Activities <i>Kosmas Kritsis, Manuel Bouillon, Daniel Martín-Albo, Carlos Acosta, Robert Piechaud and Vassilis Katsouros</i>	35
QuaverSeries: A Live Coding Environment for Music Performance Using Web Technologies <i>Qichao Lan and Alexander Refsum Jensenius</i>	41
MAIA Util: An NPM Package for Bridging Web Audio with Music-theoretic Concepts <i>Tom Collins and Christian Coulon</i>	47
Combining Collaborative and Content Filtering in a Recommendation System for a Web-based DAW <i>Jason Smith, Mikhail Jacob, Jason Freeman, Brian Magerko and Tom Mcklin</i>	53
Design of a real-time multiparty DAW collaboration application using Web MIDI and WebRTC APIs <i>Scott Stickland, Rukshan Athauda and Nathan Scott</i>	59
Soundworks - A Framework for Networked Music Systems on the Web - State of Affairs and New Developments <i>Benjamin Matuszewski</i>	65
FAUST online IDE: dynamically compile and publish FAUST code as WebAudio Plugins <i>Stéphane Letz, Shihong Ren, Yann Orlarey, Romain Michon, Dominique Fober, ElMehdi Aamari, Michel Buffa and Jerome Lebrun</i>	71
An AudioWorklet-based Signal Engine for a Live Coding Language Ecosystem <i>Francisco Bernardo, Chris Kiefer and Thor Magnusson</i>	77
Interference: Adapting Player-Music Interaction in Games to a Live Performance Context <i>Matthew Wang</i>	83

Towards Large-Scale Artistic Practice with Web Technologies <i>Luis Arandas, José Gomes and Rui Penha</i>	87
Csound Web-IDE <i>Steven Yi, Hlöðver Sigurðsson and Edward Costello</i>	92
Bringing the TidalCycles Mini-Language to the Web <i>Charlie Roberts and Mariana Pachón-Puentes</i>	98
Responsive Space. Liveness through spatial distribution of sound and image. <i>Raimund Vogtenhuber</i>	103
2 Talks	108
Creating a DJ-ready Web Player for Interactive Music <i>Attila Haraszi</i>	109
Building an intuitive web-based musical instrument <i>Ashish Dubey</i>	111
The new WAC website: towards a sustainable conference <i>Jørgen Varpe and Eigil Aandahl</i>	113
Essentia in the browser <i>Luis Joglar-Ongay, Albin Correya, Pablo Alonso-Jiménez, Xavier Serra and Dmitry Bogdanov</i>	114
Comparing apples to oranges: A comparison of AudioWorklet polyfills <i>Christoph Guttandin</i>	116
Opening the Dragon's Den: Interactive Welcome to Baton Rouge Entrepreneurship Week <i>Jesse Allison and Derick Ostrenko</i>	117
Connecting Web Audio to Cyber-Hacked Instruments in Per- formance <i>Anthony Marasco and Jesse Allison</i>	119

Flexible visualization of sound collections <i>Gerard Roma</i>	123
3 Posters	125
Ongaq JS: An elementary library for programming music <i>Hiroyuki Takakura</i>	126
A 2 Million Commercial Song Interactive Navigator <i>Michel Buffa, Johan Pauwels, Jerome Lebrun and Guillaume Pelerin</i>	128
Cooperation experiments in web-based audiovisual works <i>Balázs Kovács</i>	132
4 Demos	136
Tweakable <i>Julian Woodward</i>	137
Sounds Aware <i>Tate Carson</i>	139
Cassettes - An online audio editor <i>Xingxing Yang and Jatin Chowdhury</i>	141
Synth Kitchen <i>Spencer Rudnick</i>	143
moodplay.github.io: an online collaborative music player <i>Alo Allik, Florian Thalmann and Cornelia Metzger</i>	144
Develop WebAudio Plugins in a Web Browser <i>Michel Buffa, Stéphane Letz, Jerome Lebrun, Yann Orlarey, Romain Michon, Dominique Fober, Shihong Ren and ElMehdi Aamari</i>	145

5 Artworks 147

Generative.fm

Alex Bainter 148

Permutable

Julian Woodward 149

Wireplex, an Extended Flow of Data as Distributed Sound

Sculpture

Luis Arandas, José Gomes and Rui Penha 151

Acoustic Atlas

Cobi van Tonder and Guergana Tzatchkova 153**6 Performances 155**

Performing with QuaverSeries Live Coding Environment

Qichao Lan, Çağrı Erdem and Alexander Refsum Jensenius . . . 156

Three Pidgins: Live Coding Performance

Francisco Bernardo, Chris Kiefer and Thor Magnusson 158

Cyclic Gibbering

Charlie Roberts 160

Trondheim EMP Repository processing

*Oeyvind Brandtsegg, Anna Xambó, Trond Engum, Andreas**Bergsland and Carl Haakon Waadeland* 161

Responsive Space

Raimund Vogtenhuber 163

gravity—density

Anthony Marasco and Jesse Allison 164

map mop

Gerard Roma 166

7 Keynotes 167

Becoming latency-native
Rebekah Wilson 168

On our Past, Present, and Future with Web Audio Technologies – a participatory keynote address
Norbert Schnell 170

Recent and future evolution of the Web Audio API
Paul Adenot 171

8 Workshops 173

Designing and Performing with Live Coding Languages for Signal Processing and Machine Intelligence on the Web
Francisco Bernardo, Chris Kiefer and Thor Magnusson 174

Live coding with Makkeróni - workshop
Balázs Kovács 176

Web Audio API vs. native, closing the gap, take 2
Paul Adenot 178

Diversity Workshop
Miranda Moen and Andrea Hegdahl Tiltnes 180

Audiovisual Programming in Live Performance
Charlie Roberts 182



Papers

Facilitating Team-Based Programming Learning with Web Audio

Anna Xambó
Department of Music
Norwegian University of
Science and Technology
(NTNU)
Trondheim, Norway
anna.xambo@ntnu.no

Robin Støckert
Department of Mathematical
Sciences
NTNU
Trondheim, Norway
robin.stockert@ntnu.no

Alexander Refsum
Jensenius
RITMO
Department of Musicology
University of Oslo
Oslo, Norway
a.r.jensenius@imv.uio.no

Sigurd Saue
Department of Music
NTNU
Trondheim, Norway
sigurd.sau@ntnu.no

ABSTRACT

In this paper, we present a course of audio programming using web audio technologies addressed to an interdisciplinary group of master students who are mostly novices in programming. This course is held in two connected university campuses through a portal space and the students are expected to work in cross-campus teams. The course promotes both individual and group work and is based on ideas from science, technology, engineering, arts and mathematics (STEAM) education, team-based learning (TBL) and project-based learning. We show the outcomes of this course, discuss the students' feedback and reflect on the results. We found that it is important to provide individual vs. group work, to use the same code editor for consistent follow-up and to be able to share the screen to solve individual questions. Other aspects inherent to the master (e.g. intensity of the courses, coding in a research-oriented program) and to prior knowledge (e.g. web technologies) should be reconsidered. We conclude with a wider reflection on the challenges and potentials of using web audio as a programming environment for novices in TBL cross-campus courses and how to foster effective novices.

1. INTRODUCTION

The 21st century brings sophisticated information and communication technologies different from the 20th century, which is the reason why school curriculums are changing to a new culture of learning and the promotion of a new set of skills, the "21st century skills" [6, 11]. Programming can be considered an important component of technology literacy, which is one of the identified skills of the 21st century [11]. There are no written rules on how to teach programming, but there exist studies with recommended teaching tech-

niques, especially looking at how novices learn to program. In particular, Robins [15] distinguishes between effective and ineffective novices, and suggests to focus on promoting effective novices. A relevant factor is motivation, which can be enhanced with more interdisciplinary ways of learning, such as science, technology, engineering, arts, and mathematics (STEAM) education [7]. It has been reported that teaching programming based on STEAM promotes higher level of creativity [21] and can attract the interest of underrepresented populations in computing fields [8].

The NTNU-funded project Student Active Learning in a Two Campuses Organization (SALTO) aims to promote cross-campus learning as an open laboratory [17], where novel student-active learning strategies are investigated, such as computer-supported collaborative learning [16], flipping classroom [2] and team-based learning (TBL) [10]. Cross-site interaction is framed as a future scenario in education (e.g. teaching and learning environments) and work (e.g. working environments). The combination of simultaneous co-located vs. remote interactions, are mediated through low-latency audiovisual and communication technologies.

A new international master has been launched within this educational scheme: Music, Communication, and Technology (MCT),¹ which is a master's program in collaboration between the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, and the University of Oslo (UiO) in Oslo, Norway. The program centres around the field of music technology from a research perspective in a cross-campus setting. The students work in cross-campus teams, which are interdisciplinary, and they have different levels of expertise in STEM fields, such as programming.

As part of the master's program, most of the master courses are offered in a condensed format of two full-day weeks (eight days in total) so that students can focus on one subject at a time and can have more flexibility in their schedule. This can be especially useful for technical subjects, so that students focus on the same technology for a certain period of time. In our previous study on a physical computing workshop [20],



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹<https://www.ntnu.edu/studies/mmct>

we found that the use of hybrid technologies to teach physical computing to mixed-levels cross-campus groups was a complicated environment for novices in programming. In this paper, we present instead the results of an intense course in audio programming, only taught with web audio technologies, which is a recommended tool for novices [1].

The research question that this paper addresses is: *What are the challenges and opportunities of using web audio to teach audio programming to a mixed cohort of students mostly novices in programming?* We collected students' feedback and reflected on the students' outcomes and comments. Overall, the results indicate that we can recommend to use web audio to teach audio programming to a mixed group of mostly novices, if there is a fair combination of individual and group work, the students and teacher use the same code editor for consistency, and it is possible at all times to share the code between the teacher, smaller groups and the whole class to promote debugging in a collaborative context. However, pre-knowledge should be required in web technologies and basic programming, as well as focus the teaching on strategies instead of knowledge, in order to promote "effective novices".

2. BACKGROUND

Our previous research highlighted how collaborative music live coding (CMLC), which refers to the combination of the music improvisation practice of live coding and collaborative programming, is a promising learning strategy of programming [19]. In a more recent study on teaching physical computing to mostly novices [20], we also found that programming is a difficult skill to learn in a short period of time, as opposed to prototyping. As noticed by Robins et al.: *"It is generally accepted that it takes about 10 years of experience to turn a novice into an expert programmer."* [15, p.137] and therefore Robins et al. recommend to present material that addresses basic program design.

Web audio technologies have been successfully used in programming courses based on STEAM principles. For example, it was used in a summer programming music camp [1] to teach creative coding concepts through music by using different web audio libraries and frameworks, such as Gibber [14] and Tone.js [9]. Other courses are more focused on a particular library or environment. The Quint.js is a JavaScript library that combines visuals and audio to create interactive audiovisual programs, and was used to teach music technology concepts (e.g., trigonometry and wave motion, additive synthesis, Fourier analysis, pitch spiral, noise spiral) to fine arts students [3]. EarSketch is a learning environment and curriculum that promotes to learn how to code by making music among middle school, high school and undergraduate students, an environment reported to broaden participation in computing and music, particularly women [8]. Codecircle is a browser-based system for learning creative coding that supports real-time graphics and sound [21].

In this paper, we present a course of audio programming using web audio technologies addressed to an interdisciplinary group of master students, who are mostly novices in programming. We used a similar approach to [1] of exposing the students to different libraries and frameworks, with the final aim of creating their own project. However, our course was longer so we also included native web audio to facilitate the understanding of these higher-level environments. We promoted both individual but also cross-campus group projects.

3. THE COURSE

3.1 Context

The MCT4048 Audio Programming course is an elective master course held in Spring that aims to provide a solid foundation in digital signal processing and audio-based application development. The integration of relevant technologies and platforms plays an important part to develop user-ready applications. There are a number of reasons for choosing web audio technologies for this course, namely:

- It is written in JavaScript, one modern programming language.
- It is easy to sketch ideas and get prototypes built.
- It is easy to test, implement, and distribute.
- It showcases the fundamental concepts of audio programming.
- It gives room for artistic expression.
- It is an employable skill.
- We will be hosting the Web Audio Conference 2019.²

The students have a multidisciplinary background ranging from digital humanities, to music technology, to engineering. Their backgrounds in programming skills are varied, predominantly novices. In this course edition, the number of total students was 11 between the two campuses, with 4 students in the Oslo site and 7 students in the Trondheim site, and two women. The group was international with students from Europe and Asia.

3.2 Curriculum

This course combines lectures and hands-on activities, whose slides and code are available online.³ The lectures provide an overview of the fundamental concepts of audio programming. The hands-on workshops are based on building web applications using web audio technologies, both individually and in team. The evaluation of the course consists of measuring the daily activity and two mini-projects that incorporate the theory and practice seen in class.

Given that this course belongs to a research-based master, it combines research-based activities with development activities. The course spans 7 hours per day during 8 days (56 hours in 2 weeks). We allocate some time at the beginning of each session to set up the computers with the tools for the tutorial of the day. The first week is named "The Fundamentals" and the students are asked to develop an individual mini-project. This allows each student to work individually, familiarize themselves with the programming environment and find their way of expression. The second week is named "The Extensions" and the students are asked to develop a group mini-project. Next we show the general outline of the course by day:

1. **Introduction Day:** Paper discussion about languages for computer music [5]. Discussion about what is and why to use the Web Audio API. Discussion and exercise about pseudocode.

²<https://ntnu.edu/wac2019>

³<https://github.com/axambo/audio-programming-workshop>

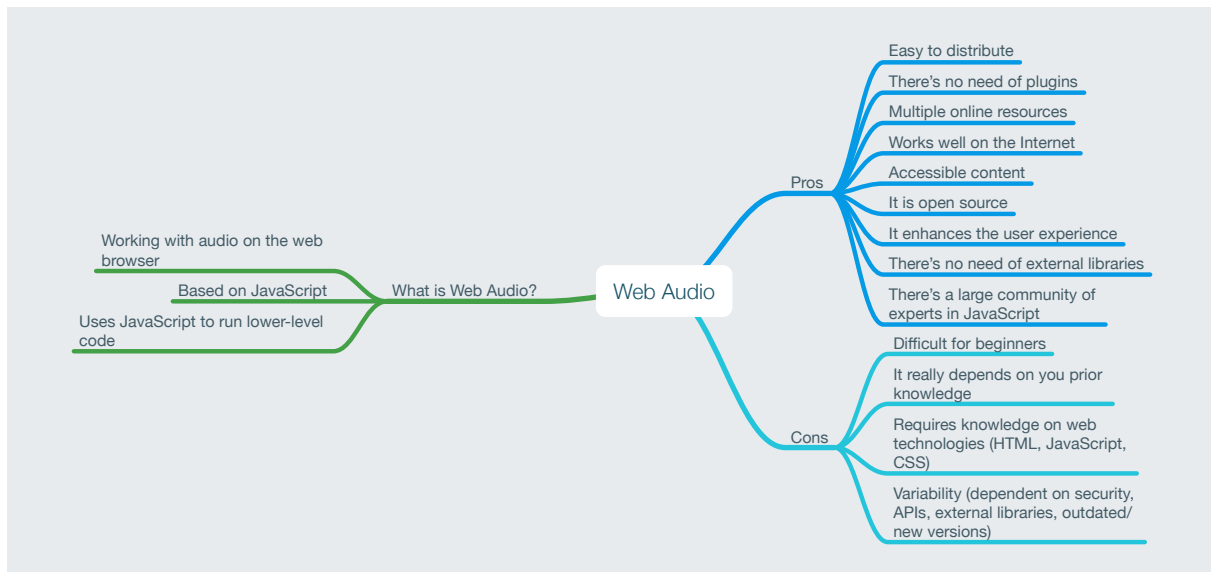


Figure 1: Mind map generated in class from a group discussion about what is web audio.

2. **Day 1:** Warm-up activity: what is Web Audio, pros and cons (see Figure 1). Tutorial of playing sounds and individual mini-project development.
3. **Day 2:** Warm-up activity: a round of one sentence each about something learned the previous day in class. Tutorial of dealing with time using Tone.js [9], and individual mini-project development. Speedy presentations of the mini-projects.
4. **Day 3:** WAC paper presentations (first half of the group, one paper per student). Tutorial of dealing with sound effects and individual mini-project development. Speedy presentations of the mini-projects.
5. **Day 4:** WAC paper presentations part 2 (second half of the group, one paper per student). Tutorial of graphical user interfaces using NexusUI.js [18], and mini-project development. Final presentations of the individual mini-projects.
6. **Day 5:** Warm-up activity: Recap quiz of week 1. Tutorial of dealing with interactivity using Web MIDI API⁴ and group mini-project development.
7. **Day 6:** Warm-up activity: what do we know about live coding. Tutorial of live coding using Gibber [14], and group mini-project development. Speedy presentations of the mini-projects.
8. **Day 7:** Tutorial of mobile music and responsive design, and group mini-project development. Speedy presentations of the mini-projects.
9. **Day 8:** Tutorial of AudioWorklets [4, 12, 13], and group mini-project development. Final presentations of the group mini-projects.

3.3 Outcomes

During the first week, the students developed a total number of 10 individual mini-projects, whilst in the second week,

⁴<http://webaudio.github.io/web-midi-api>

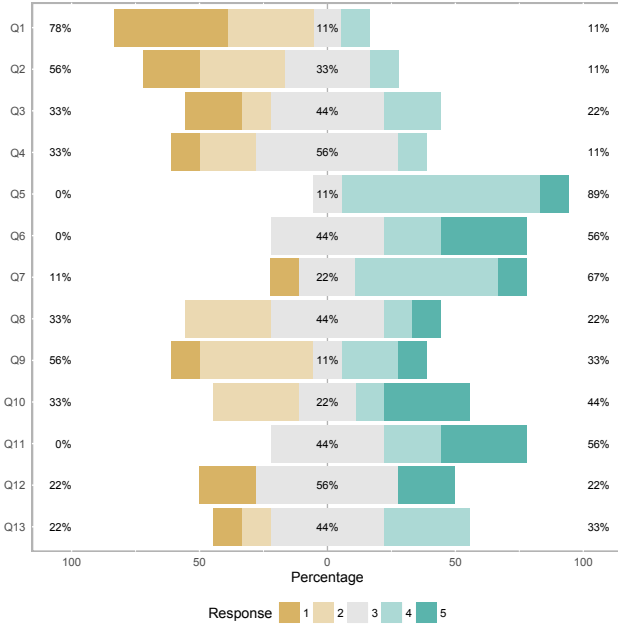
the students developed three group mini-projects. A total number of 13 blog posts about each mini-project can be found online in the student-led MCT blog.⁵

Next, a brief description of the three group mini-projects is provided:

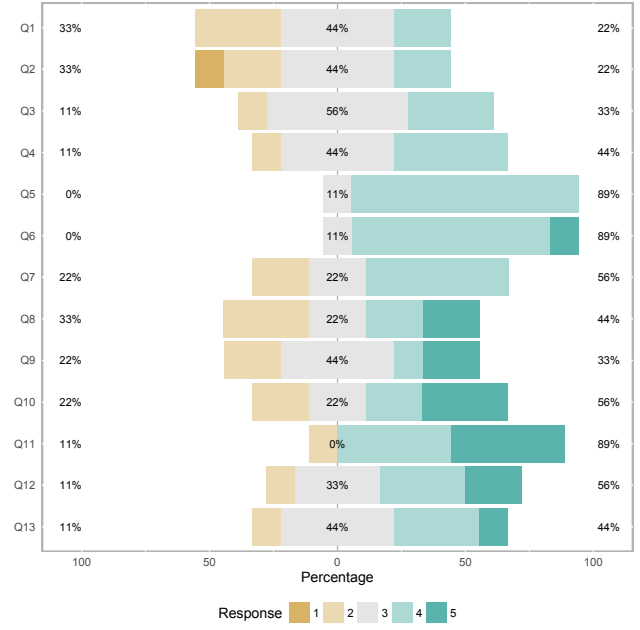
- *Touch the Alien* (Team A) – A web audio synth which offers touchscreen functionality; a combination of oscillators, FM oscillator, delay, phaser, chorus, and filter on dry/wet slider; and an interactive canvas user interface to control the filters. The technologies used include Javascript with the Web Audio API, CSS, HTML5, the audio effects library Tuna, and a smartphone or tablet.
- *The Magic Piano* (Team B) – A piano that plays the right notes of the chosen melody regardless of whether the player hits the right or wrong key. The technologies used include Web MIDI API, NexusUI.js, Tone.js, JSON, CSS and a MIDI keyboard.
- *Convolverizer* (Team C) – Real-time processing of ambient sound, voice or live instruments. The technologies used include P5.js, a sound card or audio interface, a guitar, and a Shure SM57 microphone.

The three group mini-projects were successful in completion. Although varied in theme, there are some similarities: They (1) build on the code developed individually during the previous week (except for Team C who built on knowledge from the physical computing workshop), with the challenge of merging their code and working with collaborative coding approaches, both co-located and remote; (2) use a range of web audio and web technologies, some of them seen in class but some of them explored autonomously or in previous courses, which showcases the understanding of the spirit of prototyping in finding the best tools and combining them to convey a project idea; and (3) combine software with hardware, building on previous knowledge from the physical computing workshop.

⁵<https://mct-master.github.io/audio-programming>



(a) Bar plot of the pre-questionnaire responses.



(b) Bar plot of the post-questionnaire responses.

Figure 2: Bar plot for the results of thirteen (Q1–Q13) 5-point Likert-item questions ($n = 9$).

Questions: Q1 programming; Q2 computational thinking; Q3 prototyping; Q4 instrument building; Q5 reflective practice; Q6 teamworking; Q7 individual working; Q8 continue STEM courses; Q9 continue STEM education; Q10 future use of STEM knowledge; Q11 understanding of audio programming; Q12 understanding of programming interactive musical prototypes; and Q13 programming interactive musical prototypes.

4. STUDENTS' FEEDBACK

Adapted from our previous physical computing workshop [20], we distributed among students a voluntary pre-questionnaire and post-questionnaire with the same 5-point Likert-item questions. The questions ranged from asking the level of confidence (1 = not at all confident; 2 = a little confident; 3 = somewhat confident; 4 = highly confident; 5 = extremely confident) about their ability for programming (Q1), computational thinking (Q2), prototyping (Q3), instrument building (Q4), reflective practice (Q5), teamworking (Q6), and individual working (Q7). There were also questions that asked the level of agreement (1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree) about a set of statements on their intention to continue courses related to STEM fields (Q8), to continue their education in STEM fields (Q9), and to use their STEM knowledge in their future careers (Q10). They were also asked their level of agreement of the extent to which they can understand the purpose of audio programming (Q11), describe the process of programming an interactive musical prototype (Q12), and apply the technique of programming an interactive musical prototype to their work (Q13). The questionnaire also included three open questions about what the students liked best and least about the course and how the course could be improved.

We obtained responses from 11 students in total, out of which 9 students responded the paired questionnaire ($n = 9$).

Of the course, the students liked best:

- **Learning content** (4 occurrences): “broadening the perspective” (U8); “learning new libraries and frame-

works” (U4); “learning the basic building blocks of the Web Audio API” (U2); “learning live coding” (U2).

- **Learning process and course outcomes** (3 occurrences): “more security and confidence in programming” (U3); “build applications” (U1); “collaborative working” (U5).
- **Course design** (4 occurrences): “freedom for creativity” (U7); “work hands-on on the code and learn it” (U10); “the combination of lessons and hands-on practice and prototyping” (U9); “fun to learn how to code” (U6).

The students liked least:

- **Be a novice programmer** (4 occurrences): “lack of basic programming skills” (U8); “not knowing how to program” (U3); “individual working was hard without asking permanently for help” (U5); “as a beginner the need of basic content for a longer period of time” (U9).
- **Intensive course format and remote programming in teams** (4 occurrences): “too short and condensed” (U1); “a bit too fast and packed” (U10); “only 2 weeks in the whole semester seems to be a pity!” (U9); “hard to collaborate with coding over distances” (U6).
- **Mixed groups and research-based programming course** (2 occurrences): “added complexity of working with other web technologies, which can take a bit of focus away from audio programming” (U2); “too much focus on other things than programming (blog, presentation)” (U4).

Students' suggestions on how to improve the course included:

- **Avoid intensive course format** (4 occurrences): *"making it less intense and more spread during the term"* (U8); *"extend to more than 2 weeks workshop"* (U1); *"adding another week and slowing down the process will help for absolute beginners a lot"* (U10); *"we could have learnt more if the course had run for a longer period of time"* (U9).
- **Request pre-knowledge in programming and web technologies** (3 occurrences): *"having a basic level of training in the beginning"* (U8); *"required pre-knowledge in programming from all students"* (U4); *"an introductory lecture or two in the absolute basics in coding"* (U7).
- **Satisfy both novices and experts** (3 occurrences): *"the course was well taught, even though the level was very high, more time needed to evaluate"* (U3); *"the way it was this year would suit more for students with prior understanding of programming to some level"* (U9); *"more clearly defined incremental tasks related to the curriculum, which should also have the possibility of extra challenges for those that are on a higher level"* (U2).

Figure 2 shows the percentages of the level of confidence and agreement, which tended to be more positive in the post-questionnaire ($Mdn=4$, $M=3.47$) than in the pre-questionnaire ($Mdn=3$, $M=3.09$). These results align with the results from our previous course in physical computing [20], which indicate that the pedagogical techniques are in a positive direction. The level of confidence of programming (Q1), together with the level of prototyping (Q3) and instrument building (Q4) slightly improved, providing at least little more confidence. This contrasts with the level of confidence of programming achieved in the physical computing workshop, which was one of the less developed skills. The intention to continue STEM courses (Q8) and STEM education (Q9) improves slightly, again in alignment with our previous study. The understanding of audio programming (Q11) polarised a little bit more in the post-questionnaire, probably associated with the need of learning additional tools related to web development (as discussed earlier in the open questions). The level of confidence of teamworking (Q6) and reflective practice (Q5) slightly increased from an already high score, two aspects that are explicitly promoted across the different master's courses. The level of confidence of individual work (Q7) changed from extreme to moderate positive and negative opinions. Individual work was an important component of this course during the first week, it seems to be valued by the students, but it needs to be better integrated so that both experts and novices acknowledge an improvement.

5. REFLECTIONS

This section reflects on the students' feedback presented in the previous section combined with the reflections of the teacher of the course (first author). Overall, learning how to program is a slow endeavor, and team-based programming can be a helpful approach. There are five prominent themes that are worth of discussion: (1) individual vs. group

work; (2) shared coding experiences; (3) web audio vs. web technologies; (4) research vs. code; and (5) fast-paced vs. slow-paced teaching.

5.1 Individual vs. Group Work

The sequence of individual work during the first week and group work during the second week generally worked well. The students developed an individual language and style using web audio technologies, which was helpful for the collaborative work in the following week. This is similar to the need of individual rehearsal with your own musical instrument to first define your voice before meeting with a group to collaborate musically. Prior knowledge to the course that could be acquired includes basic programming and how to merge code from different contributors. The additional asset of this course, compared to other similar courses (e.g., [1]), is using TBL to solve problems that have a higher level of complexity than when working individually. The shareable nature of web technologies (e.g. easy exchange and execution of code snippets) seems to align well with TBL activities.

5.2 Shared Coding Experiences

Although not directly reported in the students' feedback, at the beginning of the course the teacher and students decided to work with the same code editor (Visual Studio Code) and web browser (Chrome). This facilitated that all students could follow the hands-on tutorials and could debug in collaboration or show their problems to the teacher or group if needed. Considering that this course should be taught by only one teacher to cross-campus scenarios of at least one co-located group and one remote group, the real-time audiovisual and screen share communication becomes crucial. With this approach, there were occurrences of expert students helping novices, therefore a student's problem could often become a group problem, and everyone was learning from each other, in terms of both errors and discoveries. Both the infrastructure of the portal with network-connected working rooms and the use of the same programming environment and language with shareability capabilities seemed to help.

5.3 Web Audio vs. Web Technologies

Novices were sometimes overwhelmed with the need to learn not only web audio, but also web technologies. By contrast, experts were sometimes expecting more focus on web audio and put less attention to web technologies. Although web audio can be a suitable tool for learning audio programming, novices should fulfil the pre-knowledge requirements of already knowing the basics about web technologies. Similarly, the basics of programming should be also a pre-requisite in order to balance better novices vs. experts. As suggested by Robins [15], it is important to focus on teaching strategies as opposed to knowledge in order to promote "effective novices", who become proficient in solving programming problems.

5.4 Research vs. Code

Teaching an audio programming course at a research-based master implies combining hands-on practical work with theory and reflection. However, we found that novices were expecting to also learn how to code, whilst experts were expecting to code more. The meaning of a research-based course in audio programming should probably be emphasized at a master level and reinforced at a course level, so that students' expectations are closer to the nature of the master.

5.5 Fast-paced vs. Slow-paced Teaching

The students' feedback suggests to expand or reconsider the intensity of the course. An option can be to link better this course with other courses, so that the students perceive a continuous learning path across courses. Given that some students report the need of more formal teaching or more time to consolidate their knowledge, perhaps there could be curated resources that help the students to continue by their own if they want to, similar to the curriculum of EarSketch [8].

6. CONCLUSIONS

In this paper, we presented an audio programming course using web audio technologies targeted to an interdisciplinary group of master students who are mostly novices in programming. Students' feedback and teacher's reflections indicated that web audio technologies is a suitable approach to novices in programming if web technologies and basic programming are requested as prior knowledge. The projects resulted from working in teams have shown the potential of addressing complexity with creativity and collaboration, which was partly possible due to the prior individual work. It is still challenging to teach programming across two campuses, but applying techniques from collaborative live coding (e.g. sharing the screen and the code editor) can positively counterbalance the issue. We expect that in the second edition of this course we will foster better "effective novices", so that the two intense weeks can be even more fruitful. In future work, we plan to study with a larger number of students; understand which features of web audio enhance collaboration in TBL activities; and investigate whether web audio works better for TBL activities compared to native environments (e.g. C++ audio library, Max or SuperCollider). We also hope to explore more the potential of teaching audio programming with web audio for STEAM education, TBL and distance learning.

7. ACKNOWLEDGMENTS

The authors wish to thank the students who participated in the course. Also thanks to the MCT teachers Daniel Formo, Anders Tveit and Kristian Nymoen for their technical help. This work was partially supported by the NTNU SALTO project (80340480).

8. REFERENCES

- [1] J. Allison, D. Holmes, Z. Berkowitz, A. Pfalz, W. Conlin, N. Hwang, and B. Taylor. Programming Music Camp: Using Web Audio to Teach Creative Coding. In *Proc. Web Audio Conference*, 2016.
- [2] J. Bergmann and A. Sams. *Flip Your Classroom: Reach Every Student in Every Class Every Day*. International Society for Technology in Education, Eugene, OR, USA, 2012.
- [3] I. G. Burleigh and T. Schaller. Quint.js: A JavaScript Library for Teaching Music Technology to Fine Arts Students. In *Proc. Web Audio Conference*, 2015.
- [4] H. Choi. AudioWorklet: The Future of Web Audio. In *Proc. International Computer Music Conference*, 2018.
- [5] R. B. Dannenberg. Languages for Computer Music. *Frontiers in Digital Humanities*, 5:26, 2018.
- [6] C. Dede. Comparing Frameworks for 21st Century Skills. *21st Century Skills: Rethinking How Students Learn*, 20:51–76, 2010.
- [7] J. Maeda. STEM + Art = STEAM. *The STEAM Journal*, 1(1):34, 2013.
- [8] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A STEAM-based Approach for Underrepresented Populations in High School Computer Science Education. *ACM Transactions on Computing Education*, 16(4):14, 2016.
- [9] Y. Mann. Interactive Music with Tone.js. In *Proc. Web Audio Conference*, 2015.
- [10] L. K. Michaelsen, A. B. Knight, and L. D. Fink. *Team-based Learning: A Transformative Use of Small Groups in College Teaching*. Stylus Pub, 2004.
- [11] B. Pearlman. Making 21st Century Schools: Creating Learner-Centered Schoolplaces/Workplaces for a New Culture of Students at Work. *Educational Technology*, pages 14–19, 2009.
- [12] C. Roberts. Strategies for Per-Sample Processing of Audio Graphs in the Browser. In *Proc. Web Audio Conference*, 2017.
- [13] C. Roberts. Metaprogramming Strategies for AudioWorklets. In *Proc. Web Audio Conference*, 2018.
- [14] C. Roberts and J. Kuchera-Morin. Gibber: Live Coding Audio in the Browser. In *Proc. International Computer Music Conference*, 2012.
- [15] A. Robins, J. Rountree, and N. Rountree. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2):137–172, 2003.
- [16] G. Stahl, T. D. Koschmann, and D. D. Suthers. *Computer-Supported Collaborative Learning*. na, 2006.
- [17] R. Støckert, A. R. Jensenius, and S. Saue. Framework for a Novel Two-Campus Master's Programme in Music, Communication and Technology Between the University of Oslo and the Norwegian University of Science and Technology in Trondheim. In *Proc. International Conference of Education, Research and Innovation*, pages 5831–5840, 2017.
- [18] B. Taylor, J. T. Allison, W. Conlin, Y. Oh, and D. Holmes. Simplified Expressive Mobile Development with NexusUI, NexusUp, and NexusDrop. In *Proc. New Interfaces for Musical Expression*, pages 257–262, 2014.
- [19] A. Xambó, G. Roma, P. Shah, T. Tsuchiya, J. Freeman, and B. Magerko. Turn-taking and Online Chatting in Co-located and Remote Collaborative Music Live Coding. *Journal of the Audio Engineering Society*, 66(4):253–266, 2018.
- [20] A. Xambó, S. Saue, A. R. Jensenius, R. Støckert, and Ø. Brandtsegg. NIME Prototyping in Teams: A Participatory Approach to Teaching Physical Computing. In *Proc. New Interfaces for Musical Expression*, 2019.
- [21] M. Yee-King, M. Grierson, and M. d'Inverno. STEAM WORKS: Student Coders Experiment More and Experimenters Gain Higher Grades. In *Proc. IEEE Global Engineering Education Conference*, pages 359–366. IEEE, 2017.

The application of Networked Music Performance technology to access ensemble activity for socially isolated musicians

Miriam Iorwerth

University of the Highlands and Islands
Inverness, UK
miriam.iorwerth.whc@uhi.ac.uk

Don Knox

Glasgow Caledonian University
Glasgow, UK
d.knox@gcu.ac.uk

ABSTRACT

Networked Music Performance (NMP) allows musicians to play together over distances via the internet. For musicians who are socially isolated this is a valuable tool to allow musical connections despite barriers of geography or mobility. There are, however, challenges when using this technology, including how musicians cope with technical challenges (such as latency, and setting up and using NMP software), but also the challenges of communicating via potentially degraded audio and video links. By examining current research and a case study of the use of NMP at the University of the Highlands and Islands in a remote part of the UK, this paper argues that these challenges are not insurmountable. Meaningful musical relationships can be built and maintained using typical domestic equipment, and the network environment gives opportunities for musical creativity that would not be possible in a conventional rehearsal space.

1. INTRODUCTION

Playing music is an important part of cultural life throughout the world, and playing with other musicians is a vital social activity for many people, both professional and amateur. The health and social benefits of being part of a choir, for example, are clear [14], however there are barriers to participation in ensemble music for many musicians as a result of general social isolation. Factors contributing to social isolation may include long-term illness or disability, transport issues, low population density, or unemployment and economic struggles.

Networked Music Performance (NMP) allows musicians to play together across the internet, and can be achieved with minimal equipment and software. While some systems, such as LOLA [15] require specialist equipment and academic networks, domestic NMP software is also available, making it accessible for any musician with a broadband internet connection, computer, and webcam. This paper does not aim to discuss particular NMP technology solutions, rather it examines the factors affecting musicians using NMP in domestic situations and how NMP could benefit isolated musicians. An examination of these factors, and how musicians practically use NMP software, may be useful to developers of NMP software when making decisions regarding,

for example, trade-offs between the bandwidth balance of video and audio, or the how latency is dealt with.

2. SOCIAL ASPECTS OF NMP

NMP in the public eye has often been surrounded by publicity around the possibilities for technology to connect musicians – popular examples include a collaboration between Canadian band Barenaked Ladies and astronaut Chris Hadfield at the International Space Station [18], as well as numerous performances showcasing technology such as LOLA [15] at academic conferences. While useful to demonstrate the technical possibilities, these collaborations miss some of the major social benefits that are possible with NMP.

One obvious benefit is for musicians who are geographically dispersed, allowing them to work together across distances without needing to travel, and this is particularly valuable for musicians in remote and rural areas of the world who may feel isolated from others with similar interests. While performance is generally highlighted in showcased examples, all areas of ensemble music are possible using NMP, including education, improvisation, rehearsal, as well as performance. The purpose of the collaboration determines the requirements of the NMP connection, in terms of synchronisation and quality of audio and video feeds [1]. In many cases performance (which requires the highest connection quality) may be the least important of these for those wishing to make social connections through music.

Playing music as an ensemble includes social aspects: musicians often spend time together before, during, and after rehearsals and performances in non-musical social activities, building up personal relationships, and dealing with conflict. In a NMP situation, much of this time together has the potential to be lost as musicians can prepare and tune their instruments, for example, before they switch on the NMP connection, and then switch it off as soon as they have finished playing. This could allow fewer opportunities to get to know fellow musicians on a personal level, and good interpersonal relationships can benefit ensemble success [24].

Brown [4] highlights the importance of recognising that the interactions between musicians involve the negotiation of social and artistic relationships with others. In NMP musicians are not merely working with an incoming stream of audio (as they might be when recording overdubs in a studio, for example), but relationships with other musicians are as important as if they were working in the same space together. These relationships have the potential to be affected by the altered social contact in this setting. Examples of relationship-building in musical and online interactive environments are highlighted by Dillion and Brown



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

[13] in their Network Jamming project, with children working remotely on music and visual projects.

In the context of computer games, Magerkurth et al. [25] argue that network-based interactions are socially ‘inferior’ to face-to-face interactions. Wright [35], however, argues that this is only true in a musical context if attempting to reproduce traditional face-to-face musical encounters. Most musicians using NMP are not aiming to do this; rather they are using the technology as a tool enabling unique interactions, which would not be possible in a traditional setting. Schroeder and Rebelo [32] highlight the differences between traditional performance spaces and networked performances, and their corresponding strengths. The network’s strengths are the opposite of the concert hall’s, allowing for experimentation and the musically unknown.

Rather than focusing on the barriers to creating music using NMP, Dessen [12] sees the positive aspects, where the feeling of distance and intimacy become parameters that can be used within composition, and that the use of interactive scores on screen, for example, can allow musicians to work beyond the limitations of conventional paper scores. In a NMP situation there is also potential for musicians to be affected by the ‘online disinhibition effect’, where they feel more relaxed and can express themselves more freely than in a collocated situation [33].

While there are clear technical difficulties facing musicians using NMP, some challenges facing musicians in this situation are related to social aspects of the collaborations caused by the use of technology. This is particularly the case with improvisation, where the interactive dialogue as well as style and quality of the music can be influenced by the technical set-up [26]. In NMP there is no physical space where musicians work, therefore the musicians do not encounter each other physically and can only confirm another’s presence by interacting with them [16]. This is particularly true when there is no video connection, although even with a video link, an image on a screen of another musician is not enough to know they are available and ready to collaborate with. This introduces additional challenges to musicians, who may be used to playing relatively spontaneously when in a room together.

Easy access to online technology allows musicians to build up communities and NMP systems can be treated as community spaces to visit to connect with other musicians. Relationships between musicians can easily form in this situation, and rather than seeing NMP as an extension of traditional music making, NMP can be seen as a community-based model, where the collective skills of the community are a valuable asset [2]. These communities are particularly valuable to those who do not have easy access to other musicians, and transcend geographic and cultural borders.

When working with NMP, musicians may need to be aware of the social issues they are facing and ensure they are making the best use of the technology available to them. This might include making time before, during, and after working on musical aspects of their work to get to know their fellow musicians on a personal level, and using other technology such as email and messaging services to build up distance relationships. Equally, the strengths of the community aspects of NMP should not be overlooked, particularly where musicians are faced with challenges that make traditional forms of music making problematic.

3. NMP IN DOMESTIC SETTINGS

3.1 Technical Considerations

Given that the internet is not designed for the real-time transmission of audio, using NMP results in a different experience to playing in a room with other musicians. Audio and video connections can be used, which may or may not be synchronised, and these both suffer from latency. If more than two musicians wish to connect at one time then there may also be multiple latencies from different connections. Despite these challenges, musicians are very good at adapting to different acoustic environments, and therefore arguably the network environment. Several different approaches can be taken to these challenges, including the master/slave approach [8], which relies on musical leadership to maintain the flow of the music, or the acceptance of the network as having an acoustic itself and working with the artefacts it produces [9]. The effects of latency on synchronisation between musicians are well-documented, and the success of NMP for particular musicians, particularly in domestic settings, relies on this issue being accepted as a feature of NMP.

Some systems have aimed to get around the problems of latency through the use of distributed metronomes [2] or deliberately delayed signals that match with the beat of the music [10], however these prevent musicians from being free to choose their own musical content which may include tempo changes, or free rhythms. The simplest domestic systems may include an audio connection using software such as Soundjack [7] or JackTrip [5], and a separate video connection using typical video-conferencing software, such as Skype.

A further consideration for musicians using NMP is the technical expertise required to set up and use the software. Current dedicated NMP software, such as Soundjack [7] and JackTrip [5], require technical knowledge of networks, for example, and therefore are difficult to use for many musicians. There are opportunities for developers to use web interfaces to make NMP more accessible for musicians with limited technical knowledge. Considerations of latency, and the complexities around dealing with multiple latencies, for example, must be taken into account, which may require creative solutions. An example of a system that is accessible to those with minimal technical knowledge is NINJAM [10], which uses a web interface and deals with synchronisation by adding latency to match the pulse of multiple musicians. Further details on the addition of latency to enable synchronisation can be found relating to the Online Orchestra project [31].

Despite the challenges, NMP offers a huge advantage to musicians who are isolated in some way – they can access other musicians anywhere in the world (although geographically closer will result in less impact of latency) without needing to leave their home. Working as a duo is the most accessible form of NMP for musicians, both in terms of minimising multiple latency issues but also in terms of organisation and planning. Musical sessions online can be treated as jam sessions or rehearsals with musicians connecting via webcams for video, and either computer microphones or separate microphones and audio interfaces, depending on the equipment available and desired audio quality.

3.2 The Challenges of Domestic NMP for Musicians

3.2.1 The Musicians’ Experience

While NMP may offer an accessible solution to the problem of isolation for musicians, the experience of working online is

undoubtedly different to traditional ensemble playing: musicians may be working in their own homes; they will be using microphones and monitors, rather than hearing the other musician directly; and views of the co-performer may be limited by use of a webcam. In addition, any interactions are clearly defined by switching on and off the connection at the start and end of the session.

Successful ensembles have positive interpersonal relationships [28], and the social interaction between musicians is altered by the use of NMP, before, during, and after the musical content of a session. Existing relationships between musicians can help with the interaction, as performers trust, respect and support their fellow musicians while playing online [20], however for the truly isolated musician, co-performers may never meet in person. In these cases, effort must be made by both parties to build socio-emotional relationships by allowing time for informal discussion and conversation as part of a musical session, much as would happen in a typical rehearsal or jam session.

In addition, musical leadership can help musicians deal with latency problems. This requires communication and negotiation (which may be deliberate or subconscious) between musicians, and how well musicians adapt to the actions of others depends on their shared knowledge and rules [34], which develop over time. This suggests that over time, musicians will get to know one another's particular playing styles and negotiate methods for dealing with any technical difficulties. It is, therefore, important that NMP is not discounted as an option for isolated musicians if they initially face difficulties when using the technology.

3.2.2 Communication

Communication between musicians is a two-way process, with musicians' body movements and the music itself transmitting information, and other musicians receiving this information via their gaze and listening. When playing in a room together, this happens in ways that are conventional to the genre, and are well rehearsed. In NMP, however, the communication is mediated by an audio and video link. The impact of this interface is two-fold: it causes degradation in the transmission of the body movements and music (by limiting the area the receiving musician sees, as well as a reduction in video quality and addition of latency, for example), as well as the display of the image and reproduction of the sound (by using small video monitors and artefacts caused by compression, for example) [21].

This degradation will impact on the musicians' performances in several ways. Musicians may have difficulty producing coherent performances, as coordination of timing, dynamics and expression may be affected by the interruption to communication. Blending and tuning are particularly difficult when musicians are separated, for example [20].

An aim of any ensemble musician is to create a coherent performance with their co-performers, including coordination of timing, tuning and dynamics, requiring communication between musicians [22]. In a typical performance, the musicians can easily judge the success of this, because they hear roughly the same as what the audience hears (distance from instruments and therefore relative loudness notwithstanding). In NMP this is not the case: the idea of a coherent performance is nebulous, as each participant will have heard a slightly different overall performance.

It is therefore important to define ensemble coherence for a NMP situation, and who judges this. Synchronisation of musicians cannot be considered a prerequisite in this situation, however coordination of musicians is still important. This may mean that

the musicians do not play to exactly the same beat (one may be delayed), but a regular temporal delay between the parts may be considered coherent. Other musical factors may also be taken into account when defining ensemble coherence, including coordination of dynamic and tempo changes.

While a visual connection between musicians is considered by many to be vital for coherent performances, musicians tend not to use the video connection in NMP for the coordination of musical content [6,27]. Despite this, musicians do use video links when they are provided, but more often for discussions around the music, rather than when playing [21], a vital part of the rehearsal process [3,17]. This is particularly relevant in domestic situations, where high-quality video may not be available, and may not be synchronised with a separate audio feed.

3.2.3 Musical issues

The NMP environment may also affect the music produced, as communicative and technical difficulties impact on the musicians. The rhythmic content of the music will affect the success of an NMP session, with both steady, predictable, rhythmic music, and free improvised music being suitable for the networked environment. Steady, predictable rhythms allow musicians to use leadership – the 'master/slave' approach [8] – to maintain synchronisation, while freely improvised music can embrace the 'acoustic' of the network and work with latency and echoes, etc., to form part of the characteristics of the music produced, what is known as the 'internet performance style' [30].

Creativity and risk-taking, important parts of the ensemble rehearsal and performance processes, are reduced when musicians are physically separated [20], although this may improve as musicians become accustomed to working in NMP environments. Musical expression is also likely to be affected in NMP due to several factors that impact on expression. These include reduced visual contact [23], and therefore reduced perception of body movements [11], and lack of ensemble cohesion [22].

Despite these impacts on the music produced, many musicians working in domestic NMP environments are likely to be improving musicians, rather than professionals, where critical aspects of ensemble playing (such as details of ensemble coherence and expression) are still to be developed. Therefore the NMP environment may be better suited to these musicians than those whose ensemble skills are better developed. It is also possible that these skills could be developed over time, which specifically meet the needs of the NMP environment.

4. A CASE STUDY IN THE SCOTTISH HIGHLANDS AND ISLANDS

4.1 Context

The University of the Highlands and Islands (UHI) is an institution in Scotland that provides traditional university education, distance learning, and blended learning (a combination of face-to-face, online, and video-conferenced learning) in the most northerly part of the country. The population density of the area is low, meaning that many students study at a distance while staying in their local area. The four-year, multi-genre undergraduate Applied Music degree was set up in 2012 to meet the needs of the many musicians in the area, who may want to study alongside a professional touring schedule, for example, or have other personal commitments that mean they do not wish to leave their local communities. As a result, there are now around 80 students based around Scotland and the rest of the UK, who are able to study wherever they are in the world, the majority of

whom work from home. In addition, the lecturers who deliver the course are distributed around the Highlands and Islands.

To facilitate community-building and foster a sense of belonging, the students and staff meet face-to-face four times a year in locations around Scotland, and meet several times per week via video-conference. Students receive individual instrumental lessons, which they may choose to do either face-to-face, or via video-conferencing, depending on their personal preference or situation.

4.2 Access to Ensemble Playing

Ensemble playing is a vital aspect of the students' music education [29], and students are expected to participate in (and facilitate) ensemble activity in their local area. For some students, particularly those in larger communities, this is straightforward, however for students who are more isolated, whether geographically (some students live on remote islands with limited ferry access, for example) or due to their genre (with few local musicians sharing similar musical tastes, for example) this can be more problematic.

NMP has been considered a solution to the problem of bringing student musicians together from the initial stages of designing the degree. Online collaboration is part of several of the modules in the first two years of the course, equipping students with the technical skills to allow themselves to share and collaborate on compositions and performances. File sharing has been the most commonly used method, with students building up recordings by recording themselves playing, and adding tracks to others' recordings. This has been particularly successful when there have been difficulties accessing internet connections that are fast enough for synchronous connections, however does not allow for spontaneous creativity that is possible with synchronous NMP. With internet access improving all the time, currently many students have access to superfast fibre-optic broadband, making synchronous NMP an achievable and attractive form of ensemble activity.

In their third and fourth years of study, students can choose to complete projects using NMP, as part of their research, and in these cases are encouraged to explore the possibilities of synchronous NMP, both within Scotland and beyond. Examples of student projects include collaboration between community groups of traditional fiddlers in the Scottish Borders and Shetland, as well as investigations into instrumental teaching via the internet. The scope of these projects are not limited to working with fellow students within the Highlands and Islands: it is particularly notable that when students have decided to use NMP in their projects they have approached musicians outside the student body in order to foster musical relationships within the wider community, despite the risks and technical difficulties involved.

4.3 Addressing the Challenges

While students are accustomed to using video-conferencing technology both in their education and personal lives, has not naturally developed to include music. NMP is technically possible for students, however currently there are several barriers that prevent it from being more integrated their day-to-day musical activities.

Firstly is the technical ability of students to set up the network connections needed for using specialist NMP software such as Soundjack [7]. This requires some knowledge of router settings and port forwarding, which may be problematic, particularly if

students are involved in community projects that may require the organisation of many connections, and are working from home. In addition there are mental barriers, particularly in relation to the belief that latency makes NMP impossible in domestic settings, which as discussed, is not the case. Given the role that video plays in social interaction in NMP, and the limited role it has for musical coordination, students can be encouraged to prioritise the audio connection when playing (by switching off the video connection and allowing the bandwidth to be used for audio, therefore increasing audio quality), and re-establish video for discussion and other social interaction. Including NMP as an option on the curriculum has helped students to engage with the technology and therefore question their assumptions around latency, however some students are still reluctant to participate in NMP.

Students who are keen proponents of NMP often attempt ambitious projects, working with large groups in multiple locations. As previous discussed, this causes additional challenges of multiple latencies for musicians, and therefore smaller, one-to-one ensemble activity has more chance of success, which can then be built on as confidence and ability in NMP increases.

Possible solutions to these issues include further educating students on the challenges and opportunities of NMP, while placing them in the centre of finding solutions, and encouraging them to be innovative and creative in problem-solving. In particular there are many creative opportunities for using the 'acoustic' of the network (such as echoes and latency) in imaginative ways, rather than focusing on any limitations of NMP on the music played.

In addition, it is important that NMP in this context is used to enhance, rather than replace face-to-face collaborations. It is likely that NMP works better when there are pre-existing relationships between musicians, so for the best possible outcomes it is used in conjunction with traditional forms of ensemble music making, both locally and at residencies. This particular application of NMP may help overcome any negative impact on creativity, by allowing musicians to work both face-to-face and online on preparing for particular performances, or writing new music.

4.4 Future Plans

Although synchronous NMP as a tool for ensemble activity is currently in its infancy on the Applied Music course, it is likely that NMP will play a larger role in the future. Initially, this must start with further encouragement of students to become active participants in NMP and see it as a useful tool for ensemble work, rather than a novelty. Helping students to engage with the academic research in this area, as well as become participants in research projects (both their own, and their lecturers') may help them to see the opportunities that NMP provides them. In addition, a regular timetabled NMP ensemble activity (similar to a traditional music departments' ensemble activities, such as choirs and orchestras), with dedicated technical support for setting up connections, may encourage participation.

NMP also extends the possibilities of collaboration with other institutions, to enhance student cultural exchanges, with links made before, during and after any physical exchanges. This allows for longer-term relationships, whole-community partners, and with less need to travel, which has both economic and environmental benefits.

5. IMPLICATIONS FOR WEB AUDIO

The simplest and most accessible form of NMP is pairs of musicians, both from the musicians' and developers' point of view, as this limits issues with multiple latencies, and these recommendations are based on this context. Previous studies [19] have shown there are some issues that may be useful for NMP software developers to consider in their designs. Firstly the intended musical content of sessions will affect the design of the software. If, for example, it is intended that musicians play as naturally as possible, as if they are in the same room together, then using a delayed feedback approach [8] to maintain a consistent latency that musicians can work with, for example, may be less appropriate than if the musicians are expected to adapt their playing in some way. It is likely that in an informal, duo situation that musicians may not want to consider and adapt to technical challenges by changing the musical content to fit the NMP context, therefore making low audio latency a priority may be more appropriate than managing the latency in other ways. Alternatively, musicians may be happy to embrace the 'network acoustic' and work creatively with latency. Whichever approach is taken, it should be made clear to musicians, so they can adapt their approach accordingly and select an NMP system that is most appropriate to their needs.

Software developers may also want to consider whether to include video in an NMP system, and whether any video is synchronised to the audio feed. In the context of informal NMP, musicians tend to use the video for discussion around musical issues, rather than when playing [21]. If the decision is made to keep audio latency as low as possible (by therefore probably avoiding compression), it may be possible to prioritise the audio feed over the video feed when the musicians are playing (either reducing the quality of the video when playing, or removing it altogether), but restore it during discussion. This has the advantage of increasing the bandwidth allocation for the audio, when the musicians need it most. If no video is included, musicians have the option of using other proprietary video conferencing software alongside their NMP, although this will reduce the bandwidth available for the NMP software.

6. CONCLUSIONS

There is a perception that NMP is only successful with high-quality audio and video equipment, and extremely fast network conditions, to allow for real-time musical interactions that mimic those in a single physical space. This paper argues that this is not the case: meaningful musical relationships can be built and maintained using typical domestic equipment and internet connections (that are improving all the time). This means that NMP is a useful tool for musicians who are isolated physically (by geography, for example) or socially (through illness, for example). Minimal equipment is required, although some technical expertise is needed to set up connection initially, with the biggest barrier to use being the connection of the NMP software itself.

NMP may have an impact on the music played, in terms of what is suitable for the network environment, but this environment also gives opportunities for musical creativity that would not be possible in a conventional rehearsal space. These opportunities are available for musicians playing any instrument, and at any level, and are potentially more suited to those who do not have highly developed ensemble skills in a traditional face-to-face setting. The online environment also has particular opportunities for community-building around the musical content, which would be beneficial for isolated musicians.

The Applied Music degree at the University of the Highlands and Islands has given an opportunity for students and staff to successfully explore the possibilities for the use of NMP for isolated musicians. Recent improvements in broadband infrastructure is now allowing synchronous NMP to be explored further, particularly in relation to preparation for ensemble work between face-to-face residencies, and to continue the community- and relationship-building that currently takes place. This model has the possibility to be extended into community music, and more informal ensemble practice.

7. REFERENCES

- [1] Chrisoula Alexandraki. 2019. Experimental investigations and future possibilities in network-mediated folk music performance. In *Computational Phonogram Archiving. Current Research in Systematic Musicology*, Rolf Bader (ed.). Springer International Publishing, Cham, Switzerland, 207–228. <https://doi.org/10.1007/978-3-030-02695-0>
- [2] Chrisoula Alexandraki and Demosthenes Akoumianakis. 2010. Exploring new perspectives in network music performance: The DIAMOUSES framework. *Computer Music Journal* 34, 2: 66–83. <https://doi.org/10.1162/comj.2010.34.2.66>
- [3] M. Blank and Jane W. Davidson. 2007. An exploration of the effects of musical and social factors in piano duo collaborations. *Psychology of Music* 35, 2: 231–248. <https://doi.org/10.1177/0305735607070306>
- [4] Andrew R Brown. 2010. Network Jamming: Distributed performance using generative music. In *NIME 2010 Proceedings of the International Conference on New Interfaces for Musical Expression*, 283–286.
- [5] Juan-Pablo Cáceres and Chris Chafe. 2010. JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research* 39, 3: 183–187. <https://doi.org/10.1080/09298215.2010.481361>
- [6] Juan-Pablo Cáceres and Robert Hamilton. 2008. To the edge with China: Explorations in network performance. In *ARTECH 2008, 4th International Conference on Digital Arts*, 61–66.
- [7] Alexander Carôt. 2015. SoundJack. Retrieved October 5, 2015 from <http://www.soundjack.eu>
- [8] Alexander Carôt and Christian Werner. 2009. Fundamentals and principles of musical telepresence. *Journal of Science and Technology of the Arts* 1, 1: 26–37. <https://doi.org/10.7559/citarj.v1i1.6>
- [9] Chris Chafe. 2011. Living with net lag. In *AES 43rd International Conference*, 1–6.
- [10] Cockos Incorporated. 2018. NINJAM. Retrieved July 17, 2018 from <https://www.cockos.com/ninjam/>
- [11] Jane W. Davidson. 2005. Bodily communication in musical performance. In *Musical Communication*, D Miell, R. a. R. Macdonald and David J. Hargreaves (eds.). Oxford University Press, Oxford. <https://doi.org/10.1093/acprof>
- [12] Michael Dessen. 2010. New polyphonies: Score streams, improvisation and telepresence. *Leonardo Music Journal* 20: 21–23. https://doi.org/10.1162/LMJ_a_00007
- [13] Steve Dillon and Andrew R Brown. 2010. Access to meaningful relationships through virtual instruments and ensembles. *Proceedings of the ISME Commission for*

Community Music Activity: CMA XII Harmonizing the Diversity that is Community Music Activity, July.

- [14] Genevieve A. Dingle, Christopher Brander, Julie Ballantyne, and Felicity A. Baker. 2013. "To be heard": The social and mental health benefits of choir singing for disadvantaged adults. *Psychology of Music* 41, 4: 405–421. <https://doi.org/10.1177/0305735611430081>
- [15] Carlo Drioli, Claudio Allocchio, and Nicola Buso. 2013. Networked performances and natural interaction via LOLA: Low latency high quality A/V streaming system. In *Information Technologies for Performing Arts, Media Access, and Entertainment. ECLAP 2013. Lecture Notes in Computer Science*, P. Nesi and R. Santucci (eds.). Springer, Berlin, 240–250. https://doi.org/10.1007/978-3-642-40050-6_21
- [16] Golo Föllmer. 2005. Electronic, aesthetic and social factors in Net music. *Organised Sound* 10, 03: 185–192. <https://doi.org/10.1017/S1355771805000920>
- [17] Jane Ginsborg and Elaine C. King. 2012. Rehearsal talk: Familiarity and expertise in singer-pianist duos. *Musicae Scientiae* 16, 2: 148–167. <https://doi.org/10.1177/1029864911435733>
- [18] Elizabeth Howell. 2013. Astronaut and musician perform 1st original duet from space and earth. *Space.com*. Retrieved September 3, 2018 from <https://www.space.com/19694-hadfield-song-duet-space.html>
- [19] Miriam Iorwerth. 2019. Playing together, apart: An exploration of the challenges of Networked Music Performance in informal contexts. Glasgow Caledonian University.
- [20] Miriam Iorwerth and Don Knox. 2019. Playing together, apart: Musicians' experiences of physical separation in a classical recording session. *Music Perception* 36, 3: 289–299.
- [21] Miriam Iorwerth and Don Knox. 2019. Musicians' gaze in Networked Music Performance. *In preparation*.
- [22] Peter E. Keller. 2014. Ensemble performance: Interpersonal alignment of musical expression. In *Expressiveness in music performance: Empirical approaches across styles and cultures*, Dorottya Fabian, Renee Timmers and Emery Schubert (eds.). Oxford University Press, Oxford, 260–282. <https://doi.org/10.1093/acprof>
- [23] Peter E. Keller and Mirjam Appel. 2010. Individual differences, auditory imagery, and the coordination of body movements and sounds in musical ensembles. *Music Perception* 28, 1: 27–46. <https://doi.org/10.1525/mp.2010.28.1.27>
- [24] M. C. Lim. 2013. In pursuit of harmony: The social and organisational factors in a professional vocal ensemble. *Psychology of Music* 42, 3: 307–324. <https://doi.org/10.1177/0305735612469674>
- [25] Carsten Magerkurth, Timo Engelke, and Maral Memisoglu. 2004. Augmenting the virtual domain with physical and social elements: Towards a paradigm shift in computer entertainment technology. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 163–172.
- [26] Roger Mills. 2010. Dislocated sound: A survey of improvisation in networked audio platforms. In *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010), Sydney, Australia*, 186–191.
- [27] Roger Mills. 2011. Ethernet Orchestra: Interdisciplinary cross-cultural interaction in networked improvisatory performance. In *The 17th International Symposium on Electronic Art*.
- [28] J Keith Murnighan and Donald E Conlon. 1991. The dynamics of intense work groups: A study of British string quartets. *Administrative Science Quarterly* 36, 2: 165–186.
- [29] Quality Assurance Agency for Higher Education. 2016. *Subject Benchmark Statement - Music*.
- [30] 30. Alain Renaud, Alexander Carôt, and Pedro Rebelo. 2007. Networked music performance: State of the art. In *AES 30th International Conference*, 1–7.
- [31] Michael Rofo and Federico Reuben. 2017. Telematic performance and the challenge of latency. *Journal of Music, Technology and Education* 10, 2+3: 167–183. <https://doi.org/10.1386/jmte.10.2-3.167>
- [32] Franziska Schroeder and Pedro Rebelo. 2009. Sounding the network: The body as disturbant. *Leonardo Electronic Almanac* 16, 4: 1–10.
- [33] John Suler. 2004. The online disinhibition effect. *CyberPsychology & Behavior* 7, 3: 321–326. <https://doi.org/10.1089/1094931041291295>
- [34] Gualtiero Volpe, Alessandro D'Ausilio, Leonardo Badino, Antonio Camurri, and Luciano Fadiga. 2016. Measuring social interaction in music ensembles. *Philosophical Transactions of the Royal Society B: Biological Sciences* 371, 1693. <https://doi.org/10.1098/rstb.2015.0377>
- [35] Matthew Wright. 2005. Open Sound Control: an enabling technology for musical networking. *Organised Sound* 10, 03: 193. <https://doi.org/10.1017/S1355771805000932>

Sounds Aware: A Mobile App for Raising Awareness of Environmental Sound

Tate Carson
Louisiana State University

ABSTRACT

Sounds Aware is a web application that runs on a smartphone and uses machine learning to detect human-made sound (anthrophony) and masks it with ambient music as a user walks around their environment. A study was completed to determine if this app is an effective means of shifting a user's attention away from anthrophony and to biological (biophony) and geophysical (geophony) sounds while walking and encouraging environmental awareness. Though the model is pre-trained with the author's local environmental sounds, the user can train the model further on their unique soundscape so that each user gets a personalized experience. After the training process, the user can listen to ambient music based on traits of the surrounding anthrophony. If the app senses less anthrophony and more biophony or geophony, then the music fades away, bringing the user's attention to the anthrophony.

1. INTRODUCTION

Composer David Dunn poses an inspirational question:

To what extent might the technologies of communication, art and entertainment serve as 'protheses' that would provide us with experiences of wilderness that would not only enrich our human identity but help us to preserve and expand the domain of the non-human world [5]?

The goal of *Sounds Aware* is to bring the user's awareness to the geophonic and biophonic soundscape, which is often so masked by noise pollution that it has fallen out of awareness for many of us. *Sounds Aware* seeks to shift the user's concept of nature to something that has no starting or ending point; it is all around us. The app brings awareness by focusing attention on the environment. Because of the predominance of eye culture [3], our reliance on seeing rather than listening as a primary means of sensing the world, it is

a lot to ask of a person who might be uninterested in acoustic ecology to "just listen" to their environment. But, if you give them a tool that urges listening in the quieter places, where the natural world will be more audible, there is a better chance of them engaging with those sounds because the app focused their perception. *Sounds Aware* is a means of technologically mediated "ear cleaning," as described by R. Murray Schafer in *Ear Cleaning: Notes for an Experimental Music Course* [13].

A 2011 World Health Organization (WHO) report found that "there is overwhelming evidence that exposure to environmental noise has adverse effects on the health of the population [10]." *Sounds Aware* shifts a user's attention away from noise pollution and to nature, which may help mitigate adverse health effects caused by noise pollution. Psychologist Stephen Kaplan found that stress reduction can be aided by the experience of the natural environment by providing a 'restorative environment' that reduces the fatigue caused by directed attention [8]. Kaplan did not mention sound directly, but a recent study by Eleanor Ratcliffe *et al* [12] has extended his research to show that certain bird sounds may provide restorative benefits. While a reduction in environmental noise at the source would be the best way to solve noise pollution, masking the noise is a stopgap solution. A masking solution has been implemented by several projects [9, 15] but not yet with a mobile device. *Sounds Aware* implements a similar idea but with a mobile phone.

2. MOTIVATION

2.1 Soundscape

R. Murray Schafer suggests that we should listen to the environment as a musical composition. He describes urban and rural soundscapes as lo-fi and hi-fi. A rural landscape is hi-fi because there is a low noise level and allows one to hear more clearly. When in lo-fi (urban) soundscapes we are dealing with a lot of sound masking and getting less discernible aural information [14]. *Sounds Aware* brings attention to that urban noise by masking it with music, possibly reducing its negative effects as described by the WHO report [10]. The music of *Sounds Aware* and Schafer's 'environment as musical composition' combine as a duet to create a new unheard work.

Through our personal listening devices (smartphones,



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

iPods etc.) many of us try to block out the environment with music. Perhaps this is because we mostly hear noise pollution? This tendency makes *Sounds Aware* a viable tool for focusing attention because we are already used to making the choice to use personal listening devices. This is a tool that lets you continue to do that but brings your attention to the nature you might otherwise miss [18].

What could be gained besides a reverence for nature from *Sounds Aware*? Many of us use music as a way of blocking out silence. Rather than listening to the “silence,” which might give us a chance to be silent in our minds, we fill up the space with music [18]. Perhaps *Sounds Aware* can re-contextualize nature sounds to be engaging enough to pay attention to, to make the city soundscape seem uninteresting compared to the natural soundscape. *Sounds Aware* satisfies both ideas. By focusing the users’ attention on environmental sounds, they will hopefully gain a greater appreciation for those sounds and then seek to design their world in a way such that my app is unnecessary. This redesign could take the form of urban planning that takes into account research from soundscape ecology [11].

2.2 Audio walks

Sounds Aware is further classified as an audio walk because it is a soundwalk mediated by technology. Johanna Steindorf describes audio walks as “experiments and works that combine walking and listening to a mediated soundscape over headphones [16].” That an audio walk takes place using headphones is important because it adds “a second layer of private sound to any place and situation, therefore transforming or enhancing the current spatial experience [16].” *Sounds Aware* mediates the public space through remediation of noise. The unique part of this audio walk is that the user is meant to be more aware when the composed ambient music is off. It is almost like a negative audio walk.

2.3 Ubiquitous listening

Urban dwellers are already often walking around with headphones in, trying to mediate the acoustic environment. Iain Chambers writes about the first entirely private listening experience, which provides a starting point for my app, the Walkman. This private experience that many of us first experienced with a Walkman, then CD players, is now experienced with mobile phones:

With the Walkman strapped to our bodies we confront what Murray Schafer in his book *The Tuning of the World* calls a “soundscape,” a soundscape that increasingly represents a mutable collage: sounds are selected, sampled, folded in and cut up by both the producers (DJs, rap crews, dub masters, recording engineers) and the consumers (we put together our personal play lists, skip some tracks, repeat others, turn up the volume to block out the external soundtrack or flip between the two). Each listener/player selects and rearranges the surrounding soundscape, and, in constructing a dialog with it, leaves a trace in the network [7].

Chambers notes that the music people were listening to was collage based, as was the way they were listening to it. He surmised that the way people would listen while walking around would change depending on the sounds of the outside

environment. The sounds around the listener became part of the collage of the sounds playing on the Walkman.

3. RELATED WORK

The Quiet Walk [1], by Alessandro Altavilla and Atau Tanaka, is locative audio walk artwork for explorations of the urban landscape, where the goal is to find the quietest place in an urban location. The app notifies users if the surrounding sounds are too high. It also records the GPS locations of quiet places that are found so that the user can view a sound map of their walk. This might be the project with the closest concept to *Sounds Aware*, but there are some key differences. *The Quiet Walk* only records loudness levels and does not categorize sounds. Because of that, a loud anthropogenic sound is treated the same as a sound not human-made, which probably produced false positives. This more intelligent system was proposed in the conclusion and was probably not tried because of technological limitations of the time.

Hazzard, Benford, and Burnett created *Ambient Walk* [4], a mobile application that encourages mindful walking through sonification of biophysical data. The app plays ambient music dependent on users breathing patterns and the pace of walking. The authors intend for the music to keep a user aware of their ‘balancing status’ between walking and breathing. The intentions of *Ambient Walk* are very similar to *Sounds Aware*, though its intention is to raise the user’s awareness of one’s own mindfulness, instead of the soundscape.

There are many examples of smartphone apps that use GPS to teach the user about a historical subject. *Sounds Aware* does not use GPS but has similarities to apps that do. *Walk With Me* is a site-specific musical smartphone app that uses GPS to create geo-tagged markers to give form to a composition. This app is like *Sounds Aware* because it uses a microphone to sense the environment, but *Walk With Me* takes the step to alter that environment and turn those environmental sounds into part of the composition in a more conscious way. Specific places are chosen for their auditory characteristics and the piece is composed by giving certain locations specific acoustic qualities. The proximity to this geo-tagged location alters the signal processing, increasing the intensity as a user gets closer. *Walk With Me* does not have the same effect as *Sounds Aware* in raising awareness of a soundscape because it alters the soundscape instead of reframing it [17].

4. USER INTERACTION

Sounds Aware is accessed by going to <https://walking.netlify.com> in a web browser on a smartphone. It requires Internet access to download the default data set. After that, the app will work offline, so it is appropriate for various network conditions. Headphones are required so that the microphone on the phone does not pick up the music playback. Headphones with a microphone are preferred so that if the user wants, they can put their phone in their pocket while walking.

When a user first opens the application, they will see it guess the surrounding sounds based on a pre-trained data set. When assured that the microphone is working, the user can then start the music by clicking the toggle switch (see Figure 1a). The music now responds to what the surround-

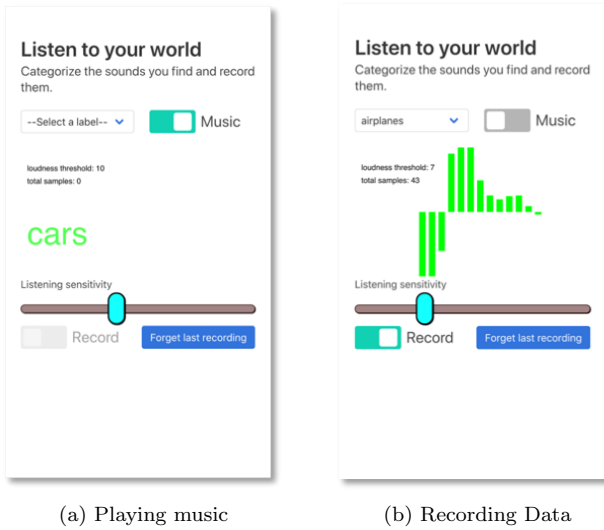


Figure 1: Two states of the application

Tag category	Tags
Geophony	wind, other weather, rain
Anthrophony	cars, construction, human-speech, AC, airplanes
Biophony	insects, birds, large animal

Table 1: Data Tags

ing sounds. The user can adjust the listening sensitivity of the microphone to their liking to match the acoustic environment if it is particularly quiet or noisy. After testing the success of the system in interpreting the user’s environment, the user can now add their own training data (see Figure 1b). For this, the user will select a sound category such as a car. Then the user will wait for a car to drive by and then record it by clicking the record toggle. This will make the system more accurate in listening to the user’s specific environment. Users are not currently able to add their own sound category tags.

5. DESIGN

5.1 Technical design

Sounds Aware is built with Tone.js, a Web Audio API framework. There are a few downsides to web-based apps, such as cross-browser compatibility issues with microphone input, but a web app was chosen because a user might be more likely to try it if they do not have to download an app.

Each user starts off with an author-defined database of tagged sounds. To create this database each recording made was tagged (see Table 1) with a general sound category. Those sounds were then placed into broader categories of origin. This allowed the composition to treat sounds from different sources—geophony, anthrophony, and biophony—differently. Table 2 shows the default data set. Footsteps were treated as silence because the system needed to work when a user was moving. If they had been added to the anthrophony the system would only work if the user was still and silent; this was a trade-off to allow for user mobility, allowing them to find various soundscapes. More data

Tags	Data points
wind	358
footsteps	361
birds	108
rain	107
cars	284
construction	62
Total	1280

Table 2: Training Data







Category	Effect	
Geophony 	amplitude mapped 	amplitude modindex harmonicity
Anthrophony 		amplitude -3
Biophony 		amplitude -60

Figure 2: Mapping of tag category to music

could have been added to this data set to make tags more accurate, but for this proof of concept the most important tags were accurate enough for users to hear a result in the sounds.

The app uses machine learning to identify sounds. The algorithm used is k-nearest neighbor. A Mel-Frequency Cepstral Coefficients (MFCC) audio feature was used to compare sounds, implemented by Meyda.js¹, a JavaScript feature extraction library. Using Meyda.js allowed the processing to be done on the client, allowing *Sounds Aware* to work offline when necessary.

5.2 Composition design

The musical composition of *Sounds Aware* is influenced by ambient music. Synthesized sounds were used that would not be too jarring to jump in and out of and did not have an obvious beginning or ending. This type of synthesized sound blends in with the surrounding acoustic environment as a composition. The synthesized sounds are simple frequency modulation synthesis with reverb and delay effects. They are tuned to just intervals so they are more likely to coincide with tunings in nature, influenced by Aeolian practices [2] and La Monte Young [6].

The app maps the loudness² of the acoustic environment to the amplitude of the ambient composition (see Figure 2). The previous 200 loudness values are averaged and the am-

¹<https://meyda.js.org/audio-features>

²A perceptual feature from <https://meyda.js.org/audio-features.html>

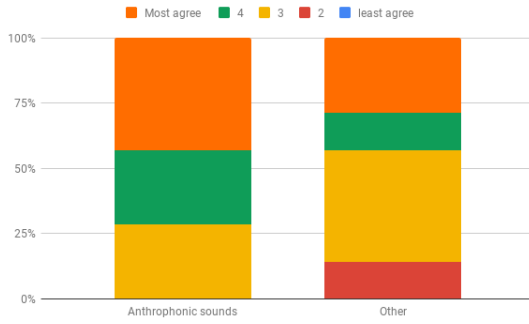


Figure 3: Sounds a user noticed in their neighborhood

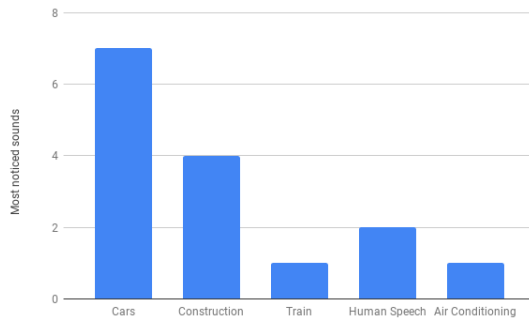


Figure 4: Noticed anthropophonic sounds

plitude ramps to a given value over one second for signal smoothing. If *Sounds Aware* hears an anthropophonic sound, the amplitude is faded up to -3 dB. If it hears a geophonic sound, the amplitude of the geophonic sound is mapped onto the synthesized sounds amplitude, creating a wind chime effect. Geophonic sounds also affect the modulation index and harmonicity of the frequency modulation synthesis, creating a variety of timbres depending on the character of the current external soundscape. If a biophonic sound is recognized, the amplitude of the ambient wash is faded down to -59 dB, which is perceptibly silent when listened to in an urban environment.

6. EVALUATION

The study begins with a survey designed to understand prior knowledge of environmental sounds and likelihood of interest in the project. Seven users took part in the study. The participants were anonymous and no demographic data was collected. The survey was conducted through a web-based question form. The participants were found through a university class and word of mouth.

Users were first asked, “When walking in your neighborhood do you notice a greater percentage of human-produced (anthrophony) sounds over biological (biophony), geophysical (geophony)?” Most users noticed anthropophonic sounds and slightly less noticed geophonic and biophonic sounds (see Figure 3). When asked the question, “Which anthropophonic sounds in your neighborhood are the most noticeable?” users noted cars and construction the most (see Figure 4).

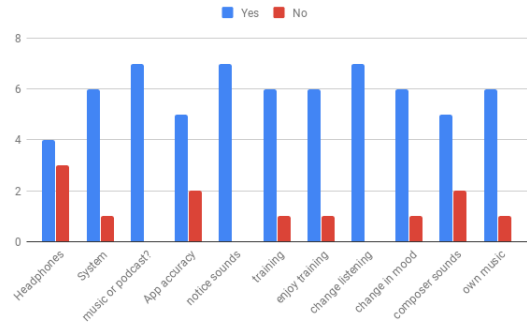


Figure 5: Exit Survey

For the exit survey, a number of questions were asked about how the app worked for the users and if it changed their awareness. The following questions were asked and represented in bar graph form (see Figure 5):

- Do you often walk around your neighborhood with headphones in?
- If there was a system to playback music when anthropophonic sound was detected but stop the music when biophonic or geophonic sound was detected, would you use it?
- Would it be more useful if you could pick the music, or even listen to a podcast?
- Did the app respond fast enough or accurately enough to cover up anthropophonic sounds?
- When the app did not detect any anthropophonic sounds, so did not play music, were you more likely to notice the geophonic and biophonic sounds?
- If the app was not accurate at first were, you able to train it with more data to make it more accurate to your soundscape?
- If you did try to train the system, did you find enjoyment in the act of training it?
- If you trained the app, did the act of training change the way you listened to the environmental sounds around you?
- Did you notice a change in mood or thought processes after having your attention drawn away from the noise and to the sounds of nature?
- Did you enjoy the composer-created sounds?
- If not, would the application be experienced better by being able to replace noise with your own music?

One issue with the study might be that a number of participants actually do not often walk around with headphones in. I suspect that in a much more urban environment than Baton Rouge, Louisiana, many more people would report walking while using headphones. Most participants were positive about the chance to try the app and also would like a way to use their own media with it. Five out of the seven users also thought the app responded quickly enough

to be meaningful and the musical response matched closely with the current natural soundscape. All of the participants responded that they were likely to notice geophonic and biophonic sounds when there was a silence, which is a positive finding. Only one user responded that they were not able to retrain the app to make it more sensitive to certain tags and most enjoyed training the system and changed their listening during training.

7. CONCLUSIONS AND FUTURE WORK

Sounds Aware is a web application meant to increase a user's awareness of the biophonic and geophonic sounds in their urban environment. In limited studies, users reported increased awareness of environmental sounds. Having people use it over a longer period of time, and then surveying with more questions, would be a future avenue of research.

The app could have been made more robust by using native phone capabilities. A future version of *Sounds Aware* could be built using Capacitor³, a native bridge for cross-platform mobile apps. This would allow the targeting of multiple platforms (web, iOS, Android) with the same code base and more robust access to native functionality such as the microphone. Usability improvements could be made such as the ability to add user defined tags so a user can record unique sounds in their location.

Other future work includes using an edge TPU computing board, such as the Coral Dev Board⁴, to do machine learning computation on. An edge TPU (Tensor Processing Unit) would allow for more powerful machine learning capabilities in a portable form factor. This will make the system much more accurate and be able to recognize many more types of sounds. A study will be done to find out if this extra piece of hardware provides more benefits than costs.

8. REFERENCES

- [1] A. Altavilla and A. Tanaka. The quiet walk: Sonic memories and mobile cartography. In *Proceedings of the 9th sound and music computing conference*, pages 157–162, Copenhagen, Denmark, 2012.
- [2] R. Bandt. Taming the wind: Aeolian sound practices in australasia. *Organised Sound*, 8(2).
- [3] J.-E. Berendt. *The Third Ear: On Listening to the World*. H. Holt, New York, 1992.
- [4] S. Chen, J. Bowers, and A. Durrant. 'Ambient walk': a mobile application for mindful walking with sonification of biophysical data. In *Proceedings of the 2015 British HCI Conference*, pages 315–315, Lincoln, United Kingdom, 2015. ACM.
- [5] David Dunn. Wilderness as Reentrant Form: Thoughts on the Future of Electronic Art and Nature. *Leonardo*, (4):377, 1988.
- [6] K. Gann. La monte young's the well-tuned piano. *Perspectives of New Music*, 31(1):134–162, 1993.
- [7] Iain Chambers, Christoph Cox, and Daniel Werner. The Aural Walk. In *Audio Culture: Readings in Modern Music*, pages 129–133. Bloomsbury Academic, revised edition edition, 2017.
- [8] S. Kaplan. The restorative benefits of nature: toward an integrative framework. In: *Journal of Environmental Psychology*, 15:169–182, 1995.
- [9] G. Licitra, M. Cobiainchi, and L. Brusci. Artificial soundscape approach to noise pollution in urban areas. *Proceedings of the Proceedings of INTER-NOISE*, page 11, 2010.
- [10] W. H. Organization. *Burden of disease from environmental noise: quantification of healthy life years lost in Europe*. World Health Organization, 2011.
- [11] B. C. Pijanowski, L. J. Villanueva-Rivera, S. L. Dumyahn, A. Farina, B. L. Krause, B. M. Napoletano, S. H. Gage, and N. Pieretti. Soundscape ecology: the science of sound in the landscape. *BioScience*, 61(3):203–216, 2011.
- [12] E. Ratcliffe, B. Gatersleben, and P. T. Sowden. Bird sounds and their contributions to perceived attention restoration and stress recovery. *Journal of Environmental Psychology*, 36:221–228, Dec 2013. 00146.
- [13] R. Schafer. *Ear Cleaning: Notes for an Experimental Music Course*. Clark & Cruickshank, 1969.
- [14] R. M. Schafer. *The soundscape: Our sonic environment and the tuning of the world*. Simon and Schuster, 1993.
- [15] D. Steele, E. Bild, C. Tarlao, and C. Guastavino. Soundtracking the public space: Outcomes of the musikiosk soundscape intervention. *International journal of environmental research and public health*, 16(10), 2019.
- [16] J. Steindorf. Walk-Along with a Mediated Presence: The Audio Walk as a Mobile Method. *Wi: Journal of Mobile Media*, 2017.
- [17] R. Van Rijswijk and J. Strijbos. Sounds in Your Pocket: Composing Live Soundscapes with an App. *Leonardo Music Journal*, pages 27–29, 2013.
- [18] K. Wrightson. An introduction to acoustic ecology. *Soundscape: The journal of acoustic ecology*, 1(1):10–13, 2000.

³<https://capacitor.ionicframework.com/>

⁴<https://coral.withgoogle.com/products/dev-board/>

Composing Spatial Music with Web Audio and WebVR

Cem Çakmak

Rensselaer Polytechnic Institute
110 8th Street, Troy NY
cakmao@rpi.edu

Rob Hamilton

Rensselaer Polytechnic Institute
110 8th Street, Troy NY
hamilr4@rpi.edu

ABSTRACT

Composers have been exploring complex spatialization techniques within multi-channel sound fields since the earliest days of electroacoustic and electronic music. However the reproduction of such works outside of highly specified concert halls and academic research facilities, or even their accurate reproduction within those spaces, is difficult and unpredictable at best. Tools such as Omnitone combine the reach and simplicity of web browsers with the flexibility and power of higher-order ambisonics (HOA) and binaural rendering, ensuring greater accessibility for existing spatial electronic musical works as well as acting as a platform upon which future works for virtual sound fields can be implemented. This paper describes the technical design and artistic conception of one such spatial composition for binaural listening and immersive visuals on the web - *od* - produced in the CRAIVE-Lab, an immersive audio-visual facility.

1. INTRODUCTION

Spatial, or multi-channel, approaches in electronic music composition are as old as the practice itself. As such works furthered experimentation in sound, light and multimedia, they surrounded, and sought to immerse the audience in unique spatiotemporal experiences. Distinct as these performances were, today they are often talked about while only a handful of people have truly experienced them on site. Spatial electronic music often depends on research institutions and patronage due to their experimental nature and funding required, while the audience ranges from highly trained ears to unassuming visitors. Furthermore, such works have always remained a niche and their recreation unfeasible. This paper breaks down a recent multimedia work, *od*, to illustrate a more accessible and inclusive way to experience spatial electronic music based on Web Audio and WebVR technologies. In order to achieve this, the authors seek to bring together a number of online and offline tools together in an artistically meaningful way.

Emmerson identifies two traditions of diffusion in electronic music: the idealist and realist approaches [5]. In short, the idealist approach works toward conveying the

composer's vision as accurately as possible to the listener, whereas the realist approach emphasizes the diversity of the hearing experience among individuals within the listening space. This difference in experience is due to a variety of factors that include audience seating, architecture of the space, or external disruptions; since such works focus on timbre and space as opposed to traditional musics that rely predominantly on pitch and rhythm, they are much less tolerant to disturbances, faulty equipment, or lower qualities of production [15].

Regardless of the compositional approach, spatial works remain poorly documented in terms of the audience experience, and their reconstruction remains a challenge. Thus the main goals of the research are as follows:

- Utilize contemporary web technologies to facilitate the composition and dissemination of a novel, spatial music piece.
- Exercise the above-mentioned idealist approach in a way that includes all listeners in an idealized listening situation.
- Enable access to a specific physical site that is typically difficult to visit.
- Augment the immersiveness of a real-world location with VR production techniques.

2. BACKGROUND

Beyond stereophony lies an infinite playground; one where loudspeakers work together not only to form a directional image, but to encompass the listeners and innovate new spatial complexities. While ambisonic systems aim to recreate a spherical soundscape as accurately as possible, composers also use unconventional speaker arrangements, where loudspeakers are treated as instruments that contribute to the music with their characteristics and form a relationship with their surroundings. The "acousmonium", first designed in 1974 at GRM [7], is an asymmetrical approach in multi-channel loudspeaker distribution where the focus is on combining loudspeakers that vary in size, shape, response and function together, and showcase not only auditory but also visual novelty. Furthermore, Iannis Xenakis' 1971 polytope *Persepolis* was set in the dark of night outdoors, among the ruins of the ancient Persian city of the same name. In addition to the 8-channel composition diffused over 59 speakers, the performance utilized lasers, spotlights, as well children parading with torches over the hills [10]. Besides real-world locations, specific constructions were designed and built,



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

bringing together composers, architects, and technicians in order to create unique yet ephemeral structures for spatial performances; notable examples include the Philips Pavilion in EXPO '58 Brussels, or the Pepsi Pavilion in EXPO '70 Osaka [16].

Although the works mentioned above are often mythicized as significant achievements in electronic music history, recreating such pieces as they were meant to be listened to is a delicate task, if even possible. Some of these compositions have been released in stereo, but restaging *Persepolis*, for example, a piece riddled with themes of Zoroastrian fire-worshipping, could possibly have extra-musical ramifications in modern day Iran. As for the EXPO pavilions, there were a number of attempts to physically and virtually reconstruct the Philips Pavilion and the Edgard Varèse's piece *Poème Electronique* performed inside the structure in order to assess the works retrospectively [14]. These however, are often expensive undertakings with dubious fidelity to the original.

With the ongoing development of VR and web technologies, researchers are challenged to create new, innovative forms of spatial interactions. As these technologies become more widespread, new spatial artworks can potentially become more accessible and less ephemeral. Fyfe et al. [6] combine web audio streams with motion tracking and binaural audio to create telepresence for networked collaborations. Kermit-Canfield [11] proposes a configurable diffusion tool for building virtual acousmonia and encode the output using ambisonics for transparent speaker arrangements. Çamcı et al. [4] introduce a stylized interactive system with UI controls and visual representations for composing detailed virtual sonic environments with an implementation of Web Audio API. Lastly, personalized head related transfer functions (HRTFs) for spatial sound reproduction are being implemented in Web Audio; Geronazzo et al. propose a framework based on Unity and the Web Audio API for new immersive experiences on mobile VR [8]. Likewise, this research brings together a number of tools and instruments available to construct a contemporary spatial music piece, available to experience through desktop, mobile or head-mounted devices.

3. TOOLS AND INSTRUMENTS

3.1 Omnitone

Written with Web Audio API, Omnitone¹ is a JavaScript implementation for ambisonic decoding up to the third order and binaural rendering on the web browser. Illustrated in Fig. 1, multi-channel ambisonic files are streamed in via `AudioBufferSourceNode`, and the head position is translated to a rotation matrix via either user interaction or sensor data (or in this project's case, A-Frame camera position). Binaural rendering is handled by `Convolver` and `GainNode` interfaces, both native to Web Audio. In line with Google's spatial media specifications, the decoded ambisonic signals for each ear pass through head related transfer functions (HRTFs) based on SADIE filters² to simulate binaural hearing.

¹<https://googlechrome.github.io/omnitone/>

²<https://www.york.ac.uk/sadie-project/GoogleVRSADIE.html>

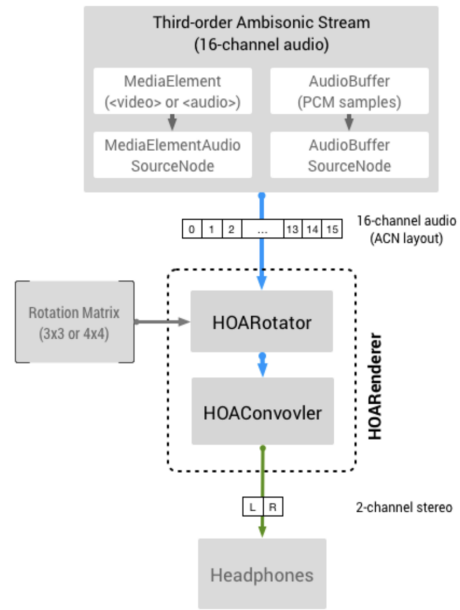


Figure 1: Diagram of Omnitone's third-order ambisonic to binaural rendering

3.2 A-Frame

The A-Frame library³ is an open source, three.js framework for WebVR. Developed by the Mozilla VR team in order to make VR content development more accessible, A-Frame provides higher-level coding within a special HTML tag, `<a-scene>`, to implement and manipulate VR content without having to manage complex WebGL code. Furthermore, A-Frame works across a range of mobile, desktop and head-mounted devices [12]. The A-Frame framework has an entity-component system architecture, common for game development applications; most code lives within registered components that are plugged into entities to describe their specific attributes. Unfortunately, A-Frame does not support WebM⁴ as of today, an extremely efficient media format for video playback on browsers.

3.3 CRAIVE-Lab

Collaborative-Research Augmented Immersive Virtual Environment Laboratory, or CRAIVE-Lab [13], is a state-of-the-art interactive immersive environment operating in Rensselaer Technology Park. The surround screen, shaped as a rounded rectangle, totals a resolution of 15360x1200 pixels. The panoramic image is front-projected by eight Canon REALiS WUX400ST Pro short-throw LCoS projectors mounted on the grid above. These projectors are calibrated and warped with Pixelwix to create a smooth continuous image along the entire projection surface. Furthermore, along the back of the screen is a 128-channel wave field synthesis array set up with additional speakers mounted on the above grid for HOA projection support, although the on-site sound setup is not employed in this research.

This audiovisual environment is often used in an architectural context and research in interaction, where a specific

³<https://aframe.io/>

⁴<https://www.webmproject.org/>

site is reproduced visually as well as aurally; given that impulse responses taken from the real site modeling its reverberance is possible. Furthermore, some projects add in new designs within the site footage, creating hyper-real immersive landscapes. But, in the context of this project, the goal is to export CRAIVE-Lab from its real-world location and enable people to virtually experience works made within it, as opposed to its main function where real-world sites are placed within the immersive space.

3.4 Max & SPAT5

IRCAM’s *Spatialisateur*, or SPAT [3], is a library of Max objects written in C++ for spatializing sound in real-time for ambisonics or nearfield binaural synthesis, and generating artificial reverberations for room effect. SPAT offers a dynamic 3D environment for organizing and manipulating sound sources, in addition to modeling loudspeaker arrangements for real-time synthesis and diffusion. The SPAT library can be used for live performances, mixing, post-production, installations, VR and other applications. The 5th version of SPAT [2] implements the open sound control (OSC) protocol for the processors throughout the library, along with more detailed documentation, improved HOA features, cross-operability with VR SDKs and other aspects.

3.5 GoPro Omni

GoPro’s Omni rig encapsulates and powers six GoPro Hero 4 cameras on each side of its cube-shaped structure in order to record spherical scenes. By uploading a special firmware provided by the manufacturer, all cameras can be synced and controlled through a single remote control. To stitch all the footage together to construct a spherical image, we used Kolor softwares AutoPano Giga 4.4 and AutoPano Video Pro 2.6, all deprecated since September 2018. Finally, the stitched footage is further processed with MantraVR⁵, an Adobe After Effects plugin for VR content production.

4. COMPOSITION

The composition process involves navigating through different environments using the tools described in section 3. This task is mainly split into two parts due to their respective media types: the audible and the visual. As illustrated in Fig. 2, the design of the audio and visuals are mostly independent from each other during composition, joined together and synced between A-Frame and Omnitone.

4.1 Auditory Design

Despite the strong influence of 20th century spatial works on the aesthetics of the project as mentioned in 2, the central musical idea comes from a stereo drone installation by La Monte Young, *The Base 9:7:4 Symmetry in Prime Time When Centered above and below The Lowest Term Primes in The Range 288 to 224 with The Addition of 279 and 261 in Which The Half of The Symmetric Division Mapped above and Including 288 Consists of The Powers of 2 Multiplied by The Primes within The Ranges of 144 to 128, 72 to 64 and 36 to 32 Which Are Symmetrical to Those Primes in Lowest Terms in The Half of The Symmetric Division Mapped below and Including 224 within The Ranges 126 to 112, 63 to 56 and 31.5 to 28 with The Addition of 119*, exhibited at the Dream House, located in downtown Manhattan [17].

⁵<https://www.mettle.com/product/mantra-vr/>

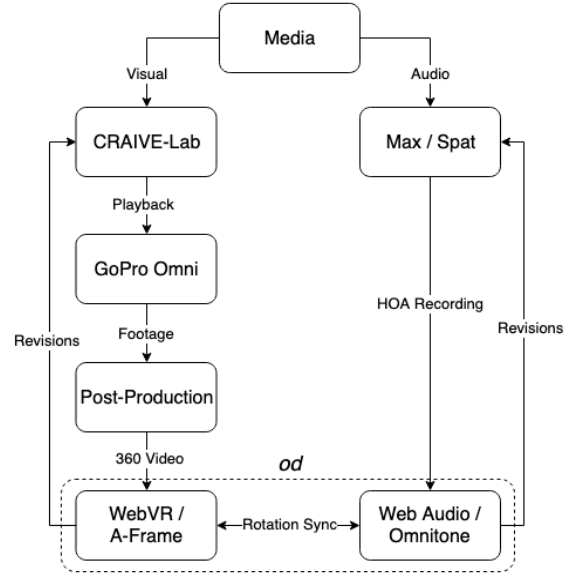


Figure 2: Composition workflow for *od*

The sonic properties of the drone are strictly related to the listener’s body, position and movement; as long as the individual stands still, the drone is constant, but when the head, or the body is displaced, certain partials diminish and new one emerge, activating the drone. The artistic motivation was thus composing a spherical drone in a virtual space, where the user’s change in viewpoint activates the uniform soundscape and sets timbre in motion.

In the context of our project, Max/MSP with SPAT externals offered a composition workflow where:

- An ambisonic scene is composed and visually monitored via `spat5.viewer` object, as seen in Fig. 3,
- The scene is normalized with the N3D scheme for spherical harmonic components and streamed in the HOA encoder of third order,
- The HOA stream is then recorded into a 16-channel sound file and simultaneously transcoded to binaural for headphone listening,
- Finally, the scene is transformed in yaw, pitch, and roll in order to monitor and interact with the soundscape binaurally

This ambisonic scene consists of 20 virtual sound sources arranged as the vertices of a dodecahedron surrounding the listener, as seen in Fig. 3. Instead of utilizing prime frequencies as in Young’s piece, we have randomly distributed three spectral clusters of low, mid and high frequencies that are centered around 150, 1500, and 6000 Hz. Although starting with pure tones, we have found triangle waves more suitable as sound sources, since they contain overtones that enhance to the timbral quality without weighing down the fundamentals as much as sawtooth or square waves. Finally, for further effects, we attached Perlin noise generators to these sound sources to add smooth subtle vibrations, and make the auditory scene feel more organic.

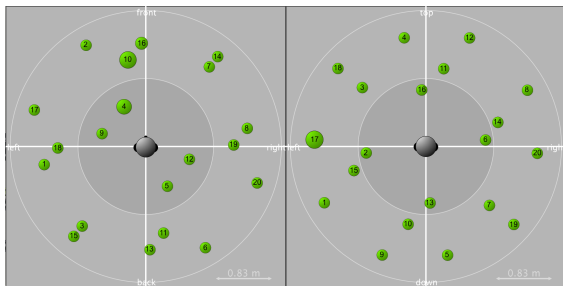


Figure 3: Snapshot from the spat5.viewer interface with 20 sound sources distributed in a dodecahedral pattern around the listener

4.2 Visual Design

As mentioned in section 3.3 above, the main goal of the visual design is to render the CRAIVE space accessible through the browser to experience virtually. In order to achieve this, we shot a number of 360° video recordings using a GoPro Omni rig consisting of six GoPro Hero 4 cameras set to 4:3 4k resolution. The camera array was placed at the center of the room in all three axes, or the “sweet-spot”. Extending from the author’s previous 43-channel acousmonium performance held at EMPAC Studio 1⁶, the projected video screen consisted of footage from the concert, stretched, re-produced and manipulated for the 15360x1200 screen. While a number of formats were experimented with for the projection screen of unusual dimensions, we have found the .webm format significantly more efficient than other encodings in terms quality, of file size and smooth playback, despite the fact that it meant for media online.

The individual camera recordings were then stitched together with proprietary software to construct a 8000x4000 spherical image from the center of CRAIVE-Lab. On the horizontal, the projection screen is continuous apart from the entrance; but above and below the screen remained large blank areas. In order to augment the immersive qualities of the room, we strategically placed virtual mirrors on the spherical image, as seen in Fig. 4, to expand the projected image in all directions using Mantra VR plugins. The final, augmented spherical video is then converted to .mp4 and specified as an A-Frame asset to be played back on the web browser.

4.3 Implementation

The HRTF filters for SPAT’s binaural rendering were default KEMAR filters. These differ from the SADIE filters for Omnitone’s rendering in terms of a number of attributes; subjects comparing HRTF qualities in a recent research have rated KEMAR as brighter, richer, better externalized and overall more preferable compared to SADIE database samples [1]. While this did not have a significant effect on the timbral variations emerging upon perspective change, it nonetheless demands the composer to be mindful of the differences in overall timbre during the compositional process and the final web Audio implementation. This is one of the causes for revisions and repetition of the procedure in a loop as illustrated in Fig. 2. The recorded composition, a fixed multi-channel wave file, is then piped into the Om-

⁶<https://vimeo.com/308695695>

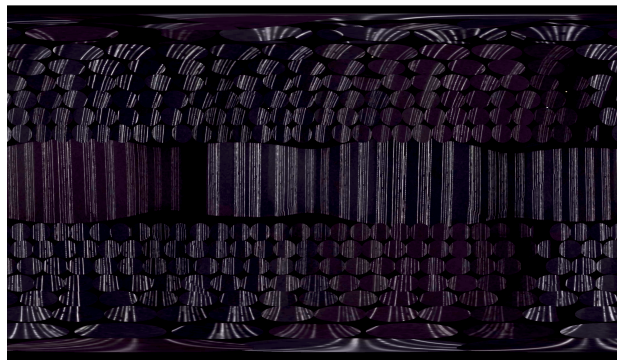
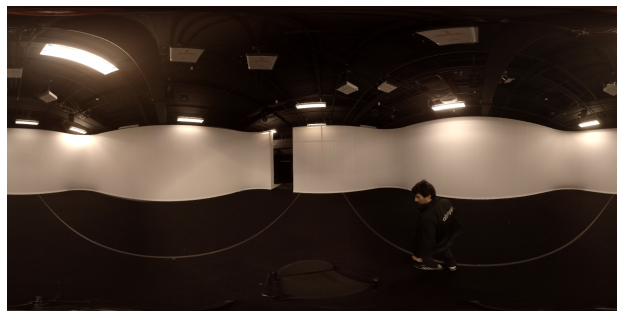


Figure 4: Spherical image of the CRAIVE panoramic screen (above) and the room augmented with virtual mirrors in post-production (below)

nitone HOA decoder as an audioContext via *AudioBufferSourceNode* and rendered as binaural signals on the browser with reference to the rotation matrix updates from A-Frame.

Communication between A-Frame and Omnitone is achieved by updating values from registered components; in our case, an entity attached as a camera component with a .tick() handler grabs the current rotation values at every frame in Euler angles. The azimuth and elevation values are converted to a 3x3 matrix by a standard Open-GL “View” Matrix calculation; the matrix is then set as the Omnitone rotation matrix, updating the HOARenderer as illustrated in Fig. 1.

5. CONCLUSION AND DISCUSSION

This paper documented a contemporary research in electronic music composition and multimedia design that resulted in *od*, a spatial composition for binaural listening on the web. We have used a number of online and offline technologies to not only document a physical space accurately, but to further augment its immersive audiovisual qualities with an artistic approach. New spatial compositions continue to emerge at research institutions, studios and art exhibitions, yet there are still no standard methods to archive and share such works. Since there are no standard methods of documentation for such works, we hope to demonstrate a favorable approach to document spatial music that may be of use to other composers. The tools exist and are available; with the use of ambisonic microphones and Web Audio-based renderers like Omnitone, composers can easily and accurately share their recorded works online for binau-

ral listening. Making these accessible to an online audience could facilitate healthier communication among composers and listeners and expand the community. Furthermore, Web Audio and WebVR technologies will hopefully reduce the need for expensive equipment and performance spaces; this will likely evolve the practice into a more inclusive and accessible one, enabling more people to contribute.

With these thoughts in mind, we can address some issues we faced for future improvements. Firstly, we have found Google Chrome to be the only reliable browser during the development phase for both Web Audio and WebVR components. Moreover, 360° videos and ambisonic files are large in size due to the high amount of information they contain; thus, it takes considerable time on the user side to load and play the composition. Utilizing other formats (such as .webm videos instead of .mp4) for smoother and longer playback with better spatial fidelity would be a significant performance improvement. Streaming the spatial media rather than downloading is another possible future development; as of today A-Frame only supports a component for streaming with the Vimeo API, but this is available only for paying members.⁷ HRTF filters differ from one another in terms of their spectral qualities and spatial perception, as mentioned in section 4.3. Providing multiple HRTF libraries to select and compare within the Web Audio renderer would facilitate further musical experimentation. Methods of modeling one's own ear and creating their personal HRTF filters are possible and available today [9]; so in addition to the standard libraries, implementing individualized HRTFs would also be an interesting feature.

Finally, our general impression from available Web Audio API resources is that they essentially speak to web developers with extra-musical concerns rather than creative investigators. Now supported by a wide range of browsers and devices, Web Audio and WebVR contain much potential for artistic research, but a shared language for spatial media composition is yet unestablished. Therefore with this paper, we hope to demonstrate new possibilities for this emerging field and contribute to its critical discussion.

6. ACKNOWLEDGMENTS

We would like to thank Jonas Braasch for giving us unrestricted access to CRAIVE-Lab, Shawn Lawson for providing information and discussion on VR technologies, Sam Chabot for assistance with the CRAIVE infrastructure and camera array, Görkem Özdemir for executing the visual design, Büşra Tunç for help with issues on panoramic projections and immersive video, and lastly Barış Demirdelen for help and assistance with coding.

7. REFERENCES

- [1] C. Armstrong, L. Thresh, D. Murphy, and G. Kearney. A Perceptual Evaluation of Individual and Non-Individual HRTFs: A Case Study of the SADIE II Database. *Applied Sciences*, 8(11):2029, Oct. 2018.
- [2] T. Carpentier. A new implementation of Spat in Max. In *Proceedings of the 15th Sound and Music Computing Conference*, pages 184 – 191, Limassol, CY, July 2018.
- [3] T. Carpentier, M. Noisternig, and O. Warusfel. Twenty Years of Ircam Spat: Looking Back, Looking Forward. In *Proceedings of the 41st International Computer Music Conference*, pages 270 – 277, Denton, TX, Sept. 2015.
- [4] A. Çamcı, P. Murray, and A. G. Forbes. A Web-based System for Designing Interactive Virtual Soundscapes. In *Proceedings of the 42nd International Computer Music Conference*, pages 579–585, 2016.
- [5] S. Emmerson. Diffusion-Projection: The Grain of the Loudspeaker. In *Living Electronic Music*, pages 143–170. Routledge, Leicester, UK, Sept. 2017.
- [6] L. Fyfe, O. Gladin, C. Fleury, and M. Beaudouin-Lafon. Combining Web Audio Streaming, Motion Capture, and Binaural Audio in a Telepresence System. Berlin, DE, Sept. 2018.
- [7] E. Gayou. The GRM: landmarks on a historic route. *Organised Sound*, 12(03), Dec. 2007.
- [8] M. Geronazzo, J. Kleimola, E. Sikstroöm, A. de Götzen, and S. SeraiñAn. HOBA-VR: HRTF On Demand for Binaural Audio in immersive virtual reality environments. In *Audio Engineering Society Convention 144*, Milan, IT, May 2018. Audio Engineering Society.
- [9] S. Ghorbal, R. Séguier, and X. Bonjour. Process of HRTF individualization by 3d statistical ear model. In *Audio Engineering Society Convention 141*, Los Angeles, CA, 2016.
- [10] M. A. Harley. Music of sound and light: Xenakis's polytopes. *Leonardo*, 31(1):55–65, 1998.
- [11] E. Kermit-Canfield. A Virtual Acousmonium for Transparent Speaker Systems. Hamburg, DE, Aug. 2016.
- [12] S. Neelakantam and T. Pant. Introduction to A-Frame. In *Learning Web-based Virtual Reality*, pages 17–38. Apress, Berkeley, CA, 2017.
- [13] G. Sharma, J. Braasch, and R. J. Radke. Interactions in a Human-Scale Immersive Environment: the CRAIVE-Lab. *Cross-Surface 2016, in conjunction with the ACM International Conference on Interactive Surfaces and Spaces*, Nov. 2017.
- [14] S. Sterken. Reconstructing the Philips Pavilion, Brussels 1958: Elements for a Critical Assessment. In D. v. d. Heuvel, M. Mesman, W. Quist, and Bert Lemmens, editors, *Proceedings of the 10th International DOCOMOMO Conference*, pages 93–98, Rotterdam, NL, Sept. 2008. IOS Press.
- [15] S. Waters. Timbre composition: Ideology, metaphor and social process. *Contemporary Music Review*, 10(2):129–134, 1994.
- [16] S. Williams. Osaka Expo '70: The promise and reality of a spherical sound stage. In *Proceedings of inSONIC2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, Karlsruhe, DE, November 2015.
- [17] L. M. Young and M. Zazeela. Dream House Opens for the 2016-2017 Season - Our 24th Year. url-<http://www.melafoundation.org/DHpressFY17.html>, 2016.

⁷<https://github.com/vimeo/iframe-vimeo-component>

From the museum to the browser: Translating a music-driven exhibit from physical space to a web app

S. M. Astrid Bin^{1,2}, Christina Bui², Benjamin Genchel², Kaushal Sali², Brian Magerko², Jason Freeman¹

Center for Music Technology¹ & Expressive Machinery Lab²
Georgia Institute of Technology
Atlanta, GA

a.bin, cbui6, bgenchel3, ksali3, magerko, jason.freeman @gatech.edu

ABSTRACT

This paper describes the process of developing a browser-based version of GrooveMachine, a tangible museum exhibit that aims to foster interest in computer science (CS) through the music-driven exploration of a computational system. GrooveMachine is aimed at kids aged 10-14, and specifically targets learners from groups currently under-represented in computing by demonstrating CS applications that challenge stereotypes. While an observational study suggests that GrooveMachine triggers situational interest, long-term engagement with CS requires this interest to be deepened and developed. To provide an opportunity for interest development, we have implemented a browser-based GrooveMachine. This not only offers the opportunity for learners to continue their exploration of CS through creative interaction, but provides a pathway to other music and CS learning platforms where they can deepen this interest. In this paper we describe the theoretical underpinnings of interest, how it relates to CS, and how it intersects with identity. We also describe the differences between the museum and browser contexts. We detail the design and implementation of GrooveMachine in the museum and explain how we translated it to the browser, including the rationale behind our central design decisions and a discussion of our technical implementation. In this way we provide valuable insight for researchers who want to reach larger audiences by developing browser-based versions of physical installations.

CCS Concepts

•Human-centered computing → User interface design; •Applied computing → Sound and music computing; •Information systems → Browsers;

Keywords

collaborative computing, user interfaces, interfaces for music, children, CS learning

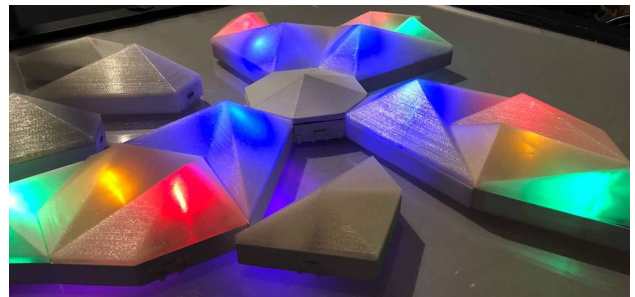


Figure 1: GrooveMachine's tangible blocks

1. INTRODUCTION

GrooveMachine is a tangible, interactive museum exhibit that aims to foster interest in computing through music-driven exploration of a computational system. It is aimed at children aged 10-14, and places specific focus on children from groups currently under-represented in computing. This paper describes how we translated GrooveMachine from a physical museum exhibit to a browser-based web app. First, we discuss why generating interest in computer science (CS) is important, the role music can play in this process through identity, and how this relates to broadening participation in computing. We also discuss the differences between the museum and browser contexts and their implications for design. Second, we describe the museum implementation of GrooveMachine, and briefly describe an observational study that suggests triggered situational interest. Third, we detail the design decisions made in the browser implementation of GrooveMachine, including our approach to technical implementation, in order to provide insight for researchers interested in broadening participation in computing through browser-based interactive environments.

2. BACKGROUND

2.1 Why interest matters

Computer programmers invent, design and develop the technology that affects all of our lives. Despite its broad impact, women and people of colour (PoC) are vastly under-represented in the technology-creating population [9, 22]. As a result, women and PoC are not reaping the economic benefits of CS careers; the technology that affects us all is



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

designed by and for a small segment of the population, and untold human potential is going unrealised. As Bennedson states, “introducing students to computing is still one of computing education’s grand challenges” [5]. Strategies for greater inclusion are needed.

Joining the technology-creating population requires a high degree of intrinsic interest in CS, meaning a desire to engage in computing not because one has to, but for the sake of it [2]. CS courses are challenging with high failure rates [17], but intrinsic interest is a major factor persevering through obstacles, including poor teaching and a lack of support [22].

Though intrinsic interest is sometimes viewed as an in-born trait, Hidi and Renninger propose that this interest can be developed. They suggest that there are two broad types of interest, *situational* and *individual*, and that interest develops over four phases. Situational interest, which has Triggered and Maintained phases, begins with environmental stimuli and is highly motivating, but does not last. However, this Situational Interest can be developed into Individual Interest (through the Emerging and Well-Developed phases), producing an enduring motivation to re-engage with the subject matter in question [14].

2.2 Using music to affect computing identity

Though women and PoC are under-represented in the technology-creating population, there is not a lack of engagement with technology in these groups. Many young Black men have a passion for video games yet don’t engage with learning CS [8], and girls in high school may be enthusiastic about but computing but few pursue it at the college level [12] (and those that do are more likely to drop those

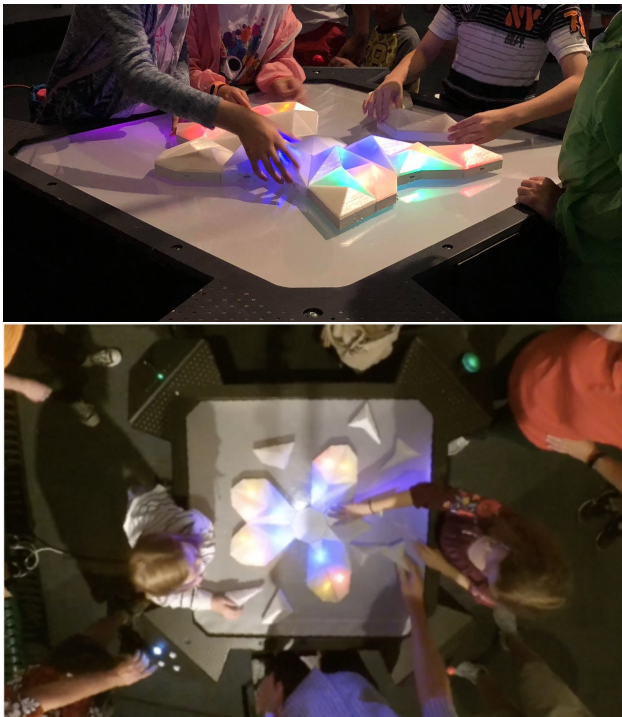


Figure 2: Top: Players exploring GrooveMachine in the museum. Bottom: Overhead view of GrooveMachine in use.

classes [7], and those remaining are more likely to leave academic careers at early stages [16]).

Instead, stereotypes about technology creators cause people in these under-represented groups to *disidentify* with CS. Computer scientists are stereotyped as male and socially awkward [24], which conflicts with young Black men’s concept of masculinity [8], and stereotypes of CS as a solitary pursuit clashes with young women’s tendency to want to work with others [22]. The result is that many young people assume that CS is “not for me” [23].

It is here that STEAM approaches (using the Arts to drive engagement with Science, Technology, Engineering and Maths) can bridge the gap between interest and identity. GrooveMachine builds on existing research that demonstrates the effectiveness of music as a driver for CS engagement, particularly among under-represented groups [11, 20], because music’s wide cultural relevance gives learners an opportunity to engage with CS via a creative activity that is meaningful to them. Further, music-driven engagement demonstrates the creative potential of computing, which challenges the stereotypes that typically serve to disidentify.

2.3 Fostering interest in the museum

GrooveMachine is a museum exhibit. The museum context is ideal for exploratory learning because museums are focused on fostering curiosity [19, p 33]. They are full of novel, hands-on, multisensory exhibits, where visitors can follow their own interests without obligation or constraint [1]. However, these same characteristics that make museums rich learning environments also present significant challenges for designing learning experiences: lots of exhibits are vying for attention and an exhibit must continuously engage learners, because they are free to walk away from any exhibit they find uninteresting.

The nature of CS presents another dimension of challenge for the museum context. CS is a process of identifying a problem, devising a detailed plan to solve it, and translating those instructions into a language a computer can understand [25]. A pre-existing, robust understanding of the nature, makeup, capabilities and function of computational systems and ways of understanding and approaching computational problems must be in place before writing any code, let alone seeing the outcomes and potential rewards of learning CS. As museum learners follow their own interests, voluntarily participate, and can leave at any time, focusing on the scaffolding knowledge necessary to meaningfully use a computer – knowledge which has little relation to the act of programming [17], let alone the compelling possible outcomes – is unlikely to retain learners.

Because of these challenges in maintaining attention in the museum, considerable interest has developed in active prolonged engagement (APE) [1, 15], which refers to visitor engagement with interactive exhibits. APE has a number of characteristics, such as positive collaboration, meaningful discussion, and prolonged engagement with the exhibit. Our observational study of GrooveMachine (discussed in Section 3) measures one of these aspects, prolonged engagement.

2.4 The museum and browser contexts

The situational interest triggered in the museum is motivating, but if it is not further developed it soon dies off. Because GrooveMachine is part of an online, music-driven CS

learning ecosystem ¹, there are two opportunities presented by a browser-based version: to provide the opportunity to develop interest through further exploration of GrooveMachine, and to provide a pathway to discovering these other platforms for deeper exploration.

The most influential factor in translating GrooveMachine from the museum to the browser is the differences in *context*, as both places are not only very different in terms of their physical, social, interactive and cognitive affordances, but also in what people expect from interaction.

Context is a primary consideration in interface design. Bannon [3] established the concept of a *human actor* in HCI theory, proposing that the context in which a system is used is a primary consideration in how it should be designed. There are stark contrasts between the museum and browser contexts, across the axes of who the person is with, why they do things in that context (their motivations), and the features of the experience:

Who are they with? Science museum exhibits are designed to support collaborative, group interaction. By contrast, a computer browser has one set of controls and is designed for a single user.

Why are they here? Museums are places of learning, and visitors voluntarily take part based on their personal interests. Browsers, however, have a multiplicity of uses: they are used to play games, engage in communities, search for information, communicate with others, complete tasks.

How do they experience it? There are three main differences in experience. First, museums are full of novel, unique experiences and visits are occasional, whereas browsers are familiar interfaces that most of us use them every day. Second, museums are multimodal, using “full sensory and expressive capabilities including visual, sonic, haptic, and kinesthetic/proprioceptive” [6], whereas browsers are primarily experienced visually and aurally. Third, museum interaction is embodied, as visitors “physically explore concepts and systems by moving within and acting upon an environment” [6], while a browser is largely disembodied with users interacting through the physical manipulation of a mouse to control a pointer on the screen.

3. MUSEUM IMPLEMENTATION

3.1 Tangible and musical interaction

GrooveMachine is a tangible tabletop exhibit. The interface is based on a step sequencer, with the table divided

¹<https://tunepad.live>, <http://ears sketch.gatech.edu>



Figure 3: Rendering of the GrooveMachine exhibit.

into eight radial “steps”. In the middle of the table is a hub. The stepper moves around the table, and lights in the hub indicate which section is active.

To construct musical patterns, players attach tangible blocks to the central hub. There are two types of blocks, samples and modifiers. Samples can attach to the hub, and modifiers can be added to the samples. When a block is attached successfully it illuminates from the inside, indicating that it is recognised (see Figure 1).

The shape of these tangibles are derived from the Islamic system of geometry (specifically a four-fold star pattern). This approach was chosen for a number of reasons: these patterns are beautiful and deeply mathematical [18], tessellation has been shown to be an effective method of engaging children in mathematical exploration [10], and the symmetry of this layout means that players can look to the actions of others to gain intuition of what to do.

Most importantly, tessellation is a method of assembling instructions that is dependent on exploration, and not on a pre-existing knowledge of computational syntax. In the museum environment, exhibits must continually engage learners in order to retain their attention and curiosity, and here we give curious players the means to discover the nature of this system not through learning its prerequisites, but instead through the opportunity to act [26].

3.2 Physical form

GrooveMachine is a square table. The square shape fosters collaboration (by encouraging players to distribute themselves around it as there are four obvious places to stand), and also places constraints on the player (as not everything is within easy reach). Since players cannot easily reach all the steps of the sequencer, they must negotiate with others.

Each of the table’s four corners features arcade controls, chosen because they are familiar to kids and invite interaction. Each of these sets of controls affects a different global variable of the GrooveMachine system: volume, the genre of the music, tempo, and the direction in which the step sequencer is travelling.

3.3 Connection to computing

The CS content in GrooveMachine is delivered via embodied metaphor. As learners interact, they engage with metaphors for computational systems: the loop (the step sequencer), computational objects (the samples), parameterisation (the modifiers), variables and variable scope (the arcade controls). In this way we establish and reinforce mental models of computing through a fun experience driven by music.

3.4 Technical implementation

GrooveMachine contains an embedded hardware network of four Arduino Megs, and a central Mac Mini. The Arduino Megs track the samples and their modifiers as they arrive and leave. The Mac Mini controls audio playback, moving the stepper around the table and playing audio samples based on the tangibles that are present in a given step. The audio playback, which uses pre-recorded samples and some real-time processing, is implemented using Pyo (a Python-based DSP library).

3.5 Hold time study

In July 2018 we conducted a preliminary hold time study at Chicago’s Museum of Science and Industry, where we installed GrooveMachine on the open museum floor. We observed groups of visitors use the installation, noting the number and approximate ages of group members, and timing their interaction. We wanted to determine if the lengths of time that our target audience spent with GrooveMachine could be considered “prolonged”: since one of the characteristics of situational interest is focused attention triggered by the environment [13], long hold times could be a first indication. In the seminal APE studies the average time spent at APE exhibits was 3:18 [15, p 13] (similar to findings in [4]), so we used this as our threshold for “prolonged” engagement.

We observed 72 groups, with an average hold time of 4:26. We performed a one-tail t test to determine if this hold time was higher than the threshold for “prolonged” interaction, and found that it was statistically significantly higher ($M=4:26$, $SD=2:59$, $t(71)=3.2596$, $p<0.01$).

33 of these groups included at least one child in our target age range of 10-14, and this group had an average hold time of 5:34. We performed an independent-samples one-tail t test to compare the average hold time of this subset with the 3:18 threshold, and found that it was statistically significantly higher ($M=5:34$, $SD=2:57$, $t(32)=4.38101$, $p<0.01$).

More in-depth study is needed to determine the quality of and reasons for this effect, but this does suggest that GrooveMachine is triggering situational interest. In the museum context learners engage voluntarily, can leave at any time if they are uninterested, and there are plenty of exhibits competing for their attention, but GrooveMachine captures the interest of learners in our target age range for significantly longer than the average “prolonged engagement” time threshold.

This suggestion of situational interest is promising, but gave us pause. Hidi and Renninger indicate that though motivating, situational interest will die off if not deepened through further exploration [14]. Because our broadest aim is to foster interest in computing among people under-represented in computing, just triggering situational interest through music-driven exploration is not enough. A browser-based version of GrooveMachine was the next logical step to create an opportunity to further develop interest in CS through music, and to provide a pathway to other learning platforms that can take learners further.

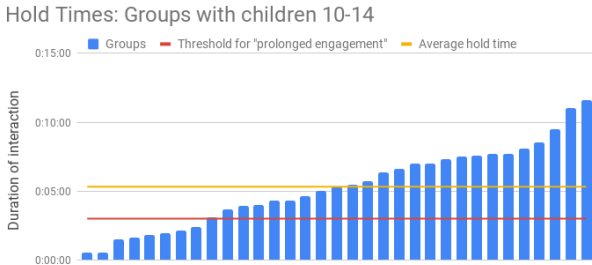


Figure 4: Hold time durations of the 33 observed groups containing at least one child in the 10-14 age range. Red line: Prolonged engagement threshold. Yellow line: Average group hold time.

4. MOVING TO THE BROWSER

As discussed in Section 2.4, there are profound differences between the museum GrooveMachine and the browser GrooveMachine, and these are a function of differences in context. In order to translate between the two contexts, we first had to determine the GrooveMachine’s core aspects that are independent from context, both *formal* (design features that are stable), and core *functional* (effects on the person as a result of using it). Through team discussion we determined that the core formal aspects are the step sequencer interface that is activated through adding tessellating blocks, and that the core functional aspects are exploration and discovery.

4.1 Formal aspects: Interaction and UI

The interface of the web-based GrooveMachine is an octagonal space with a central hub. On the top on the left are outlines of the GrooveMachine blocks. Along the bottom are controls for the global variables, as well as a button to save a groove and buttons to toggle the code visualisation (see Figure 5).

The browser version differs from the physical installation in a number of specific ways:

Octagon shape: In the museum, GrooveMachine’s physical form is square, to encourage collaboration. In the browser these collaborative and physical aspects are not relevant, so we adopted an octagonal interface to reinforce the step sequencer’s loop metaphor.

Tangible drawers: GrooveMachine’s tangible blocks are placed on and around the table allowing learners to explore their shape and try them out. In the browser we wanted to avoid visual clutter but still maintain the element of discovery, so we placed these into expandable “drawers” that expand when the shape is clicked.

Connection to computing: Using GrooveMachine in the museum requires learners to engage in physical metaphors for computing, and we have the benefit of being able to add printed material around the exhibit to drive this connection. In the browser we have the opportunity to link this music-computer relationship more directly by including code visualisation. When the code toggle at the bottom is clicked, code is visualised on top of the interface. This is updated in real time, and includes both system state and executing functions (see Figure 5, bottom).

Variables: In the museum, GrooveMachine’s arcade controls allow learners to manipulate global variables, and place them outside the interaction to embody a metaphor for their global scope. In the browser embodied metaphor is not possible, but we did make these controls readily accessible at the bottom of the interface (Figure 5).

The core mechanic of GrooveMachine is placing blocks onto the steps of the sequencer, and a block causing the system to produce a specific sound when that step is active. We maintained this interaction, so learners drag blocks from the collapsible drawers and onto the interface. The difference in the interactions is spatial: In the museum the spatial manipulation of tangibles is an important element of the interaction, but this isn’t possible in a browser. In the browser version, learners drag blocks from the drawer to the interface, and as a block passes over the interface it automatically orients itself to fit in a given step.

4.2 Functional aspects: Exploration and discovery

GrooveMachine is designed with exploration and discovery in mind, and though future work is needed to determine the precise connection between this and extended hold times we wanted to preserve these features. Exploration is a key factor in experiencing this computational system, and for this reason both the museum and the browser have no required onboarding, and instead are tolerant to learners trying things out and visually responding when they get it right (in the museum version the tangibles light up, in the browser the blocks are outlines when being dragged, and filled in when attached). (Though there is no onboarding for the browser, we have included a simple Help box that provides minimal instruction, as there are no other learners to watch or guides to ask.)

In both versions, discovery is key. In the museum learners can move between exploring the tangibles and working the arcade controls, can learn from watching others, and can move to other vantage points if they wish. These physical aspects aren't relevant in the browser, but we included discovery by locating the tangibles in collapsible drawers. All other controls are readily available, but to find tangibles learners need to find them.

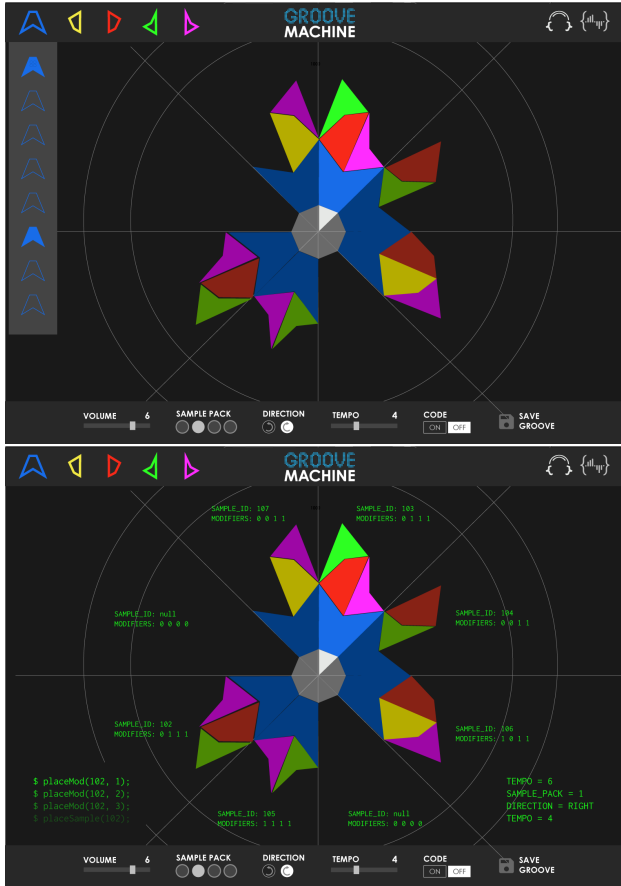


Figure 5: GrooveMachine browser interface. Top: Interface with drawer open. Bottom: Interface with code layer visible.

4.3 Technical implementation

To build the browser-based interface we used PixiJS², a 2D WebGL rendering library that uses HTML5 and JavaScript. This allowed us to build an interface where pieces could be dragged and dropped, emulating the physical table interaction.

Though Pixi is powerful and easy to use, a caveat is that it is not ideal for designing UI elements. For example, the collapsible drawers holding the blocks are implemented using a combination of Pixi and HTML. Slide-out drawers allow for discovery and are straightforward to implement in HTML, but since the tangibles are Pixi elements they must be contained in the canvas. To mitigate this, the expansion is triggered by an HTML element (the shapes at the top of the interface), but the expanding drawer is a Pixi element.

All audio in the browser is handled by Tone.js [21], described as “a framework for creating interactive music in the browser”. Tone.js is built on top of the Web Audio API³, and provides scheduling, synths, effects, and buffers for reading and playing back audio files.

At present, all sounds are read from a set of audio files that are downloaded from a database while the application loads. Certain effects that can be applied to the base samples via effects tangibles, namely high pass and low pass filtering, are not realized in real time but stored in the database as static files as well. A file exists for the high-passed and low-passed versions of each base sample. Other effects, such as sample reversal, are provided by Tone directly and applied in real time.

The timing for the entire application is driven by Tone’s internal clock. When the application loads, a Tone.js Sequence object is created that invokes a call back function 8 times with the iteration step as argument. The callback function has two advantages: it advances PIXI’s game clock, eliminating the need for PIXI’s own internal clock which is timed by frame count, and it plays the sample and associated effects for whichever step of the virtual table is currently active. Additionally, it checks for and applies global changes such as tempo and volume change. The Sequence object runs in an indefinite loop.

4.4 New affordances from the browser

Though some physical aspects of GrooveMachine in the museum are lost when we move to a web app, the browser does offer some unique advantages.

Easy iteration and expansion. A drawback of physical installations is that production is labour-intensive and expensive, and changing aspects compounds this. The browser-based GrooveMachine, however, does not have these constraints, and we can easily and quickly iterate on its design and function in response to testing.

Located in a computer. One of the challenges of the GrooveMachine exhibit has been connecting this experience with computing. This has been addressed somewhat with printed didactic material around the exhibit, but making that connection without causing disidentification because of negative stereotypes is challenging. In the browser, this connection is obvious. Browsers are ubiquitous and familiar, and seeing code on a screen is not out of place. In this way we can direct interest further towards computing instead of

²<https://www.pixijs.com/>

³<https://www.w3.org/TR/webaudio/>

simply having a fun experience.

Easy connection to other platforms. GrooveMachine is part of an ecosystem of online CS learning environments that are music-based (EarSketch and TunePad). These are considerably more in-depth but offer much deeper exploration of CS through music. A browser-based GrooveMachine offers an easy connection to these platforms, enabling learners to develop their interest in CS to a more profound level than is possible in a museum interaction.

5. CONCLUSIONS AND FUTURE WORK

GrooveMachine is designed to foster interest in CS, particularly among children from underrepresented groups. There are preliminary indications that it triggers situational interest in the museum, but for this interest to last it must be developed. In order capture and deepen this triggered interest we developed a browser-based version of GrooveMachine. Through careful consideration of GrooveMachine's core features we translated this exhibit from museum to browser, and we leveraged web-based tools to implement it.

Our next steps are to refine this interface through testing, to develop a pathway from museum to the browser, and to build infrastructure between GrooveMachine, EarSketch, and TunePad (such as single sign-on, exporting and importing projects, etc). To broaden participation in computing experiences that challenge computing stereotypes are needed, as well as ways for these learners to develop lasting personal interest in CS.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the National Science Foundation for their ongoing support of this project (grant 1612644).

7. REFERENCES

- [1] S. Allen. Designs for learning: Studying science museum exhibits that do more than entertain. *Science education*, 88(S1), 2004.
- [2] M. Apiola, M. Lattu, and T. A. Pasanen. Creativity and intrinsic motivation in computer science education: experimenting with robots. In *SIGCSE*. ACM, 2010.
- [3] L. J. Bannon. From human factors to human actors: The role of psychology and human-computer interaction studies in system design. In *Readings in HCI*. 1995.
- [4] C. Barriault and D. Pearson. Assessing exhibits for learning in science centers: A practical tool. *Visitor Studies*, 2010.
- [5] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming: 12 years later. *ACM Inroads*, 2019.
- [6] D. Birchfield, H. Thornburg, M. C. Megowan-Romanowicz, S. Hatton, B. Mechtley, I. Dolgov, and W. Burleson. Embodiment, multimodality, and composition: convergent themes across hci and education for mixed-reality learning environments. *Advances in Human-Computer Interaction*, 2008.
- [7] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. *SIGCSE Bulletin*, 2006.
- [8] B. DiSalvo, S. Yardi, M. Guzdial, T. McKlin, C. Meadows, K. Perry, and A. Bruckman. African american men constructing computing identity. In *CHI*. ACM, 2011.
- [9] B. J. DiSalvo, K. Crowley, and R. Norwood. Learning in context: Digital games and young black men. *Games and Culture*, 2008.
- [10] R. S. Eberle. The role of children's mathematical aesthetics: The case of tessellations. *The Journal of Mathematical Behavior*, 35, 2014.
- [11] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, and E. Fruchter. Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch. In *SIGCSE*. ACM, 2014.
- [12] S. Graham and C. Latulipe. Cs girls rock: sparking interest in computer science and debunking the stereotypes. In *SIGCSE Bulletin*. ACM, 2003.
- [13] S. Hidi. Interest: A unique motivational variable. *Educational research review*, 2006.
- [14] S. Hidi and K. A. Renninger. The four-phase model of interest development. *Educational psychologist*, 41(2), 2006.
- [15] T. Humphrey and J. P. Gutwill. *Fostering active prolonged engagement: The art of creating APE exhibits*. Routledge, 2017.
- [16] M. Jadidi, F. Karimi, H. Lietz, and C. Wagner. Gender disparities in science? dropout, productivity, collaborations and success of male and female computer scientists. *Advances in Complex Systems*, 2018.
- [17] T. Jenkins. On the difficulty of learning to program. In *LTSN for Information and Computer Sciences*, 2002.
- [18] D. H. Kaplan, Craig S & Salesin. Islamic star patterns in absolute geometry. *ACM Transactions on Graphics*, 23(2), 2004.
- [19] G. D. Lord and B. Lord. *The manual of museum management*. Rowman Altamira, 2009.
- [20] B. Magerko, J. Freeman, T. McKlin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A steam-based approach for underrepresented populations in high school computer science education. *Transactions on Computing Education*, 2016.
- [21] Y. Mann. Interactive music with tone.js. In *Proceedings of the 1st annual Web Audio Conference*. Citeseer, 2015.
- [22] J. Margolis, A. Fisher, and F. Miller. The anatomy of interest: Women in undergraduate computer science. *Women's Studies Quarterly*, 2000.
- [23] J. Margolis, J. Goode, and D. Bernier. The need for computer science. *Educational Leadership*, 68(5), 2011.
- [24] A. Master, S. Cheryan, and A. N. Meltzoff. Computing whether she belongs: Stereotypes undermine girls's interest and sense of belonging in computer science. *Journal of Educational Psychology*, 2016.
- [25] D. Sleeman. The challenges of teaching computer programming. *Communications of the ACM*, 1986.
- [26] H. Van der Meij. Principles and heuristics for designing minimalist instruction. *Technical communication*, 1995.

Join my party! How can we enhance social interactions in music streaming?

Alo Allik, Florian Thalmann, Cornelia Metzger, Mark Sandler
Centre for Digital Music, Queen Mary University of London
{a.allik, f.thalmann, c.metzig, mark.sandler}@qmul.ac.uk

ABSTRACT

In this paper we examine ways to encourage social interactions in online music streaming platforms and discuss the challenges that emerge when deploying browser-based music mixing systems. With the example of an interactive online application, that allows users to choose music collaboratively based on mood, create their own personal parties, as well as share their favourite tracks with other participants, we explore new alternatives for musical experiences in the context of social media on the one hand and music streaming on the other.

1. INTRODUCTION

With the streaming paradigm becoming the prevalent method of our daily musical experiences there are potential new ways to introduce more collaborative and socially interactive experiences to music listening in the age dominated by various social media platforms. Collaborative playlists have become standard on most streaming platforms, yet most music sharing seems to happen through posted links in social chat applications. We explore a way to select and share music collaboratively in a sample web application using higher level musical concepts such as mood. The selected music is automatically DJ mixed using the Web Audio API with the help of content-based audio features that assist in matching tempo, key, beats and volume of the tracks to create a continuous and evolving stream.

The sample web application we are going to use as a demonstrator of the different ideas explored in this paper, moodplay.github.io, has grown out of two earlier, now defunct music players. The original system - Moodplay[3] - was designed as an interactive installation for public spaces where participants could experience and interact with the system and each other on location. This made for an immediate and engaging experience, but at the same time significantly constrained public accessibility for the same reasons. It also involved remarkable time and effort to deploy, involving audio and visual systems, and complex interaction between components that made it unfeasible for any sort of frequent deployment. This Moodplay should not

be confused with its namesake web application accessible at <http://moodplay.pythonanywhere.com/>, which is not related to the family of music players described here, yet contributes an interesting and valuable approach to the area of music mood similarity[2]. A web application also happened to be the next iteration of our Moodplay[1] in attempts to solve the problem of public accessibility, while also focusing on personalisation, discovery and playlist creation. However, various issues with music licensing, deployment infrastructure and, most importantly, the loss of the social aspect of the application forced the development to be abandoned.

Having been faced with the joys and challenges of developing and deploying these applications, a larger picture has gradually started to intrigue about how we listen to and share music in an era of music streaming services and social media, in which we are perpetually connected online and what kind of opportunities could this environment potentially afford. How users on various streaming services and online stores select music depends on the interface they are presented and affects their choices. We are interested in exploring if there are viable alternatives to the established ways of designing music selection interfaces and how to combine music selection with social aspects of listening to music, in particular, how to make music selection a collective interaction and how to enrich the sharing experience in web environments.

2. MUSIC SELECTION

The most common music selection interfaces on digital platforms, be it offline music players, online streaming services or retail stores, present choices on the level of artists or tracks in form of lists. Even curated playlists that usually include tracks on higher level notions of popularity, style, genre, mood or activity use the same format. A number of alternative representations of musical tracks or artists could help break from the standard by displaying a collection of such entities in the shape of networks, clusters or scatter plots, for example. The organising principle behind these kinds of displays would derive from some kind of similarity measure based on collection and analysis of contextual or content-based information. Figure 1 illustrates a network layout of a hypothetical track collection, in which each node represents a track, the colors of nodes identify an artist, the size of nodes could indicate popularity of each track, the connections are made based on content-based similarity, which could also serve as a probability weight when selecting a next track, for example. Large collections of data about music tracks, from which similarity models could be designed, can



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).



Figure 1: Tracks could be displayed as a network in a music player interface.

be accessed from various publicly available sources. There are a number of music-related APIs, both community-run and commercial, from where rich troves of information can be gathered about musical entities like tracks, albums and artists, that can then be used to enrich the music discovery and selection for users. Connecting various data sources can also have a qualitative effect on musical experiences. Discogs¹, Gracenote², MusicBrainz³, AcousticBrainz⁴, MusicStory⁵, and Spotify⁶ are just a few specifically music-related data sources that a music player could rely on for information, and there are many more.

In case of moodplay.github.io, the arrangement of the tracks, shown in Figure 2, is inherited from the original system and is based on crowd-sourced data from the ever popular Last.fm API (<https://www.last.fm/api>). Despite the gradual reduction of the site’s functionality over the last few years, it still provides valuable information about how millions of users have described these entities by providing various tags related to genre, mood, instrumentation, geographical location, lifestyle and many other typical notions, but at the same time almost anything expressing a personal opinion. The tracks are arranged in the space by two mood coordinates: horizontally from negative to positive (designated as valence in dimensional models of emotion) and vertically from calm to excited (also arousal or intensity). The process that converts user tagging data to the scattering of tracks in the space involved first determining a set of most often occurring mood tags, which can also be extracted from the service. Then querying the API for counts of how many times each tag has been applied by users to each track produced a multidimensional space, in

¹<https://www.discogs.com/developers/>

²<https://developer.gracenote.com/web-api>

³<https://musicbrainz.org>

⁴<http://acousticbrainz.org/data>

⁵<http://developers.music-story.com/developers>

⁶<https://developer.spotify.com/documentation/web-api/>

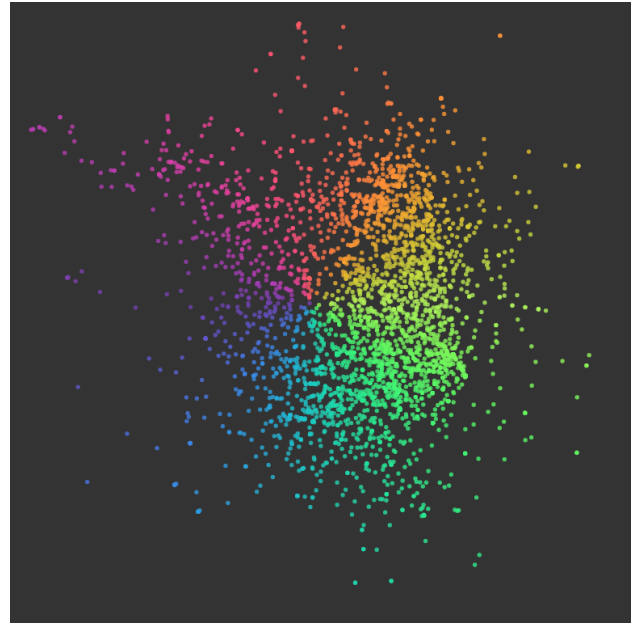


Figure 2: Representation of moodplay.github.io tracks in 2D space, arranged in terms of general mood from negative (left) to positive (right) horizontally and calm (bottom) to excited (top) vertically.

which each track is associated with a vector of tag counts. This enables calculating coordinates for each track in a 2D space by applying dimensionality reduction techniques. The full process of track selection together with the models of emotion and valence-arousal mapping of tracks has already been discussed more in depth in previous Moodplay-related papers.

The audio content for the player originates from Deezer API⁷. Due to licensing limitations, the current system can only access the 30-second previews of each track. The dataset included in the current version of Moodplay is a subset from the original system, that has been whittled down through a process of finding matching Deezer identifiers of tracks and further by including only the ones that have a valid preview URL. In the end there are 3,497 tracks by 2,314 artists left of the original collection.

Users can explore the mood space by selecting a location on the interface and are then displayed the corresponding mood tag for that area. The space is tessellated using the Voronoi algorithm⁸ with the mood tag coordinates as the set of points as shown in Figure 3. The tessellation does not appear in the interface (which is described in more detail in Section 3 and shown in Figure 4), but partitions the space into mood regions in the background. Once a desired mood has been located, it can then be selected by clicking on the popup label, which means the user preference is communicated to the server and added to other users’. Thus users never select music directly by artist or track, but by emotion that the closest track to the preference matches.

⁷<https://developers.deezer.com/api>

⁸https://en.wikipedia.org/wiki/Voronoi_diagram

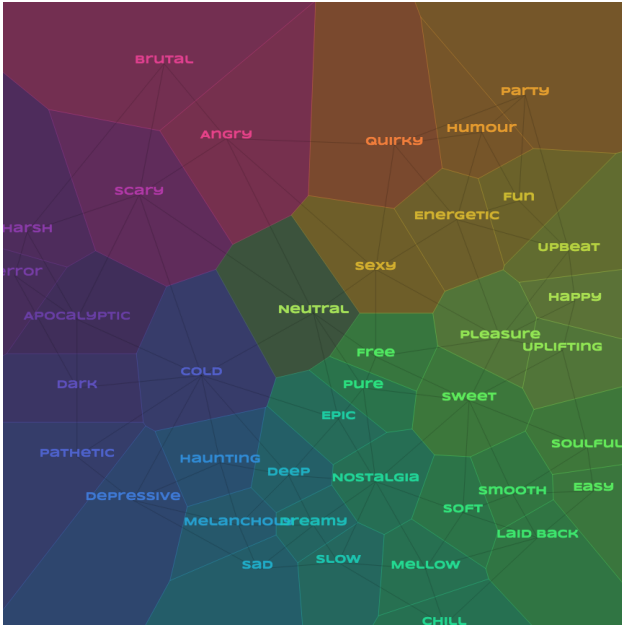


Figure 3: Voronoi tessellation of the 2D mood space that underlies the mood selection interface

3. SOCIAL INTERACTION

Social Web platforms accessible to anyone with a portable device have drastically changed the way we communicate and interact with each other. This also provides new opportunities in the context of music streaming, that perhaps have not yet been explored to their full potential. Collaborative music selection and sharing could be enhanced and encouraged through new types of music player interfaces that we discussed in the previous section. Social interaction on popular music streaming platforms is currently for the most part encouraged through collaborative playlists. Spotify used to provide a messaging functionality in the early versions that allowed users to send music links to each other, but that functionality was removed. Dubtrack.fm⁹ is a social online radio where users can create their own rooms or join existing ones and collaboratively queue tracks they would like to listen to. Each room has a specific theme that represents a community of music listeners, like "chillout" or "The Eighties" for example, and code of conduct including what to play, what not to play, and active hours. Tracks can be queued from public music services like YouTube and SoundCloud. Playlist¹⁰, a social music platform for iOS only available in select territories (i.e. not UK), enables, in addition to collaborative playlist creation, a synchronised listening and a chat functionality to participants. These are fine examples of communal music sharing that enhance user participation and music discovery, although still centred around the notion of a linear playlist of tracks as the core musical entities. An alternative to creating cumulative linear playlists could be based on consensus and involve users voting for their preference from a set of options. If we envisioned a network of tracks as suggested in the previous section, the users could be presented a selection of tracks to choose from based on

⁹<https://www.dubtrack.fm/>

¹⁰<https://www.playlist.com/>

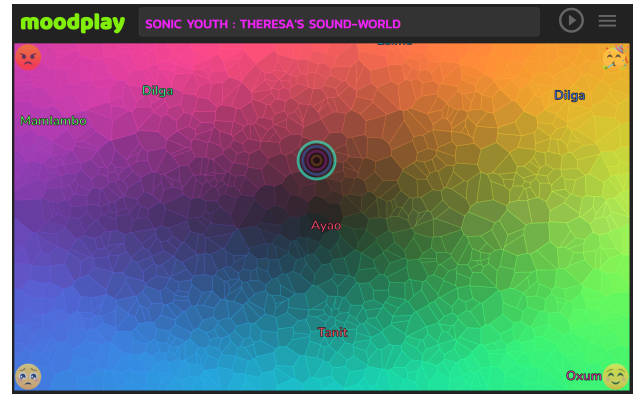


Figure 4: moodplay.github.io interface with 5 active user names displayed alongside the circular player cursor indicating the average mood of the party

feature similarity and the music stream is determined by which tracks receive the most votes. If the selection happens by a higher-level concept such as mood, genre or activity, for example, then the selection of tracks is determined by the system that aggregates user preferences, like in moodplay.github.io. Since the Moodplay web application can be accessed by any number of users simultaneously, the interaction becomes more immediate between potentially larger number of participants. Each user can select a preferred location in the mood space, which remains registered for a certain period of time until it expires. This creates a continuously changing average mood that is displayed to users by the circular player cursor as illustrated in Figure 4. The preference can be changed at any time, although if done while the last preference is still active, it is overwritten by the new selection, i.e. each user can only have one preference at a time.

For users who would like to explore the player on their own or with a select group of friends, we have introduced the notion of a party, somewhat similarly to the concept of a room in Dubtrack.fm. Every user who arrives at the front page is immediately added to the global Moodplay party where they can see the votes of all other participants. However, they are also able to create their own personal party and share the link to it with whomever they choose. All members of a party are able to vote on their select private musical sequence and always hear the same track at the same time, mixed together in the same way, regardless of where they are geographically. If they are in the same room, the users can decide to have only one of their devices play the music while still all of them can vote on silent devices.

The synchronization between different audience members of a party is ensured by socket.io¹¹ - a JavaScript library that enables real-time, bidirectional and event-based communication between clients and the server. A server gathers votes and push updates to all participants in a party at a specific interval which can be configured separately for every party. The updates contain information about currently active users, the average mood of the party, and information about the track that is playing. If the system detects that there are no currently active participants, it activates

¹¹<https://socket.io>

a number of bots that generate automatic votes to keep the party going. Interacting with other participants by selecting music according to mood is one way moodplay.github.io encourages social interaction, but it also facilitates sharing music between participants.

4. MUSIC SHARING

Users can also choose to add their own music that is not part of the official moodplay.github.io repertoire for use at their private party. They can simply drag and drop audio files onto the browser window. Using a statistical method a custom track's mood can be automatically inferred from audio features, which can be extracted directly in the browser using the framework *piper-js*.¹²

Users who want to share their music can use the mood plane to visualize of the music they are sharing. We explore two options. The first is to attribute mood variables valence and arousal in an automated way, using features extracted with vamp plugins, and the mood tags by humans for the existing dataset. We use various statistics of both high and low level features, like moments and autocorrelation. We trained a random forest classifier on these feature vectors to predict valence and arousal, and identify the features with the highest importances as the informative features for valence or arousal respectively. With those, we construct a distance measure between tracks. This distance is uses only extracted features, so it can be used to new uploaded tracks without mood tag. We use it to calculate the nearest neighbours of a track, separately for arousal and valence. The averaged mood values of the neighbours are then used as valence and arousal tag for an uploaded track. This approach works well however has some limitations. One is to choose the distance measure: Since the feature vector by which each song is represented has 400 dimensions or more, the problem of hubness is present (i.e. song songs have a low distance to many others because of the aggregation over many dimensions, but not because of apparent audio similarity). To reduce hubness, we reduce the number of dimensions to the ones of the most informative features (separately for valence and arousal). In addition we use a method from [4] that constrains the number of neighbours of hubs to the mutually closest neighbours. The attributed mood variables depend on these choices, as well as on the used data.

A second option is to let the user tag the uploaded song himself. This can be complementary to the first method, or be assisted by the first method, such that the system suggests mood coordinates, which the user can then accept, or modify. Even if it is not possible to place a track with high precision, it is useful to give a quick first description of the music a use wants to share. Another possibility is to use automated mood tags as suggestion which region of the mood plane to explore, and to find tracks with similar mood to an uploaded track, without the need to describe it or attribute genre tags to it.

5. AUTOMATIC MIXING

One of the advantages of recent developments in Web technologies and especially the Web Audio API is that the ways in which audio content can be presented to users online are much more varied than they used to be. Instead of simply playing back music files linearly, one can now easily create

¹²<https://github.com/piper-audio>

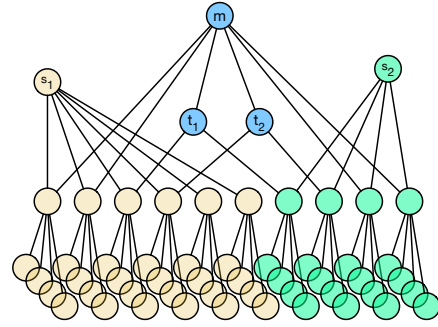


Figure 5: A simplified representation of a sample mix m from one piece s_1 to the next s_2 , where the mix features bars from both pieces as well as combined transition parts t_1, t_2 .

interactive applications with advanced audio functionality.¹³ Streaming services can greatly benefit from these advancements and incorporate custom music processing functionality that is tailored to individual listeners.

Moodplay.github.io demonstrates this in the form of automatic DJ mixing which consists in creating smooth and appropriate transitions between subsequent songs for individual listeners or parties. Depending on the compatibility of two subsequent songs and their tempo, harmonic, and rhythmic content one of several available customizable transitions is chosen and adapted to the specific musical situation. This functionality is based on a Node.js module that can be embedded into any Web application.¹⁴ It uses the Dynamic Music Objects framework to build DJ mixes on the fly based on features extracted from the audio[5]. The core of the package uses a decision tree to determine the best way to transition from one given song to the next depending on their degree of compatibility which is inferred via high-level analytical descriptors derived from the audio features. These decision trees can be custom-defined or they can be learned automatically from transitions ratings by users[5]. A root object of type sequence represents the whole mix to which the parts of the pieces intended to play are then gradually added. These parts can be of various types and may vary depending on the type of transition decided on. Figure 5 shows a simplified representation of a mix containing parts of two pieces. Depending on which root object is passed to the player, one can get it to either play the individual original songs, or the created mix, or all at the same time.

The auto-dj node module has a simple interface with a few public functions: `isReady()` which returns a Promise that resolves once the module is initialized, `getBeatObservable()` which returns an RxJS Observable which emits an incremented number whenever a beat is played (can for example be used for animations), `getTransitionObservable()` through which one can obtain an Observable that emits whenever a transition is started, `transitionToSong(audioUri: string)` which transitions to the song at the uri passed as an argument, and `playDjSet(audioUris: string[])` which mixes a whole set of songs. Anything else is done internally and automati-

¹³see for example <https://tonejs.github.io/demos>

¹⁴<https://www.npmjs.com/package/auto-dj>

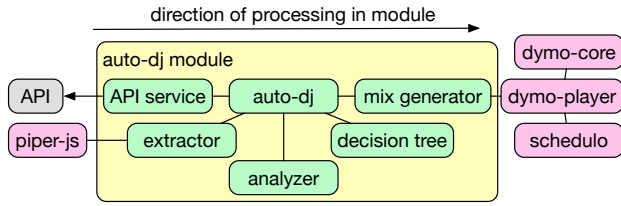


Figure 6: The current structure of the mixing functionality of moodplay.github.io.

cally, including loading the audio, extracting features, calculating higher-level descriptors, deciding on a transition, gradually adding to the mix structure, and playing it back. moodplay.github.io also makes use of the option of providing a custom feature service when initializing auto-dj, which provide pre-extracted features for all songs in the standard collection which significantly reduces browser load. For any custom songs (Section 4) a *feature extractor* extracts audio features directly in the user’s browser using *piper-js*.

Figure 6 shows the structure of the current version of the automatic mixing module. An *analysis unit* calculates and buffers the high-level descriptors, a *decision unit* is capable of performing various decisions, and a *mix generator* contains templates for all the transition types, adds hierarchical song structures to the *dymo-core* triple store, and creates the mix object as described above. This object is then navigated by the *dymo-player*,¹⁵ a playback module optimized with Web Workers and based on the dynamic scheduling module *schedulo* built around Tone.js¹⁶ or *web-audio-scheduler*¹⁷.

6. SUMMARY

moodplay.github.io is an online music streaming platform that strives to encourage social interactions in music streaming by allowing users to collaboratively choose music according to mood. Since it is a web application, it is accessible on any platform or operating system that supports web browsing. Users can participate in the global Moodplay party where everyone is added upon arrival, but they can also create personal parties and control who are invited. There is functionality that allows participants to share their favourite tracks with other invited participants. The system analyses the uploaded tracks by audio features to find their corresponding mood coordinates, so that the automatic DJ module can incorporate the new additions to the continuous mix. One of the many challenges we have faced implementing and deploying this system has been managing and optimizing the audio processing with the Web Audio API. Apart from trying to keep the CPU load at reasonable levels, we have encountered interesting dilemmas regarding where the different phases of processing should happen. Currently the audio is accessed directly from Deezer API and the server only deals with track metadata, user and party coordinates and determining which track should be mixed in next, leaving all the audio processing to the client devices. Alternatively, given sufficient server resources, the audio mixing could also take place on the server, that sends out the same stream to all the clients.

¹⁵<https://github.com/dynamic-music/dymo-player>

¹⁶<https://tonejs.github.io>

¹⁷<https://www.npmjs.com/package/web-audio-scheduler>



Figure 7: The structure of moodplay.github.io

Technically, the system consists of an Angular¹⁸ front-end accessible at <https://moodplay.github.io> and Express.js¹⁹ server application, <https://moodplay-data.herokuapp.com/> that stores track metadata, mood coordinates and audio features for the auto-dj module. Both components are developed as open source projects under GNU General Public License v3.0. The code repository for the front-end can be accessed at <https://github.com/darkjazz/moodplay> and the back-end is available at <https://github.com/darkjazz/moodplay-server>.

7. ACKNOWLEDGMENTS

This work was supported by EPSRC Grant EP/L019981/1, “Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption”. We would like to acknowledge the contribution of Mathieu Barthet and György Fazekas, who created the original installation version of Moodplay.

8. REFERENCES

- [1] A. Allik, G. Fazekas, M. Barthet, and M. Sandler. myMoodplay: an interactive mood-based music discovery app. In *Proc. of the 2nd Web Audio Conference (WAC)*, 2016.
- [2] I. Andjelkovic, D. Parra, and J. O’Donovan. Moodplay: Interactive mood-based music discovery and recommendation. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP ’16*, pages 275–279, New York, NY, USA, 2016. ACM.
- [3] M. Barthet, G. Fazekas, A. Allik, and M. B. Sandler. Moodplay: an interactive mood-based musical experience. In *Proceedings of the Audio Mostly 2015 on Interaction With Sound, AM ’15, Thessaloniki, Greece, October 7-9, 2015*, pages 3:1–3:8, 2015.
- [4] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer. Using mutual proximity to improve content-based audio similarity. In *ISMIR*, volume 11, pages 79–84, 2011.
- [5] F. Thalmann, L. Thompson, and M. Sandler. A user-adaptive automated dj web app with object-based audio and crowd-sourced decision trees. In *Proceedings of the 4th Web Audio Conference, Berlin*, 2018.

¹⁸<https://angular.io>

¹⁹<http://expressjs.com>

iMuSciCA: A Web Platform for Science Education Through Music Activities

Kosmas Kritsis
Athena R. C., Greece
kosmas.kritsis@athenarc.gr

Manuel Bouillon
University of Fribourg,
Switzerland
manuel.bouillon@unifr.ch

Daniel Martín-Albo
Wiris, Spain
dmas@wiris.com

Carlos Acosta
Leopoly, Hungary
carlos.acosta@leopoly.com

Robert Piéchaud
IRCAM, France
robert.piechaud@ircam.fr

Vassilis Katsouros
Athena R. C., Greece
vsk@athenarc.gr

ABSTRACT

In this paper we present the iMuSciCA web platform which addresses secondary school students with the aim to support mastery of core academic content on STEM subjects (Physics, Geometry, Mathematics, and Technology) alongside with the development of creativity and deeper learning skills through the students' engagement in music activities. Herein we focus on the technical implementation of the various music related tools and Activity Environments hosted by the iMuSciCA workbench, which are exclusively developed with modern web technologies.

Keywords

Web Audio, WebGL, STEAM education, music interaction

1. INTRODUCTION

The iMuSciCA platform is a European funded project which aims at providing a web-based workbench for the deeper learning of STEM (Science, Technology, Engineering and Mathematics) subjects by bringing Arts and especially encouraging learners in co-creative music activities [6]. Music plays a crucial role in cognitive development of humans since the early years of life, supported by multiple studies which report that participation in music lessons is associated with higher academic abilities of students [1, 9].

Music and STEM resonate with each other within the iMuSciCA educational framework and work as a real paradigm of how the art creativity is fostered in to STEM creativity and vice-versa. iMuSciCA uses inquiry-based science education (IBSE) phases [4, 10]: engage, imagine, create, analyze, communicate and reflect. Thus, providing real evidence of the positive impact of the interaction between arts and science, on the creativity and innovation thinking of learners. The main objective of the iMuSciCA project is to develop a set of practical activities, such to give learners the opportunity to explore different phenomena and laws of

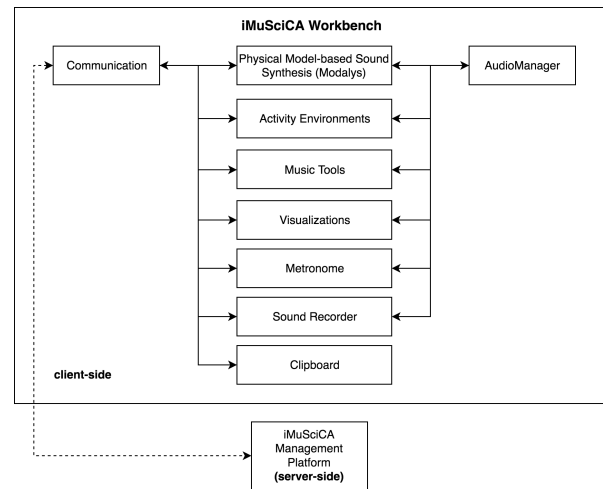


Figure 1: General overview of the system architecture.

physics, geometry, mathematics and technology through creative music activities, to examine them from various viewpoints and to increase integration among various curriculum subjects contributing to innovative cross-disciplinary educational approaches.

2. THE IMUSCICA WORKBENCH

In this section we briefly present the general architecture of the iMuSciCA web platform and its main components, focusing on the technical aspects of the various implemented Activity Environments (AEs) and tools, and specially those where the music activities take place.

2.1 General Architecture

The iMuSciCA Workbench¹ is the main web platform where the user is able to perform STEAM-related activities according to the iMuSciCA pedagogical framework. It provides a set of AEs and Tools, categorized according to the different STEAM domains in music, science and mathematics, engineering and technology. The various tools and AEs are hosted in different web servers; hence the key role of Workbench is to operate as the parent HTML document

¹<https://workbench.imuscica.eu/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

that loads the AEs as child *IFrame* elements and to provide a common communication framework for supporting their interoperability (Figure 1). Most of the provided services are written in JavaScript and run on the user’s browser (i.e. client-side). Furthermore our system provides a cloud storage for saving user-generated data, such as 3D virtual instrument models and audio recordings. This functionality is handled by a server-side service, named the iMuSciCA Management Platform (IMP).

2.2 Communication framework

The communication framework implements an internal protocol that was developed for exchanging information across the various AEs and tools. The communication protocol was implemented based on the *Postal.js*² asynchronous in-memory message bus library, in order to facilitate the different performance delays of the AEs. Moreover, since all AEs and tools are *IFrame* elements, Workbench handles and federates the allowed communication channels with the *postal.xframe* plugin. The following code snippets demonstrate the postal unique identifier registration of the 3D Instrument Interaction environment, its subscription to Workbench clipboard channel for loading data as well as the publish function for copying data to clipboard.

```
// unique identifier registration
postal.instanceId("performance");

// sending data to clipboard
postal.publish({
  channel: "clipboard",
  topic: postal.instanceId() + ".export.receive",
  data: {content: someData}
});

// receiving data from clipboard
postal.subscribe({
  channel: "clipboard",
  topic: postal.instanceId() + ".import.receive",
  callback: function(data, env){
    // do something with the loaded data
  }
});
```

2.3 Audio Manager

Since multiple tools generate and exchange audio data, there was a need to develop the Audio Manager framework in order to have a centralized control throughout the iMuSciCA Workbench. This was achieved by employing the *WebAudio* API for implementing the audio routing between the various audio modules, which can be working either as transmitters or receivers (see Figure 2). When a tool or environment has to generate audio, it needs to get the *AudioContext* from the central Audio Manager of the iMuSciCA Workbench, create a transmitting node, and finally share it with the central Audio Manager by calling the following function.

```
// generate audio
audioManager.receiveAudioFromNode(
  tool.transmitterNode
);
```

²<https://github.com/postaljs/postal.js>

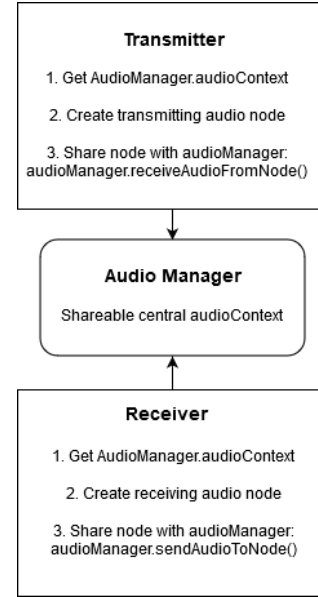


Figure 2: The Audio Manager framework.

A similar process is followed in the case where a tool or environment wants to receive audio from the central Audio Manager.

```
// receive audio
audioManager.sendAudioToNode(
  tool.receiverNode
);
```

2.4 Sound Recorder and Metronome Tool

The Sound Recorder (SR) tool enables the user to record the generated audio from the various AEs and tools that produce sound, as well as the input from the microphone. After the end of a recording, the user can playback the audio caption, save it to the IMP or copy it to the Workbench Clipboard. Additionally, the recorded audio is represented by a Blob object (i.e. raw data), that requires further compression in order to minimize its size and facilitate its transmission between the AEs via the communication framework as detailed previously. To this end, all the compression and decompression processes are implemented with the *LZ-string*³ compression library, which produces encoded UTF-16 strings that can be safely included in a *Postal.js* publish message.

The Metronome tool (MT) provides the basic functionalities of an original metronome, including start, stop, variable time signature (numerator and denominator) and resolution. The MT timer leverages the accuracy of the *AudioContext.currentTime* property to precisely calculate the beat intervals of the selected tempo. A tool can retrieve from the Workbench MT many pieces of information, regarding the time and rhythmical configurations, as well as its events (pulses) in order to synchronize and schedule audio events.

³<https://github.com/pieroxy/lz-string>

2.5 Physical Model-Based Sound Synthesis

The physical model-based sound synthesis engine utilizes a web port of Modalys [3], which tries to mimic the sound of natural instruments as closely as possible. Under this paradigm, sounding objects (strings, plates, bars, membranes or tubes), which are described in physical terms of geometry, material and other properties, can be connected with one another through specific interactions, such as striking, bowing, and blowing. The evolution of the physical model system is processed in real-time according to the complex physics equations that rule the corresponding phenomena of both objects and interactions, resulting in a very subtle and lively sound.

The code base of the engine was originally developed in C++, and it was ported to JavaScript using the *Emscripten*⁴ transpiler in order to utilize it in HTML5 environments. Initially the *Modalys.js* was deployed as a server-side service, however this approach introduced extra latency due to the network communication overhead. We then used various optimization strategies until we were eventually able to perform a musical instrument in a satisfactory way, without perceptible latency. This was achieved by developing a wrapper library based on the *Web Workers* API that enables the communication of the *WebAssembly*⁵ module, with the Workbench Audio Manager and the rest of the AEs that employ the Modalys engine.

2.6 Audio Visualizations

The iMuSciCA platform provides three visualizations, including the Snail, the 3D Spectrogram and the 2D visualizations, which are briefly described as follows.

2.6.1 Snail

The Snail [5] is a real-time visualization application that incorporates an original spectral analysis technology, combined with a display on a spiral scheme, as it is depicted in Figure 3a. The center of the spiral corresponds to the lowest frequencies, while the perimeter to the highest frequencies. Furthermore, each turn represents one octave, so that the tones are organized with respect to angles. The spectrum analysis is displayed according to perceptive features, in a way that the loudness of the corresponding frequencies are mapped to both the line thickness and its brightness. The original implementation of the Snail was developed in C++ and it was ported to JavaScript using the *Emscripten* transpiler. The FFT algorithm runs on a *WebAssembly* module, that communicates with the Audio Manager through an *ArrayBuffer*.

2.6.2 3D Spectrogram Visualizer

The 3D Spectrogram Visualizer (see Figure 3b) utilizes the FFT of an *AnalyzerNode* in order to retrieve the frequency components that comprise the input signal originating from the main audio output of the Audio Manager. Moreover, the 3D graphics were developed with the *three.js*⁶ 3D library that uses a *WebGL* renderer. On each frame, the rendering function checks if new data has arrived from the *AnalyzerNode* in order to update the displayed graphics with the current spectral values.

⁴<http://emscripten.org>

⁵<https://webassembly.org/>

⁶<https://threejs.org/>

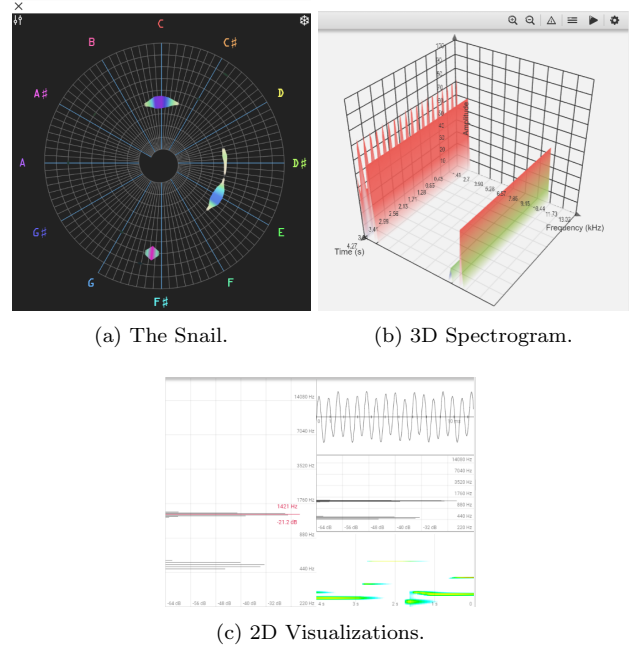


Figure 3: The visualizations provided in the iMuSciCA web platform.

2.6.3 2D Visualizations

The 2D Visualizations (see Figure 3c) display the audio data of the Audio Manager main output node in three different HTML canvases; the first canvas displays the waveform in time domain, while the second and third canvases display the frequency domain analysis, as it is computed from an *AnalyzerNode*. Moreover, the second canvas, presents the amplitude of the different frequency harmonics of the input signal (FFT), while the third canvas displays the variation of the frequencies over time (2D spectrogram).

2.7 Activity Environments

The iMuSciCA workbench provides nine AEs in total. However in this section we focus only on those environments that support musical activities, including the Musical Whiteboard, Performance Sampler, Tone Synthesizer, 3D Instrument Design and Interaction environments as well as the Acouscope.

2.7.1 The Musical Whiteboard

The Musical Whiteboard (MW) is a web-based environment that enables the free drawing of music on touch-enabled computers [2]. The x-axis represents time and the y-axis is mapped to the frequency domain, that is displayed on the right in Hertz with a correspondence in notes on the left (see Figure 4f). The user can draw on the canvas using either the mouse, his finger or a stylus on a touch-sensitive computer and the MW produces a live sonification of the drawn strokes. The various colors represent different sound types (sinewave, triangle, sawtooth and square waveforms), thus enriching the overall music creativity. Once the tune-drawing is complete, the user can playback the whole creation from left to right. For a wider range of frequencies the user can use the zoom buttons to increase the frequency

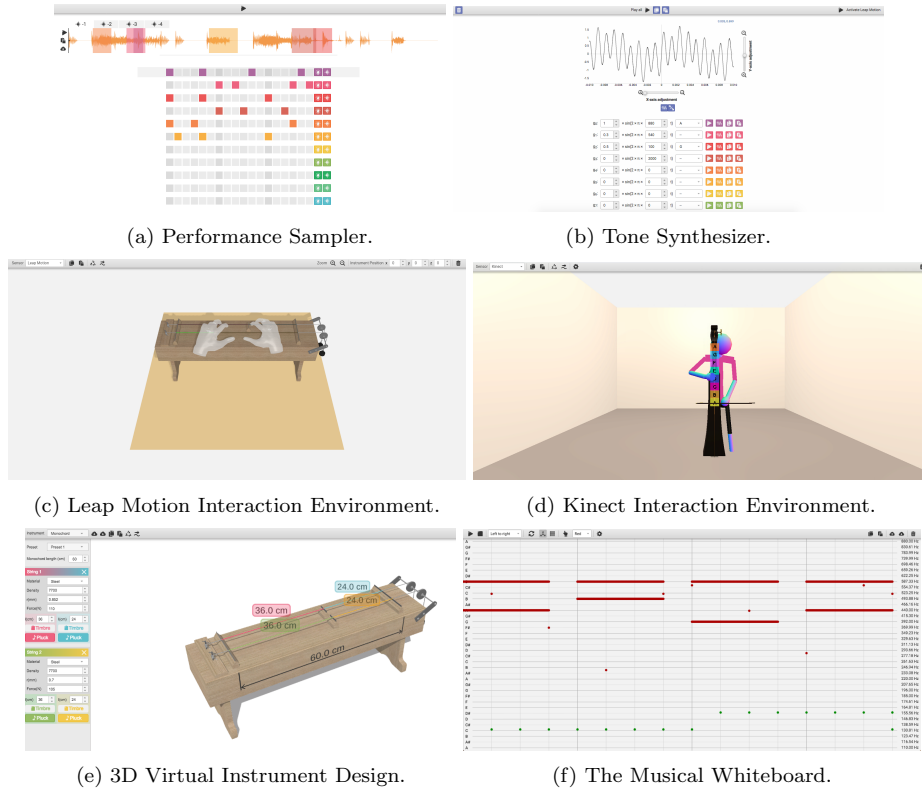


Figure 4: Screenshots of the music related Activity Environments provided by the iMuSciCA Workbench.

and time spans. Various options are available in the settings menu, such as activating the loop playback or the “snap-to-line” option, which constrains the strokes to be drawn on the note lines. The playback speed, time signature and resolution is controlled from the Workbench MT.

2.7.2 Performance Sampler

The Performance Sampler is a tool that allows users to load up to four recorded waveforms (either from the clipboard or from the IMP), select parts from those recordings and “program” the activation of each sample at specific time intervals through a sequencer matrix (see Figure 4a). This tool is intended to allow exploration of compositions that can emerge from recordings created with the 3D Instrument Interaction tool (described later), where users interact with virtual instruments in real-time. The user can use up to 11 samples (number of rows in the sequencer matrix); the number of columns in the sequencer matrix depends on the time signature and time resolution selected by the Workbench MT and represents a duration of one bar. Additionally, the above factors affect the width of each cell in the sequencer matrix and the spacing between consecutive cells, giving a visual interpretation of the metrical setup of the Workbench MT. The user is also given the ability to employ random selection of sample parts and activations, either for each individual sample or for all samples simultaneously. Sample selection and manipulation was implemented based on the “Region” plugin of the *Wavesurfer.js*⁷ library.

⁷<https://wavesurfer-js.org/>

2.7.3 Tone Synthesizer

The Tone Synthesizer (see Figure 4b) is an environment for investigating the audio and visual behavior between combinations of sinusoidal functions. Moreover, the user can activate up to 10 sinusoidal waveform generators with a given frequency and amplitude, listen the results and visualize the waveform. It is also possible to load a “timbre” object exported from a virtual instrument, where the first 10 partials of the instrument are “assigned” to each sinusoidal element, where the frequency and amplitude of the individual partials are adopted by the respective sinusoidal elements. In this sense, the waveform visualisation is an analytic interpretation of the sinusoidal functions rather than a representation of the actual produced waveform. The user can also manipulate the analytic waveform visualisation by zooming in/out, moving horizontally/vertically and applying optimal zoom, which focuses horizontally on two periods of the minimum frequency and vertically on the total amplitude of the waveform. Furthermore, the Tone Synthesizer is designed to function as a theremin-like digital musical instrument by employing the Leap Motion sensor. Specifically, the amplitude and frequency values of the 10 sinusoidal elements are mapped to the x and y positions of the user’s fingertips, as they are calculated by the Leap Motion JavaScript SDK. The user needs to extend a finger in order to activate the respective sinusoid, whilst elements corresponding to non-extended fingers have zero amplitude. When the user activates the Leap Motion-enabled interaction, a graphical visualisation provides real-time information about the frequency and amplitude of each extended finger.

2.7.4 3D Virtual Instrument Design

The 3D Virtual Instrument Design environment (see Figure 4e), enables the user to design 3D graphical models of predefined virtual instruments without requiring any advanced skills. Specifically, the environment provides six predefined instrument models including circular and square-shaped membranes, a two string monochord, a guitar, a tromba marina (bass monochord) and a xylophone. All 3D models are represented by the *glTF*⁸ format, which facilitates the data transmission throughout the iMuSciCA platform. In addition to the 3D editing functionalities, the user is able to modify and experiment with several physically-based modeling parameters of the instruments such as size, material density and tension, which can be sonified by the Modalys engine.

The modular core engine is written in C++ and it utilizes the *OpenGL*⁹ graphics library, thus enabling a cross platform architecture. The modeling engine is already ported to several platforms, including, desktop, mobile and web-based versions. The proposed system benefits from the JavaScript port of the 3D modeling engine, produced with the *Emscripten* transpiler in order to enable the 3D design services to run on any HTML5-compatible web browser. The core modeling engine utilizes the *WebAssembly* framework for improving the overall performance of the environment. From a technical perspective, the engine and the GUI are separate instances, meaning that the HTML UI communicates with the core engine through a proprietary API.

2.7.5 3D Instrument Interaction

The 3D Instrument Interaction environment enables the user to perform the 3D virtual music instruments, by utilizing two motion sensors, including the Leap Motion and the Microsoft Kinect sensors. According to the selected sensor the 3D Instrument Interaction environment loads a different subsystem with the corresponding back-end architecture. The different sub-environments are depicted in Figures 4c and 4d.

The goals and technical details of the Leap Motion enabled performance environment have been previously presented in [8]. Furthermore, previously developed heuristic-based interaction methods have been updated [8], while experimental deep Neural Network architectures have been employed for improving the gesture recognition module [7]. In this regard, new interactions were developed, by introducing the virtual 3D models of a set of mallets, drumsticks as well as a bow, in order to interact with the virtual xylophone, membranes and the tromba marina respectively. Specifically, when the user selects to perform the xylophone or any of the two membranes, the system maps the palm position and its orientation to a virtual mallet (xylophone) or drumstick (membrane). On the other hand, performing the tromba marina entails continuous interaction between the movement of the bow and the string of the instrument, thus requiring the user to perform a natural gesture as holding a real bow in order to control the horizontal movement of the virtual bow. Other supported modules that enrich the educational and creative aspects of the environment include a gesture recorder and a musical/rhythmical quantizer, enabling the user to edit his/her recordings while giving a

deeper insight of his/her performances by reproducing the same visual and auditory feedback.

Regarding the Kinect-enabled environment, we have developed a local server written in C#, that uses the *WebSocket* protocol for broadcasting the skeletal tracking data to the client, which in our case is the Kinect-enabled web environment. The server executable is available online¹⁰. Furthermore, the interactions designed for the Kinect-enabled environment can support both hands as primary, in addition to a virtual fingerboard for selecting different chords when performing the guitar. Regarding the xylophone and the circular membrane, the instruments appear in front of the player's avatar for facilitating the interaction that happens by using the hands as mallets. In the case of the tromba marina, the user's primary hand controls the virtual bow, whilst the non-primary hand controls whether a note is played or not, depending on its position on the virtual fingerboard. The movement velocity of the primary hand also modifies the amplitude of the note. The system has been designed to support collaborative performances where 2 players are able to perform different instruments [11].

All virtual instruments, excluding the tromba marina, produce sound by utilizing the "trigger" function of the Modalys engine, which is much faster and more computationally efficient, thus consuming less resources from the user's computer and improving the overall user experience of the AE. Since the tromba marina is a bowed instrument, it requires a continuous interaction approach for simulating the excitation of the string from the friction of the bow as it moves.

2.7.6 Acouscope

The Acouscope Environment employs a hardware device called HyVibe¹¹, that uses state-of-the-art actuation technology for identifying the frequency response of any surface. As it is presented in Figure 5a, the hardware comprises of two transducers: a) an electric actuator that is used for applying force on a surface and b) a piezo sensor, for sensing and converting the reactive vibrations of the surface to an electric, measurable current. The transducers are connected through cable to a microcontroller, that runs an embedded algorithm for analyzing the frequency response of the surface. The web interface (see Figure 5b) communicates with the hardware through the Web Bluetooth API, in order to trigger the chirp sound that is sent to the attached surface via the electric actuator and then retrieve and display the FFT analysis of the vibration response, as it is sensed from the piezo sensor. Finally, the corresponding eigenfrequencies are visualized as a note sequence on a stave based on the *VexFlow*¹² music notation JavaScript library.

3. DISCUSSION AND CONCLUSIONS

In this paper, we present the iMuSciCA web platform, an innovative pedagogical framework with cutting-edge technologies for carrying out STEAM activities. Furthermore, the workbench provides a set of musical AEs, as well as visualizations and various tools for supporting and enhancing the interdisciplinarity of the learning process.

⁸<https://www.khronos.org/glTF/>

⁹<https://www.opengl.org/>

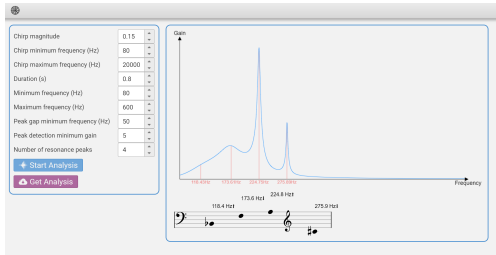
¹⁰<https://athena.imuscica.eu/software/kinect/websocket/kinectImuscica.zip>

¹¹<https://www.hyvibe.audio/>

¹²<http://www.vexflow.com/>



(a) The HyVibe device.



(b) Acoscope Web Interface.

Figure 5: The Acoscope system.

From a technical perspective, our goal was to develop an easily accessible and OS independent platform. In this sense, we decided to implement the iMuSciCA Workbench as a web application by employing state-of-the-art web APIs and libraries. The various tools and AEs are hosted in different web servers which are loaded as child *IFrame* elements within the iMuSciCA Workbench parent HTML document. In order to support their interoperability and handle the different performance delays, we developed a common communication framework based on the *Postal.js* asynchronous message library. Our system leverages the modular design of the WebAudio API for implementing the Audio Manager framework, that functions as a centralised controller for routing audio data across the AEs and tools. Additionally, the MT timer employs the accuracy of the *WebAudio* *AudioContext.currentTime* property for calculating the beat intervals of the selected tempo.

Large JavaScript objects, such as 3D instrument models (gITF format) and audio recordings (Blob objects), are compressed using the *LZ-string* library. This approach facilitates their transmission and minimizes their memory footprint. Computational heavy AEs and tools were initially running as server-side services; however the additional network overhead was an important drawback, affecting the overall user experience. After testing out various system architectures, we decided to employ the *Emscripten* transpiler and the *WebAssembly* standard in order to run these programs as client-side services, while the *Web Workers* API provided us the foundations for implementing the appropriate wrapper libraries. The tremendous evolution of the available web technologies during the last years, allowed us to fulfill the ambitious goals of the iMuSciCA platform.

4. ACKNOWLEDGMENTS

The iMuSciCA project has been fulfilled and funded by the European Union’s Horizon 2020 research and innovation program under the grant agreement No 731861.

5. REFERENCES

- [1] J. W. Bequette and M. B. Bequette. A place for art and design education in the stem conversation. *Art education*, 65(2):40–47, 2012.
- [2] M. Bouillon, F. Simistira, R. Ingold, and M. Liwicki. Drawme: Drawing canvas for music creation - a new tool for inquiry learning. In *International Conference On Learning And Teaching (ICLT 2018)*, Singapore, 2019.
- [3] R. E. Causse, J. Bensoam, and N. Ellis. Modalys, a physical modeling synthesizer: More than twenty years of researches, developments, and musical uses. *The Journal of the Acoustical Society of America*, 130(4):2365–2365, 2011.
- [4] W. S. Gershon and O. Ben-Horin. Deepening inquiry: What processes of making music can teach us about creativity and ontology for inquiry based science education. *International Journal of Education & the Arts*, 15(9):1–38, 2014.
- [5] T. Hélie and C. Picasso. The Snail: a real-time software application to visualize sounds. In *International Conference on Digital Audio Effects (DAFx-17)*, Edinburgh, United Kingdom, 2017.
- [6] V. Katsouros, E. Fotinea, R. Frans, E. Andreotti, P. Stergiopoulos, M. Chaniotakis, T. Fischer, R. Piechaud, Z. Karpati, P. Laborde, D. Martín-Albo, F. Simistira, and M. Liwicki. imuscica: Interactive music science collaborative activities for steam learning. *Designing for the User Experience in Learning Systems*, pages 123–154, 2018.
- [7] K. Kritsis, A. Gkiokas, M. Kaliakatsos-Papakostas, V. Katsouros, and A. Pikrakis. Deployment of IstmS for real-time hand gesture interaction of 3d virtual music instruments with a leap motion sensor. In *International Conference on Sound and Music Computing (SMC 2018)*, Limassol, Cyprus, 2018.
- [8] K. Kritsis, A. Gkiokas, Q. Lamerand, R. Piechaud, C. Acosta, M. Kaliakatsos-Papakostas, and V. Katsouros. Design and interaction of 3d virtual music instruments for steam education using web technologies. In *International Conference on Sound and Music Computing (SMC 2018)*, Limassol, Cyprus, 2018.
- [9] E. G. Schellenberg. Examining the association between music lessons and intelligence. *British journal of psychology*, 102(3):283–302, 2011.
- [10] J. Trna and E. Trnova. Inquiry-based science education in science and technology education as a connectivist method. In *8th International Conference on Education*, Samos, Greece, 2012.
- [11] A. Zlatintsi, P.-P. Filntisis, C. Garoufis, A. Tsiami, K. Kritsis, M. Kaliakatsos-Papakostas, A. Gkiokas, V. Katsouros, and P. Maragos. A web-based real-time kinect application for gestural interaction with virtual musical instruments. In *Audio Mostly 2018 (AM 2018)*, Wrexham, Wales, 2018.

QuaverSeries: A Live Coding Environment for Music Performance Using Web Technologies

Qichao Lan
RITMO
Department of Musicology
University of Oslo
qichao.lan@imv.uio.no

Alexander Refsum Jensenius
RITMO
Department of Musicology
University of Oslo
a.r.jensenius@imv.uio.no

ABSTRACT

QuaverSeries consists of a domain-specific language and a single-page web application for collaborative live coding in music performances. Its domain-specific language borrows principles from both programming and digital interface design in its syntax rules, and hence adopts the paradigm of functional programming. The collaborative environment features the concept of ‘virtual rooms’, in which performers can collaborate from different locations, and the audience can watch the collaboration at the same time. Not only is the code synchronised among all the performers and online audience connected to the server, but the code executing command is also broadcast. This communication strategy, achieved by the integration of the language design and the environment design, provides a new form of interaction for web-based live coding performances.

1. INTRODUCTION

Live coding, when used in a musical context, refers to a form of performance in which the performers produce music by writing program code rather than playing physical instruments [4]. During the past decade, dozens of live coding languages have emerged.¹ These languages run in various environments, such as the desktop, the browser, and embedded systems (Raspberry Pi, BeagleBone, etc.). The number of programming languages developed for live coding can, in some ways, indicate that performers want to develop their subjective language syntaxes tailored to their musical expressions. Some examples of such new syntaxes are Tidal-Cycles [18], ixi lang[11], Lich.js [6], and Mercury [17].

There have been some discussions about how live coding languages relate to musical instruments [3], but relatively little attention has been devoted to analysing how it is possible to ‘transduce’ electronic instrument knowledge to the syntax design itself. That is, what types of symbols should be used for what musical purposes, how should different elements be connected, and so on. Instead, the syntax in most

live coding languages is mainly borrowed from other programming languages. For example, the use of parentheses is ubiquitous in programming languages, and it is adopted in almost every live coding language. That is the case even though live coding without the parentheses is (more) readable for humans [12].

Inheriting standard programming syntax may create difficulties for non-programmers who want to get started with live coding. We have therefore been exploring how it is possible to design a live coding syntax based on the design principles of digital musical interface design. This includes elements such as audio effect chains, sequencers, patches, and so on. The aim has been to only create a rule by 1) borrowing from digital musicians’ familiarity with the digital interfaces mentioned above, or 2) reusing the syntax from existing programming languages to help the parser to work. The design principle is also aligned with the concept of *ergonomics*, namely the ‘application of work processes from one domain to another’ [13]. The goal of this principle is to lower the learning curve of our language syntax, especially for non-programmers.

The second aim of our current exploration is to develop a live coding language that is usable for a larger group of people. This can be seen as part of the trend of ‘musical democratisation’ [8]. Our experience with running workshops for larger groups of university students or pupils in schools is that software that needs to be installed locally makes it much more difficult to get started making music quickly. We, therefore, see browser-based interfaces as the best solution for minimal setup time.

Finally, as part of our interest in exploring the blurring of roles between performers and perceivers, we have also looked at how it is possible to include the ‘audience’ in live coding. An online deployment makes it possible to not only share the code among performers, but it also makes it possible for the audience to easily join into the online live coding performance. This requires a delicate organisation of a stack of web technologies.

In the current paper, our main research question is:

- How can web technologies influence the music interaction between performers and the audience?

From this two sub-questions emerge:

- How can we design a live coding environment that makes the audience part of the performance? How should the environment be modified to meet this requirement?



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

- How can we use knowledge from digital musical instrument design when developing the syntax of a live coding language? What are the trade-offs that we have to make to achieve this goal?

The paper starts with a background section in which we introduce some related work. Section 3 presents the new domain-specific language, elaborating on how the parser is designed and its semantics. In Section 4, we describe the interface design and explain the communication strategies based on the Firebase real-time database. Finally, in Section 5, we discuss how the environment fits the language design, and how it experimentally changes the relation between performers and audience.

2. BACKGROUND

Many existing live coding environments are installed locally and use the SuperCollider music programming language as the sound engine [15]. With the advent of the Web Audio API, there has been a shift towards developing live coding environments with web technologies. In the following, we will reflect on these two approaches to live coding.

2.1 Live coding with SuperCollider

SuperCollider consists of a programming language called *sclang* and an integrated development environment (IDE). In the IDE, users can boot an audio server (*scsynth*) in the background, and write the code following the syntax of *sclang*. The code, when executed, will be compiled into Open Sound Control (OSC) messages, and sent to the *scsynth* server to control the music sequence. One typical workflow in SuperCollider is to define the synthesiser architecture with the keyword `SynthDef`, and then play the *Synth* in a SuperCollider music sequence (*Pattern*).

Several live coders have chosen to design their syntaxes on top of SuperCollider. For instance, TidalCycles (Tidal) is a domain-specific language written in the Haskell programming language [18]. During live coding, the Tidal code will be interpreted and sent as OSC messages to control the sound engine called *SuperDirt* running in SuperCollider. FoxDot follows a similar architecture but uses Python as the host programming language [9]. Additionally, Troop is a collaborative environment developed for both Tidal and FoxDot, which allows users at the same network to co-edit and share the code on the screen [10].

An inconvenience with the above-mentioned environments, relying on one or more programming languages in addition to SuperCollider, is that it requires several steps of installation. A more user-friendly solution, then, is Sonic Pi, which is also built on SuperCollider audio engine, but offers a single, complete installation package [1]. Even though several OSes are supported, it does not currently run on desktop Linux or Chrome OS. In our experience, this makes it less ideal for schools. And for quick introductory workshops to live coding, we find that having to rely on software installs is less than ideal. For such situations, a web-based solution is more feasible and scalable.

2.2 Web-based live coding

Web-based or browser-based live coding environments only require an up-to-date browser to get started with live coding. With the rapid progress of the Web Audio API, the sound synthesis possibilities and timing capabilities for

browser-based live coding have matured quickly. Two good examples of this are the JavaScript-based Gibber environment [20], and the Lisp-style language Slang.js [21]. Although the latter currently does not support collaboration, its parser, written in Ohm.js, provides a valuable example for our development. Another inspiration for us is from EarSketch [16], a music producing environment mainly designed for normal programming education, and its use of the Firebase real-time database pointed us in the direction of a collaborative live coding solution [23].

Some other web-based environments serve as interfaces for other languages. Estuary is a system built for live coding with Tidal in browsers [19]. It has several unique features: collaboration in four different text fields, the support for both SuperCollider and the Web Audio API, and so on. Estuary makes it possible to live code together from different locations, and has been shown to work reliably in cross-continental live coding.

As can be summarised from the brief review of existing live coding environments, the programming languages, syntaxes, and interfaces are diversified. In our exploration, we have been borrowing parts from many of these when designing our syntax and environment.

3. LANGUAGE DESIGN

The syntax design of QuaverSeries is based on a functional programming paradigm.² The following sections will describe its syntax and how Ohm.js and Tone.js have been used to implement the parsing and semantics.

3.1 Note representation

The note representation is probably the element that is varied the most among live coding languages. QuaverSeries is based on the idea of a music sequencer. Our prototype syntax looks like this:

```
60 _62 63_64_65_ 66_67_68_69
```

The *sequence* has only three elements: numbers, underscores and blank spaces. The numbers refer to MIDI notes, with 60 being ‘middle C’. A blank space indicates a separation of individual *notes*, while an underscore denotes a musical *rest*.

A *sequence* will always occupy the duration of a whole note, and all the *notes* will be divided equally. To illustrate, the one-line *sequence* above will be divided into four *notes*: 60, _62, 63_64_65_, and 66_67_68_69_, with each of them occupying a quarter note length. Each *note* can be further (equally) divided by the total number of MIDI notes and rests. On the example above, _62 means that an eighth MIDI note 62 will be played on the off-beat, after an eighth rest. Likewise, 63_64_65_ means eighth note triplets.

As can be seen from the examples above, we create the syntax rule by referring to the musical sequencer, and add extra programmability to the syntax using the dividing algorithm invented in TidalCycles [18]. One direct influence here is that we form a left-to-right typing flow. For the sake of consistency, this flow is kept in other parts of the syntax design as well. Hence, there is no pairing symbol such as parentheses and quotation marks in the syntax.

The sequence can then be connected to a sound generating module using the double greater-than sign (`>>`) which is

²<https://github.com/chaosprint/QuaverSeries>

to keep the default value of a parameter. For instance, the `adsr` function has a value of zero for the *sustain* parameter, which means that there is no need to write the *release* value. Hence, we can use an underscore to represent the release.

The second block demonstrates a concept called *reference*. With a tilde-prefix (`~`), a function name becomes a reference that can link two signal chains with one signal chain modulating a parameter of the other. In the example above, the cut-off frequency of the low-pass filter is modulated by a low-frequency oscillator (`lfo`). Hence, to keep the consistency, it is suggested users add a reference at the beginning of every function chain.

3.4 Semantics

The semantics part of QuaverSeries defines how the code should be executed after being parsed. In Ohm.js, the parsing and semantics definitions are separated. Thus after the parser reads through the code and identifies several valid functions, Ohm.js needs further instructions on how to deal with these functions. For instance, when the parser detects a number, the parser will return the number as a string character. It is therefore necessary to write the semantic action as a JavaScript function that converts the string to a float in JavaScript, so that it can be used for numerical operations.

The semantics definition of QuaverSeries is written with Tone.js, a JavaScript audio and sound synthesis library based on the Web Audio API [14]. Currently, the functions are categorised into three parts: *control*, *effect* and *synth*. In the semantics definition, each function is organised into different tracks. Each track has its attributes, including *note*, *synth*, and *effect*.

Once a `run` message is received, the parser will read through the whole page, and convert every function to Tone.js code based on the semantics definition. For instance, when the `loop` function is detected, a Tone.js *Sequence* instance will be created. Likewise, if a `synth` function is identified, a Tone.js *Synth* instance will be created. If audio effects are found, the relevant Tone.js effect instances can be created. Finally when the `amp` is detected, the `connect` method of the *Synth* instance will be called to connect all the effects, with the amplifier (`Tone.Master`) at the end of the effect list.

As a summary, when the `run` command is given, the parser will read through the whole page and identify the functions. Next, semantics action will be executed by constructing Tone.js instances and calling their methods. The `update` command also reads the whole page, and updates each node that is playing, although it will first be effective at the beginning of the next bar.

4. ENVIRONMENT DESIGN

Collaboration has been an important motivation when developing the QuaverSeries live coding environment.³ The aim is to create a web application that live coders can use to collaborate in different *virtual rooms*, and where the audience can go to a particular room to watch an ongoing performance, albeit with a different level of access (see Figure 2).

4.1 Collaboration support

³<https://quaverseries.web.app>

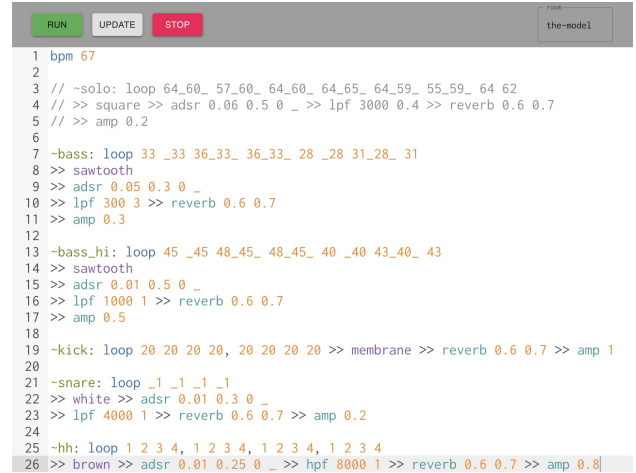


Figure 2: The QuaverSeries interface prototype. The syntax highlight has been implemented as a Ace editor theme. The buttons (Run, Update and Stop) are mapped to the keyboard shortcuts `Command/CTRL + Enter`, `Shift + Enter`, and `Command/CTRL + Period(.)`, respectively. The keyboard shortcut `Command/CTRL + Slash(/)` is for commenting out lines of code, which can be useful for muting a track during the performance.

Tools and algorithms such as Firebase and Operational Transformation have made the implementation of real-time code sharing much more approachable than it was only some years ago [5]. Firepad is an open-source tool that mainly uses Firebase realtime database and Operational Transformation algorithm. Thus, it provides a solution for synchronising code and sharing the cursor position between clients. In QuaverSeries, Firepad is used to share the code, while a customised strategy is designed to broadcast the related `run` and `update` commands to every client connecting to the database, including both the performers' and the audience members' clients. In this way, a live coder can control the sound running in all the browser clients. This is a similar strategy to what can be found in the Hydra synth, an environment developed for sharing visuals in the browser [7].

In the server database, two entries are storing the states of the `run` and `update`. Hence, once a user sends the `run` command by clicking the button or using the keyboard shortcut, the value of the entry `run` in the database will be set to the Boolean value `true`. As each client connecting to the database has its monitoring function for the value, once the Boolean value `true` is detected, each client will execute a relevant handling function. This function will do two things: 1) execute the code in the editor, 2) set the `run` entry back to the Boolean value `false`. Here is an example to illustrate this in pseudo code:

```
# the server
if Server get "run":
    send "run" to every client

# the client
if Client get "run":
    execute Music Code

# the interface
```

```
if Button "run" is pressed:
    sent "run" to Server
```

The principle of `update` is almost the same as `run`. The only difference is that `update` is used to renew the piece while the music is already on, that is, to calculate the current time and schedule what to play from the beginning of the next bar.

How does the system work in terms of stability? Fortunately, the transmission of only code between clients makes it possible to run the system over connections with very limited bandwidth. Furthermore, since the system is based on looped sequences, and an updating strategy per bar, allows for a considerable network delay without necessarily influencing the final musical result. This system design can, of course, be problematic if an `update` message is sent at the end of a bar. Still, the worst-case scenario is a one-bar offset among different locations. In our real-world testing so far, however, this has not been a problem in practice.

4.2 Live coding democratisation

Musical democratisation—in the sense of making music available to larger audiences—has been a growing trend ever since the invention of the phonograph [22]. Up until now, live coding has been an activity practised by relatively few. This is not only because it has been technically difficult to get started, but also because the community has been comparably small, and access to venues has been limited. Web-based live coding may help to address both of these problems, at the same time making it easier to provide access for online performances to get started.

QuaverSeries is a novel music streaming solution in that it does not stream audio or video but rather focuses on *code streaming*. Thus, instead of watching the audio/video of a performer's screen, the audience can enter a virtual room, watch new code appear on the screen, and have the musical sound rendered locally in the user's own browser. The audience can unidirectionally receive the `run` and `update` message from the server. This makes it possible to stop the rendering of music in the local browser at any time without influencing any other instances running on the machines of other performers or audience members.

The main difference between the performer and audience modes is that in the latter the code is not editable. Technologically speaking, though, every audience has the complete instrument locally, with the performers triggering the code. Thus every audience member could be seen as a collaborator and parttaker in the musicking.

5. CONCLUSION

In this paper we have presented the three parts of QuaverSeries: 1) a new domain-specific language, 2) an interface to edit and run the code, and 3) a new way of collaborating using 'virtual rooms.'

The environment draws extensively on new web technologies, utilising the power of local rendering in the user's browser thanks to the advance of the Web Audio API. This makes it possible to easily and quickly share live coding with lots of people, hence allowing the audience to become 'active participants' in a live coding session. Sharing only code between users makes it possible to create a collaborative environment with low network bandwidth. To minimise the risk of delay in the network, we have employed two strategies:

1) transmitting only code, and 2) calculating the updating based on musical bars. The trade-off of this approach, however, is that it also limits the musical possibilities of the system.

At the moment, the system is based on looped sequences only, resulting in beat-based music. Still, we have found that it is possible to create fairly complex musical results by adding multiple layers in the code. In future development, our first priority is to explore how it is possible for the audience to participate more in online performances. One approach is to build a chatting system in which the audience can write their own code, and propose this code to the performers. It would be up to the performer whether to accept the proposed code or not, similar to the way a *fork* works in the git version control system (such as used on GitHub).

QuaverSeries is currently at a prototype stage. It is fully functional, and we have explored it in a number of jam sessions. The aim now is to test it more in different performance contexts. We also plan to perform a more systematic user study, to understand more about how it works for beginning live coders.

6. ACKNOWLEDGMENTS

This work was partially supported by the Research Council of Norway through its Centres of Excellence scheme, project number 262762 and by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

7. REFERENCES

- [1] S. Aaron. Sonic pi—performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, 12(2):171–178, 2016.
- [2] S. Aaron and A. F. Blackwell. From sonic pi to overtone: creative musical experiences with domain-specific and functional languages. In *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, pages 35–46. ACM, 2013.
- [3] A. F. Blackwell and N. Collins. The programming language as a musical instrument. In *PPIG*, page 11, 2005.
- [4] N. Collins, A. McLean, J. Rohrerhuber, and A. Ward. Live coding in laptop performance. *Organised sound*, 8(3):321–330, 2003.
- [5] C. A. Ellis and C. Sun. Operational transformation in real-time group editors: issues, algorithms, and achievements. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 59–68. Citeseer, 1998.
- [6] T. Hoogland. Mercury: a live coding environment focussed on quick expression for composing, performing and communicating. 2019.
- [7] O. Jack. Livecoding networked visuals in the browser. <https://github.com/ojack/hydra>, 2019.
- [8] D. Kim-Boyle. Network musics: Play, engagement and the democratization of performance. *Contemporary Music Review*, 28(4-5):363–375, 2009.
- [9] R. Kirkbride. Foxdot: Live coding with python and supercollider. In *Proceedings of the International Conference on Live Interfaces*, 2016.

- [10] R. Kirkbride. Troop: A collaborative tool for live coding. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 104–9, 2017.
- [11] T. Magnusson. ixi lang: a supercollider parasite for live coding. In *Proceedings of International Computer Music Conference 2011*, pages 503–506. Michigan Publishing, 2011.
- [12] T. Magnusson. Code scores in live coding practice. In *TENOR 2015: International Conference on Technologies for Music Notation and Representation*, volume 1, pages 134–139. Institut de Recherche en Musicologie, 2015.
- [13] T. Magnusson. *Sonic writing: technologies of material, symbolic, and signal inscriptions*. Bloomsbury Academic, 2019.
- [14] Y. Mann. Interactive music with tone.js. In *Proceedings of the 1st annual Web Audio Conference*. Citeseer, 2015.
- [15] J. McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68, 2002.
- [16] S. McCoid, J. Freeman, B. Magerko, C. Michaud, T. Jenkins, T. Mcklin, and H. Kan. Earsketch: An integrated approach to teaching introductory computer music. *Organised Sound*, 18(2):146–160, 2013.
- [17] C. McKinney. Quick live coding collaboration in the web browser. In *NIME*, pages 379–382, 2014.
- [18] A. McLean. Making programming languages to dance to: live coding with tidal. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 63–70. ACM, 2014.
- [19] D. Ogborn, J. Beverley, L. N. del Angel, E. Tsabary, and A. McLean. Estuary: Browser-based collaborative projectional live coding of musical patterns. In *International Conference on Live Coding (ICLC) 2017*, 2017.
- [20] C. Roberts, G. Wakefield, M. Wright, and J. Kuchera-Morin. Designing musical instruments for the browser. *Computer Music Journal*, 39(1):27–40, 2015.
- [21] K. Stetz. Slang: An audio programming language built in js. <https://github.com/kylestetz/slang>, 2018.
- [22] T. D. Taylor. The commodification of music at the dawn of the era of “mechanical music”. *Ethnomusicology*, 51(2):281–305, 2007.
- [23] A. Xambó, P. Shah, G. Roma, J. Freeman, and B. Magerko. Turn-taking and chatting in collaborative music live coding. In *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*, page 24. ACM, 2017.

Dependencies	Version
React.js	16.2.8
Ohm.js	0.15.0
Tone.js	13.4.9
Material-ui	1.0.0
Firebase	7.0.0
Firepad	1.5.3
Ace	1.4.6

B. PARSER

This is how the parser of QuaverSeries is currently set up.

```

Quaver {

    Piece = Piece #"\\n"? #"\\n"? #"\\n"?
           Block —stack
           | Block

    Block = comment | Track

    comment = "/" " c+

    c = ~"\\n" any

    Track = funcRef? ":"? Chain

    Chain = Chain ">>" Func — stack
           | Func

    Func = funcName listOf<funcElem,
           separator>

    funcElem = para | funcRef

    para = para subPara — combine
          | subPara

    subPara = number | "_"

    number = "-"? digit* "." digit+ —
            fullFloat
            | "-"? digit "." — dot
            | "-"? digit+ — int

    funcRef = "~" validName

    funcName = validName

    validName = listOf<letter+, "_">

    separator = ","? space
}

```

APPENDIX

A. TECHNICAL STACK

The following table lists the main dependencies of QuaverSeries, and their current version number.

MAIA Util: An NPM Package for Bridging Web Audio with Music-theoretic Concepts

Tom Collins^{1,2}

¹Music, Science and
Technology Research Cluster
Department of Music
University of York, UK
tomthecollins@gmail.com

Christian Coulon²

²Music Artificial Intelligence Algorithms, Inc.
P.O. Box 73004
Davis, CA 95617
christianco@gmail.com

ABSTRACT

The Web Audio API and associated JavaScript packages have enabled developers to design interfaces where, to an unprecedented extent, the elements of music are at users’ fingertips. When these interfaces are intended to help users to understand those musical elements, it can be useful to calculate or display manifestations of music-theoretic concepts, such as the key of an excerpt, the segmentation and labeling of chords, and melodic and harmonic intervals.

The MAIA Util package contains JavaScript code for executing these calculations. The input music representations are assumed to be symbolic, coming from MIDI or MusicXML files, or having been estimated from audio. This paper introduces the contents of the package and the music-cognitive research on which some of its constituent algorithms are based.

Some of the methods are of a more basic nature, such as for cyclic permutation of arrays or estimation of the pitch and octave of a note given its MIDI number and surrounding context. We have found use for these methods often enough during music interface development that they are included too.

The MAIA Util package is available for use from <https://www.npmjs.com/package/maia-util>

1. INTRODUCTION

In the domains of sound and interface design, packages such as Tone.js [25, 24] and NexusUI [3, 2] have increased the ease with which programmers can develop and prototype applications that make use of the Web Audio API [1, 29]. In the domain of music information retrieval (MIR), progress continues to be made with automatically analyzing and generating music-related data, especially with regards the training and testing of deep learning algorithms, and much of it implemented in open-source Python libraries. In the domain of computational music theory, a library called music21 [15] – also in Python – has proved popular for importing, displaying, and calculating features of symbolic music representations. All the while, the domain of music cog-

nition proceeds with experimental investigation and computational modeling of how we perceive, create, and respond to music [17]. The broad premise of this paper is that while exciting progress is being made *within* each of these domains, exciting progress can also be made by improving the bridges that exist *between* them.

A more specific premise of the paper, and the MAIA Util package described below, is that often in sound and interface design, it can be useful to run and even display the results of MIR, music-theoretic, and music-cognitive algorithms. The remainder of the paper is structured around introducing and contextualizing implementations of some of these algorithms.

2. HELLO WORLD AND INPUT/OUTPUT

A MAIA Util “hello world” demo can be explored at <https://tinyurl.com/y3rz5q73>, and shows how the package can be used to analyze and display phenomena arising from an incoming symbolic representation, obtained from automatic transcription of the audio signal [18, 22]. For the sake of demonstrating this bridging of Web Audio, MIR, music theory, and music cognition, we show how MAIA Util can be used to calculate and graph the empirical distribution of estimated MIDI note numbers (MNN), as well as the distribution of estimated events throughout the average measure, quantized at the 16th-note level. Two of the MAIA Util functions used in this demo are `count_rows`, which is useful for calculation of possibly-multidimensional empirical distributions, and `bar_and_beat_number_of_ontime`, which takes an incrementing measure of time in a song (referred to as *ontime*) and an array consisting of the initial time signature and any subsequent changes, and returns the corresponding measure (bar) and beat numbers.

The input symbolic representation for the “hello world” demo is a custom-made, JSON format called a *Composition object*.¹ The exact format of the input is not of particular importance, however, since most of MAIA Util’s methods operate on a point-set representation of the music,

$$E = \{e_1, e_2, \dots, e_n\}, \quad (1)$$

with elements

$$e_i = (x_i, y_i, z_i, w_i, u_i, v_i), \text{ where } i \in \{1, 2, \dots, n\}, \quad (2)$$

and which is relatively straightforward to obtain from Node

¹See <https://crunchy.musicintelligence.co/composition/> for more details.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Package Manager (NPM) MIDI or XML parsers.² In terms of implementation, E is stored as a nested numeric JavaScript array. For a given point e_i as in (2),

- x_i is an *ontime*, which is an incrementing measure of time in a song or piece counted in quarter-note beats from 0 for measure 1 beat 1;
- y_i is an MNN, which is the numeric position of a key on the piano keyboard;
- z_i is a morphetic pitch number (MPN) [26], which is the numeric height of a note on the staff (see Section 5 for more details);
- w_i is a duration measured in quarter-note beats;
- u_i is a channel or staff number;
- v_i is a velocity in the range 0 (silent) to 1 (maximum loudness).

As mentioned in the introduction, the MAIA Util package is intended to bridge existing libraries built on the Web Audio API with both MIR algorithms (such as the audio-to-symbolic demo mentioned above) and music-theoretic concepts. As such, there is no one common or definitive output format. The guiding principle has been to make any array output convenient for subsequent use by JavaScript’s built-in `map`, `filter`, and `reduce` methods, Tone.js’ scheduling methods, NexusUI’s GUI elements, and by the HTML5 canvas in general. The API can be explored at <https://musicintelligence.co/api/maia-util/>. It is documented with JSDoc and tested using Mocha. There is potential for optimization, and the approach at present is more functional than object-oriented.

The next three sections address the most important music-theoretic components of MAIA Util, while emphasizing that the applicability of these methods extends beyond purely music-theoretical concerns.

3. KEY ESTIMATION AND KEYSCAPIES

A considerable amount of research in the music-cognitive literature has been devoted to the perception of key and tonality [14, 23]. Complementing this work are numerous efforts in the music-cognitive and music information retrieval literatures to automatically estimate the key of an excerpt from input audio or symbolic representations. While the use of key signatures in staff notation can be considered a music-theoretic concept, the above-mentioned work underlines the psychological reality or validity of key and tonality, as well as how the perceived key sometimes differs from the notated key. Therefore, being able to calculate, display, or otherwise visualize the key of an excerpt in a web-based music interface can be useful, irrespective of whether that interface involves staff notation.

The key-estimation algorithms of Collins et al. [14] and Krumhansl [23] are based on listening studies involving the perception of tonality. The former operates directly on the audio signal, involves simulation of the inner ear and auditory cortex, and is implemented in Matlab, whereas the latter – known as the Krumhansl-Schmuckler key-finding algorithm – operates on symbolic input and involves correlating

²E.g., <https://www.npmjs.com/package/midiconvert> and <https://www.npmjs.com/package/xml2js>

a pitch-class histogram of incoming music data with experimentally derived, idealized representations of the pitch-class content of each major and minor key. Due to its relative simplicity and speed, we implemented the latter algorithm in MAIA Util as the function `fifth_steps_mode`.

A common observation regarding musical structure – broadly construed – is that it is *hierarchical*. With regards tonality and key, this observation applies, say, to an excerpt that begins and ends in C major, with a modulation to G major partway through. On the highest level, it is accurate to state that the excerpt is in C major, but this overlooks the more fine-grained detail, where a modulation occurs from C to G and back again. Sapp [30] introduces the concept of a *keyscape*, which is a pyramidal representation of the output of a key-estimation algorithm that captures the hierarchical nature of tonality and key. Colored blocks toward the top of the pyramid represent key estimates of larger segments of an input excerpt, while blocks towards the base of the pyramid represent key estimates of smaller, more momentary segments of the excerpt. Key estimates of segments of equal length are represented by blocks in the same row, and, in the same row, a block to the left represents a segment that occurs earlier than one to the right. Keyscape calculation itself is not built into MAIA Util, but an interactive demo of keyscales available from <https://tinyurl.com/y5q8um4q> demonstrates that it is relatively straightforward to calculate and display keyscales based on the use of MAIA Util’s `fifth_steps_mode`.

4. CHORD LABELING

Automatic chord labeling from input audio or symbolic representations is a problem of long and consistent interest in music computing [28, 6, 31, 16]. As a primary use case, the output of an accurate chord labeling system provides guitarists and other musicians a convenient means to begin playing along to and/or learning a song.³ The function of chords in sequences is a topic that has received much attention from music theorists [8] and cognitive scientists [7, 21], but compared to work on key-estimation algorithms, there is less overlap between music-cognitive and MIR literatures when it comes to automatic chord labeling.

With measure-length granularity (or shorter), the lowest parts of a keyscape can be thought of as a chord-labeling system. For instance, if a measure (or less) of music were labeled as being in C major, then from a music-theoretic standpoint this would not be considered sufficient material to properly establish a key – rather it would probably contain some or all the pitch classes of the C-major triad and possibly some non-chord tones, i.e. a C-major chord. That is, the lowest parts of a keyscape provide chord-like labels.

What the keyscape does not provide explicitly, however, is a *segmentation* of the music. For instance, if an excerpt consisted of three-and-a-half measures articulating a C-major triad, followed by a change to a G-major triad for half a measure, then the correct segmentation and labeling would be (0, “C major”), (14, “G major”), where (x, l) denotes a chord label l beginning at ontime x . The information underlying the keyscape, however, would likely be (0, “C major”), (4, “C major”), (8, “C major”), (12, “C major”). That is, the first three-and-a-half measures are not combined into a single label, and the existence of the G-major chord is not

³E.g., <https://chordify.net/>

evident, with content in ontimes 12-14 (C major) and 14-16 (G major) being described by a single label (“C major”).

Pardo and Birmingham [28] suggest that any viable chord analysis system ought to have both segmentation and labeling components. They propose a linear-time algorithm called HarmAn, which explores a subspace of all possible segmentations of an input symbolic representation, according to how well those segments score against predefined chord templates (pitch-class sets, such as $\{0, 4, 7, 10\}$ for “C 7”). Each predefined template has an associated label, and so the template giving rise to the maximum score for the segment under consideration will receive that label. Again giving priority to the algorithm’s relative simplicity and speed, we implemented HarmAn in MAIA Util as the function `harman_forward`. The term “forward” refers to the subspace of all possible segmentations explored by the algorithm.

Partition point and *minimal segment* [28] are two important concepts involved in the early stages of HarmAn that have uses beyond chord labeling. We have implemented them in a corresponding function called `segment`. A partition point is an ontime in the input symbolic representation where a note either begins or ends. A minimal segment is the set of notes that sound between one partition point and the next. (One note may belong to more than one minimal segment.) With respect to HarmAn, the minimal segments are the building blocks of the segmentation. The scoring function favors the appending of two minimal segments if their combined score against the chord templates is greater than or equal to the sum of the scores for the two minimal segments considered in isolation. Returning to the example of three-and-a-half measures articulating a C-major triad, followed by a change to a G-major triad for half a measure, this process of considering combined or isolated minimal segments is how HarmAn would arrive at the correct segmentation and labeling of (0, “C major”), (14, “G major”).

Beyond HarmAn, minimal segments can be used to:

- Provide a measure of monophony – the proportion of minimal segments in an excerpt that contain either one or zero note(s) can be used to measure the extent to which that excerpt is monophonic [9, 20];
- Extract the top (or bottom) line from polyphonic material, known as *skylining*;
- Group notes so as to retrieve all instances of a specified harmonic interval in an excerpt [9]. This is discussed further in the next section;
- Group notes so as to calculate empirical probability distributions for one or more excerpts, for analytic [12] or generative [13, 11] purposes.

5. MELODIC/HARMONIC INTERVALS

Identifying the interval between two consecutive notes (melodic interval) or between two notes that sound together (harmonic interval) is a fundamental topic in music theory. Quantitative and qualitative investigations of those intervals have been the source of much discourse. Representing a melody in relative (as intervals) rather than absolute (as note names or MIDI numbers) terms is also psychologically relevant and computationally advantageous: regarding psychological relevance, most listeners have relative rather than absolute means of pitch perception and production, i.e. we

can recognize or attempt to sing “Happy birthday” irrespective of starting pitch; regarding computational advantage, most audio- and symbolic-based music recognition systems use differences between pitches (intervals) rather than the pitches themselves as a means of querying a database [5, 32].

For users with music-theoretic knowledge, a frustrating aspect of existing music-intervallic calculators is that the interval between the note pair (C4, F#4) might be mislabeled as a diminished fifth, or vice versa that between the note pair (C4, Gb4) might be mislabeled as an augmented fourth, when the opposite is true. Both of these intervals are six MNNs, and if MNNs alone are used as the basis for the difference calculation, then both pairs are represented as (60, 66), so it is impossible to distinguish an F#4 from a Gb4, and impossible to distinguish an augmented fourth from a diminished fifth.

There is a useful, complementary numeric representation of pitch called morphetic pitch number (MPN) [26] that helps to resolve this issue. Figure 1 contains some examples of (MNN, MPN)-pairs in the neighborhood of “middle C” $\equiv C4 \equiv (60, 60)$. Whereas MNN is the numeric position of a key on the piano keyboard, MPN is the numeric height of a note on the staff. So while F#4 and Gb4 map to the same key on the keyboard, they are notated on different lines of the staff, and so taking their MNNs and MPNs together, they are distinguishable: F#4 maps to (MNN, MPN)-pair (66, 63), while Gb4 maps to (MNN, MPN)-pair (66, 64).⁴

Similarly, we can associate the difference between two (MNN, MPN)-pairs with correctly named musical intervals, meaning that entities such as “augmented fourth” and “diminished fifth” are distinguishable from one another. For example, two pitches p_1 and p_2 , where p_1 is assumed lower in pitch than p_2 , form the interval of a diminished fifth if and only if the difference between their corresponding (MNN, MPN)-pairs is (6, 4); they form the interval of an augmented fourth if and only if the difference between their corresponding (MNN, MPN)-pairs is (6, 3), and so on. The MAIA Util functions for converting between pitch and octave number, and (MNN, MPN)-space are called `pitch_and_octave2midi_note_morphetic_pair` and `midi_note_morphetic_pair2pitch_and_octave`.

Identifying all instances of a specified harmonic interval is slightly more difficult than the corresponding task for a melody, because (a) two notes forming the relevant interval may not begin at the same time, even though this is how we typically think of rudimentary harmonic intervals, and (b) there may be notes that occur in between them in a chord, meaning that we cannot rely on comparisons between adjacent notes when the notes are placed in ascending order. Problem (a) can be solved with use of `segment` (see previous section), because if two notes do form the specified harmonic interval, they will occur together in at least one minimal segment. Problem (b) can be solved as follows:

1. Let us notate the point-set representation of notes belonging to this minimal segment as $E = \{e_1, e_2, \dots, e_n\}$, and then let us project this set down to a space where only the dimensions of MNN and MPN are retained, notating this projected set as $D = \{d_1, d_2, \dots, d_m\}$, where m may be less than n . So

⁴In mathematical terms, we say there is a bijection between pitch and octave number, and (MNN, MPN)-space.

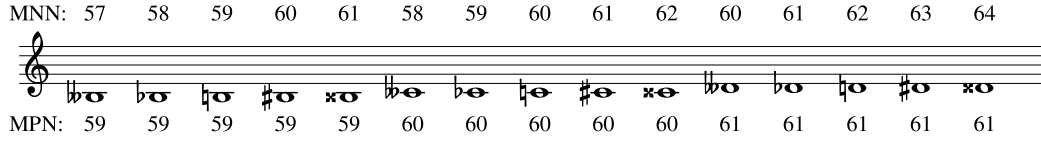


Figure 1: Some MIDI note numbers (MNN) and morphetic pitch numbers (MPN) close to C4 (“middle C”).

$\mathbf{d}_i = (y_i, z_i)$ for some MNN y_i and MPN z_i ;

- Let $\mathbf{v} = (v_y, v_z)$ be the (MNN, MPN)-difference corresponding to the specified harmonic interval. Then the specified harmonic interval occurs in this minimal segment if and only if there exist some $i, j \in \{1, 2, \dots, m\}$ satisfying $\mathbf{d}_i + \mathbf{v} = \mathbf{d}_j$. In other words, \mathbf{d}_i translates to \mathbf{d}_j by the vector \mathbf{v} .
- Meredith [27] defines the *maximal translatable pattern* (MTP) of the vector \mathbf{v} in the point-set D as the set of all points in D that are translatable to another point in D by the vector \mathbf{v} , and MAIA Util contains an implementation of this concept as the function `maximal_translatable_pattern`. So we can solve problem (b) by calculating E, D as described above and then seeing if the array returned by `maximal_translatable_pattern(v, D)` is populated (specified interval occurs) or empty (specified interval does not occur).

Algorithms for calculating MTPs in an unsupervised manner form the basis of geometric pattern discovery in music, enabling the discovery of repeated elements such as riffs, motifs, themes, and sections [27, 10]. A discussion of these algorithms is beyond the scope of this paper, but their representational and functional bases are present in MAIA Util.

6. UTILITY FUNCTIONS

This section addresses a couple of methods that are very widely used in our interfaces, but not as complex or closely connected to music cognition as those described above.

6.1 Pitch estimation

During the development of several web-based music interfaces, we have found it necessary to estimate the pitch of given MNNs in a surrounding musical context. Such a need tends to arise because MIDI is a ubiquitous input format, but lacks accurate pitch and octave information – e.g., is MNN 66 an F#4 or Gb4? – and we strive for music-theoretic correctness. Just as it is frustrating when an interval is mislabeled (see the discussion of diminished fifths and augmented fourths in the previous section), so it annoys musically trained users when pitch sequences such as (C, D, E, Gb, G) appear in an interface, which probably ought to be (C, D, E, F#, G). These misspelt sequences tend to result from oversimple MNN-to-pitch estimation algorithms. From an educational perspective, mislabeling of pitches and intervals is also detrimental to novice users who are learning, implicitly or explicitly, about music theory via use of an interface.

A recent discussion about pitch estimation from MNN on the music21 Google Group [4] underlines that music21 does not have built-in functionality for addressing this problem.

The solution suggested on the forum involves a base-40 numeric representation [19], which has less psychological validity compared to the solution involving MNN-MPN space, where MNNs model our perception of chromatic pitch and MPNs model our perception of diatonic pitch.

MAIA Util contains a function called `guess_morphetic`, which works by providing an MNN and the key of an excerpt (see discussion of `fifth_steps_mode` above), and then mapping the provided MNN to the (MNN, MPN)-pair that is most likely to occur in the key. For instance, F#’s are generally more numerous in pieces in C major than are Gb’s, so if the provided MNN and key are 66 and C major, respectively, then 66 will map to the (MNN, MPN)-pair (66, 63) for F# and MPN 63 will be returned, rather than mapping to the (MNN, MPN)-pair (66, 64) for Gb and returning MPN 64. Due to the bijection between pitch and octave, and (MNN, MPN)-space (see Section 3), once we have the (MNN, MPN)-pair, we can map unambiguously to a pitch and octave using `midi_note_morphetic_pair2pitch_and_octave`. That is, (MNN, MPN)-pair (66, 63) maps to F#4. This is how MAIA Util can be used to convert an input MNN sequence (60, 62, 64, 66, 67) to the correctly spelled pitch and octave sequence (C4, D4, E4, F#4, G4).

Pitch estimation algorithms in the literature tend to be more complex than that based on MAIA Util’s `guess_morphetic` (e.g., [26]). Informally, we have not observed a big tradeoff in accuracy, however, and so favor a simple and speedy approach at present.

6.2 Projection, sorting, and deduplication of nested numeric arrays

Since point-set representations featured prominently in Section 2 and have been present in other sections too, it seems appropriate to conclude the tour of MAIA Util with some of the point set-specific functionality. There is a function `orthogonal_projection_not_unique_equalp` that can retain/remove a specifiable selection of dimensions from an input point set, which is a form of *projection* operation in mathematics. The sorting of a point set can be achieved with `sort_rows`, which comprises calling the built-in `sort` method with a helper function `lex_more` that returns +1 if the first input is *lexicographically more* than the second input, and −1 otherwise. The `sort_rows` function returns both the sorted array as well as the indices of the elements from the input array. Another function, `unique_rows`, returns a sorted, deduplicated version of the input array, as well as the indices of the elements from the input.

7. DISCUSSION

Considerable progress has been made in recent years concerning complex topics and problems within the domains of web audio, MIR, music theory, and music cognition. In some cases, this progress has manifested in usable, if circumscribed, algorithmic solutions. The current paper repre-

sents a modest attempt to bring together some of these algorithms and associated insights. It describes an NPM package called MAIA Util that contains implementations of methods that allow the algorithms to interface with each other. One outcome consists of interactive demos that bridge the above-mentioned domains in potentially novel and interesting ways.

Through introducing some of MAIA Util’s methods and use cases for key estimation, chord labeling, and interval identification, we have suggested that it can be useful to calculate and sometimes display manifestations of music-theoretic concepts, even if the interface that displays these manifestations is not intended for exploring music theory or staff notation per se. We have also emphasized how it can be frustrating and potentially detrimental to users when music interfaces show “unmusical” pitch spellings or intervals, and we have demonstrated how our use of MNN-MPN space helps to address some of these issues. Where our key estimation, chord labeling, and interval identification algorithms draw on findings from the literature on music cognition, this has been pointed out because it is reasonable to assume that use of algorithms for which there is some underlying psychology validity will lead to calculations and/or visualizations that listeners find to be more convincing or realistic.

Up until now, our criterion for inclusion of a method in MAIA Util has been that the method is required in multiple, in-development music interfaces. Future work is likely to consist of including more algorithms for quantization, score following, pattern discovery, and music generation. Whether these algorithms warrant their own packages – either because they are too complex or because they are not widely required – or are added to MAIA Util remains to be seen. Either way, there is something exciting about the potential for wide and rapid dissemination afforded by the web, combined with contributions from several different intersecting music research domains that are coalescing in web audio space.

8. ACKNOWLEDGMENTS

We are grateful to Jeremy Yu, with whom the first author worked on a framework for running multiple MIR algorithms and integrating their results into Composition objects. This work formed the basis of the “Hello world” demo.

9. REFERENCES

- [1] P. Adenot, R. Toy, C. Wilson, and C. Rogers. Web Audio API. <https://www.w3.org/TR/webaudio/>. Retrieved November 10, 2017.
- [2] J. Allison, W. Conlin, D. Holmes, Y. Oh, and B. Taylor. NexusUI GitHub repository. <https://github.com/nexus-js/ui/>. Retrieved August 5, 2015.
- [3] J. Allison, W. Conlin, D. Holmes, Y. Oh, and B. Taylor. Simplified expressive mobile development with NexusUI, NexusUp and NexusDrop. In *Proceedings of the New Interfaces for Musical Expression Conference*, 2014.
- [4] T. Anders and C. S. Sapp. Converting MIDI pitch or pitch class numbers into enharmonically most likely pitches depending on a key. <https://tinyurl.com/y232n8jw>. Retrieved June 5, 2019.
- [5] A. Arzt, S. Böck, and G. Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 433–438, 2012.
- [6] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 304–311, 2005.
- [7] E. Bigand, B. Poulin, B. Tillmann, F. Madurell, and D. A. D’Adamo. Sensory versus cognitive components in harmonic priming. *Journal of Experimental Psychology: Human Perception and Performance*, 29(1):159–171, 2003.
- [8] W. E. Caplin. *Analyzing classical form: an approach for the classroom*. Oxford University Press, New York, NY, 2013.
- [9] T. Collins. Stravinsky/De Montfort University at the MediaEval 2014 C@merata task. In *Proceedings of the MediaEval Workshop*, 2014.
- [10] T. Collins, A. Arzt, H. Frostel, and G. Widmer. Using geometric symbolic fingerprinting to discover distinctive patterns in polyphonic music corpora. In *Computational Music Analysis*, pages 445–474. Springer, 2016.
- [11] T. Collins and R. Laney. Computer-generated stylistic compositions with long-term repetitive and phrasal structure. *Journal of Creative Music Systems*, 1(2), 2017.
- [12] T. Collins, R. Laney, A. Willis, and P. H. Garthwaite. Modeling pattern importance in Chopin’s mazurkas. *Music Perception*, 28(4):387–414, 2011.
- [13] T. Collins, R. Laney, A. Willis, and P. H. Garthwaite. Developing and evaluating computational models of musical style. *AI EDAM*, 30(1):16–43, 2016.
- [14] T. Collins, B. Tillmann, F. S. Barrett, C. Delbé, and P. Janata. A combined model of sensory and cognitive representations underlying tonal expectations in music: From audio signals to behavior. *Psychological Review*, 121(1):33–65, 2014.
- [15] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 637–642, 2010.
- [16] W. B. De Haas, J. P. Magalhães, F. Wiering, and R. C. Velkamp. Automatic functional harmonic analysis. *Computer Music Journal*, 37(4):37–53, 2013.
- [17] D. Deutsch. *The psychology of music*. Academic Press, San Diego, CA, 3rd edition, 2013.
- [18] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.
- [19] W. B. Hewlett. A base-40 number-line representation of musical pitch notation. *Musikometrika*, 4:1–14, 1992.
- [20] B. Janssen, T. Collins, and I. Ren. Algorithmic ability to predict the musical future: Datasets and evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference*. In press.

- [21] S. Koelsch. Music-syntactic processing and auditory memory: Similarities and differences between eran and mmn. *Psychophysiology*, 46(1):179–190, 2009.
- [22] F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 227–232, 2013.
- [23] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Oxford University Press, New York, NY, 1990.
- [24] Y. Mann. Tone.js GitHub repository. <https://github.com/Tonejs/Tone.js>. Retrieved August 5, 2015.
- [25] Y. Mann. Interactive music with Tone.js. In *Proceedings of the Web Audio Conference*, January 2015.
- [26] D. Meredith. The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35(2):121–159, 2006.
- [27] D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.
- [28] B. Pardo and W. P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.
- [29] C. Rogers. Web Audio API. <https://www.w3.org/TR/2012/WD-webaudio-20121213/>. Retrieved November 10, 2017.
- [30] C. S. Sapp. Visual hierarchical key analysis. *ACM Computers in Entertainment*, 3(4):1–19, 2005.
- [31] D. Tymoczko. Review of Michael Cuthbert, music21: A toolkit for computer-aided musicology (<http://web.mit.edu/music21/>). *Music Theory Online*, 19(3), 2013.
- [32] A.-C. Wang and J. Smith. System and methods for recognizing sound and music signals in high noise and distortion, 2012. Patent US 8,190,435 B2. Continuation of provisional application from 2000.

Combining Collaborative and Content Filtering in a Recommendation System for a Web-based DAW

Jason Smith
Georgia Tech Center for Music
Technology
840 McMillan Street
Atlanta, Georgia USA
jsmith775@gatech.edu

Mikhail Jacob
Georgia Tech Center for
Interactive Computing
85 5th Street NW
Atlanta, Georgia USA
mikhail.jacob@gatech.edu

Jason Freeman
Georgia Tech Center for Music
Technology
840 McMillan Street
Atlanta, Georgia USA
jason.freeman@gatech.edu

Brian Magerko
Georgia Tech School of
Literature, Media, and
Communication
686 Cherry Street
Atlanta, Georgia USA
magerko@gatech.edu

Tom Mcklin
The Findings Group
2646 Woodridge Drive
Decatur, Georgia USA
tom@thefindingsgroup.org

ABSTRACT

EarSketch is a web-based audio production and education platform that uses an online coding environment and the Web Audio API to teach introductory programming and music production to students. One of the main challenges of implementing an educational online music production platform is providing users with a variety of foundational audio loops to use in order to foster creative personal expression. EarSketch aims to achieve this through the inclusion of a sound browser for users to navigate and select sounds to use as part of their compositions.

This paper describes the implementation and evaluation of a hybrid recommendation engine, combining collaborative and content filtering, designed to guide users through the sound browser and promote diversity in student compositions. The paper also presents a preliminary analysis of the impact of different recommendation strategies on user sound selection, and how the application of recommendation strategies can inform the design of EarSketch and other web-based DAWs.

CCS Concepts

•Applied computing → Sound and music computing;

Keywords

music, recommendation systems

1. INTRODUCTION

EarSketch [10] is an online platform that integrates Python and JavaScript coding environments with a Web Audio API-based digital audio workstation (DAW). It helps to teach students coding and music production through the manipulation of audio loops from a large sound library [11].

EarSketch engages diverse student populations in computing through a curriculum and learning environment that is authentic in both the computing domain (i.e. industry-standard programming languages) and the music domain (i.e. interface and API designs resembling digital audio workstations and the inclusion of audio loops created by professional artists) [3]. Previous research has found student perceptions of *authenticity* to be correlated with attitudes towards computing and intention to persist in the field, which EarSketch is designed to reinforce through its focus on pervasive music production paradigms, programming languages, and musical styles [13].

A critical component of this authentic learning environment is the ability of students to find personally relevant, expressive loops that fit musically with the compositions that they are creating through code. EarSketch includes a sound library of nearly 4000 audio loops across a variety of popular music genres, created for the platform by sound designer Richard Devine and hip-hop engineer and DJ Young Guru. These sounds form the building blocks of student compositions, and are designed to promote a wide range of creative expression in styles that are personally meaningful to students.

An analysis of 20,000 non-tutorial user scripts, both from experienced and novice users, collected in spring 2018 [16] reveals a significant under-utilization of a majority of the sound library. As seen in Figure 1, fewer than 200 sounds were used in over 1% of scripts. Under 20 sounds were used in over 10% of scripts. We hypothesize that this relative lack of usage of most sounds in the library stems from the difficulties users face in searching for and finding sounds in the browser interface, which has been limited to simple filters and text search. In fact, over half of the sounds most

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

often used in scripts closely align with those found in sample tutorial scripts.

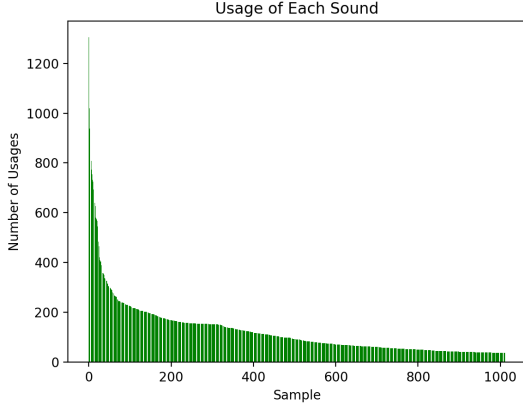


Figure 1: The 1,000 most commonly used EarSketch sounds in 20,000 user scripts [16].

We explored the implementation of a hybrid recommendation system [16] in order to a) address the cause of this usage problem, b) improve the diversity and coverage in the representation of the sound library by user scripts, and c) to better facilitate student creativity, potentially leading to higher perceptions of authenticity of EarSketch as a creative environment.

This project emphasizes the inclusion of an intelligent recommendation system in a web-based music production software. The guiding principle behind this that when a program gathers statistical data from its user base’s creative decisions, it can generate its own decisions that are representative of its users. With a large sound library and user base, this allows for intelligent recommendations that approximate the collective design choices of users. Such recommendations may be of particular importance in web-based music production systems, which are typically targeted to novice users who may desire more guidance in locating and selecting sounds from a massive library of choices.

Common approaches to evaluating hybridized recommendation systems focus on the domain of listening preferences for songs. These examples use previous listening history and ratings [19] or social media [18] to collect user preferences. The domain of full songs differentiates these methods from our approach, which gathers preferences for short loops used in combination, and for composition instead of listening.

An example of ongoing research into feature analysis of shorter audio loop is Groove Explorer [2]. This drum loop visualization tool measures similarity between rhythmic vectors and evaluates performance using genre labels. Our recommendation system does not currently use these labels, but the inclusion of textual metadata may be used to improve future performance (see future work).

In this article, we present our research on a recommendation system for discovering new sounds for use in EarSketch, informed by a user-centered design study and proposed recommendation engine described in [16]. The main contributions discussed are:

- The implementation of a hybrid recommendation system using collaborative filtering of previous user

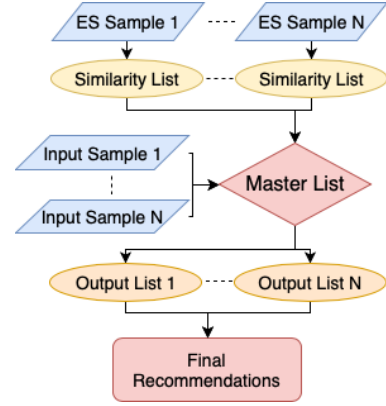


Figure 2: Program flow of recommendation generation

scripts and analysis of audio similarity between sounds in the sound library.

- The redesign of the EarSketch sound browser to accommodate recommendations, and to improve user experience in navigating options.
- An evaluation methodology and preliminary results from two studies designed to assess the relative performance of four variations of the recommendation algorithm.

2. IMPLEMENTATION OF THE RECOMMENDATION SYSTEM

The base model of the hybrid recommendation system (Figure 2) is comprised of six steps.

1. Each sound in EarSketch is used as an input sound during a separate iteration of the recommendation process.
2. The system generates a *co-usage list* (representing the most commonly used sounds with the input) for each EarSketch sound, with usage statistics generated from an analysis of a collection of 20,000 user scripts.
3. The system generates a *similarity list* for each input sound, using content-based filtering of audio features as well as the co-usage data to compare it to every other sound in EarSketch and generate a recommendation score.
4. The system then combines each *similarity list* into a *master list* and uploads it to EarSketch.
5. When a user script is active in EarSketch, all of its component sounds are indexed in the *master list*.
6. The system combines the *similarity lists* for each input sound in order to generate weighted recommendation scores, and displays the sounds with the highest scores to the user.

This model is used, as explained in [16], to provide recommendations based on acoustic similarity as well as the habits of users without collecting personally identifiable data in conformance with EarSketch’s privacy policy.

2.1 Collaborative Filtering

The *co-usage list* is generated from a collection of previous user scripts [16]. Every sound in each script containing the input has its score increased, so the sounds that most commonly appear with the input have the highest score.

This system is intended to regularly update with new user scripts, as improved sound representation in user scripts due to the addition of the recommendation system will increase the diversity of coverage scores. The system will increase in computational time as more scripts are added, but this will not be an issue due to it being an offline process. Scripts will be removed from the system over time, with a maximum set of scripts to be defined by future comparative studies.

2.2 Content-based Filtering

The system employs precomputed feature distances to quickly calculate recommendations for each sound, using each sound as input to create the *co-usage list* and *similarity list*, which are used to form the *master list*. The distance between every combination of EarSketch sounds is recorded for two common audio features, STFT and MFCC [7]. The features are calculated as *fingerprints* representing short sections of the samples, generated from an extended version of Kyle McDonald’s AudioNotebooks code [12].

2.2.1 STFT

Comparison of Short-time Fourier Transform vectors, representing spectral features of each sound on a frame-by-frame basis, assesses temporally-based similarity. [8]. This time-series data is used in dimensionality reduction and clustering techniques to group and visualize sounds [5].

In Figure 3, STFT fingerprints are represented in a two-dimensional space using t-SNE visualization techniques [9] as inspired by the Infinite Drum Machine [17], with colors representing genre labels.

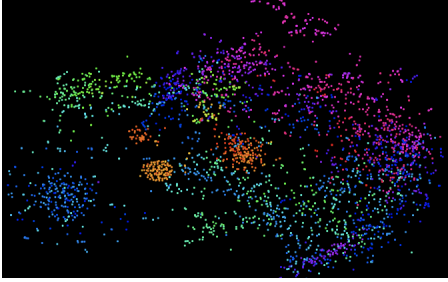


Figure 3: Short-time Fourier Transform used to cluster and represent EarSketch sounds in 2D space.

2.2.2 MFCC

Mel-frequency Cepstrum Coefficients represent sounds in the domain of frequency distribution. They are created by decorrelating the spectral information between frames and can be used in temporally-independent timbral speech analysis and genre recognition [8].

2.3 Limiting Factor

The recommendation system includes a limiting factor to save computational resources, both when generating the *master list* and when generating recommendations in real time. The *similarity list* for each sound stores only the 50 highest recommendation scores, and uses only the highest 10 values of the *co-usage list* to generate the scores.

2.4 Combined Recommendation Scores

The recommendation scores for each (*input*) and *output* sound pair, split into component scores labeled R_A , R_B and R_C which are added to form the final score R , are generated as follows:

$$R_A = D_{co-usage}(hc, input) + D_{STFT}(output, hc) + D_{MFCC}(output, hc) \quad (1)$$

The base model calculates the feature distances between samples with the highest co-usage scores to generate R_A , which can be interpreted as similarity to a sound hc that is highly co-used with the input.

$$R_B = D_{co-usage}(output, input) \quad (2)$$

R_B is co-usage between the recommended sound and the input found in the collection of stored user scripts.

$$R_C = D_{STFT}(output, input) + D_{MFCC}(sound, input) \quad (3)$$

R_C is the feature similarity between the recommended sound and the input, calculated using precomputed MFCC and STFT distances.

$$[R_A, R_B, R_C] = [R_{A1} + \dots + R_{AN}, R_{B1} + \dots + R_{BN}, R_{C1} + \dots + R_{CN}] / \sqrt{N} \quad (4)$$

With multiple sounds in a user script, the system accommodates for multiple inputs. It adds the component scores for each input and divides by the square root of the number of inputs (N) when generating recommendations. This is in order to balance - the scores recommended highly by multiple input sounds receive a score increase which cannot be achieved by using a mean of multiple scores, but without the hard increase of pure summation that would negate single recommendations.

$$R = R_A + C_U * R_B + S * R_C \quad (5)$$

The model adds the three component scores together to generate the final recommendation score R . Tuneable parameters C_U and S can be set to -1 or 1 to maximize or minimize co-usage and similarity respectively. Minimization is chosen over neglect to allow for more parameter combinations that make use of the negative, such as intentionally choosing the lowest co-usage and the highest similarity or vice versa. These combinations generate the following labeled recommendation categories: *highest co-usage*, *highest similarity*: “Others Like You Use These Sounds”, *lowest co-usage*, *highest similarity*: “Sounds That Fit Your Script”, *highest co-usage*, *lowest similarity*: “Discover Different Kinds of Sounds”, and *lowest co-usage*, *lowest similarity*: “Are You Feeling Lucky?”.

2.5 Sound Browser Redesign

The EarSketch sound browser (Figure 4) has been redesigned to display recommendations as they are generated.

The addition of openable and closeable sound folders allows users to see a larger amount of sound types in immediate succession and to explore the options available to them through navigating the list. Recommendation folders appear at the beginning of the list when they are generated, with the type of recommendation indicated by a highlighted label.

The sound browser updates with new recommendations when a change is detected in the user script, such as when a user types in a new sound name, pastes from the sound browser, saves a script, or switches tabs. The sound names found in the active script are entered as the input, and if none are detected, a user’s previous scripts are used to avoid the ‘cold-start’ problem [18] faced by usage-based systems.

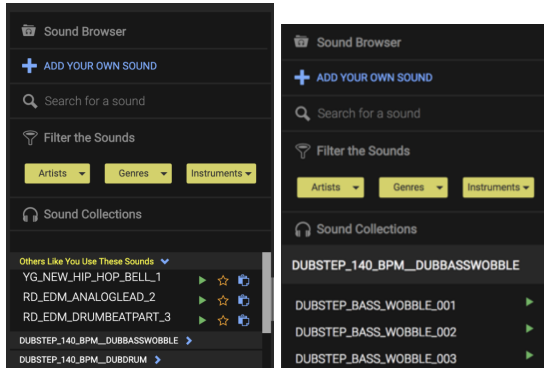


Figure 4: Sound Browser redesign (left) featuring open and closed folders and recommendation labels, with original Sound Browser design (right).

This design arose from a user study [16] that discovered that users wanted to find sounds based on categorical recommendations, highlighting the similarity or difference to their current work. Users also expected a degree of novelty or serendipity [1], which are emphasized by the recommendation labels.

3. EVALUATION

The recommendation engine and updated user interface were implemented and deployed to EarSketch users in May 2019. Our preliminary evaluation of the system seeks to understand the utility of the sound recommendations across the four categories (‘Others Like You Use These Sounds’, ‘Sounds That Fit Your Script’, ‘Discover Different Kinds of Sounds’, and ‘Are You Feeling Lucky?’) that differentially weight co-usage and similarity. The following research questions motivated our evaluation design: 1) Which recommendation category do a general pool of adult subjects prefer for completing a given musical fragment? 2) Which recommendation category do EarSketch users prefer while composing music in a web-based DAW?

This section presents two initial studies addressing these questions, with the intent that the findings will guide iterative design of the EarSketch recommendation system and potentially inform the design of sound recommendation systems in other web-based DAWs as well.

3.1 General Subject Pool Study: Ranking Recommendation Categories

An initial study was conducted based on the research question: which recommendation category do a general pool of adult subjects prefer for completing a given musical fragment? The study was conducted outside the context of a web-based DAW in order to understand relative user preferences between the four recommendation categories among a large number of subjects from a general population. Rather than actual EarSketch users, study participants consisted of adults from a general subject pool with a range of experiences in various musical activities (see Table 1). Subject experience levels with musical activities were divided into categories of never, rare (once in their lifetime or a few times a year), and regular (a few times a month, week, or day). Participants ($N = 919$ subjects in total) were recruited using the Amazon Mechanical Turk crowd-worker platform¹ and were offered \$0.25 to participate in the study.

Table 1: General Subject Musical Experience

Activity	Never	Rare	Regular
Listening To Music	2	8	908
Playing Music	436	266	215
Writing Music	634	170	113
Amateur/Professional DJing	768	94	57
Read Musical Notation	593	182	141
Music Production Software	650	175	91

The study was designed with a within subjects repeated measures configuration. Each participant was asked to first listen to an audio track of a partial composition representing an in-progress musical composition on the EarSketch DAW. They were then asked to listen to four options for recommended audio loops, each obtained from a different type of recommendation. Recommendations were generated for each example before the test was published, and each subject was given the same four recommendations. They were asked to rank the options from one (best fit) to four (worst fit) in terms of how well they thought the suggested audio loop fit with the partial composition. Ties in ranks and missing ranks were grounds for rejecting participant responses as they were explicitly not permitted in the task.

The rankings obtained from subjects was considered to be ordinal data with one independent variable and four levels (1 to 4). The hypothesis (H_1) for the study stated that significant differences were present in the rankings between the four recommendation types. The null hypothesis (H_0) stated that there were no significant differences between the rankings for each recommendation option. The options were labeled as A - “Sounds That Fit Your Script”, B - “Others Like You Use These Sounds”, C - “Discover Different Kinds of Sounds”, and D - “Are You Feeling Lucky?” for convenience of analysis.

The individual rank distributions for each option were first run through a Shapiro-Wilk test for normality [15] to see if they could be analyzed using parametric tests. All four distributions of responses were classified as non-normal. Therefore, the Friedman’s rank sum (omnibus) test [4] was used to test whether there were significant differences between

¹<https://www.mturk.com>

the ordinal ratings data for individual types of recommendations. The results of the Friedman’s test allowed us to reject H_0 at a significance level $\alpha < 0.05$, showing significant differences between the distributions with $p = 2.7 \times 10^{-20}$.

Table 2: Pairwise Nemenyi Post-hoc Tests

	A	B	C
B	9.70×10^{-12}	-	-
C	2.61×10^{-14}	2.00×10^{-1}	-
D	1.06×10^{-3}	4.91×10^{-3}	7.16×10^{-7}

Pairwise Nemenyi post-hoc tests [14] were conducted on the data after finding significance with the Friedman’s omnibus test. The results are shown in table 2 with significant differences highlighted in bold, showing that all pairs except for types B vs. C are statistically different. Median rankings, mean rankings, and standard deviations for all four distributions can be seen in table 3 with lower values being better and the best rankings highlighted in bold. The relative rankings and significant difference results placed the recommendation types ordered as $B, C > D > A$. However, a computation of effect size using the Kendall’s W test of concordance [6] resulted in values of $W = 0.03$. This showed that our study was overpowered and the statistical significance observed was likely due to the large sample size.

Table 3: Recommendation Category Rankings

Recommendation Category	Median	Mean	Standard Deviation
A	3	2.80	1.22
B	2	2.37	1.11
C	2	2.25	1.04
D	3	2.57	1.02

Our results from studying a general subject pool of adult users with varying experience in musical activities did not reveal a clear preference between categories of recommendations. However, the study was conducted on a population unlike the target audience for EarSketch. Additionally, the context for ranking the different recommendation categories was far removed from the context of music composition in a web-based DAW. Therefore, a second study was conducted using aggregated data from actual EarSketch users composing music within EarSketch, in order to understand if significant differences would be observed in the relative usage of the different recommendation categories.

3.2 EarSketch Users Study: Comparing Relative Usage of Recommendation Categories

Over a period of 14 days in June 2019, we collected aggregate data from 21,368 EarSketch users. Users were randomly assigned to a recommendation type such that they only saw one category of recommendations in the sound browser (not all four). Users received single random category assignments, so they would see the same category on all sessions during the trial period. For each of the four recommendation categories, a web-based analytics engine collected aggregate data on a) the total number of recommendations displayed to users, b) the number of recommendations previewed by users (suggesting that they saw the recommendation and were intrigued by the sound’s name), and c) the

number of recommendations pasted by users into their code (suggesting that they found the recommendation to be a good fit for their project and proceeded to experiment with it, potentially using it in their final code).

Calculation of the number of recommended sounds pasted into user code relative to the number of recommendations previewed functioned as a simple comparison of the four recommendation types. A greater percentage of previewed sounds that are pasted into code may suggest that users find the recommendations better suited for their projects.

Due to the aggregate nature of this data collection, this system is limited in its inability to recognize sounds that have been pasted without being previewed. The general assumption made is that a user is first previewing a sound before pasting it into their script, but this assumption cannot be validated without collecting personal account data.

Table 4: EarSketch User Study Results

Recommendation Category	Pastes	Previews	Percentage
1:“Others Like You Use These Sounds”	173	1043	16.587%
2:“Sounds That Fit Your Script”	225	1448	15.539%
3:“Discover Different Kinds of Sounds”	214	1033	20.716%
4:“Are You Feeling Lucky?”	96	767	12.516%

Category 3 is found to be significantly higher than 2 and 4, using a general proportion test with a pairwise post-hoc test [14]. However, 3 is not significantly higher than 1, and 1 and 2 are not significantly higher than 4. The two significantly less used categories were the two with minimized co-usage scores, suggesting that showing the user intentionally rare combinations of sounds does not lead to higher usage of these sounds. EarSketch users showed a significant increase in usage for sounds in the ‘Discover Different Kinds of Sounds’ category above all other categories, unlike the general subject pool.

The difference in population of EarSketch’s student body and adult subjects, as well as the difference in context between general EarSketch usage and selecting sounds for the completion of an example script, can explain the statistical variation between tests. The general subjects’ task may have led them towards sounds that were more acoustically similar to the ones in the example, while EarSketch users free to create were more inclined to try sounds presented as intentionally different to what they were using.

The labeling and presentation of the categories is a possible cause of this difference, in that the ‘Discover Different Kinds of Sounds’ category appeals most to the desire for novel recommendations from users. Users in past interviews suggested that recommendations of sounds similar to previously used sounds was of lesser importance to them [16]. The statistical variation between subject groups reflects the importance of evaluating recommendations in context. The different tasks created different reasons for users to consult sound recommendations, and other web-based systems should tailor their evaluations to user behavior.

4. FUTURE WORK

As the studies in this section compared usage of the four categories in completing scripts without user input on recommendation quality, future explicit comparison between styles is necessary. A possible format for this evaluation is having participants rank sample recommendations for active scripts in a blind procedure in terms of relevance, novelty, diversity, and serendipity, in order to determine the strengths and weaknesses of each category beyond usage statistics.

Once enough time has passed for data collection, coverage analysis using a new set of post-recommendation user scripts will be conducted. The changes in the usage of each sound compared to Figure 1 will be used to evaluate the long-term effectiveness of the recommendation engine. Additionally, paste-to-preview percentages (as found in Table 4) for non-recommended sounds can provide a baseline for comparison.

Other changes to the recommendation system can be made to better adhere to user interests and evaluate performance. Metadata can be used to improve recommendation relevance or provide more contextual recommendations. Finally, the recommendation engine will be re-seeded with a larger collection of user scripts. The effects that re-seeding using scripts built with recommendations has on usage statistics will be evaluated as the system improves.

5. CONCLUSIONS

As our sound recommendation system has been integrated with EarSketch, a web-based music production system, we can evaluate performance in terms of user trends and studies designed to compare categorical recommendations. Performance goals defined by user study and performance analysis have concrete influences on design choices, such as the integration of recommendation systems into EarSketch and other web-based DAWs.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation Award No. 1814083. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Additional thanks to Nashawn Cherry for assistance in creating precomputed feature distance vectors for the EarSketch sound library. EarSketch is available online at <https://earsketch.gatech.edu>.

7. REFERENCES

- [1] C. C. Aggarwal et al. *Recommender systems*. Springer International Publishing, 2016.
- [2] F. Bruford, M. Barthet, S. McDonald, and M. Sandler. Groove explorer: An intelligent visual interface for drum loop library navigation. In *Joint Proceedings of the ACM IUI 2019 Workshops*, 2019.
- [3] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, and E. Fruchter. Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 85–90. ACM, 2014.
- [4] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.
- [5] S. Haskey, B. Blackwell, and D. Pretty. Clustering of periodic multichannel timeseries data with application to plasma fluctuations. *Computer Physics Communications*, 185(6):1669–1680, 2014.
- [6] M. G. Kendall and B. B. Smith. The problem of m rankings. *Annals of mathematical statistics*, 1939.
- [7] A. Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Wiley-IEEE Press, 1st edition, 2012.
- [8] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–289. ACM, 2003.
- [9] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [10] B. Magerko, J. Freeman, T. McKlin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A steam-based approach for underrepresented populations in high school computer science education. *ACM Transactions on Computing Education (TOCE)*, 16(4):14, 2016.
- [11] A. Mahadevan, J. Freeman, B. Magerko, and J. C. Martinez. Earsketch: Teaching computational music remixing in an online web audio based learning environment. In *Web Audio Conference*, 2015.
- [12] K. McDonald. Audio notebooks. <https://github.com/kylemcdonald/AudioNotebooks>, 2016.
- [13] T. McKlin, B. Magerko, T. Lee, D. Wanzer, D. Edwards, and J. Freeman. Authenticity and personal creativity: How earsketch affects student persistence. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 987–992. ACM, 2018.
- [14] P. Nemenyi. Distribution-free multiple comparisons (doctoral dissertation, princeton university, 1963). *Dissertation Abstracts International*, 25(2):1233, 1963.
- [15] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.
- [16] J. Smith, M. J. Dillon Weeks, J. Freeman, and B. Magerko. Towards a hybrid recommendation system for a sound library. In *Joint Proceedings of the ACM IUI 2019 Workshops*, 2019.
- [17] M. Tan and K. McDonald. Infinite drum machine. <https://experiments.withgoogle.com/ai/drum-machine/>, 2017.
- [18] A. Vall and G. Widmer. Machine learning approaches to hybrid music recommender systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 639–642. Springer, 2018.
- [19] D. Wu. Music personalized recommendation system based on hybrid filtration. In *2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pages 430–433. IEEE, 2019.

Design of a real-time multiparty DAW collaboration application using Web MIDI and WebRTC APIs

Scott Stickland

The University of Newcastle, Australia
School of Creative Industries (Music)
scott.stickland@uon.edu.au

Rukshan Athauda

The University of Newcastle, Australia
School of Electrical Engineering and
Computing (Information Technology)
rukshan.athauda@newcastle.edu.au

Nathan Scott

The University of Newcastle, Australia
School of Creative Industries (Music)
nathan.scott@newcastle.edu.au

ABSTRACT

Collaborative music production in online environments has seen a renewed focus as developers of Digital Audio Workstation (DAW) software include features that cater to limited synchronous participation and multiparty asynchronous collaboration. A significant restriction of these collaboration platforms is the inability for multiple collaborators to effectively communicate and seamlessly work on a high-fidelity audio project in real-time.

This paper outlines the design of a browser-based application that enables real-time collaboration between multiple remote instantiations of an established, mainstream and fully-featured DAW platform over the Internet. The proposed application provides access to, and modification and creation of, high-fidelity audio assets, real-time videoconferencing and control data streaming for communication and synchronised DAW operations through Web Real-Time Communication (WebRTC) and Web MIDI Application Programming Interfaces (APIs). The paper reports on a proof-of-concept implementation and results, including several areas for further research and development.

1. INTRODUCTION

Present online collaboration, facilitated by recent DAW-specific and -generic approaches, can be classified as either synchronous or asynchronous in design and operation. Synchronous approaches focus on sharing and editing music data in real-time. For effective and meaningful collaboration, this data consists of high-fidelity audio files and streams. However, current Internet bandwidths struggle to handle the real-time transfer of such data-intensive streaming and maintain the audio's high fidelity. Thus, these collaborations are limited to a few participants at most. Conversely, asynchronous approaches forgo real-time online interactions in preference for increased numbers of remote participants and maintenance of the audio assets' fidelity. Central to asynchronous collaboration is the uploading and downloading of lossless audio files to and from cloud storage linked to the DAW applications, or directly to and from collaborators' local storage drives.

Our previous research examined four widely-available approaches to DAW-integrated online collaboration [1]. Two synchronous collaboration platforms, Source Elements' DAW-generic *Source-Connect Pro* and Steinberg's *Cubase Pro*-specific *VST Connect Pro* and *VST Connect Performer* bundle, not only restrict the number of participants but also access to the DAW project itself [2, 3]. Synchronous collaboration necessarily requires compression of the project's audio streams to reduce the effects of latency and jitter

inherent in Internet data transport. Applying lossy audio codecs, though, result in a degradation of the audio's fidelity. The two synchronous platforms do provide optional features to record, store, and stream lossless Pulse Code Modulation (PCM) audio files of a remote performer to replace jitter-affected files post-session.

Data- and bandwidth-intensive audio streaming, and the complication of varying latencies between multiple, simultaneous connections, obliged Avid and Steinberg to approach online collaboration asynchronously, adopting a similar concept to that utilised by Source-Connect Pro's, and VST Connect Pro's, automatic PCM audio upload features. The two asynchronous platforms, Avid's *Cloud Collaboration* and Steinberg's *VST Transit*, are included features of their respective DAWs, *Pro Tools* and *Cubase* [4, 5]. By eschewing real-time collaboration, these approaches significantly increase participant numbers and provide universal access to a DAW project. Employing lossless audio codecs to compress the project's PCM audio files before uploading maximises the available cloud storage capacity, reduces transfer times, and maintains the original audio's high fidelity.

Web browser-based platforms, such as Spotify's *Soundtrap*, AmpTrack Technologies' *Amped Studio 2*, and BandLab Technologies' *BandLab* also offer asynchronous collaboration methods for multiple participants [6-8]. All provide users with the ability to share a DAW project with others, issuing invitations by either co-opting existing social media applications, such as Facebook and Twitter, email and Short Message Service (SMS), or in-application messaging to distribute a link to the project. In *Amped Studio 2* and *BandLab*, sharing a DAW project creates multiple versions, or forks, of the project, one per collaborator, so that the integrity of the original project is preserved [7, 9]. *Soundtrap* adopts a collaboration model similar to *VST Transit* and *Cloud Collaboration*, where users can contribute and save to a cloud version of the project, then sync their local project with the one in the cloud asynchronously.

Nevertheless, transferring large lossless audio files over the Internet takes time and bandwidth, meaning collaborators can only access, listen to, and assess project contributions post factum. Asynchronous collaboration can slow progress in this environment; contributions cannot be auditioned or modified in real-time, so the possibility of revisions, or indeed discarding material altogether, is significantly more likely. *Soundtrap*'s approach to minimising these consequences is to integrate a videoconferencing feature using the WebRTC API [10]. Lind and MacPherson [10] explain that a video chat feature enables 'instant feedback with collaborators as they create projects together in real-time.' It is worth pointing out that while the video chat feature is indeed synchronous, sharing contributions to, and modifications of, a project remain asynchronous.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Georgia Institute of Technology's Centre for Music Technology's *EarSketch* is an amalgam of a browser-based Python and JavaScript Integrated Development Environment (IDE), and a rudimentary DAW for the playback and manipulation of audio samples via scripts [11]. Their recent developments focus on collaborative script editing through the use of an operational-transform algorithm via WebSockets, and time synchronisation utilising Network Time Protocol's (NTP's) clock-synchronisation algorithm [12]. It, however, lacks the requisite processing capabilities for professional mixing and mastering activities.

Currently, mainstream and web browser-based DAW platforms cannot facilitate synchronous, multi-party collaboration and communication, where all participants, irrespective of location, can simultaneously access, edit and contribute to professional, specialised activities such as post-production mixing and mastering. Repositioning such activities as an online collaborative undertaking presents several challenges and requirements.

Latency, Jitter, Bandwidth and High-Fidelity audio assets: Expert post-production activities require the use of high-fidelity audio files. In particular, professional mixing of multitrack recordings for commercial CD and streaming-service release require audio files with a sample rate of 44.1 or 48 kHz, at the very least, to avoid heavy anti-aliasing filtering. Similarly, audio file bit depths of 16- or 24-bit are necessary to reduce the noise floor and create more headroom. Streaming these assets in real-time is challenging considering high-fidelity audio is, by its very nature, bandwidth-intensive. Furthermore, streaming over the Internet utilising reliable transport layer protocols such as Transmission Control Protocol (TCP), increases latency, whereas, the use of 'best-effort service' protocols such as User Datagram Protocol (UDP) can reduce delays (latency) but does not guarantee reliable delivery, resulting in lost packets (jitter).

A benefit of existing asynchronous collaboration methods is the ability to preserve the DAW project's high-fidelity audio files; however, existing synchronous approaches rely upon lossy audio streams [2, 3]. Therefore, a way to access and modify high-fidelity audio files in synchronous multi-party collaborative environments is needed.

Access to, and synchronous operation of, a DAW platform: Existing asynchronous DAW-specific multi-party solutions grant each participant equal access to the music production project via their local DAW instantiation. Providing such equal access to a collaborative, specialised DAW platform in real-time, though, is currently lacking.

Real-time communication methods: Cloud Collaboration, VST Transit, Amped Studio 2 and BandLab, while catering for multiple collaborators, only provide asynchronous text messaging communication [4, 5, 7, 8]. While the VST Connect Pro/Performer bundle and Soundtrap include effective real-time video streaming, the communication is just peer-to-peer (P2P) [2, 6]. Videoconferencing is highly desirable in a multi-party, real-time collaboration environment.

The design of the DAW collaboration application presented in this paper aims to address the challenges and requirements outlined above.

2. THE COLLABORATION FRAMEWORK

Figure 1 illustrates the infrastructure and data flow of a proposed music production collaboration framework.

Each collaboration endpoint runs a local DAW instantiation. Every participant downloads and opens a previously uploaded

collaborative DAW project in cloud storage to each collaborator's DAW instantiation before the collaboration session. The Signalling Server negotiates and instigates the WebRTC peer connections, while the Media Server is used to synchronise and manage the real-time communication and MIDI control data streams among multiple collaborators. The following sections discuss the architecture in detail.

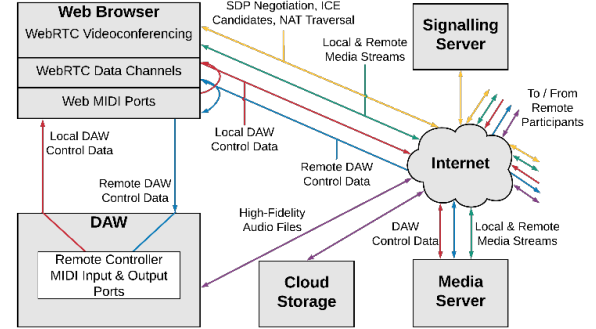


Figure 1. The proposed collaboration framework's architecture and data flow.

The proposed framework (see Fig. 1) avoids high-fidelity audio streaming for the collaboration project's playback and source of new audio material. The framework employs cloud storage and file sharing to distribute the DAW project and its audio assets to all participants before a collaboration session begins. The participants employ their local DAW instantiation's playback for monitoring the project's audio. Streaming lower-volume timecode facilitates collaborative synchronous operation and aligns the playback of remote DAW instantiations' locally-stored high-fidelity audio assets. While not eradicating latency, this approach dramatically reduces its effects by avoiding bandwidth-intensive, high-fidelity audio transmissions in real-time. Each participant is ignorant of the slight differences in timing across the collaboration.

2.1 Achieving synchronicity

DAW instantiations are capable of synchronising their playback utilising three mechanisms: machine control, clock source and timecode [13]. Together, they deliver transport commands, positional references in time, and speed references. The MIDI paradigm includes MIDI Timecode (MTC) and MIDI Machine Control (MMC) protocols. MTC has the additional benefit of functioning as a clock source; therefore, MTC and MMC together institute synchronised playback by establishing a master/slave configuration across the collaboration (see Fig. 2).

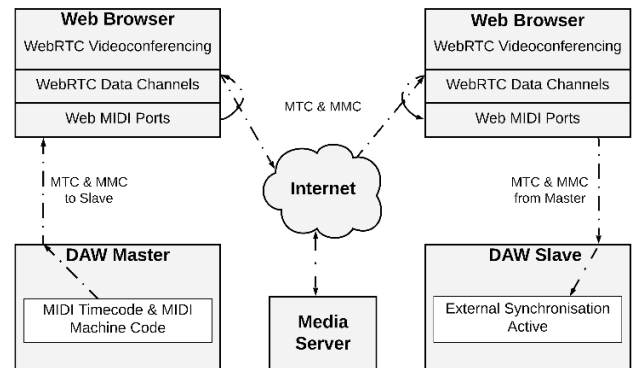


Figure 2. Synchronised playback of DAW instantiations through the streaming of MTC and MMC.

Incumbent on the framework is providing an equitable and inclusive ability to edit the collaboration project and its audio assets. All of the collaboration’s DAW instantiations must also execute the on-screen modifications, made by any individual end-user to their local DAW project, in real-time. Doing so provides synchronised editing and navigation across the collaboration. We have chosen Cubase Pro 10 to be the framework’s DAW platform, as discussed in the next section.

2.2 Cubase Pro 10

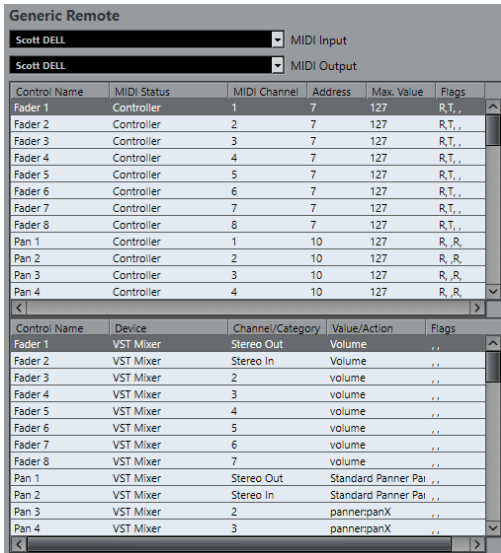
The rationale for choosing Cubase Pro 10, beyond its calibre as a professional, industry-standard DAW for post-production activities, includes the ability to create user-defined MIDI command maps implementing its *Generic Remote* feature, the ability to create user-defined keyboard shortcuts (*Key Commands*) and its MTC and MMC external synchronisation capabilities.

2.2.1 Generic Remote

The Generic Remote (GR) feature allows users to tailor the operation of a generic MIDI controller to most any of Cubase’s functions [13]. For our purposes, the GR provides significant utility for transmitting and receiving MIDI control data mapped to the DAW’s functions, navigation and transport. The GR creates bespoke MIDI maps, linking specific MIDI Continuous Controllers (CCs) and notes to DAW commands and operations (see Fig. 3). Users can assign the GR’s MIDI input and output ports from a list of the computer’s available MIDI devices.

2.2.2 Key Commands

Keyboard shortcuts are a common feature of many software applications, designed to enhance productivity by reducing the number of mouse moves and clicks. Cubase Pro includes numerous keystroke combinations linked to DAW operations, including an increasing number of homogeneous DAW-generic combinations. Cubase Pro’s key commands can be tailored to map specific keystroke patterns to almost any DAW function or operation.



Generic Remote					
		MIDI Input		MIDI Output	
		Scott DELL		Scott DELL	
Control Name	MIDI Status	MIDI Channel	Address	Max. Value	Flags
Fader 1	Controller	1	7	127	R, T, ..
Fader 2	Controller	2	7	127	R, T, ..
Fader 3	Controller	3	7	127	R, T, ..
Fader 4	Controller	4	7	127	R, T, ..
Fader 5	Controller	5	7	127	R, T, ..
Fader 6	Controller	6	7	127	R, T, ..
Fader 7	Controller	7	7	127	R, T, ..
Fader 8	Controller	8	7	127	R, T, ..
Pan 1	Controller	1	10	127	R, R, ..
Pan 2	Controller	2	10	127	R, R, ..
Pan 3	Controller	3	10	127	R, R, ..
Pan 4	Controller	4	10	127	R, R, ..

Control Name	Device	Channel/Category	Value/Action	Flags
Fader 1	VST Mixer	Stereo Out	Volume	..
Fader 2	VST Mixer	Stereo In	Volume	..
Fader 3	VST Mixer	2	volume	..
Fader 4	VST Mixer	3	volume	..
Fader 5	VST Mixer	4	volume	..
Fader 6	VST Mixer	5	volume	..
Fader 7	VST Mixer	6	volume	..
Fader 8	VST Mixer	7	volume	..
Pan 1	VST Mixer	Stereo Out	Standard Panner Pos	..
Pan 2	VST Mixer	Stereo In	Standard Panner Pos	..
Pan 3	VST Mixer	2	pannerpanX	..
Pan 4	VST Mixer	3	pannerpanX	..

Figure 3. Cubase Pro 10’s Generic Remote page.

2.3 Control data streaming

One of the framework’s significant efficiencies must be its ability to effectively and reliably stream MIDI control data to all participants over the Internet. Successful tests of Cubase Pro’s GR MIDI mapping utilised the OSX MIDI network driver on Mac and

Tobias Erichsen’s *rtpMIDI* driver software [14] on Windows computers to establish remote network connections between MIDI ports. Employing the RFC 6295 Real Time Protocol (RTP) payload format for MIDI messages (RTP-MIDI), the drivers map MIDI 1.0 data onto RTP streams over UDP [15]. These ports were assigned to their corresponding GR’s input and output ports (see Fig. 4), and each GR instance was configured identically by importing a mutual XML mapping file.

RFC 768 UDP [16] is an inherently best-effort transport service that is suited to real-time transmissions. UDP lacks reliable data transfer characteristics. Nevertheless, RTP offered a degree of reliability through error correction and concealment strategies to deal with lost packets when combined with the Real-Time Control Protocol (RTCP) [17].

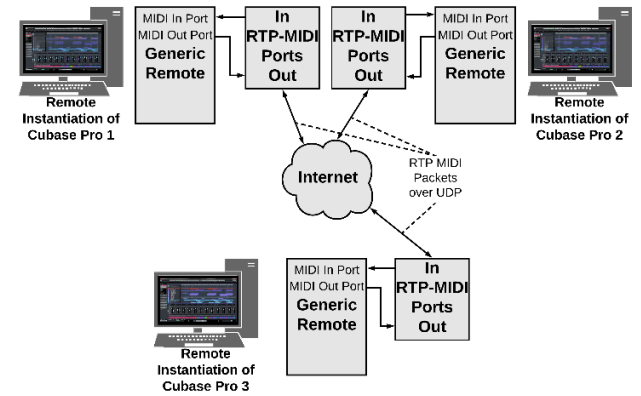


Figure 4. Using connected network MIDI ports over the Internet to test Cubase Pro’s Generic Remote interactivity.

3. WEBRTC AND WEB MIDI APIs

This section outlines the use of Web RTC and Web MIDI APIs to implement the proposed architecture. In order to implement the framework, the application needs to: (a) access a collaborator’s computer webcam and microphone; (b) create secure connections between the online participants; (c) provide a secure videoconferencing capability; (d) gain access to MIDI ports assigned to Cubase Pro’s GR; (e) create secure, semi-reliable, configurable data channels between the online participants; and (f) route MIDI control data to and from Cubase Pro and the data channels for synchronous streaming over the Internet. Exploiting the WebRTC and Web MIDI APIs can achieve all of these requirements.

3.1 WebRTC

3.1.1 getUserMedia method

The `getUserMedia()` method is one of the most common ways to access local webcam and microphone media devices, thus creating a local media stream [18]. In the interests of privacy, only once a user gives permission, the browser can access the local media devices.

3.1.2 RTCPeerConnection API

WebRTC’s `RTCPeerConnection` interface is its fundamental basis and establishes a connection between two endpoints, or peers, over the Internet. Once established, the connection provides direct bidirectional P2P communication without requiring an intervening server. Reducing the distance data needs to travel similarly reduces the latency it incurs. For the interface application, the `RTCPeerConnection` can facilitate both media and data flow between collaborators.

3.1.3 MediaStream API

A media stream comprises of two tracks, one each for video and audio, with each track comprising of one or more channels; for example, a stereo audio track consists of separate left and right channels. The `MediaStream` interface creates an object by grouping the local media tracks, thus defining each participant's media stream. The flow of `MediaStream` objects over an `RTCPeerConnection` facilitates the collaboration framework's videoconferencing.

3.1.1 RTCDATAChannel API

The WebRTC `RTCDATAChannel` interface creates an additional bidirectional channel over an `RTCPeerConnection` for the simultaneous transmission of arbitrary data with similarly low latency and high throughput [19]. The `RTCDATAChannel` interface was modelled closely on the WebSockets API, and consequently, their methods (e.g. `send()`) and handlers (e.g. `onmessage`) behave similarly [18]. The transmission between participants of the framework's MIDI control data occurs over these data channels.

3.2 Web MIDI

3.2.1 MIDIACCESS API

Web MIDI's `MIDIACCESS` interface supplies methods to list the MIDI input and output ports available to the browser and provide access [20]. The `navigator.requestMIDIAccess()` method and `onMIDISuccess` handler allows the framework's interface application, and by extension the participants, to nominate input and output ports that correspond to Cubase Pro's GR ports, consequently establishing the crucial link between Cubase Pro and the interface application.

3.2.2 MIDIInput and MIDIMessageEvent APIs

The success of the browser-based application to act as an interface between Cubase Pro and the larger collaboration infrastructure depends upon bidirectional transfer of MIDI control data to and from a DAW's GR MIDI ports, to and from a created `RTCDATAChannel`.

The `MIDIMessageEvent` interface achieves one direction when passing an event object, in this case, a MIDI 1.0 message, to a `MIDIInput` port's `onmidimessage` handler upon receiving control data from the GR output port. Each event object consists of a `Uint8Array` comprising MIDI message data bytes and a high-resolution timestamp [20] and is transmitted via the `send(event)` method on the `RTCDATAChannel` (see Fig. 5).

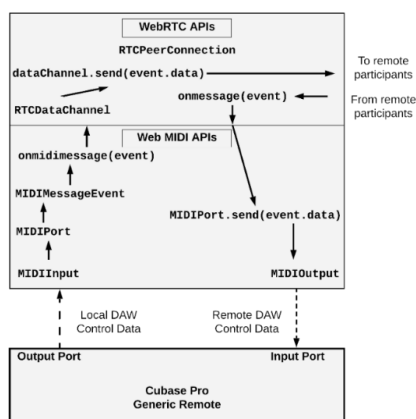


Figure 5. The interface application's use of WebRTC and Web MIDI APIs, methods and handlers.

3.2.3 MIDIOutput API

The Web MIDI `MIDIOutput` interface and its `send()` method realise the application's other directional interfacing, together with the `RTCDATAChannel` interface's `onmessage` event handler. The data channel's `onmessage` handler receives a control data event object, which transmits the MIDI data bytes and timestamp over the `MIDIOutput` interface's MIDI port (see Fig. 5).

3.3 Virtual MIDI ports

The most productive way to link the GR and corresponding interface application input and output ports are via virtual MIDI ports. For the tests conducted so far, a third-party MIDI driver, *LoopBe30* by *nerds.de* [21], has created ports for the internal connections between Cubase Pro and the interface. However, plans are for the application to feature its own virtual MIDI driver and ports in the future.

3.4 Stream Control Transmission Protocol (SCTP)

WebRTC data channels utilise the RFC 4960 Stream Control Transmission Protocol (SCTP) for their implementation and delivery. Johnston and Burnett state that SCTP "provides useful features not available in TCP, including reliable or semi-reliable delivery, non-ordered delivery of packets, multiple streams within an SCTP association, and an ability to send messages." [18]. Of particular importance to the collaboration framework is SCTP's semi-reliable and non-ordered delivery of data packets, in addition to the `RTCDATAChannel` interface's `maxPacketLifeTime` and `maxRetransmits` unsigned shorts and ordered Boolean attributes [22]. Further testing and tuning are needed to determine optimal values for such parameters.

4. CONNECTION ARCHITECTURES

While WebRTC is primarily designed to establish a P2P connection, three different approaches can create multi-party collaborations: (a) Mesh, (b) Mixing, and (c) Routing [23].

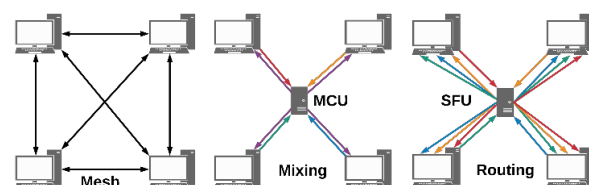


Figure 6. The mesh, mixing and routing architectures.

4.1 Mesh

As the name suggests, each participant constructs the mesh architecture by establishing a peer connection with every other participant (see Fig. 6). While it is relatively simple to implement and requires no backend infrastructure, it is limited in its ability to scale to a large number of participants and is CPU- and bandwidth-intensive as the number of participants increases [24].

4.2 Mixing

A mixing architecture requires the integration of a Multipoint Control Unit (MCU) into the multi-party architecture. Its construction involves each participant establishing a peer connection with the MCU only (see Fig. 6). It is the MCU's task to receive and mix the individual media streams, then send the mixed-media stream to the participants [24]. Each endpoint assumes it is interacting with another single endpoint.

4.3 Routing

A routing, or relay, architecture requires the integration of a Selective Forwarding Unit (SFU) where each participant establishes a peer connection with the SFU only (see Fig. 6). Unlike an MCU, an SFU forgoes transcoding of the media streams, instead deciding which of the media streams to forward on to the participants [23]. Each participant receives the routed media and data streams of all other participants in the collaboration.

5. PROTOTYPE IMPLEMENTATION

This section describes the results of a prototype implementation of the DAW collaboration application. The prototype implementation utilises a mesh architecture (which is the simplest to implement) to interact with peers. Although mesh architecture is not scalable to a large number of participants, the prototype demonstrates the feasibility of the proposed application. Table 1 summarises the resources used in the implementation.

Table 1. Resources for the P2P Mesh Test.

Computer 1 (Peer 1, Signalling Server)	i5-7300U 2.6 GHz CPU; 16 GB RAM; Windows 10 Enterprise OS
Computer 2 (Peers 2, 4, 6, 8)	i7-8700K 3.7 GHz CPU; 32 GB RAM; Windows 10 Pro OS
Computer 3 (Peers 3, 5, 7)	i7-3610QM 2.3 GHz CPU; 12 GB RAM; Windows 7 Home Premium OS
LAN Speed	1 Gbps
Browser	Chrome 75
DAW	Cubase Pro 10
Application-Layer Protocol	HTTPS
Signalling Server	Built using socket.io on node.js

The application's media and data streams were stable up to and including the addition of the sixth peer, though the time taken for the signalling process to establish each connection was noticeably longer with each new addition. While there was a perceptible increased latency in the playback and execution of functions across the DAW instantiations, this perception was only due to having all three computers in the one physical space.

The addition of the final two peer connections, however, saw a marked deterioration in stability, including frozen video streams, increasingly distracting audio stream jitter, and a progressive lack of DAW responsiveness and extended delays in executing data-heavy functions such as level fader operations. Figure 7 plots Computer 1's transmission and reception rates of WebRTC-related packets.

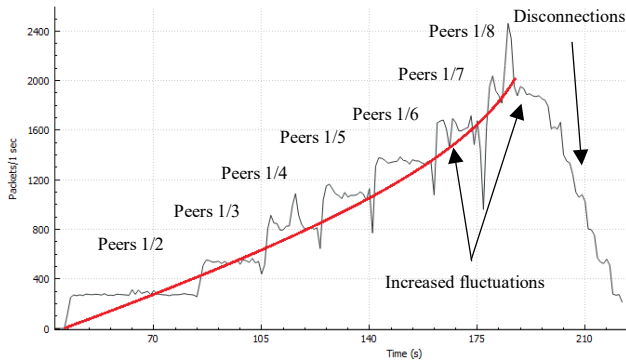


Figure 7. Peer 1 packets transmitted and received per sec.

Latency across the mesh architecture increased exponentially as each new peer joined the collaboration, as demonstrated by Figure 8's plot of Peer 1's packet delivery times. With a single P2P connection, the average delivery time was 0.47 ms and increased by 0.18 ms and 0.15 ms with the addition of the next two peers, respectively. However, the addition of connections to Peers 7 and 8 saw increases of 0.82 ms and 2.09 ms, respectively.

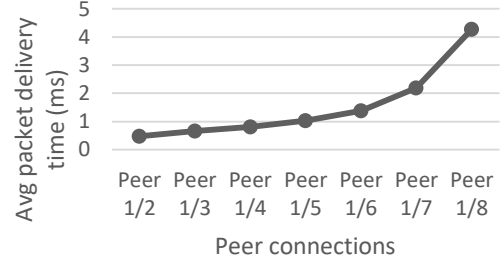


Figure 8. Peer 1 connections: average packet delivery times.

Figure 9 plots the percentage of Computer 1's overall CPU capacity utilised by the framework's three processes, namely Chrome.exe, Cubase10.exe, and Node.exe. At peak consumption, which coincided with 15 mesh connections and data-intensive Cubase Pro 10 operation, the processes accounted for 66.43% of the 2.6 GHz CPU. At the same time, the computer's overall percentage of CPU usage measured just over 93%.

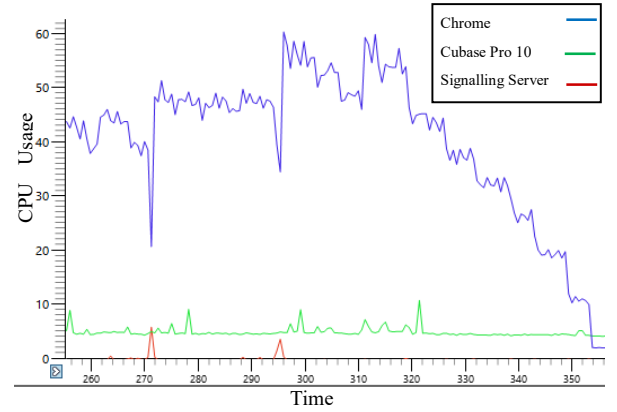


Figure 9. Peer 1 CPU usage percentage: Chrome Browser (blue), Cubase Pro 10 (green), Node Signalling Server (red).

6. CONCLUSION AND FUTURE WORK

This paper presented the design of a browser-based DAW collaboration framework that aims to provide multi-party real-time collaboration. The framework addresses many of the shortcomings of existing approaches – including access to high-fidelity audio assets by collaborators, equal access to a DAW project, multi-party real-time video-conferencing and others. The paper also outlined a prototype implementation using Web RTC and Web MIDI APIs as a proof-of-concept. The results are promising.

Future work will determine the most reliable and timely delivery methods, having demonstrated the ability to replace the transport of RTP-MIDI packets over UDP with MIDI bytes and timestamps over SCTP, via WebRTC data channels, successfully.

At present, Cubase Pro's GR feature limits the transmission of MIDI control data to commands and functions commonly featured on mainstream control surfaces. This restriction is due to control

surfaces requiring feedback from the DAW reflecting on-screen execution operations, such as Mixer Console and Transport functions, Insert, Send and VST instrument plug-in parameters and channel-strip EQ settings. Future testing will encompass the use of keyboard commands with keystroke-to-MIDI translation to address the shortcomings of the GR's range of functions.

The architecture has been implemented only in a limited scenario and with mesh architecture only. Future work will implement and analyse results of mixing and routing architectures and their scaling capabilities through the inclusion of a media server. Testing will also expand to include implementation over the Internet to measure latencies and determine an acceptable delay threshold.

The MIDI 2.0 protocol, upon its release and mainstream integration, could provide enhancements to the transport and delivery of the application's MIDI control data. Information published by the MIDI Manufacturers Association (MMA) has signposted an increase in the resolution of control messages from 7 bits up to 32 bits, and MIDI packets will include a jitter reduction timestamp to improve timing accuracy [25]. Future work will integrate the MIDI 2.0 messaging protocol to determine the scale of improvement in the transport of data and the accuracy of received data streams.

7. ACKNOWLEDGMENTS

This research is supported by the Australian Government Research Training Program Scholarship.

8. REFERENCES

- [1] Stickland, S., Scott, N. and Athauda, R. 2018. A Framework for Real-Time Online Collaboration in Music Production. In *Proceedings of the ACM2018: Conference of the Australasian Computer Music Association* (Perth, Australia, 6-9 December, 2018).
- [2] Steinberg Media Technologies GmbH. 2018. VST Connect Pro. Retrieved 6 May 2018 from https://www.steinberg.net/en/products/vst/vst_connect/vst_connect_pro.html
- [3] Source Elements. 2018. Source-Connect. Retrieved 20 June 2018 from <http://source-elements.com/products/source-connect>
- [4] Steinberg Media Technologies GmbH. 2018. VST Transit: The World of Music Cloud Collaboration. Retrieved 25 July 2018 from https://www.steinberg.net/en/products/vst/vst_transit.html?et_cid=15&et_lid=22&et_sub=VST%20Transit
- [5] Avid Technology Inc. 2018. Avid Cloud Collaboration for Pro Tools: How It Works. Retrieved 5 June 2018 from <http://www.avid.com/avid-cloud-collaboration-for-pro-tools/how-it-works>
- [6] Spotify AB. 2019. Soundtrap - Make music online. Retrieved 9 September, 2019 from <https://www.soundtrap.com/>
- [7] AmpTrack Technologies AB. 2019. Amped Studio 2 | Online Beatmaker and Music Studio. Retrieved 9 September, 2019 from <https://ampedstudio.com/>
- [8] BandLab Technologies. 2019. BandLab: Music Starts Here. Retrieved 9 September, 2019 from <https://www.bandlab.com/>
- [9] Guitar Player Magazine. 2017. BandLab Collaborative App @ NAMM 2017. Video. (31 January, 2017). Retrieved 9 September, 2019 from <https://www.youtube.com/watch?v=hYaOZl1999g>
- [10] Lind, F. and MacPherson, A. *Soundtrap: A collaborative music studio with Web Audio*. Queen Mary Research Online, City, 2017.
- [11] Web Audio Conf. 2018. Collaborative Coding with Music: Two Case Studies with EarSketch by Avneesh Sarwate. Video. (28 September, 2018). Retrieved 10 September, 2019 from <https://www.youtube.com/watch?v=0qBVSCRpogg>
- [12] Sarwate, A., Tsuchiya, T. and Freeman, J. 2018. Collaborative Coding with Music: Two Case Studies with EarSketch. In *Proceedings of the Web Audio Conference 2018* (Berlin, Germany, 19-21 September, 2018).
- [13] Bachmann, C., Bischoff, H., Harris, L., Kaboth, C., Mingers, I., Obrecht, M., Pfeifer, S., Schütte, B. and Sladek, M. (2018). *Cubase Pro 10, Cubase Artist 10 Operation Manual*, Hamburg, Germany: Steinberg Media Technologies GmbH., Retrieved 16 November 2018, from https://steinberg.help/cubase_pro_artist/v10/en/Cubase_Pro_Artist_10_Operation_Manual_en.pdf.
- [14] Erichsen, T. 2016. rtpMIDI. Version 1.1.8. Computer Program. Retrieved 15 April, 2018 from <https://www.tobias-erichsen.de/software/rtpmidi.html>
- [15] Lazzaro, J. and Wawrzyniek, J. (2011). *RTP Payload Format for MIDI* (RFC 6295), The IETF Trust, Retrieved 29 April, 2018, from <https://tools.ietf.org/pdf/rfc6295.pdf>.
- [16] Postel, J. (1980). *User Datagram Protocol* (RFC 768), Internet Engineering Task Force, Retrieved 18 February, 2018, from <https://tools.ietf.org/pdf/rfc768.pdf>.
- [17] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V. (2003). *RTP: A Transport Protocol for Real-Time Applications* (RFC 3550), The Internet Society, Retrieved 19 February, 2018, from <https://tools.ietf.org/pdf/rfc3550.pdf>.
- [18] Johnston, A. B. and Burnett, D. C. 2014. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Digital Codex LLC., St. Louis, United States of America.
- [19] Dutton, S. 2012. Getting Started with WebRTC. *HTML5Rocks*. Retrieved 24 July, 2018 from <https://www.html5rocks.com/en/tutorials/webrtc/basics/>
- [20] World Wide Web Consortium. 2015. Web MIDI API. (March 17) Retrieved 30 July 2018 from <https://www.w3.org/TR/webmidi/>
- [21] Schmitt, D. 2019. LoopBe30. Version 1.6. Computer Program. Retrieved 10 April, 2019 from <https://www.nerds.de/en/loopbe30.html>
- [22] World Wide Web Consortium. 2018. WebRTC 1.0: Real-time Communication Between Browsers. (September 20) Retrieved 14 September, 2018 from <https://www.w3.org/TR/webrtc/>
- [23] Levent-Levi, T. 2019. WebRTC Multiparty Architectures. *BlogGeek.Me*. (15 April, 2019). Retrieved 2 June, 2019 from <https://bloggeek.me/webrtc-multiparty-architectures/>
- [24] Bernardo, G. G. 2014. WebRTC beyond one-to-one communication. *webrtcH4cKS*. (4 February, 2014). Retrieved 2 June, 2019 from <https://webrtcH4cKS.com/webrtc-beyond-one-one/>
- [25] The MIDI Association. 2019. Details about MIDI 2.0, MIDI-CI, Profiles and Property Exchange. *MIDI News*. (1 June, 2019). Retrieved 3 June, 2019 from <https://www.midi.org/articles-old/details-about-midi-2-0-midi-ci-profiles-and-property-exchange>

Soundworks

A Framework for Networked Music Systems on the Web

State of Affairs and New Developments

Benjamin Matuszewski
CICM/musidance EA1572, Université Paris 8
STMS Ircam-CNRS-Sorbonne Université
Paris, France
benjamin.matuszewski@ircam.fr

ABSTRACT

This paper presents a novel major version of *soundworks*, a framework dedicated at developing distributed multimedia applications on the web and entirely written in *javascript*. Since its first release in 2015, the framework has served as a basis for numerous artistic and research projects such as concerts, installations, workshops, teaching or experimental setups. These diverse use cases and situations permitted to validate numerous aspects of the framework but also showed some limitations—particularly in terms of inclusion of non-expert developers such as artists and researchers—leading to the novel version presented here.

The paper first presents some applications developed in the last year and show that, despite their idiosyncrasies, recurring problems have emerged during their elaboration and development (e.g. state-management). Second, we present new design and implementation aspects of the framework developed to overcome these issues. Finally we describe a simple testbed application—designed to summarize a number of recurring features and constraints encountered in Network Music Systems—and some elements of its implementation within *soundworks*.

We believe that this novel version will provide solid foundations for the design and implementation of higher-level tools dedicated to non-expert developers, and thereby, foster new artistic, technological and epistemic areas. The *soundworks* framework is open-source and released under BSD-3-Clause license.

1. INTRODUCTION

The recent specification and development of the WebAudio API—together with other APIs such as WebSockets [17] and the developments of ubiquitous computing [16] with smartphones and nano-computers—enabled novel possibilities in the area of Networked Music Systems. These novel tools, alongside with the possibilities offered by a full-featured and interactivity-centered language such as *javascript* permit to envision these technologies from several points of view. First, they can be considered as a new development and a natural extension in the long history of multi-source electro-acoustic music. [13] Second, they can provide a novel platform for composition and performance. [3] Third, they

enable a wide range of possibilities in the creation of new interfaces for musical expression. [8] In all cases, these trends tends to show that these technologies—with their simplicity, ubiquity and inherent networked nature—can play a central role in the evolution of Networked Music Systems. [2, 15]

In this context, the development of a dedicated framework, adapted to and designed for the specificities of the web platform is essential. In the last years, a number of such frameworks have been proposed by the community. [1, 7] For now, however, these solutions are far from the maturity of environments such as *Max/MSP* or *PureData*. While these environment have developed a number of key concepts along the years, [9, 10] we think however that adapting these concepts directly to the web^{1, 2} tends to neglect the main specificity of the platform (and the novel possibilities it unfolds): the *network*.

soundworks^{3, 4} is a framework dedicated to the development of distributed multimedia applications on the web. The initial version of the framework, released in 2015, has been written by S. Robaszkiewicz and N. Schnell. [11] Since then, the framework has known two major revisions (in 2016 and 2017) and has served as a basis for numerous artistic and research projects (e.g. concerts, installations, workshops, pedagogical or experimental setups). Thereby, *soundworks* has permitted to explore Network Music Systems in many directions such as: participative performances, use of smartphones as speaker array or as new instruments, measures of movements in collective settings. While these achievements tends to prove the efficacy of the framework considered as an experimental platform, [12] they also permitted to highlight some inherent and recurring difficulties. *soundworks#v3* aims to address some of these difficulties as well as to provide solid foundations upon which environments that facilitate the inclusion and agency of non-expert developers can be built.

In Section 2, we review three recent *soundworks* applications implemented in close collaboration with multiple stakeholders, and present of some of their similarities despite their different aims and goals. In Section 3, we present the main conceptual and technical aspects of *soundworks#v3*, designed to provide a better support to the recurring needs described in 2. Finally, in Section 4, we describe a simple, yet not trivial, application that we consider representative of recurring aspects of Networked Music Systems, and shortly present elements of its implementation using *soundworks*.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹<https://github.com/petervdn/webaudiotool>

²<https://github.com/Fr0stbyteR/webaudio-patcher>

³<https://github.com/collective-soundworks/soundworks>

⁴<https://www.npmjs.com/org/soundworks>

2. ACHIEVEMENTS AND RATIONALES

In this section we shortly present three applications that has been designed and used in several contexts (e.g. workshops, performances, installations, scientific experiments) in the last year. These descriptions focus on the features and strategies implemented to support the needs, agency and workflows of non-expert developer users (e.g. artist, researcher, performers) in working situations. We conclude with a formalization of these recurring patterns, leading to the developments presented in section 3.

2.1 Elements

*Elements*⁵ is an application that has been specifically designed to conceive and prototype movement-based distributed Interactive Machine Learning scenarios. [5] The application has been iteratively tested and developed in several contexts such as workshops, artworks and performances⁶ or scientific experiments.⁷

The key aspects for the appropriation of the application by non-expert developer users stand in two complementary elements:

- A JSON file dedicated at configuring the different clients in terms of interface, type of synthesis or mapping.
- A controller (see Figure 1, left) that allows for both remote monitoring (e.g. plot sensors, decoding of the ML algorithm) and remote control of each client (e.g. mute, volume).

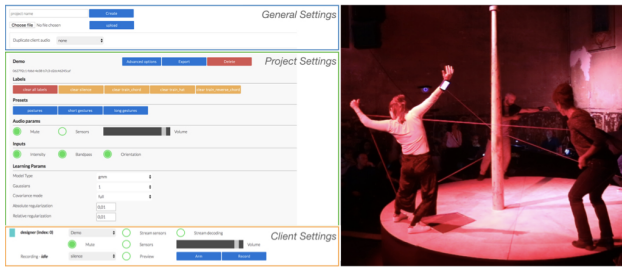


Figure 1: On the left, *Elements*' controller interface highlighting the different possibilities of monitoring and control. On the right, performance of *Cordas* composed by Michelle Agnes Magalhaes.

2.2 Future Perfect

Future Perfect is an immersive 3D audio visual performance and installation work developed by Garth Paine during a residency that took place in 2018 between Ircam and ZKM.⁸ The application allows the composer to perform on the audience smartphone's speakers using several dedicated interfaces. Figure 2 shows the composer in performance situation with three iPads as well as screenshots of the different interfaces designed and used for composing and performing.

In this application, many strategies have been implemented to provide the composer a dynamic environment in which he could test sonic material (i.e. dynamic update of sound files, creation of presets), simply configure many aspect of the synthesis (e.g. granular synthesis parameters, fade times), but also have useful feedback on the state of audience's smartphones (e.g. loading states, position in concert hall). Again, the key elements here were the remote monitoring and control interfaces that enabled a rapid feedback loop in both composition and performance situations.

⁵<https://como.ircam.fr/apps/elements>

⁶https://www.youtube.com/watch?v=c6Flruf_Udc

⁷<https://www.unige.ch/cisa/emodemos/>

⁸<https://www.ircam.fr/person/garth-paine/>

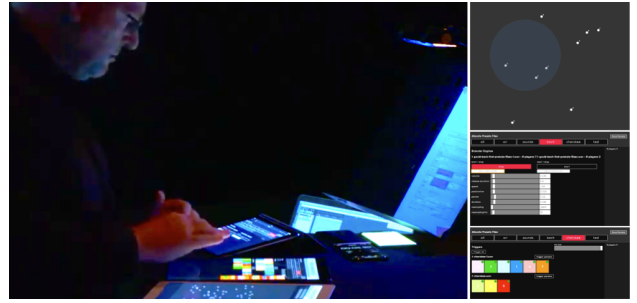


Figure 2: On the left, Garth Paine performing *Future Perfect*. On the right, screenshots of the different interfaces used during composition and performance.

2.3 Biotope

Biotope is an generative and interactive installation composed by Jean-Luc Hervé, realized at Ircam and exposed at the Centre Georges Pompidou, Paris in the context of the exhibition "La fabrique du vivant".^{9,10} The installation is composed of 27 *Raspberry Pi* nano-computers (see Figure 3, left) running soundworks clients written using NodeJs. [4] The audio synthesis is achieved using a NodeJs wrapper on top of libpd.¹¹

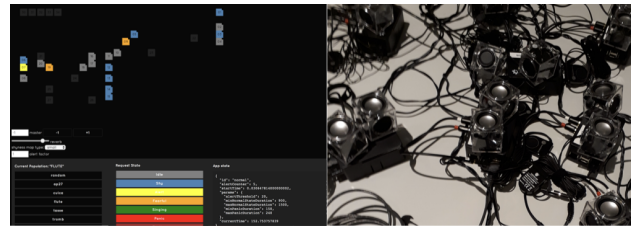


Figure 3: On the left, screenshot of the centralized controller. On the right, photograph of the musical agents, running on Raspberry Pi, created for the *Biotope* installation composed by Jean-Luc Hervé.

In this system, many strategies have been implemented to provide a dynamic and testable environment to the composer and to the computer music designer. Among them, the more important ones are: a mean to easily update audio content, and a centralized controller dedicated at both monitoring the state of the application (for example, each square in Figure 3 right, represents a musical agent in its relative position in the exhibition space, the different colors give an overview of their state in real-time) and at controlling the state and parameters of each client in real-time.

2.4 Recurring Patterns

We can see that these different examples—which span across a wide range of applications (from experimental system to performance or installation)—share common strategies. First, they all provide a dedicated client that allows to monitor and take control over every client of the system in a simple way. This point stands to be of primary importance to maintain the agency of the user working in a complex setup composed of many devices. Second, they all allow—at different levels of maturity and usability—to update content, mappings and synthesis parameters dynamically or from configuration files. Indeed, these applications implement a similar architectural pattern where the state of each client is synchronized

⁹<https://www.ircam.fr/agenda/biotope/detail/>

¹⁰<https://youtu.be/RmSujqdT6L0>

¹¹<https://github.com/ircam-jstools/node-libpd>

in some way with the server, allowing to update every part of the distributed application from a centralized point.

This pattern—that appeared very effective for implementing versatile and adaptable tools fostering creativity—provide insights on the functionalities our framework must facilitate. More precisely, it shows the necessity of a robust and versatile distributed state management system, aimed at simplifying *remote monitoring and control* in an environment composed of many devices.

While the presented examples showed the feasibility of creating such systems using the current version of *soundworks*, the experience showed that these aspects were not properly supported by the framework, leading to overly complicated and redundant architectures. This is these drawbacks that the novel version of *soundworks* presented in the next section propose to overcome.

3. DESIGN AND IMPLEMENTATION

In this section, we present some design and implementation aspects of the third version of the *soundworks* framework. First, we present the scope and high-level aspects of the framework. Second, we describe in more depth the novel state management component that have been introduced to support recurring patterns described in Section 2. We briefly conclude by presenting the motivations and expected benefits of the novel packaging and distribution strategy.

3.1 Architecture Overview

Since its inception, *soundworks* is dedicated at simplifying the development of web-based and distributed real-time musical systems. Figure 4 presents a bird’s-eye view of a typical *soundworks* application. Applications created using *soundworks* follow a star network architecture centered around a NodeJs server. [6] Clients can have multiple responsibilities (e.g. audio rendering, visual rendering, control) and be of different kinds (e.g. mobile, desktop, nano-computers).

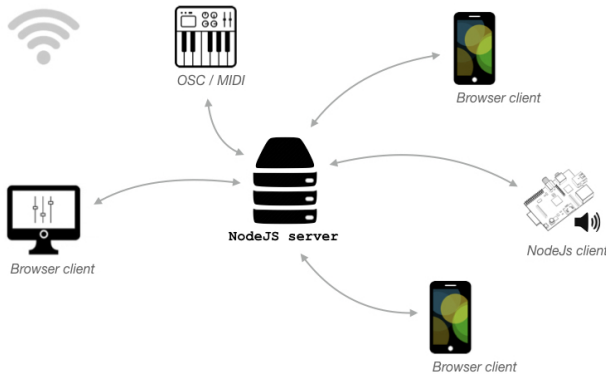


Figure 4: High-level view of the architecture of a typical *soundworks* application: clients of various types (e.g. mobile and desktop browsers, embedded hardware running a NodeJs client, external software communicating through OSC or MIDI) connected to a central NodeJs server.

Until now, the framework has mainly focused on mobile applications and has therefore privileged certain characteristics of these platforms (e.g. graphical user interface, usability). While these aspects remain important, it appears now that—to preserve its efficiency as an experimental platform and to support more and more complex applications and use-cases—the framework must evolve toward more modularity and extensibility, considering both software (e.g. integration of third party components and libraries) and hardware (e.g. integration of IoT elements).

In this objective, the scope of the framework has been refined and narrowed down to focus on three key aspects, namely: *communications*, *service management* and *state management*. As a consequence, some features such as templating, have been removed from the framework and are now delegated to external and specialized libraries. These developments also permitted to reduce the API surface area of the framework.

3.2 Communications and Services

While similar in their principles, the *communication* and *service management* components have evolved toward more simplicity and efficiency. Figure 5 summarizes the initialization process common to all *soundworks* clients:

- The *init* step consists in connecting two WebSockets to the server, one dedicated to string (JSON compliant) data and another dedicated to binary data. The API of both sockets is similar and expose a simple *publish / subscribe* interface.
- Once both sockets are connected, *soundworks* can start the services initialization. As services can depend on each others (for example, the clock synchronization process can rely on a resumed audio context), *soundworks* takes care of the services’ dependency graph and start each service accordingly.
- Finally, when all services are in *ready* state the application specific code (called *Experience* in *soundworks*’ terminology) can start.

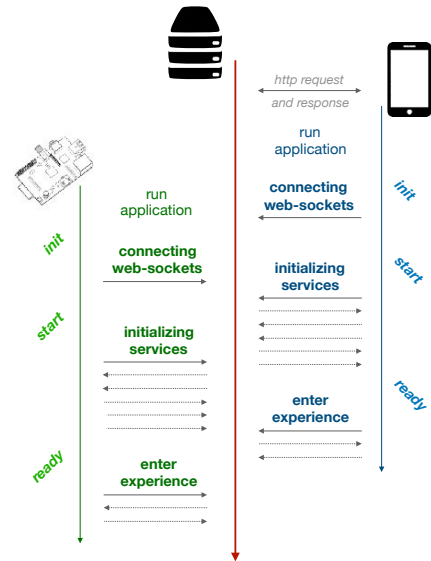


Figure 5: Initialization process of a *soundworks* clients, here a mobile browser and a NodeJs client running in embedded hardware.

Figure 5 also illustrates a novel feature of the framework that enables the seamless implementation of *soundworks* clients in any type of javascript environment (i.e. browsers or NodeJs). Indeed, while this approach has already been tested and deployed in a production setting (cf. 2.3), the complete rewriting of the framework permitted to properly integrate it by making most of the code compatible to both platforms. This novel feature should simplify the creation of applications composed of multiple kind of clients (e.g. smartphones and nano-computers), and thus allow to generalize and democratize the concept of *web of audio things* described in [4] (note that similar ideas has been proposed in [14]).

3.3 State Management

An important novel feature of *soundworks* is the integration of a state management system.

Indeed, since the introduction of the *Flux* pattern proposed by Facebook,¹² a number of state management libraries¹³ have been proposed. The usage of this pattern is nowadays widespread and considered a good practice among the *javascript* community. However, existing libraries are not firstly designed for distributed applications and are difficult to adapt to our context for two main reasons. First, they do not formalize nor integrate the notion of discrete and volatile events very common in our applications (e.g. triggering a sound). Second, they do not provide *out-of-the-box* a simple way of synchronizing states across several nodes in the network.¹⁴

To tackle these issues, we created a novel protocol and implemented a novel component, inspired by the *Flux* pattern and adapted to the particular requirements of our applications.

Concepts and Requirements

In the context of real-time, audio-centered and distributed applications, the application of such circular pattern presents certain particularities schematized in Figure 6.

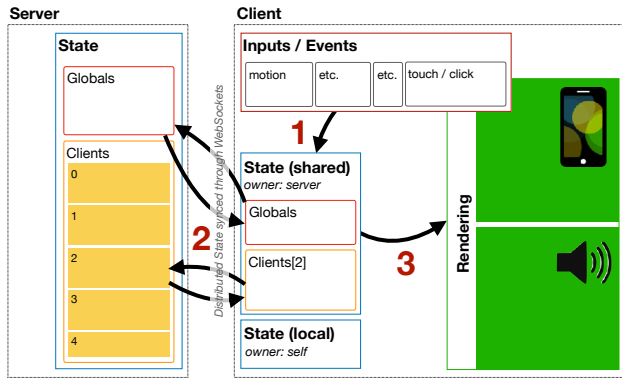


Figure 6: Conceptual overview of a circular and distributed state management system inspired by the *Flux* pattern. The main addition stands in the necessity to keep states synchronized with the server.

The Figure particularly highlights two important requirements and implications for the implementation of this pattern. First, it shows that the state of every client has to be kept synchronized server-side. The rationale for this design strategy (see Section 2 for details) stands in the need to remotely monitor and control any client of the system from a centralized point. Indeed the possibility to dynamically interact with any node of the network, and the rapid feedback loop it enables, is of primary importance in working situations. Furthermore, it appears to be crucial in exploratory contexts (such as artistic and research activities) where the final application cannot be specified beforehand and emerges from a iterative process.

Second, the Figure highlights the need of a certain granularity in the definition and synchronization of the states. Indeed, while some variables and parameters (named *globals* in the Figure) needs to be accessible to every client (e.g. master volume, mute), the particular state a client (*clients[2]* in the Figure) should not be shared

¹² <https://facebook.github.io/flux/>

¹³ For example: <https://redux.js.org/> or <https://vuex.vuejs.org/>

¹⁴ The *dop.js* (<https://distributedobjectprotocol.org/>) library propose an interesting approach, however it aims at synchronizing a single state across every node which is not optimal (particularly in terms of bandwidth) in our context.

with all its peers. It only needs to be monitored or controlled by particular types of clients dedicated to authoring and performance situations.

Protocol and API

To fulfill these requirement while preserving the idea of circular flow between actions, data and rendering proposed by the *Flux* pattern, we designed a simple protocol and implemented a new library.¹⁵ The main principles of the protocol we propose are:

- Allow any node to *create* a new state from a declared *schema*.
- Allow to keep the state *synchronized* with the server.
- Allow any node to *observe* new states created on the network.
- Allow any node to *attach* to a state created by another node.

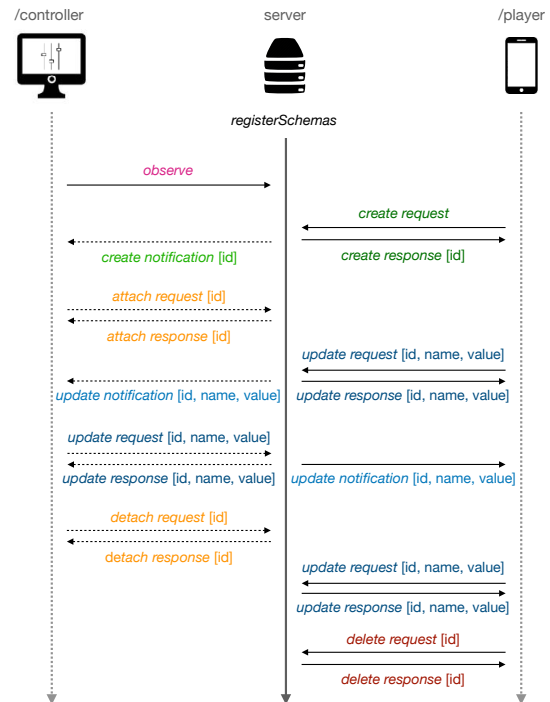


Figure 7: Example of the protocol implemented by the *StateManager*.

Figure 7 illustrates a generic scenario enabled by this protocol. A client (we name *controller*) observes the server and attach to the state created by another client (here, called *player*). When attached, the *controller* receives a notification each time the state is updated by its creator (or any other attached node), enabling *remote monitoring*. The *controller* can also update values of the attached state, enabling *remote control*. At any moment, the *controller* can detach from the state and stop to receive update notifications.

The protocol is abstracted behind a small and simple API illustrated in the pseudo-code example of Listing 8. This simple example also highlights two interesting aspects of the component:

- The complete abstraction of network communications, allowing users to focus on the application logic rather than routing of network messages.

¹⁵ While the component is for now integrated as a first class citizen in *soundworks*, it will be abstracted and released as a standalone library in a near future.

```

1 // server-side
2 const synthSchema = {
3   volume: { type: 'float', min: -80, max: 6 },
4   triggerSynth: { type: 'any', event: true },
5 };
6 const manager = new StateManager(server);
7 manager.registerSchema('synth', synthSchema);
8
9 // client-side
10 const manager = new StateManager(client);
11 const playerState = await manager.create('synth');
12 playerState.subscribe(updates => {
13   for (let [key, val] of Object.entries(updates)) {
14     switch (key) {
15       case 'volume':
16         mixer.volume = val;
17         break;
18       case 'triggerSynth':
19         synth.trigger();
20         break;
21     }
22   }
23 });
24 // later (or from any other attached node)
25 playerState.set({ volume: -6 });

```

Figure 8: Pseudo-code example of the main aspects of soundworks state manager API.

- The possibility to reflect on the schemas' declarations to generate controls and monitoring interfaces, simplifying the implementation of dynamic and complex interfaces.

3.4 Distribution: Core and Services

A final aspect that we want to present is the novel approach for the packaging and the distribution of soundworks. The framework is now distributed behind its own npm organization namespace: @soundworks¹⁶. Furthermore the core of the framework and the different services have been decoupled.¹⁷ As such, services are now imported in the application as plugins that must be registered in the ServiceManager.

We think this modular approach will facilitate future evolutions of the codebase, as well as maintenance of existing applications. Furthermore, this strategy should help to simplify the design and development of new components as well as their testing and documentation.

4. A *TODO(Noise)* APPLICATION FOR DISTRIBUTED AUDIO FRAMEWORKS

In this section we describe a simple application, inspired by the *TodoMVC* project¹⁸, that aims at providing a common basis to test and compare frameworks dedicated at building distributed audio applications. We first present the motivations and features of the application and, second, describe elements of its implementation within soundworks.

4.1 User Story

The proposed application purposely privileges the point of view of a user in a working situation (i.e. developer, designer, composer or performer) rather than the point of view of the end user (e.g. participant, audience). Indeed, while the later tends to be very application or artwork specific, we have shown in Section 2 that the former embodies common properties—the need for remote moni-

toring and control of the distributed state of the application—that can be reduced to simple features.

To illustrate these features, we have designed a basic application composed of two different clients. The first client, we call *player*, can be envisioned as the client dedicated to the end users. The application can accept any number of *players*. Each *player* has access to the following functionalities:

- can trigger a sound
- can start and stop a synthesizer
- can update a parameter (i.e. volume)

The second client, we call *controller*, is dedicated to the user in working situation (e.g. design, composition, research, performance). The application can accept any number of *controller*. A *controller* can:

- control global parameters of the application (i.e. mute, master volume)
- take control over each *player* (i.e. volume, trigger and state of the synthesizer)

Globals parameters of the application (i.e. *mute* and *master*) must stay synchronized across every clients of the application (i.e. *player* and *controller*).

We think this minimal set of functionalities provides a good reduction of important and recurring aspects of distributed audio applications. We also believe that it could, after eventual refinements, provide a good basis for testing, demonstrating and compare different frameworks and approaches.

4.2 Elements of Implementation

We implemented this application using soundworks (see Figure 9).¹⁹ The experience showed possible to implement all the specified features relying on the state management system described in 3.3, allowing to focus on application logic rather than on network communications and routing. This single fact tends to validate the addition of this component to the core of soundworks.



Figure 9: Interface of the (a.) controller and (b.) player clients of the *Todo(Noise)* application. Here, the controller duplicates the interface of the player with id 32, allowing for remote monitoring and control of this particular client.

The application is composed of only two schemas:

- The *globals* schema contains the list of connected *player* ids, the id of the remote controlled user (if any), the values of the *mute* and *master* volume parameters.
- The *player* schema contains the current value of the *player*'s local volume, the state of the synth (*started* or *stopped*) and a volatile event dedicated at triggering a sound.

¹⁶ <https://www.npmjs.com/org/soundworks>

¹⁷ <https://github.com/collective-soundworks/soundworks>

¹⁸ <http://todomvc.com/>

¹⁹ <https://github.com/collective-soundworks/soundworks-todo-noise>

The main logic of the application is implemented in the controller client. Indeed this client (cf. Listing 10), in its subscription to the `globals` state, observes the value of the `remoteControlled` parameter and attach to the state of the corresponding player. When attached to the `player` state, the controller simply instantiate the `player`'s GUI to locally create a remote and synchronized monitor and control interface.

```

1 // src/client/controller/ControllerExperience.js
2 this.globals.subscribe(async (updates) => {
3   for (let [key, val] in Object.entries(updates)) {
4     if (key === 'remoteControlled') {
5       const playerId = val;
6       // detach from previous player
7       if (this.playerState) {
8         await this.playerState.detach();
9         this.playerState = null;
10      }
11      // attach to new remote player
12      if (playerId !== null) {
13        this.playerState = await stateManager.attach(
14          'player', playerId);
15        // keep GUI synced with player state
16        this.playerState.subscribe(this.render);
17        // handle disconnection
18        this.playerState.onDetach(() =>
19          this.playerState = null);
20      }
21    }
22  }
23  this.render();
24 });

```

Figure 10: Main pseudo-code logic written in the controller to remotely monitor and control any player of the application.

5. CONCLUSION AND FUTURE WORKS

In this paper, we have presented the motivations, design and implementation of a novel version of `soundworks`, a framework dedicated at developing distributed multimedia applications on the web. First, we have presented three applications implemented or refined in the last year and discussed some of the recurring difficulties—centered on the point of view of the user in working situation—that the current version of the framework failed to properly address. Second, we have presented the novel architecture as well as a new component dedicated to distributed state management and designed to address these recurring issues. Finally, we have described a simple application—designed on the model of the *TodoMVC* project—that summarizes recurring aspects of distributed audio applications, and some elements of its implementation within the new version of `soundworks`.

While we think this new version provides solid foundations to further explore the possibilities of the web platform for Network Music Systems, it also opens large areas of new developments. First, a cli tool for scaffolding applications would be an important addition. Second, the integration of NodeJs clients in the core of the framework should simplify testing and thus help to stabilize the framework. Third, and more important, it opens many paths for creating a more dynamic working environment, facilitating the inclusion of users with different backgrounds (e.g. artists, researchers) and transdisciplinary approaches.

6. ACKNOWLEDGEMENTS

The presented work has been initiated in the *CoSiMa* research project funded by the french National Research Agency (ANR, ANR-13-CORD- 0010) and further developed in the framework of

the *Rapid-Mix* Project from the European Union's *Horizon 2020 research and innovation programme* (H2020-ICT-2014-1, Project ID 644862). It has also been supported by the Ircam project *BeCoMe*, which is featured in the *Constella(c)tions* residency of the STARTS program of the European Commission.

We would like to thank our projects partners and our colleagues at IRCAM for their precious contributions to the project.

7. REFERENCES

- [1] J. Allison, Y. Oh, and B. Taylor. NEXUS: Collaborative Performance for the Masses, Handling Instrument Interface Distribution through the Web. In *Proceedings of the NIME'13 Conference*, Daejeon, Seoul, Korea, 2013.
- [2] A. Barbosa. Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation. *Leonardo Music Journal*, 13, Dec. 2003.
- [3] J. Bischoff, R. Gold, and J. Horton. Music for an Interactive Network of Microcomputers. *Computer Music Journal*, 2(3), 1978.
- [4] B. Matuszewski and F. Bevilacqua. Toward a Web of Audio Things. In *Proceedings of the 15th Sound and Music Computing Conference*, Limassol, Cyprus, 2018.
- [5] B. Matuszewski, J. Larralde, and F. Bevilacqua. Designing Movement Driven Audio Applications Using a Web-Based Interactive Machine Learning Toolkit. In *Proceedings of the 4th Web Audio Conference*, Berlin, Germany, 2018.
- [6] B. Matuszewski, N. Schnell, and F. Bevilacqua. Interaction Topologies in Mobile-Based Situated Networked Music Systems. *Wireless Communications and Mobile Computing*, 2019, Mar. 2019.
- [7] S. Piquemal. Rhizome. <https://github.com/sebpiq/rhizome>. Accessed: 2019-06-24.
- [8] I. Poupyrev, M. J. Lyons, S. Fels, and T. Blaine. New Interfaces for Musical Expression. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, 2001.
- [9] M. Puckette. The Patcher. In *Proceedings of the International Computer Music Conference*, 1988.
- [10] M. Puckette. Combining Event and Signal Processing in the MAX Graphical Programming Environment. *Computer Music Journal*, 15(3), 1991.
- [11] S. Robaszkiewicz and N. Schnell. Soundworks – a playground for artists and developers to create collaborative mobile web performances. In *Proceedings of the 1st Web Audio Conference*, 2015.
- [12] M. Schwab. *Experimental Systems: Future Knowledge in Artistic Research*. Orpheus Institute series. Leuven University Press, 2013.
- [13] B. Taylor. A History of the Audience as a Speaker Array. In *Proceedings of the NIME'17 Conference*, 2017.
- [14] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. Internet of Musical Things: Vision and Challenges. *IEEE Access*, 6, 2018.
- [15] G. Weinberg. Interconnected Musical Networks: Toward a Theoretical Framework. *Computer Music Journal*, 29(2), June 2005.
- [16] M. Weiser. The Computer for the 21 st Century. *Scientific american*, 265(3), 1991.
- [17] L. Wyse and S. Subramanian. The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4), 2013.

FAUST online IDE: dynamically compile and publish FAUST code as WebAudio Plugins

Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Fober
GRAME, 11 cours de Verdun LYON
renshihong@hotmail.com
(letz, orlarey, michon, fober)@grame.fr

Michel Buffa, ElMehdi Ammari, Jérôme Lebrun
Université Côte d'Azur
CNRS, INRIA
(buffa, lebrun)@i3s.unice.fr,
ammarielmehdi@gmail.com

ABSTRACT

The development and porting of virtual instruments or audio effects on the Web platform is a hot topic. Several initiatives are emerging, from business enterprise based ones (Propellerhead Rack Extension running on the Web¹), to more community based open-source projects [10]. Most of them aim to facilitate adapting existing code base (usually developed in native languages like C/C++) as well as facilitating the use of existing audio DSP languages and platforms.

Our group previously presented an open format for WebAudio Plugins named WAP [11]. It aims to facilitate the interoperability of audio/MIDI plugins developed either using pure Web APIs, porting existing native code bases, or using Domain Specific Languages (DSL).

In the DSL category, we already did developments to use the FAUST audio DSP language. In this paper, we present a solution based around FAUST, its redesigned Web based editor, and the integration of a plugin GUI editor allowing to directly test, generate and deploy WAP plugins.

Recent improvements done in the toolchain, going from the DSP source to a ready-to-use WAP compatible plugin will be presented. The complete workflow, from the Faust DSP source written and tested in a fully functional editor, to a self-contained plugin running in a separate host application, will be demonstrated.

1. INTRODUCTION

There are many ways to develop software with the WebAudio API today. In pure JavaScript, the genish.js environment for instance [13] allows to develop sample level audio processing techniques². Already C/C++ written code can be transpiled to WebAssembly using Emscripten [12], or by using domain specific languages for programming DSP algorithms that also compile to WebAssembly, like the mature Csound [14] with its set of WebAudio examples³, or the recently announced SOUL DSP

language with its playground⁴. They all provide a dedicated and usually self-contained working environment.

When audio effects or audio/MIDI instruments have to be shared between several DAWs or audio environments, a plugin model is usually preferred.

Several native audio plugin formats are now popular, including Steinberg's VST format (Virtual Studio Technology, created in 1997 by Cubase creators), Apple's Audio Units format (Logic Audio, GarageBand), Avid's AAX format (ProTools creators) and the LV2 format from the Linux audio community. Although the APIs offered by these formats are different, they all exist to achieve more or less the same thing: to represent an instrument or an audio effect, and to allow its loading by a host application. In the first years after the birth of the WebAudio standard (2011), there was no standard format for high-level audio plugins. With the emergence of Web-based audio software such as digital audio workstations (DAWs) developed by companies such as SoundTrap, BandLab or AmpedStudio, it was desirable to have a standard to make WebAudio instruments and effects interoperable as plugins compatible with these DAWs and more generally with any compatible host software.



Figure 1: the virtual pedalboard host application scans multiple remote WAP plugin servers. WAP plugins can then be dragged and dropped and assembled in a graph.

Such a plugin standard needs to be flexible enough to support these different approaches, including the use of a variety of programming languages. New features made possible by the very nature of the Web platform (e.g., plugins can be remote or local

¹ <https://www.reasonstudios.com/press/275-reasons-flagship-europa-synth-now-available-as-a-plugin-for-other-days-and-on-the-web>

² <http://www.charlie-roberts.com/genish/>

³ <https://waaw.csound.com>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

⁴ <https://soul.dev>

and identified by URIs) should also be available for plugins written in different ways. To this end, some initiatives have been proposed [3, 9] and with other groups of researchers and developers we made in 2018 a proposal for a WebAudio plugins standard called WAP (WebAudio Plugins), which includes an API specification, an SDK, online plugin validation tools, and a series of plugin examples written in JavaScript but also with other languages⁵.

These examples serve as proof of concept for developers and also illustrate the power of the Web platform: plugins can be discovered from remote repositories, dynamically uploaded to a host WebApp and instantiated, connected together etc. The project includes examples of very simple plugins and host software, but also more ambitious software to validate the WAP standard: a virtual guitar "pedalboard" that discovers plugins from several remote repositories, and allows the musician to chain for example virtual audio effects pedal plugins, synthesizers, guitar amplifier simulators, drum machines etc. and to control them via MIDI in real time (Figure 1). The reader can get a "multimedia" idea of this work by watching online videos that present the results of this work⁶. Since last year, WAP now includes support for pure MIDI plugins (a GM midi synthesizer, virtual midi keyboards, a MIDI event monitoring plugin, etc⁷). For details about the WAP proposal, and how it is related to other approaches like Web Audio Modules (WAMs), WebAudio API eXtension (WAAX) or JavaScript Audio Plugin (JSAP), see [11].

In the next sections we will focus on a new online IDE we developed, that is well suited for coding, testing, publishing WAP plugins written in FAUST, directly in a Web browser. The IDE includes a graphical interface editor that allows developers to fine-tune the look and feel of the plugins. This editor offers a rich set of widgets that can be controlled by midi-learn. Once complete (DSP + GUI) the plugins are packaged in the form of standard W3C WebComponents and published on remote WAP plugin servers. The plugins will then be directly usable by any compatible host software, using their URIs. You can imagine WAP plugins as images in an HTML document, their URI is sufficient, and can be dynamically retrieved using APIs from a remote Web Service.

2. BACKGROUND CONTEXT AND TERMS

FAUST [6] is a functional, synchronous, domain specific programming language designed for real time audio signal processing and synthesis.

The FAUST compiler is organized in successive stages, from the DSP block diagram to signals, and finally to the FIR (FAUST Imperative Representation) which is then translated into several target languages. The FIR language describes the computation performed on the audio samples in a generic manner. It contains primitives to read and write variables and arrays, do arithmetic operations, and defines the necessary control structures (for and while loops, if structure etc.).

As a specification language, the FAUST code only describes the DSP part, and an abstract version of the control interface. It says nothing about the audio drivers or the GUI toolkit to be used. Architecture files are written to describe how to connect the DSP code to the external world.

⁵ <https://github.com/micbuffa/WebAudioPlugins>

⁶ <https://www.youtube.com/watch?v=pe8zg8O-BFs>

⁷ See the midi folder in the github repository of the WAP SDK, video <https://www.youtube.com/watch?v=jHfK3YxcjQ>

Additional generic code is added to connect the DSP computation itself with audio inputs/outputs, and with parameter controllers, which could be buttons, sliders, numerical entries etc. Architectures files can also possibly implement polyphonic support for MIDI controllable instruments, by automatically dealing with dynamic voice allocation, and decoding and mapping of incoming MIDI events [15].

Several prior developments have been done to use the language on the Web platform. By adding an asm.js generating backend in the compiler, and compiling the compiler itself in asm.js/JavaScript using the Emscripten transpiler, the dynamic generation of WebAudio nodes from FAUST code has been demonstrated [16] [17].

With the apparition of the more stable and better designed WebAssembly format in 2017, as a replacement of asm.js, the previous work done with asm.js has been adapted. For the Web platform, two backends have been developed to generate WebAssembly text (so-called "wast" or "wat") and binary formats (so-called "wasm") [7]. When embedded in the FAUST compiler running on the Web, they allow to dynamically compile FAUST DSP programs in pure Web applications. Additional JavaScript glue code is added to transform DSP modules in fully functional WebAudio nodes.

FAUST also allows to circumvent some important buffer size issues that we encountered in our previous works on the implementation of signal loops in WebAudio. For example, the Negative Feedback Loop (NFB) as in our push-pull tube amps simulations [1, 2] is a tricky issue due to some WebAudio API limitations and divergences/bugs in how browsers generally parse the WebAudio graphs with loops. In the WebAudio API specs, loops in the graph are required to include at least a delay node. Without this delay node, Firefox stops rendering the graph, while Chrome does not complain but adds, behind the scenes, a 3 ms delay (minimum size of an audio buffer is 128 frames hence a minimal delay of 128/sampling rate or roughly 3 ms at 44.1kHz). Now, to faithfully implement loops like the NFB with its RC network inducing short delays, finer precision at the level of some samples is required. With the current limitations, and quite strangely, a 3ms delay in the loop to conform to the specs, was bringing slightly different coloring of the amps between FF and Chrome. This example shows the need for solutions such as FAUST to circumvent these limitations of the WebAudio standard.

3. CURRENT STATE

3.1 The new FAUST Web editor

The Emscripten module was previously implemented in the FAUST IDE using a JavaScript wrapper which allowed the application to compile and transform FAUST source code into a WebAudio node. We recently restructured this wrapper, in order to take advantage of modern JavaScript development environments. An updated tool-chain is now used to ensure the efficiency and the compatibility of the wrapper to transform it into another JavaScript UMD module.

The past versions of this wrapper already provided the following features:

- Load WebAssembly version of the FAUST compiler and import its C functions into JavaScript

- Compile the code: the input is the FAUST source code, the output is the compiled WebAssembly binary version with some related data
- Load and wrap the module as a DSP processing function inside an *AudioWorkletProcessor* or a *ScriptProcessor* *AudioNode*

We added some new features into the module:

- A virtual file system: Emscripten supports a virtual file system (in memory) compatible with the C++ I/O standard library, but also usable from the JavaScript wrapper. This file system became important as the FAUST compiler searches libraries and imported source codes locally, or generates DSPs code for other targets/architectures. For instance SVG diagrams generated as additional files in the compilation process, are simply written on the fly in the VFS, then loaded, decoded and displayed.
- Data output: a callback has been added into the *AudioWorklet* node to support additional processing or analysis after the buffer has been fully calculated. This callback returns the current output buffer, the current buffer index and parameters change events. In addition, to be able to calculate audio separately with a FAUST DSP independent from the browser audio context, we created an “offline processor” which will be used exclusively for getting the very first samples calculated by a DSP. This allows us to debug the DSP code running with a different sample rate.

Based on the previous FAUST online editor, we built a code editor (Figure 2) with full IDE user experience that could provide more information and details of a DSP through graphical representation in a Web page. A DSP developer probably not only needs to hear how the DSP sounds, but also to test it with other audio inputs, or to precisely display the time domain and frequency domain data of outputs. We have added several testing, visualisation and debugging tools into a basic code editor.

The layout is responsive and configurable following the browser viewport dimensions:

- All options related to FAUST code compilation are located using controllers from the left sidebar panel
- All options and displays related to DSP runtime, such as MIDI, audio inputs and quick signal probing are placed in the right sidebar panel
- The remaining central region of the page is divided into two parts with configurable heights: a source code editor on the top and a multi-tab display panel which can display the logs from the compiler, a FAUST block diagram corresponding to the DSP code, a larger signal scope, a running GUI of the plugin being developed, and finally a GUI Builder / exporter for designing the user interface a WAP plugin version of the code, usable in external host applications

Besides UI improvements that facilitates code editing and compiling, audio probes are an important addition to the new editor. We designed four modes of signal visualisations: data table, oscilloscope (stacked and interleaved by channels), spectroscope and spectrogram, to help FAUST users to debug their DSPs. To implement all four probe modes, precise sample

values are needed. In the browser environment, we have two ways to get audio output samples.

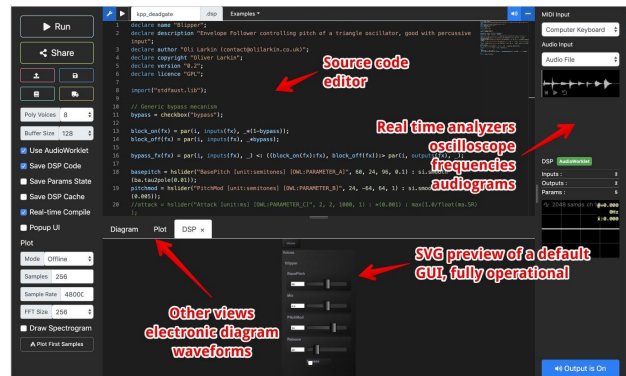


Figure 2: the FAUST IDE provides many embedded tools: oscilloscopes, spectroscope and spectrogram, functional default GUI, schema preview, etc.

The first method consists of using WebAudio *AnalyserNode* with its integrated methods:

```
getByteFrequencyData, getByteTimeDomainData,
getFloatFrequencyData,
getFloatTimeDomainData (which does not exist in Safari)
```

This method provides both sample values and a spectrum given by the FFT of the current audio buffer. However, it has several drawbacks. Firstly, the *AnalyserNode* has only one input, which means it needs an additional *ChannelSplitterNode* to retrieve the correct channel from the FAUST DSP Node. Secondly, as we cannot tell when the *AnalyserNode* does an analysis, the audio data are provided only on demand. Thus it is impossible to get precise data in a specific buffer calculated by the FAUST DSP.

The second method consists of getting the sample values directly with a callback in a FAUST DSP *AudioNode*. These values are associated with its buffer index and an event list containing all parameter changes occurring in this buffer. To get the corresponding frequency domain data, an additional FFT is required. We chose the JavaScript version of KissFFT⁸ for its high performance when compiled to WebAssembly⁹. Thus, we perform the FFT computation in the FAUST online editor with 2 overlaps using a Blackman window function.

The first method is used in the implementation of the two scopes in right sidebar as it can also probe the audio input. The second method is used for the larger scope at the bottom. It is more flexible and can adapt itself to continuous or on-demand signal display.

Developers may need to have options to select which part of the signals they want to display: we provide four modes to trigger differently the drawing function of the scopes: *Offline*, *Continuous*, *On Event* and *Manual*:

⁸ <https://github.com/j-funk/kissfft-js>
⁹ <https://github.com/j-funk/js-dsp-test/>

- *Offline*: FAUST WebAudio wrapper offers an “offline processor” which is useful to allow a DSP to calculate the first samples at any sample rate independently of the actual audio context one.
- *Continuous*: similar to normal audio scopes, this mode draws in real time the most recent samples processed by the FAUST DSP. Parameter change events will be shown in the scope. On a mainstream personal computer, the editor is able to draw up to 1 million samples continuously without significant rendering lagging.
- *On Event*: as the FAUST DSP usually comes with a GUI to control its parameters, it is important to visualize the part of signals while parameters change. In this mode, the scope draws only when it captures parameter change events, which is useful for debugging.
- *Manual*: in Manual mode, the scope displays the latest samples when a user clicks on a button.

After a FAUST DSP is tested in the editor, users can export the DSP to different architectures including WebAudio Plugins (WAPs). A dedicated GUI builder is integrated in the online IDE that receives FAUST DSP’s GUI definitions while it is compiled. Then, a default GUI is proposed and users can start customizing the GUI, testing the plugin functionalities, and finally export the plugin to a remote server. This is detailed in the next section.

3.2 The GUI Editor

FAUST code can include abstract definitions of GUI controllers, such as in this source code extract:

```
basepitch = hslider("BasePitch
[unit:semitones]", 60, 24, 96, 0.1) :
si.smooth(ba.tau2pole(0.01));

pitchmod = hslider("PitchMod
[unit:semitones]", default_pitch*2, -64, 64,
1) : si.smooth(ba.tau2pole(0.005));
```

Here, the code describes the definition of two parameters named “BasePitch” and “PitchMod” along with some data that define the default value, min, max, step, unit type, etc. These parameters can be programmatically set/computed such as “default_pitch*2” in the second example, instead of using literal values.

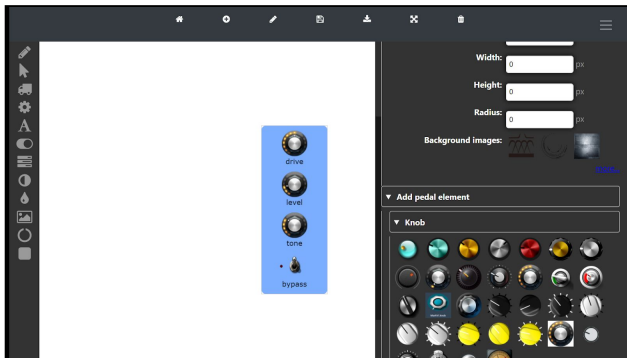


Figure 3: from the FAUST code, a WAP default GUI is proposed in the editor.

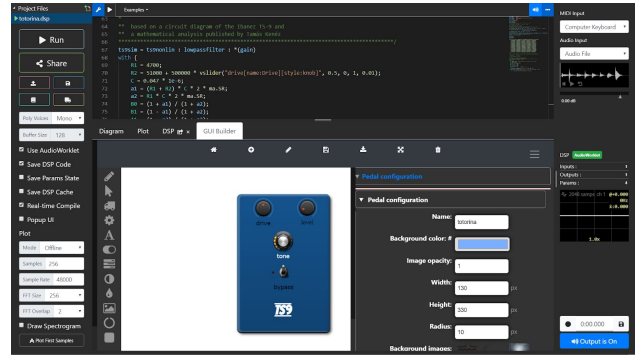


Figure 4: The default GUI can be customized: change textures, knobs, sliders, switches positions size, appearance and labels etc.

As explained in the previous sections, the FAUST DSP code is compiled to a JavaScript wrapper and a WebAssembly module. This is all done client-side. The GUI builder shares a JavaScript parameter descriptor variable that has been generated after the compilation step and that can be statically interpreted. From this parameter descriptor, a “GUI pivot descriptor” is created and a “default GUI” displayed in the GUI builder (Figure 3), that can be enriched during the GUI edition process and that will be used to generate the final GUI code (Figure 4). So far, we implemented only a generator for WebAudio plugins, using HTML/CSS/JS code that follows the W3C WebComponents specifications¹⁰.



Figure 5: Other designs for the same DSP code

At any time, the plugin (DSP + GUI) can be tested from within the IDE, without the need to download it on a local disk. It is then possible to refine the GUI, adjust the layout, appearance of the controllers among a rich set of knobs, sliders, switches (Figure 5 shows different looks and feels that can be created from the same DSP code). The editor is not yet 100% bijective with the FAUST definition of GUI controllers (that serve as a “hint” to bootstrap the GUI design process). For example, if you change the type of controller (i.e. slider to knob), it does not change the FAUST code back. However, having a way to build and customize a GUI this way is a great time saver, full sync between the FAUST code and GUI is planned as future enhancements.

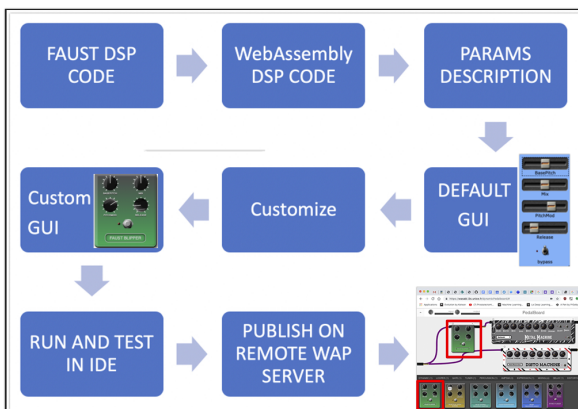
¹⁰ The WebComponents W3C standard (now in the HTML 5.2 specification) defines a way to easily distribute components with encapsulated HTML/CSS/JS/WASM code without namespace conflicts. See <https://www.webcomponents.org>

The plugin can be published on a remote plugin server, using standard Web services, this is shown in the life cycle workflow from Figure 6. A WAP plugin generated by the FAUST online IDE is a zipped archive file that contains the DSP WebAssembly module, the standard JSON WAP descriptor and the GUI code (HTML/CSS/JS) wrapped as a WebComponent. It also includes a host HTML page for trying and testing the plugin, making the plugin usable by humans as well as by client applications. In fact, once published on a server, this file is unzipped in a remote directory. The plugin is associated with a “remote URI” and can be “unit tested” by different validation tools that come with the WAP SDK¹¹ (i.e. check that their API is following the specification, that the plugin is able to save/restore its state etc.).

Figure 1 shows the “virtual pedalboard” Web application, a host for WAP plugins we developed to showcase the WAP standard, that targets guitar and keyboard players. This application scans remote WAP servers for available plugins and makes them accessible to final users that can drag and drop and assemble them in the main part of the screen. In this example, all virtual pedal effects at the bottom of the screen have been coded and compiled with their GUI designed and tested in the FAUST online IDE.

4. DISCUSSION

The authors’ short term plans are to complete and stabilize the presented workflow, to add support for polyphonic MIDI controllable instrument plugins, and to develop more features within the WAP GUI Builder that currently provides basic editing tools. The FAUST IDE itself needs to be extended to include sound file management¹², so that plugins using audio samples for instance could be implemented. To do that, we plan to expose more of the already C++ written architectures files on JavaScript side thanks to Emscripten¹³. This will also require to extend the FAUST remote compilation service. Deploying the resulting plugins in other host applications (like more traditional DAW running on the Web) should be straightforward if they comply with the WAP specification. Concluding tests have been conducted with the AmpedStudio DAW, for example.



¹¹ Normally, there should be no bad surprises as the FAUST workflow generates valid WAPs. Examples/demos of online validators can be tested online, see for example <https://jsbin.com/ieretab/edit?js,output>

¹² That is handling the language ‘soundfile’ primitive which requires to implement a proper audio resources loading architecture

¹³ C++ code using the libsndfile library can directly be compiled to WebAssembly and ported in JavaScript

Figure 6: workflow of the end-to-end design and implementation of a WebAudio plugin, from FAUST DSP code to a host application that uses the fully functional plugin with its GUI.

5. CONCLUSION

This paper presented the combined work of two teams deeply involved in the development of an audio DSP programming language and its complete ecosystem on the one hand, and the definition of a WebAudio plugin standard (WAP) and its complete surrounding environment on the other. Recent native to Web porting technologies like Emscripten and WebAssembly, as well as recognised Web standards (like WebComponents) have been heavily used. Combining client side and shared remote services is also part of the presented solution.

The complete workflow from the initial DSP source code, testing and running it in an integrated editor, polishing its user interface in another specialized GUI editor, to the finalized plugin running in an external host has been presented. Many examples of audio effects have been ported to WAPs directly by copying and pasting existing code from the Guitarix project¹⁴, from the OWL pedal project¹⁵, or from diverse open source projects, into the FAUST online IDE. Once compiled, the GUI has been customized within the GUI builder part of the IDE and published to remote WAP servers. Then, they can be tested online in the host web applications such as the pedalboard host presented in Figure 3¹⁶.

Having the authoring tools as well as the deployment platform as pure Web applications facilitates the workflow and interoperability of the components. We also think that the presented toolchain could be adapted to other plugin formats or audio DSP production tools.

6. ACKNOWLEDGMENTS

This work was supported by the French Research National Agency (ANR) and the WASABI team (contract ANR-16-CE23-0017-01).

7. REFERENCES

- [1] M. Buffa and J. Lebrun. 2017. “Real time tube guitar amplifier simulation using WebAudio”. In *Proc. 3rd Web Audio Conference (WAC 2017)*. London, UK.
- [2] M. Buffa and J. Lebrun. 2017. “Web Audio Guitar Tube Amplifier vs Native Simulations”. In *Proc. 3rd Web Audio Conference (WAC 2017)*. London, UK.
- [3] N. Jillings and al. 2017. “Intelligent audio plug-in framework for the Web Audio API”. In *Proc. 3rd Web Audio Conference (WAC 2017)*. London, UK.
- [4] M. Buffa, M. Demetrio, and N. Azria. 2016. “Guitar pedal board using WebAudio”. In *Proc. 2th Web Audio Conference (WAC 2016)*. Atlanta, USA.
- [5] M. Buffa and al. 2017. “WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications”. In *Proc. 3rd Web Audio Conference (WAC 2017)*. London, UK.
- [6] Y. Orlarey, D. Fober, and S. Letz. 2004. “Syntactical and Semantical aspects of Faust”. *Soft Computing* 8, 9 (2004). 623–632.

¹⁴ <https://guitarix.org/>

¹⁵ <https://www.rebeltech.org/product/owl-pedal/>

¹⁶ <https://wasabi.i3s.unice.fr/dynamicPedalboard/#>

- [7] S. Letz, Y. Orlarey, and D. Fober. 2018. “Faust Domain Specific Audio DSP Language Compiler to WebAssembly”. In *Companion Proc of the Web Conference, International World Wide Web Conferences Steering Committee, Lyon France 2018*.
- [8] H. Choi and J. Berger. 2013. “WAAX: Web Audio API eXtension”. In *Proc. International Conference on New Interfaces for Musical Expression (NIME’13)*. Daejeon, Korea.
- [9] Kleimola, J. and Larkin, O. 2015. “Web Audio modules”. In *Proc. 12th Sound and Music Computing Conference (SMC 2015)*, Maynooth, Ireland.
- [10] Larkin, O., Harker, A., and Kleimola J. “iPlug 2: Desktop Audio Plug-in Framework Meets Web Audio Modules”. *4th Web Audio Conference (WAC 2018)*, Berlin, Germany
- [11] Buffa, M., Lebrun, J., Kleimola, J., Larkin, O. and Letz, S. “Towards an open Web Audio plugin standard”. In *Companion Proceedings (Developer’s track) of the The Web Conference 2018 (WWW 2018)*, Mar 2018, Lyon, France. <hal-01721483>
- [12] A. Zakai, “Emscripten: an LLVM to JavaScript compiler”, In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications*, pages 301–312. ACM , 2011
- [13] Roberts, C. “Strategies for Per-Sample Processing of Audio Graphs in the Browser”. In *Proceedings of the Web Audio Conference (WAC 2017)*. London, UK.
- [14] Yi, Steven., Lazzarini, Victor., Costello, Ed “WebAssembly AudioWorklet Csound”. *4th Web Audio Conference, TU Berlin*. (WAC 2018). Berlin, Germany.
- [15] Letz, S., Orlarey, Y., Fober D., Michon R. “Polyphony, sample-accurate control and MIDI support for FAUST DSP using combinable architecture files” *In Proceedings of Linux Audio Conference*. (LAC 2017). St Etienne, France.
- [16] Denoux, S.,Orlarey, Y., Letz, S., Fober D. “Composing a Web of Audio Applications” *1th Web Audio Conference, IRCAM, Mozilla Foundation*. (WAC 2015). Paris, France.
- [17] Letz, S., Denoux, S., Orlarey, Y., Fober D. “Faust Audio DSP language on the Web” *In Proceedings of Linux Audio Conference*. (LAC 2015). Mainz, Germany.

An AudioWorklet-based Signal Engine for a Live Coding Language Ecosystem

Francisco Bernardo
Emute Lab, Dept. Music
University of Sussex
F.Bernardo@sussex.ac.uk

Chris Kiefer
Emute Lab, Dept. Music
University of Sussex
C.Kiefer@sussex.ac.uk

Thor Magnusson
Emute Lab, Dept. Music
University of Sussex
T.Magnusson@sussex.ac.uk

ABSTRACT

This paper reports on early advances in the design of an ecosystem for creating new live coding languages, optimal for audio synthesis, machine learning and machine listening. We present the design rationale and challenges when applying the Web Audio API, and in particular, an *AudioWorklet*-based solution to refactoring our digital signal processing library Maximilian.js for our high-performance signal synthesis engine. Furthermore, we contribute with a new system implementation, engineered for modern web applications, and for the live coding community to design their own idiosyncratic languages and interfaces applying our signal engine. The evaluation shows that the system runs with high reliability, efficiency and low latency.

1. INTRODUCTION

Since its introduction in 2011, the Web Audio API (WAAP) [18] has powered a substantial number of libraries and applications for interactive sound and music. Among WAAP-powered libraries, there have been varied approaches and features provided: from high-level control and abstractions over WAAP graphs and native nodes (e.g. Tone.js [12]), to abstractions with low-level digital signal processing (DSP) using optimised JS (e.g. Genish.js and Gibberish.js [14]), asm.js (e.g. Maximilian [6]) and WebAssembly (e.g. Faust [10], CSound [9]). Some of these libraries which initially implemented custom nodes using *ScriptProcessorNode*, with its inherent limitations (e.g. latency, main thread interrupts), have been shifting towards utilising *AudioWorklet* [3] (e.g. Faust, CSound, Genish.js). Such improvements have been used previously in advancing the design of live coding environments for the Web that enable custom low-level DSP and high-level control of musical processes [14], and that support end-user live coding language design [19].

Client-side Web-based machine learning middleware have shown recent and meaningful advances (e.g. Tensorflow.js [16], Magenta.js [13], ml5.js [15], RapidLib.js [7]). These libraries enable end-to-end client-side machine learning workflows with many benefits, including simplified deployment

with “zero install,” instant access across platforms, data privacy and, in the cases of Tensorflow.js -based libraries, hardware acceleration. Common to these libraries has been a strong consideration for developer-friendly experience design, particularly in relation to API abstraction level and learning resources, documentation and examples. There are also significant challenges involved, concerning varying client-side hardware and the progress of the web browser ecosystem.

We believe that the improvements in these Web technologies can afford the evolution of an ecosystem of real-time, user-defined, live coding languages that combine machine learning, machine listening and audio threads. Previous attempts to integrate real-time interactive audio and machine learning in a scalable and accessible environment such as the Web, have been fraught with technical limitations of the single-threaded JavaScript (JS) environment and the Web Audio API *ScriptProcessorNode* (SPN) [7]. We contribute a novel system design for live coding with a high-performance signal engine leveraging the *Maximilian* DSP library transpiled for *AudioWorklet*¹.

The paper is structured as follows: we first present the requirements and a proposal for a middleware architecture that integrates signal synthesis and processing, machine learning and live coding language design for client-side Web applications. Section 3 presents the elements of our design strategy and rationale for refactoring Maximilian [6] for an *AudioWorklet*-based signal engine. Section 4 describes the implementation and first use of an early signal engine prototype for a live coding performance. Section 5 discusses some of the main limitations for the audio worklet implementation and section 6 concludes with the main takeaways and future work.

2. INTEGRATED ARCHITECTURE OF SIGNAL SYNTHESIS AND MACHINE LEARNING, FOR A WEB-BASED LIVE CODING ECOSYSTEM

In 2018, we conducted a survey on language design with communities of practitioners of live coding, machine listening, machine learning and deep learning [8]. We asked participants which features they envisioned for future live coding environments and languages that could integrate machine listening and machine learning. The findings indicated a wide space of possibilities. Respondents suggested features



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹<https://github.com/mimic-sussex/Maximilian>

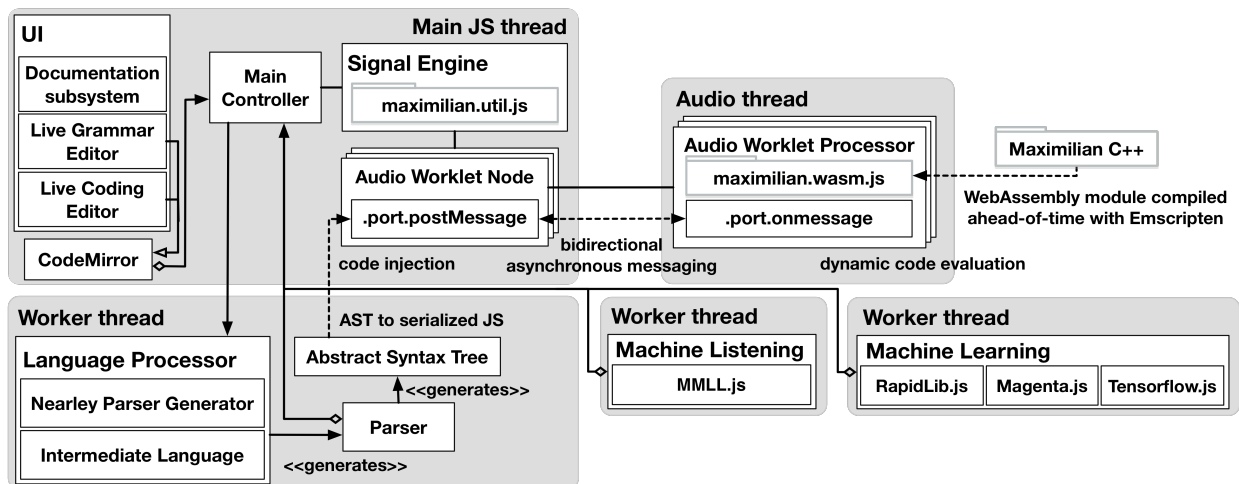


Figure 1: Diagram of the architecture of *Sema*, our live coding language design system.

such as support for hybrid approaches and multi-paradigm languages (i.e. object-oriented and functional), flexible, expressive and extensible languages and prototyping environments, good quality documentation and examples, as well as a clear and informative error report system. In a nutshell: brevity, simplicity, expressivity, flexibility, adaptability and plurality.

These findings prompted us to build *Sema*, a modern Web-based system (Figure 1) to support rapid prototyping of multiple mini-languages for live coding, enabling users to satisfy their idiosyncratic expressive preferences. It seemed wiser to make a system where the user creates or refines their favourite language, rather than trying to satisfy everyone with one general design. Considering also the history and tradition of live coders building their own systems [11], this decision would contribute to a plurality of systems in a field that is already brimming with inventive solutions. Our design exploration will help us find and refine the constraints, meta constraints and features for our system to support live coding language design.

2.1 Design requirements for a live coding system’s signal engine

In order to support a broad spectrum of live coding languages that ranges from abstract high-level languages to more powerful low-level languages, we strive for an adequate compromise between simplicity and flexibility. The following are the design principles that guided the implementation of our signal engine:

Integrated signal engine There is no conceptual split between the intermediate language and signal engine. Everything is a signal.

Single-sample signal processing Per-sample sound processing affords a broad set of sound control and synthesis processes, including more sophisticated and expressive techniques that use feedback loops, such as physical modelling, reverberation and infinite impulse response filtering [17].

Sample rate transduction It is simpler to do signal processing with one principal sample rate, i.e. the audio

rate. Different sample rate requirements of dependent objects can be resolved by upsampling and downsampling, using a ‘transducer’. The ‘transducer’ analogy enables us to accommodate a variety of processes with varying sample rates (e.g. video, spectral rate, sensors, ML model inference) within a single engine.

Minimal abstractions There are no high-level abstractions such as buses, synths, nodes, servers, or any language scaffolding in our signal engine. Such abstractions sit within the end-user mini-language design space.

We prioritise usability and learnability in the design of our signal engine and intermediate language. As such, some of these design principles prioritise usability over efficiency (e.g. single-sample processing [20]). Nevertheless, we are balancing out any performance trade-offs with a highly efficient implementation (as demonstrated in section 4).

2.2 ML and MML workers

We are designing machine learning (ML) and musical machine listening² (MML) as first-class citizens and core components of our system. Typically, both ML and MML processes have computationally intensive stages that are better suited for execution on a dedicated thread, or eventually, in an external process. Even with interactive machine learning workflows [2], where end-users build custom ML models from small, lightweight, user-created data sets, having the main JS thread freeze while an ML algorithm trains a model, causes critical usability issues and undermines the user experience of any application [1]. As such, ML and MML processes in our system follow a loosely coupled architecture based on JS workers³: threads which leverage multi-core CPUs and communicate using asynchronous message exchange and event streams with the main thread. These processes also adhere to our transducer concept, in that the sample rates from the event streams they generate are converted to and from the sample rate of the audio context.

²<https://mimicproject.com/guides/mml>

³<https://w3c.github.io/workers/>

2.3 Live code and grammar editors

After some experimentation integrating editor components such as Ace,⁴ Monaco⁵ and CodeMirror,⁶ we opted for the latter. Our choice considered multiple criteria, such as component architecture, community adoption, maintenance and support, and ease of integration with webpack⁷. CodeMirror powers two editor instances in our web-based live coding environment, which consist of the main user-facing components—the live coding and live grammar editors. A responsive live coding editor provides the user with both manual and continuous evaluation. The grammar editor enables users to specify, inspect and customise different mini-language grammar specification. Both editors will integrate with an interactive documentation subsystem comprising conceptual knowledge guides, tutorials and examples.

2.4 Language processor and intermediate language

In order to parse, validate and evaluate instances of a live coding language from textual input in the editors, we need to be able to define its formal grammar. We opted for Nearley.js, a toolkit and library that implements the Earley [5] algorithm to generate JS parsers from formal grammar specification in the Backus-Naur Form (BNF). Nearley allows a broader set of grammars than parsing expression grammars, a formalism which has been previously used in live coding language design (e.g. PEG.js [19] and Ohm-js⁸), including ambiguous grammars with left-side recursion.

The trade-off for the versatility and flexibility of Nearley is performance. Even with the integration of a tokeniser for higher parsing performance, benchmarks measured across browsers on a large JavaScript Object Notation (JSON) file,⁹ show that Nearley scores below domain specific language (DSL) and hand-written parser implementations, parsing libraries, and other parser generators. Nevertheless, preliminary tests that we present later in this paper show that its performance is reasonable for the latency requirements of live coding in which, typically, small files and code snippets are evaluated.

3. DESIGN STRATEGY: REFACTORING MAXIMILIAN FOR AUDIO WORKLET

At the core of our signal engine is Maximilian [7]. It is a free, open source and MIT-licensed audio synthesis and signal processing library implemented in C++. Maximilian was designed as a cross-platform and easy-to-use library for artists and creative computing students, and for rapid prototyping and commercial software development of interactive audio applications. The library provides a high-level interface to a comprehensive set of DSP primitives, including oscillators with standard waveforms, envelopes, filters with resonance, sample playback, delay lines, Fast Fourier Transform (FFT), granular synthesis, feature extraction, and multi-channel support.

Previously, Maximilian was transpiled to an asm.js li-

brary using Emscripten [21].¹⁰ The output consisted of single file with asm.js appended with custom JS—e.g. a pre-assembled WAAPI graph with a user-definable closure injected at the audio callback of a SPN, JS TypedArray conversion to emscripten native types and asynchronous sample loading. This library provided a simple framework and abstraction over the WAAPI graphs and nodes. Most usefully, the closure bound to the scope of the SPN audio callback provided users with a sandbox for accessing the audio buffer directly and implementing custom signal processing for interactive audio applications with Maximilian DSP classes.

While this solution revealed useful to end-users, particularly for teaching students about DSP and audio with the accessibility and convenience of a Web environment [7], it suffered from the well-known limitations of SPN [3]. In earlier studies [1], we assessed users integrating Maximilian with machine learning and different data sources for rapid prototyping interactive audio applications. The main thread interrupts at the ML-training stage resulted in audio glitches and freezes that frustrated users. These usability issues were considered critical and would clearly undermine the adoption of these tools.

In order to overcome the SPN issues, and to arrive at a more scalable, efficient and usable design for a signal engine, we decided to advance towards an AudioWorklet-based implementation, and to refactor Maximilian.js accordingly.

3.1 WebAssembly generation with Emscripten

Refactoring Maximilian.js for *AudioWorklet* implied changing the build process to generate Wasm instead of asm.js. Compiling Maximilian to Wasm caused a breaking change that rendered the Maximilian DSP types unavailable in the SPN callback at load time, due to the asynchronous Wasm module instantiation. To ensure retro-compatibility to Maximilian.js SPN users, we restructured the build process to provide conditional builds of the library. The SPN version is modularised, exports with library name, and provides embindings with smart-pointer constructors for automated memory management and garbage collection of the DSP types. The Wasm version of the module was trimmed down and now only contains the embindings for loading the DSP classes in the AudioWorkletProcessor (AWP) scope with normal constructors for finer control over object disposal. Listing 1 shows an excerpt of the embinding for *maxiOsc*, Maximilian’s native type for standard wavetable oscillators.

Listing 1: Excerpt of Maximilian maxiOsc embinding

```
class_<maxiOsc>("maxiOsc")
#ifdef SPN
    .smart_ptr_constructor("shared_ptr<
        maxiOsc>", &std::make_shared<maxiOsc>())
#else
    .constructor<>()
#endif
.function("sinewave", &maxiOsc::sinewave)
.function("saw", &maxiOsc::saw)
```

The design pattern¹¹ for *AudioWorklet* with WebAssembly provides a restricted set of Emscripten compilation flags.

¹⁰<https://gitlab.doc.gold.ac.uk/mick/maxi-js-emsripten>

¹¹<https://developers.google.com/web/updates/2018/06/audio-worklet-design-pattern>

⁴Ace Editor, <https://ace.c9.io/>

⁵Monaco Editor, <https://microsoft.github.io/monaco-editor>

⁶CodeMirror, <https://codemirror.net>

⁷Webpack, <https://webpack.js.org/>

⁸Ohm-js, <https://github.com/harc/ohm>

⁹<https://sap.github.io/chevrotain/performance/>

We found that some of the previously used compilation flags, such as `MODULARIZE` and `EXPORT_NAME` prevented the WASM module to load successfully and were removed.

```
CFLAGS--bind -O1 -s WASM=1 -s SINGLE_FILE=1
-s BINARYEN_ASYNC_COMPILATION=0 \
-s ABORTING_MALLOC=0 -s ALLOW_MEMORY_GROWTH=1 \
-s TOTAL_MEMORY=128M
```

To support the strategy of dynamic code evaluation, we found that it was necessary to add a compilation option to control the Wasm heap memory expansion. We will discuss this further in Section 5.

3.2 Audio worklet code injection

Upon the introduction of *AudioWorklet*, a few community examples emerged that use dynamic code injection (e.g. *dsp.audio*'s Worklet Editor¹² and A. Carabott's Live Coding Playground¹³). These examples de-serialise worklet processor templates in vanilla JS using Blob, manipulate them, and hot-swap them from memory into the worklet as interchangeable modules. Our first attempts explored this pattern using code generation and the additional constraint of dynamically loading the Maximilian Wasm module into the AWP scope. This constraint introduced a series of difficulties revealed by the “*Error on loading worklet: DOMException*” error message. The ambiguity of the error message and the constraints of the AWP scope caused a challenging debugging process.

To solve this problem, we considered tactics such as dynamic imports, fetch, and *wasm-through-‘message port’*, as suggested in WAAPI discussion forum. We found out that the AWP scope excluded some of the APIs that enable the first alternatives. We also found out through StackOverflow¹⁴ that the opaque origin of the worklet processor requires absolute URIs to load the module asynchronously. This unveiled another problem related to the unsuccessful parsing of the Wasm module when it was imported by processor code that was generated and loaded dynamically from a Blob. Eventually, we turned to a more creative solution.

3.3 Parsing, evaluation and execution of user-defined code

The broader design of our livecoding system will be discussed in a forthcoming paper. Here we present the solution that we have arrived at for supporting the evaluation of user-defined DSP code. Fundamentally, it consists in utilising the worklet processor as a template that imports the Maximilian Wasm module and loads its DSP classes into the AWP scope. The AWP is where the DSP code is injected and dynamically evaluated.

In our system, when users evaluate an expression in the live coding editor—i.e. by pressing Cmd-Enter after selecting an expression or placing the cursor on a given line in the editor—they are triggering a specific workflow in our system (Figure 1). First, the user-evaluated expression is parsed by the Nearley-generated parser. If the expression is valid according to the language formally defined by the BNF grammar specification, the parser outputs an Abstract Syntax Tree (AST), a tree-like data structure that breaks

down the user expression. The AST is converted into serialised JS expressions that specify which Maximilian DSP objects are used and how they are assembled into DSP functions that will run on the main audio loop *setupFunction* and *loopFunction*, respectively. These JS expressions are packed into a JS object which is posted through the processor *AudioWorkletNode* messaging port (Listing 2).

Listing 2: Code injection into the AWP scope

```
this.audioWorkletNode.port.postMessage({
  eval: 1,
  setup: userInterpretedFunction.setup,
  loop: userInterpretedFunction.loop
});
```

Once the message containing a payload of serialised DSP JS resolves asynchronously in AWP scope, the code is evaluated just-in-time (JIT) using the `eval()` function. The user-evaluated expression that was converted to a JS DSP specification with Maximilian objects is compiled and set ready to run in the AWP loop (Listing 3).

Listing 3: Dynamic code evaluation in the AWP scope

```
this.port.onmessage = event => {
  if ("eval" in event.data) {
    try {
      let setupFunction =
        eval(event.data["setup"]);
      let loopFunction =
        eval(event.data["loop"]);
      ...
    } catch (err) {
    }
  }
}
```

For instance, considering the following expression for FM synthesis (Listing 4), that is valid for of a live coding language that a user has defined through a given grammar:

Listing 4: Example of a test user-evaluated DSP expression

```
osc sin(osc sin 100 + osc tri 40)
```

The system interprets this expression and generates two anonymous functions. The *setupFunction* (Listing 5) declares and instantiates the required Maximilian Wasm objects for the custom DSP expression in the AWP scope.

Listing 5: The *setupFunction* generated from the AST

```
() => {
  let q=[];
  q.osc13 = new Module.maxiOsc();
  q.osc14 = new Module.maxiOsc();
  q.osc15 = new Module.maxiOsc();
  return q;
}
```

The original user-expression is converted to a compiled JS function, the *loopFunction* (Listing 6), which composes the Maximilian objects previously declared in *setupFunction* and runs in the AWP *process* loop (the main audio loop). The audio engine crossfades between the previous *loopFunction* and the newly generated one.

Listing 6: The *loopFunction* generated from the AST

```
(q) => {
  return q.osc13.sinewave((
    q.osc14.sinewave(100)
    + q.osc15.triangle(40)
  ));
}
```

¹²Worklet editor, <http://dsp.audio/editor>

¹³<https://acarabott.github.io/audio-worklet-live-coding>

¹⁴<http://bit.ly/2F4vZKM>

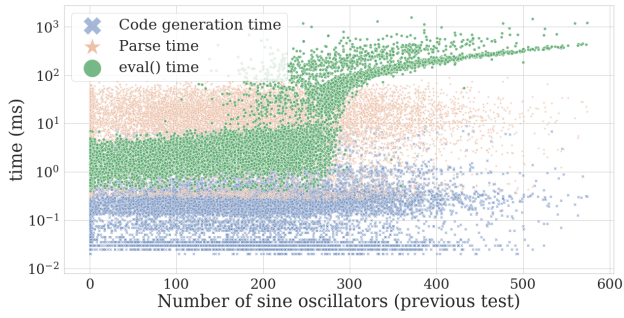


Figure 2: Load testing results

4. PERFORMANCE AND RELIABILITY EVALUATION

Good performance and reliability are central measures of the success of a live coding system, which must be able to meet the demands of live musical performance. We measured the reliability, latency and signal processing capacity of our system. To achieve this, we designed a simple testing language that was capable of creating nested trees of oscillators. An algorithm was designed to probabilistically generate code for oscillator trees with variable width and depth. An example of generated code is shown below.

```
osc sin (osc sin 960 + osc sin 961 + osc
sin (osc sin 657 + osc sin 348 + osc
sin 405) + osc sin (osc sin 1050 + osc
sin 122 + osc sin (osc sin 1052 + osc
sin 1090 + osc sin (osc sin 1088 +
osc sin (osc sin 258) + osc sin (osc
sin (osc sin 820))) + osc sin 221)))
```

This code generator could be hand tuned to change the probable size of its output. The test procedure worked as follows: (1) generate some code, (2) record the number of oscillator objects in the code, (3) run the code in the audio engine, recording times for parsing, JS code generation from the parse tree, and for compiling the code using `eval()`, (4) wait for 200ms and repeat.

All tests run in Chrome 75.0, on a MacBook Pro mid-2015, with 16Gb RAM and a 2.5GHz i7 CPU. We recorded time using `window.performance.now()`, with sub-millisecond resolution. These results figures measure the capacity of a single audio worklet, rather than the capacity of the browser's JS virtual machine, which is capable of running multiple audio worklets.

4.1 Signal Processing Capacity

Our first investigation tested the signal processing limits of the system, in a test designed to mimic real-world scenarios. The code generator was configured to create code that might overload the AWP signal processing loop. When overloading happens, this has an effect on the `eval()` time of the subsequent test, because the `eval()` command is carried out in the AWP thread and slows down if it is overloaded. We ran the test for one hour, over 14000 iterations. Figure 2 shows a scatter plot comparing the number of sine wave oscillators running in the previous test compared to the time taken for `eval()` to run in the current test. The `eval()` time is stable until around 200 oscillators, when variance starts to increase, and then above 250 oscillators we start to see an increase in latency. Parsing and code generation times remain stable throughout, as they are unaffected, running

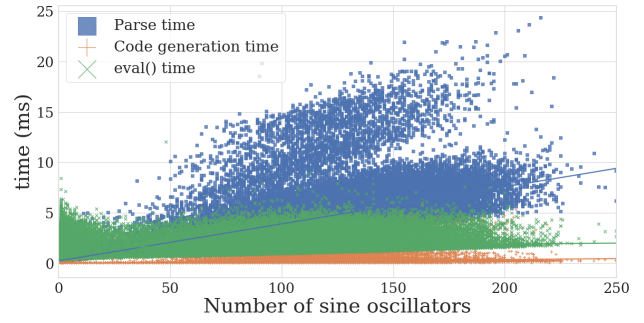


Figure 3: Latency test results: individual processes

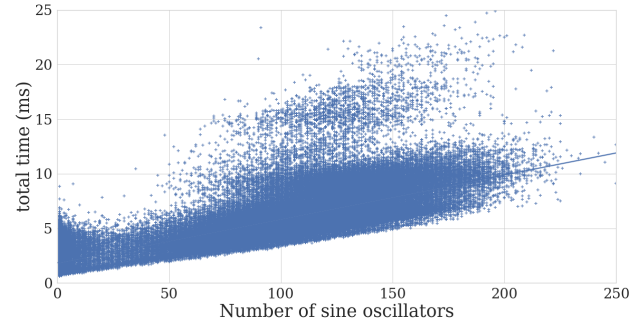


Figure 4: Latency test results: combined processes

in a separate worker thread. From these results, we can estimate that the system will run with good stability with approximately 200-250 sine wave oscillators.

4.2 Reliability and Latency

We tested the latency of the system within the stable limits discovered in the previous test. This is the time between when a user evaluates a line of code, and when the compiled code is ready to run in the next DSP cycle. The test ran for ten hours and 11 minutes, with 173422 iterations. Extreme outliers (with z-score over 20) were removed, accounting for 0.007% of datapoints. The remaining 99.993% of tests show latency of <25ms, indicating high reliability. Figure 3 shows individual latencies compared to the number of sine wave oscillators in the test code. The largest and most variable cause of latency is the Nearley parser. The combined latency times (Figure 4) demonstrate that the system is likely to run with sub-perceptual (<20ms) latencies when running within stable limits. The blue lines in both figures show a linear regression fit.

5. DISCUSSION AND DESIGN PATTERNS

The JS community often considers the use of the `eval()` function a bad coding practice [4]. However, it has been used in the Webkit console, JSBin, and the widely adopted library `lodash`¹⁵. The problems of using `eval()` have been associated with malicious attacks through code injection and cross-site scripting, hard maintenance and debugging, slow performance due to the cost of compilation, and wastefulness of resources (e.g. creating execution context collecting garbage, exporting variables). While we are

¹⁵Angus Croll: Break all the rules, JSConf 2012, <https://youtu.be/MFtjZdo>

still exploring this solution within the constraints of our scenario—e.g. client-side, immediate execution in the AWP scope—our evaluation metrics suggest that the advantages of `eval()` supersede the potential problems. One problem that we found with using `eval()` and Wasm was in the heap memory expansion. Although Emscripten provides an `ALLOW_MEMORY_GROWTH` compilation option, we found it led to inconsistent behaviours. We resolved this issue by setting a maximum memory growth threshold.

In the exploration and gradual development of our solution, we dealt with difficulties arising from the integration of technologies with different degrees of maturity. Some difficulties inherent to *AudioWorklet* were, for instance, error messaging, the availability of APIs in the AWP scope, and its impact in the Wasm module loading (e.g. for standard modularisation and module naming). Other difficulties emerged from loading these components within a webpack development environment which revealed somewhat cumbersome (e.g. adding imports for emitting Wasm without compilation, server MIME type definitions).

Our solution can provide a design pattern for people interested in re-using our stack or developing a similar approach. The first use of the signal engine was in a live coding performance at AlgoMech 2019¹⁶. The signal engine integrated with the live coding artist’s web page and supported its use as the main live coding instrument. Live coding was performed directly in Chrome Dev tools console.

6. CONCLUSION AND FUTURE WORK

We present a high-performance implementation of a signal engine for a new Web-based system for the live coding community. Evaluation shows that the system runs with efficiency and high reliability for 200-250 oscillators with sub-perceptual latency. Future work will focus on the presentation layer, user language specification support, and further development of the ML and MML workers. These developments include moving to an optimised parser implementation with WebAssembly JIT compilation, and supporting *SharedArrayBuffer* for inter-thread communication.

7. ACKNOWLEDGMENTS

This work was supported by the AHRC-funded MIMIC project, ref: AH/R002657/1 (<https://gtr.ukri.org/projects?ref=AH/R002657/1>).

8. REFERENCES

- [1] F. Bernardo, M. Grierson, and R. Fiebrink. User-Centred Design Actions for Lightweight Evaluation of an Interactive Machine Learning Toolkit. *Journal of Science and Technology of the Arts*, 10(2):2, 2018.
- [2] F. Bernardo, M. Zbyszyński, R. Fiebrink, and M. Grierson. Interactive Machine Learning for End-User Innovation. In *Proc. of the Association for Advancement of Artificial Intelligence Symposium Series: Designing the User Experience of Machine Learning Systems*, pages 369–375, 2017.
- [3] H. Choi. AudioWorklet: The future of web audio. In *Web Audio Conference*, 2018.
- [4] D. Crockford. *JavaScript: The Good Parts*. O’Reilly Media, first edit edition, 2008.
- [5] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.
- [6] M. Grierson and C. Kiefer. Maximilian: An Easy to Use, Cross Platform C++ Toolkit for Interactive Audio and Synthesis Applications. In *Proc. of The International Computer Music Conference*, number August, pages 276–279, 2011.
- [7] M. Grierson, M. Yee-king, L. McCallum, C. Kiefer, and M. Zbyszyński. Contemporary Machine Learning for Audio and Music Generation on the Web: Current Challenges and Potential Solutions. *ICMC*, 2018.
- [8] C. Kiefer and T. Magnusson. Live Coding Machine Learning and Machine Listening: A Survey on the Design of Languages and Environments for Live Coding. In *Proc. of the International Conference on Live Coding.*, Madrid, 2019.
- [9] V. Lazzarini, E. Costello, and S. Yi. Csound on the web. In *Linux Audio Developers Conference*, 2014.
- [10] S. Letz, S. Denoux, Y. Orlarey, and D. Fober. Faust audio DSP language in the Web. *Linux Audio Conference*, pages 29–36, 2015.
- [11] T. Magnusson. Herding Cats: Observing Live Coding in the Wild. *Computer Music Journal*, 38(1):91–101, 2014.
- [12] Y. Mann. Interactive Music with Tone.js. *Proc. of the 1st annual Web Audio Conference*, 2015.
- [13] A. Roberts, C. Hawthorne, and I. Simon. Magenta.js: A JavaScript API for Augmenting Creativity with Deep Learning. In *Proc. of the 35th International Conference on Machine Learning*, pages 2–4, 2018.
- [14] C. Roberts. Metaprogramming Strategies for AudioWorklets. In *Web Audio Conference*, 2018.
- [15] D. Shiffman, J. Ashe, K. Compton, H. Davis, D. Kazemi, G. Kogan, K. McDonald, and Yining Shi. ml5: Friendly Open Source Machine Learning Library for the Web. *ADJACENT*, (3), 2018.
- [16] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel, S. Bileschi, M. Terry, C. Nicholson, S. N. Gupta, S. Sirajuddin, D. Sculley, R. Monga, G. Corrado, F. B. Viégas, and M. Wattenberg. TensorFlow.js: Machine Learning for the Web and Beyond. In *Proc. of the 2nd SysML Conference*, 2019.
- [17] K. Steiglitz. *A Digital Signal Processing Primer: With Applications to Digital Audio and Computer Music*. Addison-Wesley, 1996.
- [18] W3C Audio Working Group. Web Audio API. <https://webaudio.github.io/web-audio-api>, 2019.
- [19] G. Wakefield and C. Roberts. A Virtual Machine for Live Coding Language Design. *Proc. of New Interfaces for Musical Expression 2017*, pages 275–278, 2017.
- [20] G. Wang, P. R. Cook, and S. Salazar. ChucK: A Strongly Timed Computer Music Language. *Computer Music Journal*, 39(4):91–101, 2015.
- [21] A. Zakai. Emscripten: An LLVM-to-JavaScript Compiler. In *Proc. of the ACM international conference companion on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2011.

¹⁶Algo/Mesh, <https://algoemch.com/2019/events/mesh/>

Interference: Adapting Player-Music Interaction in Games to a Live Performance Context

Matthew Wang
Princeton University
Princeton, NJ 08544
mjw7@princeton.edu

ABSTRACT

Interference is a multiplayer music game and generative music system, which is implemented as a Javascript web application and designed for live performance. It is based on the potential for dynamic music generation that exists in video games through player-music interaction. It uses a competitive multiplayer form to sustain a feedback loop in which players construct and change the music at a fine scale, while the music in turn informs players of the game state, affecting how they continue to play and therefore change the music. The design of *Interference* must also manage conflicts between games and music as contrasting media, such as presentation, length, and complexity, in order to create both a game that is engaging for its players and a musical performance that is compelling to its audience. Towards this objective, it combines elements of games that do not traditionally exist in music, such as an explicit goal-oriented structure, with features that serve strictly musical, performative purposes, allowing players to act simultaneously as performers. To support this design, it utilizes several existing web technologies to achieve tight synchronization, changeable sound synthesis, and networked interaction between players.

1. INTRODUCTION

A crucial consideration in the development of video games and especially in the creation of their music is how players interact with the music and sound of games. Due to the inherently dynamic nature of games, their music must be changeable based on how players can progress through them. For most games, this means that music starts in response to a trigger – possibly on startup, entering a new area, or encountering a certain character – and continue for its set length or indefinitely until another trigger. The player, having control over each trigger either directly or indirectly, therefore controls the music. Inversely, the music of games influences how players act. It can inform the player of a change in the state of the game, prompting them to react, or exert some emotional effect on the player, affecting their decisions or perceptions going forward.

This combination of player influence on the progression of music and musical influence on the actions of players results in a feedback system. Player action determines the progression of the music, which affects further player action in turn. But for a majority of

games, the music-making potential of this feedback system goes unrealized. This is largely unsurprising in games that do not place significant focus on music as a game element. In these games, non-musical goals are the primary concern and more complex player-music interaction, if implemented, would likely obscure those goals. However, even most music-oriented games make little use of this feedback system. Take for example the archetypal “music game” *Guitar Hero* [2]. Although it presents as a music-making game, in which the player acts as a performer, the player actually has very limited control over the music. They are only able to play or not play predetermined notes – controlling the playback of music, but not its content. Michael Liebe describes this form of music, which is typical of music games, as proactive music, which mostly occurs independently of player input and demands some response from the player, as opposed to reactive music, which reacts directly to player action, and linear music, which occurs independently of direct player input and does not demand a response [5].

With *Interference*, I aim to create a game and performable generative music system that takes advantage of player-music feedback and features music that is simultaneously proactive and reactive. From a design standpoint, this requires an understanding of approaches to sustaining such a feedback system, methods for the construction of a compelling game that uses said feedback system, and any concerns a performance context may introduce. Following a discussion of these points, I will explain the core components of the technical implementation of *Interference* as well as its gameplay and performance.

2. DESIGN

2.1 Player-Music Feedback

The main obstacle to sustaining a consistent and meaningful feedback relation between player and music is the inherent incompatibility of reactive music with proactive music. Reactive music struggles to influence player action in the same way as proactive music because players tend to interpret reactive music as their own action or as a commentary of their action rather than as an external force that demands reaction. Conversely, if the music reacts more indirectly to player action such that players feel they must respond to it, it quickly ceases to feel reactive as players become less able to purposefully influence it.

One solution to this problem is to allow the player to directly generate the music but then introduce a level of abstraction or error into that generated music that retains its reactive identity while still demanding a response from the player. An example of this sort of feedback system is *Zero Waste*, a game-like musical piece for a



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

sight-reading pianist and a computer, which has the computer present the performer with a few measures of random music to sight-read before transcribing an attempt by the pianist to play it, and presenting the transcription back again [8]. This system achieves a form of feedback that could exist in-game, but it relies on player and system error and trends. As a result, there is a somewhat deterministic quality to the process, and furthermore, change tends to occur gradually in such a system.

For this project, I sought to create a more dynamic and variable system. Therefore, to overcome this obstacle of sustaining a feedback system, rather than use error to create variation in a human-computer feedback system, I chose to design *Interference* as a multiplayer game. With multiple players, each player can contribute in a direct and transparent way to the composite generated music, but each player must also react to the contributions of every other player. This maintains both reactive and proactive features in the music.

2.2 Competitive Gameplay

Such a multiplayer game could be implemented as either a cooperative or competitive system, but I chose to create a competitive system for several reasons. First, competition, more automatically than cooperation, engages players and encourages participation because it introduces conflict and challenge without a need for high complexity or technicality. Second, concerning use in a performance context, a competitive system requires only a minimal level of skill and knowledge in the mechanics of the game in order for a performance to progress and, importantly, end.

Last, the combination of proactive and reactive musical elements in a multiplayer environment quickly becomes chaotic and complex. This complicates the implementation and balance of cooperative tasks, which require a non-player system to challenge but not overwhelm players, whereas a competitive system is largely self-balancing even in a highly complex environment. Regardless of the complexity of the system, each player starts with equal means and information and is only rewarded or limited by their own ability to use their resources in comparison to their competitors. In this way, while a player could choose to ignore proactive musical information, they would put themselves at a disadvantage to more reactive players.

2.3 Performance Context

In contrast to the competitive nature of *Interference* is its design as a performance work. While the players compete, they must also cooperate as performers and therefore may sometimes choose to take performative actions rather than actions that are competitively advantageous. Therefore, a design goal was to have performance and competitive motivations overlap as much as possible, such that actions which are advantageous in the game are also musically engaging.

An important consideration towards achieving this goal is how player action maps to sound. Ideally, game-motivated player action should correspond to musically functional sound. To the determination of strong mappings as such, attention to the time scale of game events is especially important. With some exceptions, game events that occur only occasionally best map to more significant changes and large-scale shifts in the music while changes that occur frequently map well to more subtle changes in the music. In particular, if the musical result of an action that occurs frequently in the game is too extreme, it can create conflict between the player and the performer roles. Karen Collins’ “Ten Approaches to Variability in Game Music” from *Game Sound* is a

particularly useful resource in the consideration of which elements of music in games can be effectively dynamically altered, although the actual mapping of game events to these elements is largely a matter of experimentation [1].

Interference specifically achieves some harmony between the roles of player and performer in that large modal shifts in the game correspond to paradigmatic changes in the musical texture while the most frequent game events change the music only incrementally. And perhaps most successfully, one of the strongest tactics – a technique I call “leading the sequencer step,” discussed in Section 4 – results in appropriately striking yet simple musical figures. That said, *Interference* is somewhat lacking in the range of performative expression it gives its players due to its limited use of variable sound synthesis and the relatively small amount of influence a single performer has over the composite music.

A final performance concern, aside from the harmonization of player and performer roles, is the practical execution of these roles, which are challenging for a single person to focus on simultaneously. *Interference* uses its visuals to enable its players to execute both roles. By corresponding to both game action and musical change as closely as possible, visual elements help link the game and music. Visual elements that simultaneously represent a game object and appear to produce sound allow players to act deliberately as performers. Additionally, these visuals can enhance the experience of the audience, which led to the choice in performance to display players’ screens on external monitors facing the audience (see Figure 1).



Figure 1. Setup of a five-player performance.

3. IMPLEMENTATION

3.1 Networked Game Interaction

Crucially, *Interference* is a networked game, as information about game and player states must be shared across all players. For networking game information and input between players, I used Lance, a Node.js based server and client-side library intended for multiplayer web-based games [7]. It additionally includes basic game and physics engines and synchronization strategies to handle latency. Lance does not provide for the more flexible sound synthesis or the precise rhythmic synchronization *Interference* requires but is extremely useful for essentially all other aspects of the project, and additional modules and libraries are easily incorporated.

3.2 Synchronization

Due to both the musical focus of this project and the presence of input latency in a networked game context, which exacerbates any synchronization issues for performers, implementing precise and reliable rhythmic synchronization was especially important. For this synchronization, I used Collective Soundworks’ *sync*, which provides consistent synchronization with the minor condition that players may need to wait some time for their sequencers to synchronize upon connection [3, 4].

3.3 Sound Synthesis

Finally, effectively controlling the music of *Interference* requires flexible sound synthesis. Adequately flexible sound synthesis in a game context allows for virtually any mapping of game variables to sound. For this purpose, I ultimately chose to use Tone.js, an audio framework built on the Web Audio API [6]. Tone.js has some notable limitations, such as a lack of polyphony on noise-based instruments, but as a relatively mature and widely used web audio framework, its ease of use was worth any of its inflexibilities.

4. GAMEPLAY

To balance the complex fine-scale variation of the musical and visual elements of *Interference*, the game itself is relatively simple if rather abstract. The game space consists of a series of 32 by 18 colored grids positioned in a horizontal line. Each grid represents one player's territory, their initial field of view, and a set of three step-sequencers layered on top of each other (see Figure 2). Each of these three step-sequencers runs constantly at differing rates from one another. The horizontal axis represents the sequencer time steps and the vertical axis represents the pitch of sequenced notes. Each player begins as one of five color palettes, which correspond to various harmonic sets for the sequencer pitches. Each player's goal is to convert every other player to their palette.

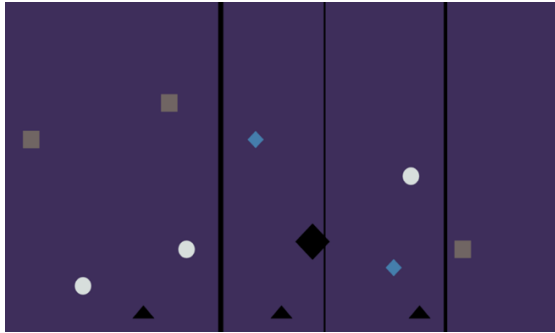


Figure 2. The view of a single player's grid during a build phase with a diamond-shaped ball in black, notes of each shape, and playheads for each sequencer. The black triangles at the bottom of the grid indicate how many notes the player has left to place.

With the exception of an outro section, the game consists of two states that alternate over its course: the "build" phase and the "fight" phase. See Figure 3 below, which provides an outline of the overall game progression structure and the mechanical and musical characteristics that define each type of phase.

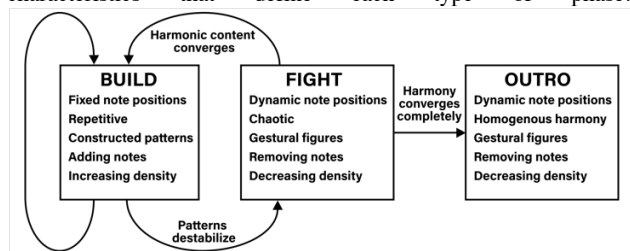


Figure 3. The progression of *Interference*. The alternation of build and fight phases creates a cycle between varying levels of density and stability as the overall harmonic content gradually converges.

The game begins in a build phase, during which players build sequences of notes. To start, a ball object spawns with a randomized position and velocity and moves throughout the space, bouncing off

its boundaries. Players can then "hit" the ball as it passes through their area to place a note in their sequencer at the position of the ball. The shape and sound of the placed note depend on the shape of the ball, which can be a circle, a diamond, or a square. After players have collectively hit the ball enough times, it breaks. The player who broke the ball then has the option to start another build phase or progress the game into a fight phase.

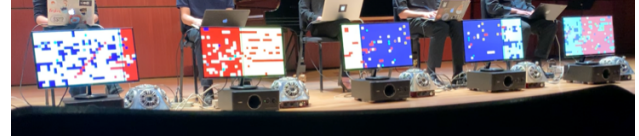


Figure 4. Players' screens during a fight phase with their views displaced from their starting position.

During a fight phase, players can move their view and placed notes as a single rigid structure through the entire game space, wrapping across boundaries (see Figure 4). Players convert cells of the grids to their color when a sequencer plays their notes on those cells, resulting in the especially effective tactic of leading each step of a sequencer playhead, such that a note plays on every step and converts a line of cells. Players can remove each other's notes by forcing collisions with their own notes. A rock-paper-scissors system using the three note-shapes determines which note to remove upon collision.

After enough notes have been removed or players have made enough inputs to force progression, the fight phase ends and the piece transitions back into a build phase. At this point, players and notes reset to their original positions and if a player's territory has been mostly overtaken by other colors, their entire territory and their notes convert to the now most prominent color. The game ends when all players have the same color palette or after some amount of time chosen beforehand, at which point all players are converted to the dominant color (see Figure 5).

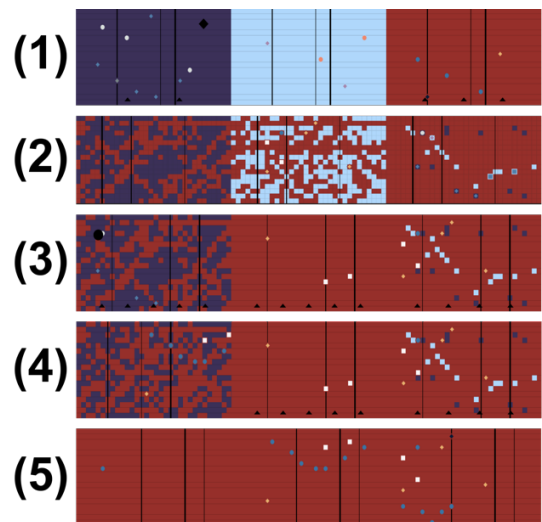


Figure 5. Progression of a three-player game. (1) First build phase. Each player has a color. (2) End of the first fight phase. The red player has converted much of the field. (3) Second build phase. The light blue player has been fully converted to red upon the transition from fight phase to build phase. (4) End of the second fight phase. The red players have converted even more of the remaining field. (5) Outro phase. The purple player has been converted to red and the game has ended.

Aside from the basic rules and progression of the game, the gameplay also intersects with the music in several ways. For example, because each player's audio output only sonifies the immediate contents of their territory, they can use their audio to identify when another player is converting their territory, assuming players use spatially separate audio output systems, as with the Princeton Laptop Orchestra's hemispherical speakers. Players can also use the music to identify the state of the game, such as the current game phase or even the dominant color palette. Players can, of course, identify these elements visually but not without a significant time commitment due to the limited scope of visual information at any given moment, so those who can react effectively to the music gain an advantage.

5. PERFORMANCE

Interference also features several strictly performative features, which generally cater to musical elements that were otherwise difficult to incorporate into the game. The outro section mentioned previously is one such case as it begins only once the actual game has ended. During the outro section, all players have the same color palette and can move freely as in a fight phase, but instead of removing each other's notes, players can input a command to remove a random note from the whole game. The purpose of the outro section is strictly musical in that it allows the performers to play freely in the harmonic progression of the final color palette and slowly time a fade-out by removal of notes to circumvent an otherwise musically abrupt ending to the game. Another performative feature is a control to progress the harmony, which during a fight phase also controls progression back into a build phase, but otherwise strictly serves a musical purpose.

6. CONCLUSION

Interference explores a form of feedback-based music generation that game environments provide but that often goes unused due to limitations games typically impose on their music. Rarely do games allow music to make heavy demands of players and when they do, as in music and rhythm games, the player generally has little control over the generation of the music. By creating a multiplayer system in which players interfere and intermingle with one another's musical and game choices, *Interference* allows its players to generate its music while simultaneously allowing the music to make demands of its players.

The concepts behind this project open up many opportunities for future development. While *Interference* features a high level of fine-scale variability and no large precomposed parts, which are more typical in games, these ideas could potentially apply to any scale of musical content. Additionally, many features of *Interference*, such as the visuals, exist mostly to make the game more accessible to players and audiences and are not necessary to the core concept of the game. A strictly audio-based game could be possible (though much more difficult to play), and as previously discussed, a cooperative game could operate under similar principles through careful design. The generative possibilities of

multiplayer music games are largely unexplored, and *Interference* is only a basic proof of their potential.

7. LINKS

Interference is playable at <https://interference.herokuapp.com/>. Server limitations may affect game performance.

Source code and a brief description of controls is available at <https://github.com/mattmora/interference>.

Video and audio recordings of the premiere performance of *Interference* by the Princeton Laptop Orchestra is available at <https://www.youtube.com/watch?v=C-5P3hXuGfs>.

8. ACKNOWLEDGMENTS

Thanks to Jeff Snyder for advising the development of this project, to the Princeton Laptop Orchestra for their assistance in testing, rehearsing, and performing it, and to my friends in the Princeton Pianists Ensemble for taking part in a second performance.

9. REFERENCES

- [1] Collins, K. 2008. *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. MIT Press, Cambridge, MA and London.
- [2] Harmonix. 2005. *Guitar Hero*. Game [PlayStation 2]. RedOctane, Mountain View, California, USA.
- [3] Lambert, J.P. sync. Github Repository. Retrieved from <https://github.com/collective-soundworks/sync>, last checked 11 March 2019.
- [4] Lambert, J.P., Robaszkiewicz S., and Schnell, N. 2016. Synchronisation for Distributed Audio Rendering over Heterogeneous Devices, in HTML5. In *Proceedings of the Web Audio Conference (WAC)*. Atlanta, USA. pp 6–11. URI= <http://hdl.handle.net/1853/54598>.
- [5] Liebe, M. 2013. Interactivity and Music in Computer Games. In *Music and Game: Perspectives on a Popular Alliance*, by Peter Moormann. Springer VS, Berlin. pp 41–62.
- [6] Mann, Y. Tone.js. Github Repository. Retrieved from <https://github.com/Tonejs/Tone.js>, last checked 11 March 2019.
- [7] Weiss, G. Lance. Github Repository. Retrieved from <https://github.com/lance-gg/lance>, last checked 9 March 2019.
- [8] Whyte, D., Didkovsky, N., and Hutzler S. 2018. Zero Waste: Mapping the Evolution of the Iterative Sight-Reading of a Piano Score. *Music Theory Spectrum* vol. 40, 2, pp 302–313. DOI= <https://doi.org/10.1093/mts/mtty019>.

Towards Large-Scale Artistic Practice with Web Technologies

Luis Arandas
Faculdade de Engenharia e
Universidade do Porto
Braga Media Arts
luis.arandas@gmail.com

José Alberto Gomes
Portuguese Catholic University
CITAR School of Arts
Braga Media Arts
jagomes@porto.ucp.pt

Rui Penha
Escola Superior de Música e
Artes do Espectáculo
INESC-TEC Porto
ruipenha@esmae.ipp.pt

ABSTRACT

In this article, we present a software architecture that explores the technological potential of the web as a programmable interface and as an interpersonal connection point in the artistic practice. This structure exposes the recently proposed Akson audio-visual (AV) environment, also raising a technical evaluation of the technologies and design used to allow the development of both the platform and the network.

1. INTRODUCTION

The main objective of this article is to expose in detail the architecture of a web-based AV environment for artistic collaboration. We propose a centralized way to explore different modes of networked interaction that can be used in distributed artistic practice. This practice is fundamentally focused on computer music and contemporary media art through digital interfaces and is designed to explore the cloud as a medium for communication. This environment provides the artist with various forms of expression in a pre-established network, explores the constraints that exist within it and applies them to digital artistic performance.

The proposed system is programmed to use the capabilities of a Chromium browser. It is allocated on the Heroku¹ [20] infrastructure and uses node.js² [26], a JavaScript (JS) runtime built on Chrome's V8 JS engine [15]. We establish two-way communication structures using the library socket.io³ [19, 34], generate audio signals using the WebAudio API⁴ [5] and graphics with the WebGL API⁵ [13]. All the connected devices are linked by default, and it is possible to dynamically change between the implemented modes of interaction (see section 6). To interact with the system the user can click on the generated graphics or use the integrated WebMIDI platform.

¹Heroku <https://www.heroku.com/>

²Node.js <https://nodejs.org/en/>

³Socket.IO library website <https://socket.io>

⁴WebAudio API <https://www.w3.org/TR/webaudio/>

⁵WebGL API <https://www.khronos.org/webgl/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

All software and interface design was developed specifically for this research and in the context of Braga Media Arts⁶. This outcome is also product of a review into collaborative interfaces [4] for AV systems and networked interaction [23, 29].

2. THE SYSTEM

Nowadays, the cloud offers enormous potential as an artistic production space [1, 30]. We reflect not only on the freedom offered by the web as an extension of the human gesture, but also as a multimodal shared musical and visual space.

In this project⁷ we contract some issues related to human embodiment and communication efficiency [18, 23] that are present in the development of communication systems. Some of those problems are addressed by much contemporary research in this area and one example of that is the Distributed Immersive Performance (DIP) project [9, 33, 38] working on latency issues for data streaming. We address this issue by adopting some techniques often found in the *live-coding* [25] practice. We base the system on socket communications with multiple events throughout the code, and use them to control the generators on each instance of the system. In this way small pieces of information are sent quickly, using the server as a mediator.

But this is just one of the multiple problems addressed. Other example might be the type of system (i.e., centralized; decentralized) and the necessary characteristics (i.e. dependencies of external libraries; hardware). In the particular case of this investigation we establish a single static server to provide the files, include all dependencies in the source code and instantiate all generators/connections in a predefined way.

When a successful link is established to a device capable of reproducing the platform (i.e. WebGL 2.0 [12]) and having included all dependencies and modules without the need for external links, the platform is structured. These include NPM⁸ and the Express⁹ [22] project, both used as frameworks for node. Regarding node modules, the source code includes: *portfinder*¹⁰ - a port scanner on the current machine; *osc-js*¹¹ package - to use the Open Sound Control

⁶<http://www.bragamediaarts.com/pt/>

⁷<https://github.com/luisArandas/akson>

⁸NPM <https://www.npmjs.com/>

⁹Express <https://expressjs.com/>

¹⁰*portfinder* <https://www.npmjs.com/package/portfinder>

¹¹*osc-js* <https://www.npmjs.com/package/osc-js>

[37] (OSC) protocol and to establish UDP connections with software like Pure-Data [27] or SuperCollider [35]; and the *winston*¹² package - a universal logging library that can help deal with runtime problems.

Before starting to build the interface [7], at the same time that the generators and controllers are instantiated, a model of communication between users is established. The established (bidirectional) sockets are defined throughout the project, accompanied by unique ID's, which in turn correspond to functions on the front end. By default, the *socket.broadcast.emit* method is set to send to all connected machines except the sending agent.

This homogeneous connection topology will later serve as a means of defining modes of interaction. Applying patterns and/or condition matrices to the code that is running on the devices, is an approach that allows for a slight and rapid state change on the various bindings. It is then possible to define communication structures in the created network, which can objectively control the scope of the user's action. How the data navigates the network is represented in figure 1 as a diagram.

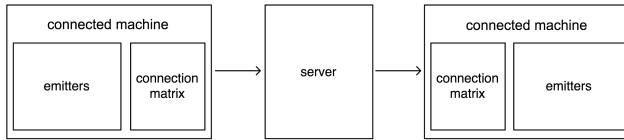


Figure 1: Communication structure between two machines connected to the server. The presented diagram shows that no data emission is performed without first passing through the established interaction matrix and the server.

3. THE INTERFACE

After establishing the connections, the type of server and the dependencies we can start thinking about the development of an interface as a mediator of the experience [32]. Of course, there are several options to be taken when it comes to developing interfaces (graphical or not), and we have decided to expose its existence by encapsulating the technical conditions of this system. Respecting the characteristics of this project as an AV environment we decided to develop two types of layers. A graphical user interface (GUI) composed by widgets, sliders and panels and also a structure that allows interacting directly in the graphical scene (see section 4/5).

This includes some JS dependencies such as: jQuery¹³ - for the use of its functions for document processing; The NexusUI¹⁴ project [2] - an open-source collection of HTML5 interfaces and JS helper functions to assist with building web audio instruments in the browser; the WUI¹⁵ project - an easy to use and lightweight collection of widgets for the web browser using vanilla JS with no dependencies; And some adapted buttons from the Bootstrap¹⁶ collection. A responsive and popular component library for quick prototyping.

¹²*winston* <https://www.npmjs.com/package/winston>

¹³jQuery <https://jquery.com>

¹⁴Nexus <http://nexus-js.github.io/ui/>

¹⁵WUI <https://github.com/grz0zrg/wui>

¹⁶Bootstrap <https://getbootstrap.com>

We have also included a way to communicate with external peripherals via the MIDI protocol. We have developed a platform for data reception and processing using Web-MIDI API¹⁷ [5] instances and a system for data emission via OSC protocol. As a web protocol that allows controllers and general electronics to communicate and synchronize through MIDI, there are some libraries that present dynamic learning systems, often found in digital audio workstations (DAW). The *SimpleMidiInput.js*¹⁸ library does an abstraction over the input *navigator.requestMIDIAccess* and we integrated it alongside *WebMidi.js*¹⁹, for data processing.

4. AUDIO ENGINE

As previously mentioned, we built the system by offering communications with small portions of information, somewhat similar to the *a.bel* system [10] (i.e., method properties, or interface changes). This is a structure that was thought to be able to control large numbers of distributed interfaces without the need of an enormous computational power also inferred by latency.

Together with native WebAudio, we also explore the *tone.js*²⁰ [21] library. It has a set of abstractions that allow the creation of generation instances, scheduling and the use of processing effects. We have linked the Nexus widgets with various methods of audio generators by assigning the corresponding IDs to back-end sockets, also with an instance of the MediaStream Recording API for direct registration on each connected instance, making it easy to document the collective performance.

By default, the way to make music is by playing notes using the graphics system. By interacting with the objects on the page, notes are generated within a scale. In order to facilitate the gestures between instances, several scales have been created, encompassing three octaves. These scales can be found in the class entitled *ScalesPlaying*.

```

var scaleMatrix = new ScalePlaying();
var scale = scaleMatrix.cMajorPentatonic();
var note = Math.floor(Math.random() *
    scale.length);
Tone.context.resume().then(() => {
    currentSynthesizer.triggerAttackRelease(
        scale[note], "4n");
});

```

This piece of code provides a way to instantiate the class and play a note within it. The *Math* object is used to perform a random operation within the array and is called a generator method. By default, all instances start in major pentatonic, and all class scales are arranged in C to facilitate phrase coordination between devices.

The group of scales that we can choose from is composed of: *cMajorPentatonic()*; *cMinorPentatonic()*; *cMajor()*; *cMinor()*; *cHarmonicMinor()*; *cMelodicMinor()*; *cAdonaiMalakh()*; *cHungarianMajor()*; *cHirajoshiJapan()*; *cIonian()* and *cLocrian()*. The way this is implemented is simply based on a dynamic way of writing to the currently used array. In the following example we instantiate a list return.

¹⁷WebMIDI <https://webaudio.github.io/web-midi-api/>

¹⁸SimpleMidiInput.js

<https://github.com/kchapelier/SimpleMidiInput.js>

¹⁹WebMidi.js <https://github.com/djipco/webmidi>

²⁰Tone.js <https://tonejs.github.io>

```

cHarmonicMinor() {
  return ['C2', 'D2', 'D#2', 'F2', 'G2',
    'G#2', 'B2', 'C3', 'D3', 'D#3', 'F3',
    'G3', 'G#3', 'B3', 'C4', 'D4', 'D#4',
    'F4', 'G4', 'G#4', 'B4', 'C5',
    'D5', 'D#5', 'F5', 'G5', 'G#5',
    'B5'];
}

```

As described above, these notes are called randomly within the scale. The software has no structure to help formalize musical structures (i.e., stochastic modelling or any other probability theory implementation [28]), often making randomness a problem. The way to choose notes by hand in a graphical interface is sometimes not enough when you want to create precise coordination between distant players. A popular and easy to implement example multiple times found in automatic music systems are the Markov chains. The *js-markov*²¹ package is a recent example of that mathematical implementation, and also the *foswig.js*²² library that works with text can be used for symbolic content generation.

5. GRAPHICS ENGINE

The graphic system is conceptually built with some similarities to the previous one, using WebGL. It also explores the generation of graphics in fixed or mobile devices using the browser, exchanging small amounts of information such as mouse clicks when interacting with the generated content.

It uses the *three.js*²³ [14] library and its custom rendering system alongside GLSL [31] shaders for post-production. We use it to pose the methods and characteristics of the renderer, instantiate the geometries and define the raycaster which is the main algorithm used for graphical interaction (instantaneously used by synthesizer).

This system is constructed to seek a synesthetic coherence between the graphical interaction and the sound generation, from the moment the artist's action occurs [8, 11]. It is possible to explore some kind of feedback between devices when they are interacting, such as color change, which is a result of a successful click. User geometries and actions are also assigned to individual IDs that will be matched on the server. The graphical interface is also populated by the visual system related methods such as camera movements, light control and instantiated geometry attributes.

6. INTERACTION MODELS

As a system focused on large-scale artistic practice, whether defined by a large number of interfaces or by large distances around the world, it is meant to explore the dynamic change of data flows in the same network.

Referencing the study area of human-computer interaction [16, 17, 24, 36] (HCI) on the development of digital collaborative interfaces, there are several options that can be taken regarding the interaction models offered to users. With the proposed architecture we can send *user data* (i.e. browser attributes), *sound data* (i.e. generator attributes; filters), *graphics data* (i.e. geometry properties; color changes) and *system info* (i.e. the machine being used; the position in

space). All these attributes are set up in matrices and arranged in such a way that they can be easily manipulated.

The connection matrix will be a validation of conditions before the sockets are transmitted anywhere else, thus allowing for its dynamic change. We then establish relationships between instances [36].

Several network topologies existing in the history of contemporary artistic performance can be implemented, always maintaining a causal relationship between the type of network and the artist's freedom.

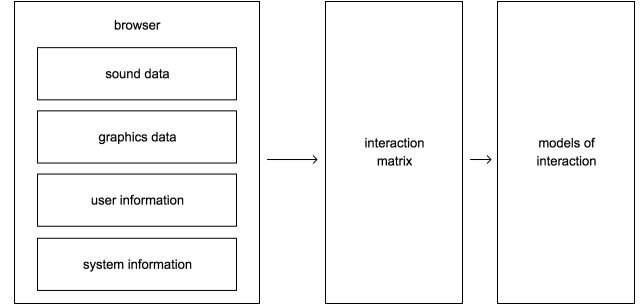


Figure 2: Data flow between browsers used to establish interaction models within the network.

7. SCALABILITY

Collaborative systems used for AV purposes in the cloud tend to require special development when all connected instances have complete freedom of action. Even if the system can (i.e. in terms of computing power) withstand hundreds or even thousands of interfaces (with or without human agents), it is necessary to create dedicated sub-systems when it comes to action management. That number of artists connected at the same time in a single performance can be interesting as an artistic phenomenon, but it can also be a problem in many ways.

Assuming that the infrastructure is set (i.e. Heroku; Amazon AWS²⁴; Microsoft Azure²⁵) and using the architecture presented throughout this article, a dynamic count of users and their automatic grouping can be a good starting point.

It is possible to solve some of the problems raised by programming a unified group, internally separated by the various modes of interaction (in sub-groups) maintaining a constant count of entries on the server (log-ins). Offering the possibility to change model to each user (including being alone), with each x number of entries in the system we can dynamically create copies of that group and allocate users there as needed (or if chosen by them). Respecting the rules of communication present in each model, we can also introduce a dedicated console in the front-end showing all these updates in real time, so that it is possible to orchestrate gestures between people and have several performances taking place at the same time separately.

Still, multiplying the various instances automatically by keeping the previously presented rules does not solve interaction freedom. Having a transmutable system can help in optimizing the environment but it is necessary to stress that the various interaction models are finite. There may

²¹js-markov <https://www.npmjs.com/package/js-markov>

²²forwig.js <https://github.com/mrsharplunto/foswig.js>

²³Three.js <https://threejs.org/>

²⁴Amazon AWS <https://aws.amazon.com>

²⁵Microsoft Azure azure.microsoft.com

be some kind of collaborative behaviour model that doesn't exist, which can lead to an easy-to-use preset and back-end control system. By providing this solution, we can have some degree of control when large numbers of people want to use the system from the same domain but do not want to interact with each other.

A public experiment already conducted with the proposed environment took place at the xCoAx international conference (Figure 3) [3]. The stage was occupied by two performers, creating an instance of the proposed system. At the beginning of the concert, it was explained to the audience how they could participate/interact, and that this would be done in a dedicated network.

The agents on stage composed the piece exploring the uncontrolled interaction of the participants for half the duration of the concert, defining what all the connected users could play. In this way the AV content was modeled, without the external participants being able to divert the purpose of this event.



Figure 3: Public performance in the International Conference on Computation, Communication, Aesthetics and X - Milan, Italy.

8. FUTURE WORK

Based on the scalability of this project and the proposed solutions to implement, we've listed some future work. We argue that this project can take different valuable paths both as a networked environment and as an AV platform of free access all over the world.

- As explored in section 7, we believe it is possible to respond to various future needs of this system by programming automation rules that also respect the freedom needed by the agents who are using it.
- Transpose this system into a decentralized architecture (i.e. using blockchain) by creating low-level instances on each device.
- Generation of AV content between machines using stochastic modeling (i.e. VMM's[6]). Allowing the creation of musical phrases between people without deterministic induction.

- A dynamic learning system that allows to create presets on back-end socket matrices to enable new interaction models
- The study of latency focused on data streaming in adaptive systems is always a valuable type of research to increase its robustness.

9. CONCLUSIONS

In this article we have presented a system made with web technologies that emerged from an investigation in computer music, media art and collaborative interfaces. We explore its existence as a digital structure, the technologies it uses, its properties as an AV platform in the cloud, and some steps that can be taken as future development.

Throughout this text, some paradigms inherent to contemporary AV performance are mentioned. The way artists interact is an issue that is mirrored in this project, and the attempt to respond to these needs is what led to use technologies such as node.js and socket.io.

The proposed system allows AV interaction from multiple devices distributed over potentially large distances across the globe, establishing communication matrices between machines.

10. ACKNOWLEDGMENTS

This article reflects and exposes part of an investigation conducted under Braga Media Arts on collaborative AV environments and web technologies. All related contributions are in it's context, part of UNESCO's²⁶ Creative Cities Network (UCCN)²⁷.

11. REFERENCES

- [1] D. Akoumianakis, G. Ktistakis, G. Vlachakis, P. Zervas, and C. Alexandraki. Collaborative music making as remediated practice. In *2013 18th International Conference on Digital Signal Processing (DSP)*, pages 1–8. IEEE, 2013.
- [2] J. T. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *NIME*, pages 1–6, 2013.
- [3] L. Arandas, J. A. Gomes, R. Penha, and G. Bernardes. Never the less: A performance on networked art. *Proceedings of the xCoAx - Conference on Computation, Communication, Aesthetics X, Milan, Italy - In preparation*, 2019.
- [4] Á. Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, pages 53–59, 2003.
- [5] J. Beggs and D. Thede. *Designing web audio*. "O'Reilly Media, Inc.", 2001.
- [6] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.
- [7] B. Bongers. Physical interfaces in the electronic arts. *Trends in gestural control of music*, pages 41–70, 2000.
- [8] R. Carvalho. From clavilux to ufabulum. *Journal of Science and Technology of the Arts*, 5(1):43–52, 2013.

²⁶UNESCO's European Website <https://en.unesco.org>

²⁷UCCN <https://en.unesco.org/creative-cities/home>

- [9] E. Chew, R. Zimmermann, A. A. Sawchuk, C. Kyriakakis, C. Papadopoulos, A. François, G. Kim, A. Rizzo, and A. Volk. Musical interaction at a distance: Distributed immersive performance. In *Proceedings of the MusicNetwork Fourth Open Workshop on Integration of Music in Multimedia Applications*, pages 15–16, 2004.
- [10] A. Clément, F. Ribeiro, R. Rodrigues, and R. Penha. Bridging the gap between performers and the audience using networked smartphones: the a.bel system. In *Proceedings of the International Conference on Live Interfaces*, 2016.
- [11] N. Correia et al. *Interactive audiovisual objects*. School of Arts, Design and Architecture, 2013.
- [12] P. Cozzi. *WebGL insights*. AK Peters/CRC Press, 2015.
- [13] B. Danchilla. *Beginning WebGL for HTML5*. Apress, 2012.
- [14] J. Dirksen. *Learning Three.js: the JavaScript 3D library for WebGL*. Packt Publishing Ltd, 2013.
- [15] J. Gray. Google chrome: the making of a cross-platform browser. *Linux Journal*, 2009(185):1, 2009.
- [16] A. Henderson. Interaction design: beyond human-computer interaction. *Ubiquity*, 2002(March):6, 2002.
- [17] S. Holland, T. Mudd, K. Wilkie-McKenna, A. McPherson, and M. M. Wanderley. *New Directions in Music and Human-Computer Interaction*. Springer, 2019.
- [18] Z. Jin, R. Oda, A. Finkelstein, and R. Fiebrink. Mallo: A distributed, synchronized instrument for internet music performance. In *Proceedings of the international conference on new interfaces for musical expression (NIME)*, 2015.
- [19] L. Kalita. Socket programming. *International Journal of Computer Science and Information Technologies*, 5(3):4802–4807, 2014.
- [20] C. Kemp and B. Gyger. *Professional Heroku Programming*. John Wiley & Sons, 2013.
- [21] Y. Mann. Interactive music with tone. js. In *Proceedings of the 1st annual Web Audio Conference*. Citeseer, 2015.
- [22] A. Mardan. *Express.js Guide: The Comprehensive Book on Express.js*. CreateSpace Independent Publishing Platform, 2013.
- [23] C. McKinney. *Collaboration and embodiment in networked music interfaces for live performance*. PhD thesis, University of Sussex, 2016.
- [24] R. Mills. Telematics, art and the evolution of networked music performance. In *Tele-Improvisation: Intercultural Interaction in the Online Global Music Jam Session*, pages 21–57. Springer, 2019.
- [25] C. Nilson. Live coding practice. In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 112–117. ACM, 2007.
- [26] C. R. Pereira. *Aplicações web real-time com Node. js*. Editora Casa do Código, 2014.
- [27] M. Puckette et al. Pure data: another integrated computer music environment. *Proceedings of the second intercollege computer music concerts*, pages 37–41, 1996.
- [28] L. R. Rabiner and B.-H. Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.
- [29] S. Rafaeli and F. Sudweeks. Networked interactivity. *Journal of computer-mediated communication*, 2(4):JCMC243, 1997.
- [30] D. B. Ramsay and J. A. Paradiso. Grouploop: a collaborative, network-enabled audio feedback instrument. In *NIME*, pages 251–254, 2015.
- [31] R. J. Rost, B. Licea-Kane, D. Ginsburg, J. Kessenich, B. Lichtenbelt, H. Malan, and M. Weiblen. *OpenGL shading language*. Pearson Education, 2009.
- [32] C. Sá. *What is an Interface? The Entification and Identification of the Interface as a Mediation Complex*. PhD thesis, Catholic University of Porto, 2011.
- [33] G. Weinberg. Interconnected musical networks: Toward a theoretical framework. *Computer Music Journal*, 29(2):23–39, 2005.
- [34] M. Williams, C. Benfield, B. Warner, M. Zadka, D. Mitchell, K. Samuel, and P. Tardy. Push data to browsers and micro-services with websocket. In *Expert Twisted*, pages 285–304. Springer, 2019.
- [35] S. Wilson, D. Cottle, and N. Collins. *The SuperCollider Book*. The MIT Press, 2011.
- [36] T. Winkler. *Composing interactive music: techniques and ideas using Max*. MIT press, 1998.
- [37] M. Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.
- [38] R. Zimmermann, E. Chew, S. A. Ay, and M. Pawar. Distributed musical performances: Architecture and stream management. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 4(2):14, 2008.

Csound Web-IDE

Steven Yi
Rochester Institute of
Technology
syyigm@rit.edu

Hlökkver Sigurðsson
Berlin, Germany
hlolli@gmail.com

Edward Costello
Music Department
Maynooth University, Ireland
edward.costello@mu.ie

ABSTRACT

Csound Web-IDE¹ is an open-source², browser-based integrated development environment (IDE) for sound and music computing using Csound[4]. The web application offers users the ability to edit and run standard, multi-file Csound projects in the same way they would do on the desktop, mobile, and embedded platforms. Enabled by modern web technologies, envisioned use cases for the Web-IDE include computer music education, music composition, and development of realtime interactive systems, as well future integration with other Web Audio-based systems.

Keywords

Csound, Web IDE, Web Audio, Web MIDI

1. INTRODUCTION

As the capabilities of the modern web platform continue to grow, so too does the ability to develop more complex multimedia software applications once only possible on traditional native platforms. Modern web APIs such as the Web Audio API and the WebAssembly runtime make it possible to create, or port existing libraries to build complex audio processing and music composition software systems. Also, the increasing availability of mature UI frameworks built on top of existing web technologies, such as Angular and React, have also greatly streamlined the development of elaborate and responsive user interfaces.

Additionally, the web platform coupled with cloud based database/application platform services such as Google's Firebase³ also offers a robust means of application distribution, document storage and online collaboration tools. This confluence of technologies has allowed us to develop a fully featured IDE—called *Csound Web-IDE*, shown in Figure 1—for the Csound language which in many ways extends the capabilities of previous Csound-based environments by leveraging the unique power of the web-based software stack.

¹<https://ide.csound.com>

²<https://github.com/csound/web-ide>

³<https://firebase.google.com/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

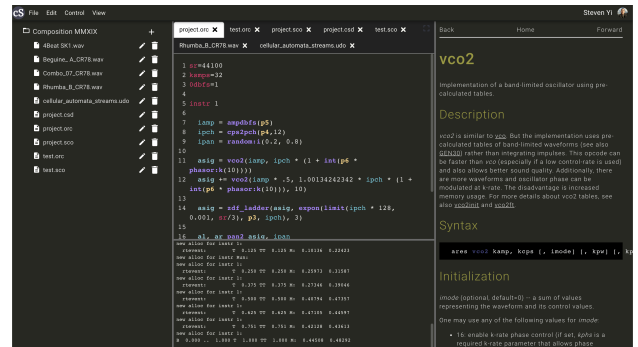


Figure 1: Primary Csound Web-IDE interface showing project file-tree and code editor.

Employing the Web Audio Csound library as the basis of the IDE enables users to be able to write, compile, and run Csound compositions on any computer with a modern web browser without installing any additional software or libraries. It also allows for the controlling of Csound-based synthesizers using the Web MIDI API and to continually recompile and update running Csound code via live coding. Users can also store their Csound instruments and compositions online giving them the ability to share and collaborate on documents with other participants. These capabilities greatly reduce any friction involved with using the Csound language and also potentially creates an environment which will allow users to easily distribute, incorporate into their own projects, and learn from an online library of Csound based instruments, effects and compositions.

2. RELATED WORK

Traditional desktop IDEs, such as Visual Studio⁴ and XCode⁵, operate within the context of their platforms where projects are generally made up of files that are organized into directories that all exist on a filesystem. IDEs like these provide development experiences where users may edit, compile, run, and debug projects all within a single application. Although the IDE may provide a unified interface to the user, they often depend upon secondary programs (e.g., compilers, linkers, debuggers) to perform operations.

Web IDEs operate much like their desktop counterparts but are developed using browser technologies. They fea-

⁴<https://visualstudio.microsoft.com/>

⁵<https://developer.apple.com/xcode/>

ture similar offerings of a unified interface for code-based projects and depend upon either server-side or client-side tools (e.g., compilers, audio engines) for executing those projects. Web IDEs offer additional features unique compared to desktop IDEs in that they are delivered over the network with zero-installation required, making them easy to deploy to end users (assuming they have a compatible browser). They also interoperate easily with servers to organize and persist projects for easy access by users across multiple systems. While recent developments of Web Audio and Web MIDI standards have allowed browser-based music applications to approach the level of features and performance found in desktop platforms, they are not quite as robust and performant as of yet. However, web applications on the browser platform offer solutions to satisfy unique use cases that traditional desktop IDEs can not do so easily or at all.

Within the Csound ecosystem, for Csound Web-IDE, we seek to create a comparable experience to the desktop CsoundQt⁶ and Cabbage⁷ applications. CsoundQt is a cross-platform IDE that works with projects on the filesystem, offers syntax-highlighting and code completion in its code editors for Csound code, uses Csound for rendering to disk and in realtime, and offers custom audio and MIDI I/O using audio and MIDI systems native to the desktop platforms. CsoundQt also offers tools such as audio scopes, UI widgets, and other features to provide a single working environment for Csound users. For Csound Web-IDE, the use cases and feature set of CsoundQt are the closest to what we are targetting for our initial phases of development.

Cabbage is a framework for audio software development that provides an IDE capable of running standalone as well as exporting binary audio plugins for VST and AU formats. Cabbage employs Csound as its audio engine via Csound's API and works within its target context, either generating audio directly to the desktop system's audio system or generating and processing signals within the host VST or AU API. Cabbage's primary standalone IDE features largely overlap with CsoundQt, though its support of industry plugin formats is unique in the Csound ecosystem. While supporting binary plugin generation is not a concern at this time for the Csound Web-IDE, we expect to have continued discussions with Cabbage's author in the future to see where compatibility with Cabbage's feature set and projects can be achieved.

Looking at web-based audio IDEs, the Faust Online Compiler^[5]⁸ (FOC) provides a browser-based web application for editing, compiling, and running Faust code. Originally a server-based solution where the compiler would run on the server-side, the system evolved to use a WebAssembly Faust compiler to run completely client-side. The Web Audio backend for the Faust compiler allows for realtime testing of Faust code for signal processing. The Faust Online Compiler simplifies development for users who may find setting up the Faust toolchain difficult on the desktop by providing a zero-install solution that is capable of working with single-file Faust projects. However, it does not save projects to the server, nor supports multi-file projects.

SOUL Playground⁹ is an online environment for devel-

oping and testing single-file projects using the SOUL¹⁰ domain-specific language for audio programming. Similar in features to the Faust Online Compiler, the playground allows compiling SOUL projects in the browser and auditioning the results using Web Audio and Web MIDI. The SOUL playground does not have import/export features like FOC, but does have a global server-side saving system that is not tied to accounts. Users can press the "Compile and Run" option to audition results, then use the "Share this playground" button to provide a link to a saved project. The ability to share projects easily through web URLs promotes a socially-rich online culture for an ecosystem and is one that we plan to implement for the Csound Web-IDE.

Looking at non-audio web IDEs, a number of systems provide multi-file project support. CodePen¹¹ describes itself as a "social development environment" and provides a system for multi-file projects with a fixed-set of files (CodePen *Pens*, made up of a single HTML, JS, and CSS file) as well as an user-definable set of directories and files (CodePen *Projects*). Both systems provide editors for files of different types, though Projects allow the user flexibility to add as many files and directories as desired. Both systems also provide options to export the project from CodePen into a zip that can be expanded and further edited and used from a desktop system. In addition to the development environment, projects can be made either public or private as well as *forked* for modification by other users. CodePen's Projects is a fully-featured IDE that compares well with desktop IDEs and maintains projects using a virtual filesystem that compares well the desktop IDE development experience. The Csound Web-IDE seeks to provide a similar feature for social development set as CodePen Projects but targetting Csound audio and music work rather than client-side web development.

Glitch provides both client-side and server-side development using multi-file projects served from per-user virtualized containers. Glitch touts easy *remixing* of projects (similar to CodePen's *forking* system), saved project history backed by Git, as well as realtime collaborative editing through its Team system. While we are not looking at these features for the current phase of Csound Web-IDE development, we do look to Glitch as a model of features that we would like to implement in future iterations of our project.

JSFiddle¹² is a fixed-set, multi-file web IDE, similar to CodePen's Pens system. JSFiddle provide easy, quick experimentation for small experiments that does not require logging in; boilerplates as starting points for projects; project history and forking; as well as easy inclusion of many popular third-party JS libraries via entering a URL or searching for them through a registry. JSFiddle's anonymous experimentation and importing of libraries are features we are interested to implement for Csound Web-IDE.

Finally, the p5.js Web Editor¹³ offers a multi-file, filesystem-based IDE for p5.js projects. Projects are all public but only accessible if one is given a URL link from its author. The Web Editor provides an easy-to-use environment for development and teaching with p5.js that serves similar goals as what we envision for the Csound Web-IDE and the

⁶<http://csoundqt.github.io/>

⁷<http://cabbageaudio.com/>

⁸<https://faust.grame.fr/onlinecompiler/>

⁹<https://soul.dev/playground/>

¹⁰<https://soul.dev/>

¹¹<https://codepen.io/>

¹²<https://jsfiddle.net/>

¹³<https://editor.p5js.org/>

Csound community.

3. PROJECT GOALS

Our goals for the Csound Web-IDE are to provide a zero-install, easy to use development environment for Csound. We believe a browser-based web application will provide many of the same benefits as using a desktop-based application, such as CsoundQt, while also providing additional features that will allow it to reach a broader audience. Use cases we considered include:

- Music composition
- Procedural Sound Generation
- Realtime, interactive audio system development
- Computer music pedagogy
- Cross-platform Csound project development

To serve those use cases, the Csound Web-IDE is currently designed to include the following features:

- Multi-file projects with support for audio assets
- Code editor with syntax highlighting and other editing features
- Project rendering in realtime as well as to disk
- Live code evaluation
- Support audio and MIDI processing through Web Audio and Web MIDI APIs
- Signal scoping (i.e., waveform and spectrum views)*¹⁴
- User-defined graphical user interfaces*
- Project collections*
- Public project and collections links making projects easy to share*

By implementing the above, the Csound Web-IDE should be an application where users may create or import projects; develop projects within the IDE; share projects with others; and export projects to be used on other platforms that support Csound (i.e., desktop, mobile, and embedded systems). Also, by targeting the browser platform, we see potential for using the IDE in places where installation of desktop software is difficult, if not impossible, to manage. Scenarios include image-based lab computers (such as found at educational institutions) and Chromebooks¹⁵ running Google's Chrome OS.

4. ARCHITECTURE

The following describes the architecture of the Csound Web-IDE application. It describes technologies used; project data representation; and interaction with Csound, Web Audio, and Web MIDI for realtime rendering.

¹⁴Items marked with asterisks are not yet implemented as of time of this writing and are planned for later phases of development.

¹⁵<https://www.google.com/chromebook/>

4.1 Technologies

4.1.1 Server

The Csound Web-IDE is built using Firebase as a backend. Rather than develop our own bespoke server application, we chose to follow a *serverless* design and chose Firebase to fulfill our requirements for authentication, document storage, binary file storage, computation, and web serving services. These requirements are met using Firebase's Authentication, Cloud Firestore, Storage, Cloud Functions, and Hosting modules respectively.

The choice to delegate server functionality to Firebase was made primarily due to ease of development and cost. In this early stage of development, we are currently working under the limits of the Spark Plan, a free-to-use tier meant for small projects. Usage and cost over time will factor into future decisions whether to continue using Firebase under paid plans or to migrate to a different solution.

4.1.2 Client

The Csound Web-IDE client is built with various Javascript libraries using Typescript. React¹⁶ and Redux¹⁷ handle the user-interface state and rendering logic. As with many modern Javascript projects, we depend on many libraries from the NPM package manager. MaterialUI¹⁸ provides many pre-styled components like buttons, menus and lists. The code editor itself is based on CodeMirror¹⁹, the tab system on GoldenLayout²⁰ and FirebaseUI²¹ provides us with ready made components for login. We use Web Audio Csound[8], a C library and an interface compiled with Emscripten[9] into JavaScript and WebAssembly, as our audio engine. This version of Csound is compiled from the same source as all of the other versions of Csound (i.e., desktop, mobile, embedded) and comes with additional JS library code that manages connections between Web Audio and Web MIDI to Csound.

While our primary goal is to deploy the client using standard web browser loading of applications over the internet, we have seen that the Web-IDE client works equally well as an Electron²² desktop application. By using Electron we have full access to a modern browser engine (Chromium) which can run Web Audio Csound. Electron also provides full access to NodeJS which gives us the possibility to use either Web Audio Csound or NodeJS native-bindings to desktop Csound. Further research of the pros and cons in using native bindings is necessary to determine the value of providing this option.

4.2 Project Data and File System

Historically, Csound projects were made up of files organized into directories. A primary .orc and .sco file or single unified .csd file was used as the entry point into a project. Additional code files may be included with Csound's pre-processor and projects may employ binary assets (e.g., audio files, text data tables). The project was loaded at the start

¹⁶<https://reactjs.org/>

¹⁷<https://redux.js.org/>

¹⁸<https://material-ui.com/>

¹⁹<https://codemirror.net/>

²⁰<https://golden-layout.com/>

²¹<https://opensource.google.com/projects/firebaseui>

²²<https://electronjs.org/>

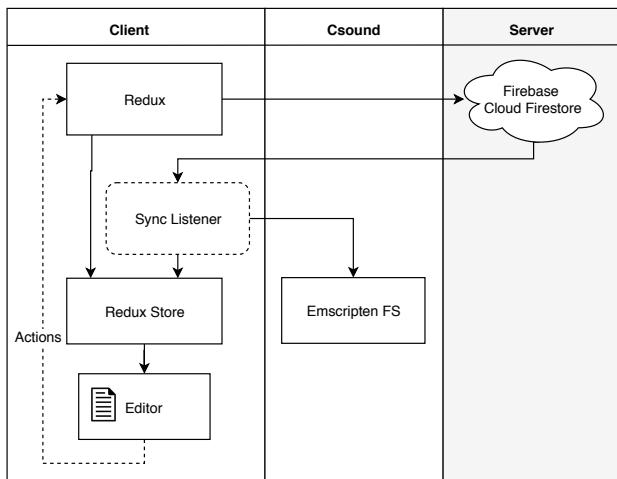


Figure 2: Diagram showing flow of project data. The Redux Store is the primary in-memory store for project data on the client. Redux Actions from the application or changes coming from Firebase mutate the Store which is synced to the client via React. Firestore changes, via a sync listener, update the Emscripten FS and Redux Store.

of Csound execution. With the introduction of the Csound API in Csound 5 and its further development in Csound 6, Csound evolved into a system that could be used as a server that begins execution without a project and where code would be evaluated live at runtime. This allowed the source of Csound project data to be determined by the host application which could, for example, store data in non-filesystem stores, such as databases, or use bespoke data file formats.

For Csound Web-IDE, we decided to follow the traditional model of project representation as a set of files and directories. Although the implementation uses in-memory datastores as a virtual filesystem, the overall effect is that the project data model appears to the user in the same way as they would experience when working with Csound on other platforms and in working with other Csound IDEs. We believe this will simplify onboarding for experienced Csound users and that the filesystem model would be easily understood by others with programming experience in other languages. Using a filesystem representation was also essential to permit Web Audio Csound to operate using C file I/O functionality with Emscripten FS, meaning no platform-specific code would be necessary in Csound’s source code for the browser platform.

Figure 2 shows an overview of how project data is stored within the Csound Web-IDE application. The primary source of truth for the client-side application is the Redux Store. Project data is loaded from Firebase (used for long-term persistence) into the store and also synced to the Emscripten FS via a synchronization listener. User edits to the project trigger Redux Actions which may directly modify the store or first write to Firebase which then triggers the synchronization flow. Once files are written to Emscripten FS, Csound can load and interpret those files.

For the user, the presentation of data appears much as it

would on a desktop IDE. The project data files and organization is shown using a file-tree. Selecting files from the tree opens up filetype-specific editors in the main editing area for the Web-IDE. Users can use the file-tree to perform standard filesystem operations such as creating, renaming, and moving files and directories. Additionally, projects have permissions associated with them that limit accessibility only to the user (private) or permit visibility to the world (public). Using a filesystem representation in Csound Web-IDE offers a well-known paradigm for Csound users. It also permits easy exporting of projects for use with Csound on other platforms and importing into the Csound Web-IDE system from those platforms.

4.3 Csound, Web Audio, Web MIDI

Originally, Csound operated strictly by rendering to disk. Csound would load projects that define instrument definitions and score data that is driven by a scheduler to trigger events (e.g., activate an instrument instance at a given time for a given duration with x additional parameters). After interpreting the project data and loading data into memory, Csound would then run to completion and exit. Users audition the resulting audio file and iterate in this cycle of editing, rendering, and testing the project until they were satisfied with the results.

Later, when realtime support was added[6], Csound operation changed to render to DAC and allow auditioning in real-time. This change also allowed for realtime input and output through text, GUI interfaces, audio signals, and MIDI data. I/O with external systems was provided with Csound via plugins. Users would select input and output devices when executing Csound and, within their projects, use Csound opcodes to read and write data from external sources. They could write code generically that would work with any kind of audio and MIDI data that was routed to/from Csound through the plugin drivers. This allowed them to write a project that could function on a desktop system as well as in the case of Cabbage—within the context of hosted VST and AU plugins. Additionally, in Csound 6[2], the API and engine were extended to support runtime evaluation of both Csound ORC and SCO code, opening up the possibilities of live coding with Csound.

Over time, Csound usage has grown to cover many use cases including music composition, sound design, development of realtime synthesizers, interactive audio applications, and more. Since the Csound library used in this project is compiled using the same C code as desktop, mobile, and embedded versions of Csound, it supports the same use cases as found on those other platforms.

The following discusses the usage of Csound in the Csound Web-IDE and interaction with Web Audio and Web MIDI APIs for realtime rendering. (Ahead-of-time rendering to disk and user-defined GUI I/O support is planned for the Web-IDE, but not yet implemented; see Section 5 for further discussion.)

4.3.1 Realtime Render

The Csound audio engine begins rendering when a user presses the “Play” button. At this time, messages are sent to Web Audio Csound to begin rendering using the file marked as the “main” project file. Users may additionally click on a file on the file-tree to change what is the “main” project or right-click on a different Csound file to execute as the

starting point of the project.

During the initialization phase, the CsoundObj API—provided by Web Audio Csound—will create a Web Audio AudioNode that is either an AudioWorkletNode or ScriptProcessorNode, depending upon what the browser supports. The node wraps a running Csound instance, reads incoming samples from Web Audio and transfers them to the instance, executes Csound enough times to fill the Web Audio buffer size, then writes outgoing samples from Csound back to the Web Audio node graph.

Csound then starts by reading files from the Emscripten FS, starting from the main file and moving through `#include'd` code files and binary assets. Once the code is read and interpreted, realtime rendering begins and continues until completion. If a project is designed in the traditional way, completion occurs when the event scheduler is clear of any pending events and Csound's audio graph is clear. If a project is designed for realtime processing, the project may run continuously until a user action explicitly turns off Csound.

4.3.2 MIDI Input

While a Csound project is rendering, users may route MIDI input from WebMIDI devices into Csound. The MIDI data may be used for instantiate notes or sending controller messages for interpretation by Csound user-code. This is done using CsoundObj's provided utility methods for instantiating and routing Web MIDI to Csound.

4.3.3 Live Coding

In addition to realtime audio and MIDI input, Csound Web-IDE's editors support live coding by realtime evaluation of both ORC and SCO code. This is done by sending selected code from the editor as text to CsoundObj which in turn routes text as arguments to one of the appropriate Csound API functions for code evaluation. This action of evaluation will bypass triggering changes to the persistent storage in both Firestore and Emscripten FS.

Evaluating Csound code at runtime permits users to modify definitions of instruments and user-defined opcodes as well as auditioning score statements. Updating a running system often yields a faster development experience compared to the classic edit, render to completion, and test cycle when composing music with Csound. Runtime evaluation also permits live coding performances with Csound Web-IDE. For performance, users may create a basic project that contains library code and audio assets they wish to use, start Csound, then use a blank editor as a starting point for live coding their performance.

Using the Csound Web-IDE for live coding, together with audio and MIDI input, covers a large number of use cases where Csound has traditionally been used in realtime on the desktop. It is a testament to the state of browser technologies that a system like Csound could be supported via WebAssembly, Web Audio, and Web MIDI, to provide a cross-platform, zero-install Web-IDE that can function much like its desktop counterparts.

5. FUTURE WORK

Csound Web-IDE is currently in an alpha-state of development. We plan to finish implementing the features mentioned in Section 3 before opening a public beta for testing and feedback. For the first release, the feature set is primarily

focused on individual usage of the IDE. Social development features, history tracking, and user-defined graphical user interface development are planned for future rounds of development.

Current work with binary assets has been limited in scope to sizes smaller than 1 megabyte. Since Web Audio Csound (as well as other versions of Csound) works with compressed files in OGG and FLAC formats, we believe the system should be able to handle any pedagogical requirements and a large number of use cases for creative purposes in the short-term. Further research is required for both maximum number of assets and maximum size of assets, both in terms of memory usage on the client-side as well as cost to support on the server-side. We believe this area of research will be one applicable to others developing Web Audio applications, particularly those using Emscripten FS.

Ahead-of-time rendering to disk is planned using a Web-Worker. Csound will write to Emscripten FS and the user will be able to download the rendered audio file as a Blob. We plan to include this functionality in the initial release of the Web-IDE.

The current editor supports syntax highlighting but does not yet support other commonly found features in IDEs. Initial work on documentation browsing and lookup of individual opcode entries of the Canonical Csound Reference Manual[7] is currently implemented in the Web-IDE and serves as the basis for future work on code completion for opcodes. Example, template, and tutorial projects are also on the roadmap for development.

In addition to the Electron-based desktop client, we plan to explore offering Csound Web-IDE as a Progressive Web Application²³. This option has already been explored with success for another Web Audio Csound-based application, *csound-live-code*²⁴, where it is installable as a desktop and Android application using the Chrome browser. We hope to show that browser-based audio applications can serve in many places traditionally handled by native desktop and mobile applications.

Once user-defined GUI components and the GUI editor are implemented, we plan to explore making Csound Web-IDE projects be publishable as Web Audio Modules (WAM)[3] and Web Audio Plugins (WAP)[1]. Offering Csound-based plugins that interoperate with other Web Audio-based applications would open up new areas where Csound may be used for musicians and developers.

Exploring interoperability with other Csound-based applications and platforms will require collaboration with developers of CsoundQt and Cabbage as well as the larger Csound community. We are excited to see where our web-based system can compliment the capabilities of existing desktop IDEs.

Finally, we are excited to see how the Csound Web-IDE can serve education for both Csound and computer music. We are hopeful that public collections of projects, together with social development features, can be a vehicle for development of content suitable for education using an active learning approach.

6. CONCLUSIONS

²³<https://developers.google.com/web/progressive-web-apps/>

²⁴<https://live.csound.com/>

The Csound Web-IDE provides a web-based development environment for Csound sound and music computing projects. This was implemented using Firebase for the server-side and numerous well-known libraries and frameworks for the client-side application. Developing a full-featured IDE for Csound on the browser platform was made possible due to Emscripten’s FS filesystem—employed by Web Audio Csound—as well as using Web Audio and Web MIDI APIs. Social development features found in other web-based IDEs are planned to encourage networking, education, sharing, and discovery of Csound work for users.

We hope the IDE proves useful for both creative and pedagogical purposes. We look forward to expanding upon the system over time to serve both computer music creators and educators in their work.

7. REFERENCES

- [1] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, G. Pellerin, and S. Letz. WAP: Ideas for a Web Audio Plug-in Standard. In *4th Web Audio Conference, TU Berlin, Berlin*, 2018.
- [2] J. P. fitch, V. Lazzarini, S. Yi, M. Gogins, and A. Cabrera. The New Developments in Csound 6. In *International Computer Music Conference 2015*, 2015.
- [3] J. Kleimola and O. Larkin. Web Audio Modules. SMC, Maynooth University, 2015.
- [4] V. Lazzarini, J. fitch, S. Yi, Ø. Brandtsegg, J. Heintz, and I. McCurdy. *Csound: A Sound and Music Programming System*. Springer, Berlin, 2016.
- [5] R. Michon and Y. Orlarey. The Faust online compiler: a web-based IDE for the Faust programming language. In *Proceedings of the Linux Audio Conference (LAC-12)*, pages 111–116, 2012.
- [6] B. Vercoe and D. Ellis. Real-time CSound: Software Synthesis with Sensing and Control. In *International Computer Music Conference 1990*, 1990.
- [7] B. Vercoe et al. The Canonical Csound Reference Manual, 2019.
- [8] S. Yi, V. Lazzarini, and E. Costello. WebAssembly AudioWorklet Csound. In *4th Web Audio Conference, TU Berlin, Berlin*, 2018.
- [9] A. Zakai. Emscripten: An LLVM-to-JavaScript Compiler. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, OOPSLA ’11*, pages 301–312, New York, NY, USA, 2011. ACM.

Bringing the TidalCycles Mini-Notation to the Browser

Charles Roberts

Interactive Media & Game Development
Department of Computer Science
Worcester Polytechnic Institute
charlie@charlie-roberts.com

Mariana Pachon-Puentes

Department of Computer Science
Worcester Polytechnic Institute
mpachonpuentes@wpi.edu

ABSTRACT

TidalCycles has rapidly become the most popular system for many styles of live coding performance, in particular Algoraves. We created a JavaScript dialect of its mini-notation for pattern, enabling easy integration with creative coding tools. Our research pairs a formalism describing the mini-notation with a small JavaScript library for generating events over time; this library is suitable for generating events inside of an AudioWorkletProcessor thread and for assisting with scheduling in JavaScript environments more generally. We describe integrating the library into the two live coding systems, Gibber and Hydra, and discuss an accompanying technique for visually annotating the playback of TidalCycles patterns over time.

1. INTRODUCTION

In canonical live coding performances, performers create artistic output by programming it in front of an audience, often projecting their source code for the audience to follow [21]. Such performances have increased in popularity over the years; for example, organizers have promoted hundreds of concerts in one genre of live coding performance, Algorave[4], where performers take turns live coding (predominantly) dance music. The browser is a popular platform for developers creating live coding systems, with many environments that primarily target music, visuals, and choreography, in addition to hybrid systems enabling performers to generate both audio and visual content (see Section 2). Our research takes the pattern mini-notation of TidalCycles[11] and brings it to JavaScript as a library that can be incorporated into browser-based live coding environments. The TidalCycles mini-notation is a terse and expressive way to describe patterns of values over time, and we hope that bringing this popular representation to the browser will encourage adoption across many systems. Section 4 describes some of the integrations we and other developers have created to date.

But first we begin by describing the state of live coding in the browser, alongside a brief introduction to TidalCycles. We describe our implementation of the parser and accom-

panying library, and their integration into various browser-based creative coding systems. We then discuss techniques for visually annotating the playback of TidalCycle patterns, taking advantage of the browser's flexible markup capabilities. We conclude with a short discussion of open questions, and possible directions for future work.

2. BACKGROUND

Live coding is growing in popularity, with new systems introduced every year and an growing number of local interest groups and performances. The browser has become a popular target for many live coding systems, thanks to its ubiquity, powerful graphics capabilities, and increasingly powerful audio features. In this section we discuss existing live coding tools and their notations for music—with an emphasis on TidalCycles—and the state of current browser-based live coding systems.

2.1 TidalCycles

TidalCycles is a domain-specific language embedded within Haskell [11]. It has proven to be one of the most popular choices for live coding, particularly in the Algorave genre, perhaps due to its strong emphasis on time and rhythm [10]. For example, in a recent four-day streamed live coding event, sixty-five of approximately one hundred and sixty performances primarily used TidalCycles[19]. The next most popular end-user language in the event was SuperCollider[9], with eighteen live coding performances¹.

TidalCycles outputs OSC or MIDI messages, and performers have used it to control a wide variety of applications. It is most often used to remotely control *SuperDirt*, an audio sample playback and manipulation engine written in SuperCollider.

2.2 Live Coding and Patterns

There are a variety of techniques for generating patterns in live coding systems. In *ixi lang*[6], a mini-language that controls the SuperCollider audio server, patterns (or *scores* in *ixi lang* terminology) are defined using strings, with the location of each token determining the timing of notes and sample playback. In both Gibber[15] and in Conductive[3], a Haskell live coding environment where the programmer controls semi-autonomous agents, there are separate patterns for values in sequences and controlling the timing of

¹Although live coding in the SuperCollider language is arguably not as popular as live coding in TidalCycles, many live coding systems provide an alternative language that uses the SuperCollider audio server for synthesis and processing.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

sequence output. For example, in Listing 1, the pattern `[0,1,2,3]` selects notes from a default global scale while the pattern `[1/4,1/2]` specifies that these notes will have alternating durations of a quarter note and a half note.

```
syn = Synth()
syn.note.seq( [0,1,2,3], [1/4,1/2] )
```

Listing 1: A sequence in Gibber

Other systems opt for more verbose temporal recursions (functions that invoke themselves over time[17]) or coroutines that provide low-level control of timing and output. For example, the *Extempore* and *Impromptu* systems use verbose temporal recursions to generate patterns, often based on sampling continuous waveforms [18]. While these recursions are lengthy to type by hand (at least in the context of a live performance), the environments include text-editing macros that make insertion and editing fast and fluid. Sonic Pi[1] uses a terser design similar to a coroutine named the `live_loop`, where loops can sleep and are editable. Gibber and Gibberwocky[13] also support temporal recursions, in addition to their shared pattern affordances.

The mini-notation in TidalCycles is based on the Bol Processor (BP2)[2]. Polyrhythms, polymeter, repetition, and rhythmic subgrouping can all be described quite tersely. For example, the pattern string `'[kd sd, ch ch oh]'` describes a pattern where a kick drum and a snare drum are each half a *cycle* in duration, while the closed hihat and open hihat sounds are each a third of a cycle, creating a two against three polyrhythm. The expression of both timing and output values in the same text string provides a terseness that lends itself well to live coding. In our experience developing and performing with Gibber, we found that while defining output and timing separately has its merits, we often wanted the expressiveness of a combined notation similar to the TidalCycles mini-notation.

2.3 Browser-based Live Coding Environments

There are at least a dozen browser-based environments for live coding performance². Some are geared heavily towards audio³, while others emphasize graphics [5, 16], and at least one targets choreography⁴. Our hope is that the research presented here will be broadly applicable to a variety of such systems. In this vein, we discuss the integration of our library into two browser-based live coding environments, Hydra and Gibber, in Section 4.

Of particular relevance to this paper is *Estuary*[12], which provides a projectional editing interface for TidalCycles (in addition to other live coding systems), and runs in the browser. The authors developed the software in Haskell, which was then compiled into JavaScript. One notable advantage of this approach is that almost all of the features of TidalCycles were immediately available to the authors, while our approach requires implementing each of TidalCycles' features individually. For this reason, many features in our library are still in development / discussion, as described in Section 5. A comparative advantage of our library is that

²for a fairly comprehensive list of live coding systems, see <https://github.com/toplap/awesome-livecoding/blob/master/README.md>

³<https://live.csound.com/>

⁴<https://github.com/sicchio/terpsicode>

it is easy to incorporate into browser-based projects without requiring the use of a Haskell compiler⁵, potentially easing adoption into other systems.

3. IMPLEMENTATION

We created two modules, designed to work in tandem, enabling the use of the TidalCycles patterns in the browser. The first module is a *parser* for the TidalCycles mini-notation. This parser was written using the *parsing expression grammar* formalism (or PEG); we have prior experience using this formalism to teach workshops on live coding language design [20]. The PEG.js library⁶ translates the TidalCycles grammar into a parser that generates appropriate JavaScript data structures for *querying*, which is handled by our second module. Following the general structure of the TidalCycles library, this module exposes a `queryArc` function which accepts three arguments: a pattern, a start time, and a duration⁷. From these three arguments a list of *values* and *timestamps* is generated; the timestamps are offsets from the starting phase argument passed to `queryArc` and the values typically correspond to either indices in a scale denoting a pitch to be played or an identifier for a sample/sound to be triggered. The code example below shows these two modules in use employing the CommonJS module syntax.

```
parser = require('./dist/tidal.js')
queryArc = require('./src/queryArc').queryArc

pattern = parser.parse( '0 [1 2]' )
events = queryArc( pattern, 0, 1 )
/*
 * events = [
 *   { value:0, arc:{
 *     start:Fraction(0), end:Fraction(1,2) }
 * },
 *   { value:1, arc:{
 *     start:Fraction(1,2), end:Fraction(3,4) }
 * },
 *   { value:2, arc:{
 *     start:Fraction(3,4), end:Fraction(1) }
 * }
 * ]
 */
```

Listing 2: Using the parser and query function

We combined the two modules together into a single `Pattern` object for easier use. The `Pattern` constructor accepts an argument pattern written using the TidalCycles mini-notation; the generated pattern object can then be queried by passing a starting phase and a duration to its `query` method. The `Pattern.query` method also sorts its output by the temporal proximity of each event to the start time of the query. Listing 3 shows this in action, assuming that the `Pattern` object has been imported into the global namespace.

⁵Despite the unfamiliarity of Haskell for most web developers, the authors of Estuary make many interesting arguments for the security and type safety of Haskell in browser-based projects.

⁶<https://pegjs.org>

⁷In TidalCycles the combination of a start time and a duration is known as an `Arc`

```
const pattern = Pattern('0 [1 2]')
const events = pattern.query( 0, 1 )
// same event output as Listing 1
```

Listing 3: Use of Pattern object

In addition to these two modules and the `Pattern` object, the repository also contains a number of demos, and a set of over eighty unit tests covering the parser.

4. INTEGRATION

A demonstration using the `Pattern` object described in Sec. 3 with an HTML `<canvas>` element is included in the repository for the project and shown in Fig. 1. This figure draws the pattern `'0 [1 2]*2 <3 4 5> [6 [7 8]]*4 9]*5'`, where each number⁸ is assigned to represent a different color. The demo queries a user-provided pattern for a selected number of cycles, and then loops through all of the generated events drawing rectangles based on the duration and value of each event.

We have also successfully integrated this library into two live coding environments: Olivia Jack’s Hydra and Gibber, developed by the first author. Developer/performer Diego Dorado also integrated the library into his browser-based live-emojizing playground⁹. In this playground, emojis are used as TidalCycles groups. A pattern such as `'[🍌*2 🍌*2?]*'` can be interpreted as a TidalCycles pattern, where each emoji corresponds to a predetermined sound. Dorado’s work uses `Tone.js`[7] for sound generation. Our repository also includes an audiovisual representation of TidalCycles patterns written with `p5.js`[8]. These two examples help demonstrate that the integration with common JavaScript libraries is relatively simple, enabling developers to bring the TidalCycles mini-notation to various browser applications.



Figure 1: Using the library in conjunction with the HTML `canvas` object. In this example, numeric values are assigned to colors and patterned using the TidalCycles mini-notation. The `*5` at the end of the visualized pattern causes the entire pattern to repeat five times.

4.1 Integrating with Hydra

Hydra¹⁰ is a popular browser-based environment for live coding visuals using the metaphor of analog video synthesis. When creating a video synthesis object, Hydra enables coders to assign functions to parameters instead of numbers;

⁸The exceptions are numbers to the right of the `*` operator, which instead controls repetition

⁹<https://diegodorado.com/en/labs/live-emojizing>

¹⁰<http://hydra-editor.glitch.me/>

these functions are then evaluated every frame to determine the parameter’s value. In order to integrate our `Pattern` object into Hydra, we first import the `pattern.js` file included in the source code repository, and then create a function that accepts TidalCycles mini-notation as an argument. Invoking this function creates a function that updates an internal phase, queries an associated pattern for events, and returns values generated by the query. The code for the `Tidal` function shown in Fig. 2 (not counting on the included `Pattern` module) is given in Listing 4.

```
Tidal = function( pattern ) {
  let value = null,
      phase = 0,
      phaseIncr = 1/120,
      events = [],
      end = -Infinity

  const p = Pattern( pattern )

  const out = function() {
    phase += phaseIncr

    if( events.length <= 0 && phase > end ) {
      phase = 0
      events = p.query( phase, 1 )
    }

    const next = events[0]
    if( next !== undefined
        && phase > next.arc.start.valueOf() ) {
      const event = events.shift()
      value = event.value
      end = event.arc.end.valueOf()
    }

    return value
  }

  return out
}
```

Listing 4: Hydra Integration

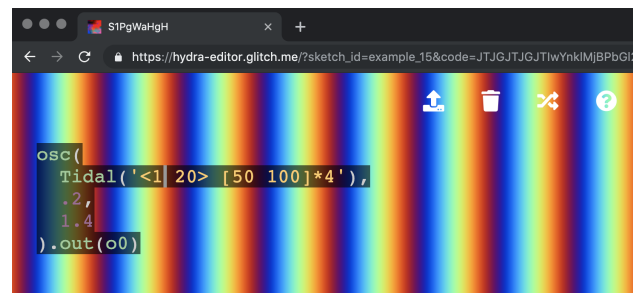


Figure 2: Using the function from Listing 4 to create time-based patterns in Hydra.

4.2 Integrating with Gibber

Gibber is a browser-based tool for live coding both music and ray-marched visuals. Our integration enables creative coders to use the Tidal mini-notation to create patterns that can be applied both domains; the TidalCycles mini-notation can be used to generate musical patterns as well as control 3D geometries. We additionally provide a unique, animated

annotation system that marks individual tokens in the mini-notation and reveals when events associated with each token are triggered.

4.2.1 Integrating with the AudioWorkletProcessor

Audio in Gibber runs in an AudioWorkletProcessor thread; this includes all sequencing and scheduling, so that these systems can also be modulated using audio-rate signals. The use of the `Pattern` object needs to be as efficient as possible to fit in the tight time constraints specified for the `AudioWorkletProcessor` by the Web Audio API (128 samples per buffer). At first, we encountered problems with the speed of parsing patterns with deeply nested tokens; however, after specifying that the parser should cache results, these problems disappeared at the slight expense of additional processing time for the caching itself. The current unit test suite of over eighty tests completes in an average of 42 ms, indicating an average test time of about half a millisecond. We are optimistic that we can optimize the grammar to improve these results, but are also relatively unconcerned given that parsing—which typically takes much more time than querying—only occurs when a user creates a new pattern, a relatively infrequent event.

4.2.2 Annotating TidalCycles Patterns

One relatively unique feature in Gibber is the use of animated annotations to reveal system state within the source code editor. These annotations and visualizations can take many forms, from changing the source code itself, to highlighting tokens in the editor, to using HTML canvas-based visualizations of modulation signals that animate over time.

In order to annotate TidalCycles patterns in a similar fashion we extended our parser to assign a unique identification number to each token depicting a value, and to also include the each token's location in the code editor. With this information we can create a `` element wrapping the token, and assign a unique CSS class to it based on its ID number. This enables us to always know the precise location of each token in our editor, even as the code is actively being edited¹¹.

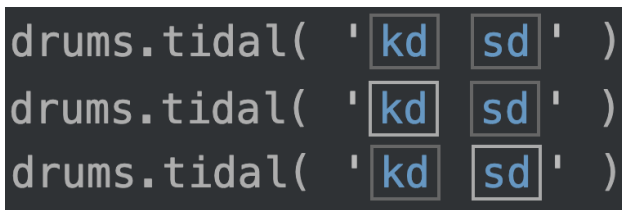


Figure 3: A simple TidalCycles pattern with three different annotated/animated states of activity. In the first line of code, no sounds are being triggered. In the second line, the kick drum (kd) has just been triggered, while the snare drum has been triggered in the final line.

The highlighting effect, in our opinion, is subtle, and while it is difficult to portray here in a static document, when animated the flashes of activity are clear and visible. Although we could use an annotation with higher contrast, our past research has indicated that many people prefer such anno-

tations to be less distracting[14], so that they don't distract from the act of programming.

```
verb = Bus2('spaceverb')
delay= Bus2('delay,1/6')
Tidal.cps = 140/120/2
drums = EDrums().connect( verb, .05 ).connect( delay, .1)

drums.tidal(`
  <kd kd*2>
  [sd kd]
  ~
  <[kd ~ sd kd]
  [kd sd]
  [kd sd sd]>
  <ch ch*2>
  <cp cp ~ cp*2 ~ kd>
`)

bass = Monosynth('bass')
bass.note.tidal( `[<0 7 -7>] 3 ~ 4 ~ [<7 0>]*2` )

pad = PolySynth('bleep').connect( verb, 1 )
pad.chord.tidal( `[0,2,4,5]` )
```

Figure 4: Multiple annotated sequences running concurrently. One kick drum token and two notes in the bass line are highlighted to indicate activity.

5. CONCLUSIONS AND FUTURE WORK

We have created a JavaScript library for parsing and querying the mini-notation of TidalCycles, and successfully integrated it into two live coding environments, Gibber and Hydra. Our research also described a novel technique for annotating the playback of TidalCycle patterns to potentially improve audience and programmer understanding of the TidalCycles mini-notation.

However, decisions and challenges remain. TidalCycles contains functions to both specify patterns and to transform them over time; these transformations are missing from our current implementation. Some are relatively simple to implement and have been included in our Gibber integration, such as shifting the placement of patterns by adding or subtracting from the current phase of the pattern. Other transformations are more difficult. In regards to integrating the mini-notation into Gibber, it is unclear whether it is preferable to use Gibber's existing pattern manipulation API to also transform patterns generated by the TidalCycles mini-notation, or if we should include a separate API that matches the TidalCycles for manipulating patterns.

Having a separate implementation of the TidalCycles mini-notation also raises the possibility of creating a new dialect. For example, the mini-notation currently doesn't have a mechanism for specifying the loudness of individual notes or sample triggers. We plan to add this to our library, creating a minor fragmentation between the two notations. Having the language available in JavaScript might encourage more developers to experiment with the grammar and add their own additional features; perhaps some of these features will eventually make their way back to the canonical Haskell implementation to the benefit of the greater community.

6. ACKNOWLEDGMENTS

Our thanks to Alex McLean and the TidalCycles community for helping us to understand the intricacies of pattern parsing in TidalCycles via online discussion.

¹¹Thanks to the exceptional <http://codemirror.net/> CodeMirror library for providing editing interface that supports this

7. REFERENCES

- [1] S. Aaron and A. F. Blackwell. From sonic pi to overtone: creative musical experiences with domain-specific and functional languages. In *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, pages 35–46. ACM, 2013.
- [2] B. Bel. *Rationalizing musical time: syntactic and symbolic-numeric approaches*, pages 86–101. Feedback Studio, 2001.
- [3] R. Bell. An approach to live algorithmic composition using conductive. In *Proceedings of LAC*, volume 2013, 2013.
- [4] N. Collins and A. McLean. Algorave: Live performance of algorithmic electronic dance music. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 355–358, 2014.
- [5] D. Della Casa and G. John. LiveCodeLab 2.0 and its language LiveCodeLang. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 1–8. ACM, 2014.
- [6] T. Magnusson. ixi lang: a SuperCollider parasite for live coding. In *Proceedings of the International Computer Music Conference*. University of Huddersfield, 2011.
- [7] Y. Mann. Interactive music with tone.js. In *Proceedings of the 1st annual Web Audio Conference*. Citeseer, 2015.
- [8] L. McCarthy, C. Reas, and B. Fry. *Getting Started with P5.js: Making Interactive Graphics in JavaScript and Processing*. Maker Media, Inc., 2015.
- [9] J. McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68, 2002.
- [10] A. McLean. Making programming languages to dance to: live coding with tidal. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 63–70. ACM, 2014.
- [11] A. McLean and G. Wiggins. Tidal-pattern language for the live coding of music. In *Proceedings of the 7th sound and music computing conference*, 2010.
- [12] D. Ogborn, J. Beverley, L. N. del Angel, E. Tsabary, and A. McLean. Estuary: Browser-based collaborative projectional live coding of musical patterns. In *International Conference on Live Coding (ICLC) 2017*, 2017.
- [13] C. Roberts and G. Wakefield. Live Coding the Digital Audio Workstation. In *Proceedings of the 2nd International Conference on Live Coding*, 2016.
- [14] C. Roberts, M. Wright, and K.-M. JoAnn. Beyond editing: Extended interaction with textual code fragments. In *Proceedings of the New Interfaces for Musical Expression Conference*, 2015.
- [15] C. Roberts, M. Wright, and J. Kuchera-Morin. Music Programming in Gibber. In *Proceedings of the International Computer Music Conference(ICMC)*, pages 50–57, 2015.
- [16] J. Rodríguez, E. Betancur, R. Rodríguez, and A. de México. Cinevivo: a mini-language for live-visuals. In *Proceedings of 2019 International Conference on Live coding*, 2019.
- [17] A. Sorensen. The many faces of a temporal recursion, 2013 (last accessed October 5th, 2019). http://extempore.moso.com.au/temporal_recursion.html.
- [18] A. Sorensen and H. Gardner. Programming with time: cyber-physical programming with impromptu. In *ACM Sigplan Notices*, volume 45, pages 822–834. ACM, 2010.
- [19] TOPLAP. *TOPLAP 15th Anniversary Concert Signup Form*, 2019 (last accessed October 5th, 2019). https://docs.google.com/spreadsheets/d/1Mksi_TReJpp9R3XunGCR0SALZRA03aMoiOEd3qUW0bo/edit#gid=0.
- [20] G. Wakefield and C. Roberts. A virtual machine for live coding language design. In *NIME*, pages 275–278, 2017.
- [21] A. Ward, J. Rohrerhuber, F. Olofsson, A. McLean, D. Griffiths, N. Collins, and A. Alexander. Live algorithm programming and a temporary organisation for its promotion. In *Proceedings of the README Software Art Conference*, volume 289, page 290, 2004.

Responsive Space – Liveness through spatial distribution of sound and image

Raimund Vogtenhuber
Institute for Computer Music and Sound Technology
Zurich University of the Arts
raimund.vogtenhuber@zhdk.ch

ABSTRACT

In this project a performance framework called "Responsive Space" has been created, which allows a flexible way of working with distributed sound and image projections. The system consists of a multichannel speaker-system, up to three video projections, a local Wifi network with a connected webserver and mobile devices. The audience is invited to log into the local network with their mobile devices. In the browser of the mobile devices sound and image is generated or streamed.

This offers the possibility of a great diversity in the spatial distribution of sound and image. As experienced in previous projects with live audio-streaming and mobile devices the spatial gesture of music as a musical parameter comes to the fore. In the performances the audio is streamed with an icecast-server and visual output is generated with javascript on the client's browser. The imperfection of time synchronization leads to very interesting effects. The awareness of space and the interaction of the audience with each other leads to the emergence of a social space in the concert.

In addition to the dimension of space this setup also examines the relationship between different media layers. This is central for the question how to achieve "liveness" - which in my opinion can also be achieved with this spatial arrangement and distribution of sound and image.

1 INTRODUCTION

There is an ongoing discussion about liveness and the use of media. Emerson argues, "whatever the origin is of what I receive, how does it address me (and others)?" [5] Fischer-Lichte doubts that liveness is possible in mediatized performances. [17] She argues it still needs the performer to address the audience directly. In contrast, Auslander argues that every performance is a mediatized performance, because it is already part of our world. [18] In loudspeaker concerts the sound generation is separated from the

sound reproduction and there is often no way to understand how these sounds are produced. In contrast to instrumental music with performers playing live, there is no certainty of witnessing a unique moment that lends something special to the present and the moment. [14]

This project follows the idea that space can make a significant contribution to giving viewers the feeling of "Liveness". In contrast to a strategy that places the physical presence of the performers in the foreground, the spatial arrangement of image and sound projections is also suitable to create liveness and presence and to bring the peculiarity of the moment closer to the audience.

In this context, it is also important how the artistic work can address the listener personally and at the same time achieve the perception of a unity with distributed image and sound sources. The experiences from earlier projects were included as well as considerations on how a distributed audio stream can be combined with visuals in space and on mobile devices.

1.1 Previous projects

In the project "The Neukoms" [1], and the follow-up project "Cloudspeakers - a mobile performance network" [2] or similar projects like "fields"[14] two phenomena were observed which came into existence surprisingly during the development and execution of the performances. The simultaneous use of a multi-channel speaker system and the integration of the mobile devices of the audience lead to an unusual listening situation that emphasizes the spatial character of music. The spectators themselves were able to determine the volume and orientation of the sound and started to interact with the mobile devices and each other. This awareness of space and the interaction with each other led to the emergence of a social space in the concert.

1.2 The sonotope

In this setting the live electronic performance is fed into the multi-channel speaker setup, which is arranged around the audience. The performing musicians are also able to decide to stream a part of their output via a local Wifi network to the mobile devices of the audience. The audience then receives the audio stream through a browser running on their mobile devices. This leads to a diverse distribution of the sound sources in the room and – due to the



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

different latencies of the mobile devices - to a time delay of the sound events. As a result, the spatial gesture of sound as a musical parameter comes to the fore. In addition, the loudspeakers and the speaker of the mobile devices have different sound characters and different impact areas. While the loudspeakers can impact the entire public performance space, the effective range of the mobile devices is very limited. The scope of these speakers covers only the very personal space and adjacent range of a listener. These different characters and effects of the sound sources lead to an unfamiliar perception situation, which has been called Sonotope [1] and are reminiscent of a forest.

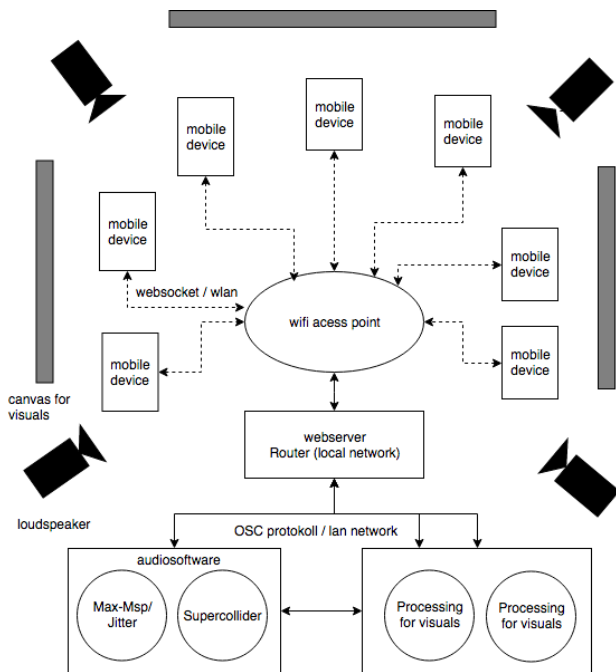


Figure 1. Communication diagram of the setup

2 RESPONSIVE SPACE

2.1. The Setup

The Setup includes a multi-channel speaker system, up to three visual projections distributed throughout the room, and a local network for the listeners' mobile devices. The audience is invited to log into a local network and use his/her smartphone. In the browser of the mobile device (or computer) sound and image is generated or streamed.

This offers the possibility of a great diversity in the spatial distribution of sound and image. The audience is invited to come with their personal smartphone, tablet or computer and log in to the network. Sound and image are generated in the room and through speakers as well as in the browsers of the viewers' mobile devices. The audio software communicates via the OSC protocol to the video software and a web server to which the mobile devices of the audience are connected. For audio-processing supercollider is used.

There are different parameter mappings from the audio source to the visual output, which is transmitted with OSC messages. This OSC messages are sent via LAN to the computers (I used separate UDOO X86 Minicomputer with Linux), which run the visual programming software processing. The web server (which was integrated into the performance computer) receives the audio program's OSC data and forwards it to the individual clients (the smartphones) via Web Socket. (figure 1). The webserver is running with a Node.js Framework called rhizome. [3]

A raspberry pi with a soundcard (pisound) is used to broadcast the audio stream. The visual output is generated with processing on the computers and p5js on the clients. The source code is available on github. [5]

2.2 Artistic works

The first piece was an audiovisual piece called "Open Form". Stochastic processes generate layers of five different sound types. The piece examines which gestural possibilities the respective sound offers, and in which temporal dimensions these tonal gestures form meaningful unities. A "gesture", I would describe as a small musical motif, which then can be altered and varied in sequences. Temporal change plays a major role in the creation of shapes and their sequences in time. Overlaying and combining the types of sound results in unpredictable configurations and shapes. The gestural forms result from the superposition of sounds and the density of events.

Every sound type is mapped to a visual form. According to the sound character of the five different sound types I tried to find a corresponding visual form. I mapped various parameters of the sound synthesis to the visual output via OSC. The mapping includes the frequency, the volume, panning, and from granulated sound I derived the grain length. They are directly connected to the position in the vertical, the brightness, the horizontal position and the density of a texture. Additionally a separated process generates a random walk which is mapped to parameters of the visual output.

There are variations of the visual output on every mobile device. And there is a difference between the visual output on the mobile devices and the projections on the screens in the room.

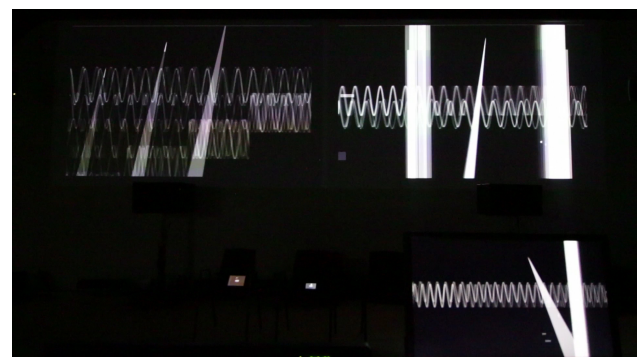


Figure 2. Open Form III (2018). Videostill.

3 THE SPACE IN CONCERT

Space was always an important parameter in noncommercial electroacoustic music. It has a long tradition to experiment with various kinds of loudspeaker setups, which explore the possibilities to distribute and move the sound in the space. The Acousmonium created by the Groupe de Recherches Musicales (GRM) in 1970 or the Birmingham Electroacoustic Sound Theatre (BEAST) are still in use. In the cinema we see a development towards 3D-Sound or similar systems to create an immersive experience. But generally spoken, music is more often perceived as an arrangement of events in time rather than in space. As Emmerson argues, time and space in music are not independent or absolute, “time needs space and space needs time”. He describes: “a performer might ask ‘where are we?’ in reference to a printed score or a given structure in the (perhaps unwritten) music, but this question can also be asked with respect to a venue, furthermore we can ask it about a historical or sociological ‘position’. It seems this question is the most comprehensive it is possible to ask!” [5].

Music theory more often deals with questions of harmonic and temporal structures of a music piece than with the question of space. It might be the case, following Emmerson’s argument, that the space could be understood in many different ways and therefore it is harder to make distinct classifications. The space in a musical piece also often means the space of the sounds within the music piece apart of the situation or context of a concert or a social context. Emmerson suggested a division of the space universe into four components of increasing scope: event, stage, arena and landscape. This description of sound areas covers the public space, but with the use of mobile devices and sound sources with a very limited range we might to go more in detail to a smaller and personal area of space.

3.1 The hidden dimensions

Dennis Smalley has used the theories of proxemic classification as a basis for describing the sound spaces created in a performance with instrumental music [6]. He refers to theories of Eduard Hall [7] where he divided zones of social interaction into four distinct zones. These four spheres of action are described with spatial dimensions that are in a very dynamic relationship to each other. Eduard Hall describes these relationships as a kind of contraction and expansion of the spheres of influence. The intimate zone includes the very close physical area in which interpersonal acts such as gentle touch but also, for example, contact sports take place. This is followed by the personal zone, which covers about one arm's length and where we directly communicate in a personal way, the social zone in which the social interaction of a group of people takes place, and finally the public zone which includes the whole space in which an event takes place. Accordingly, Smalley describes zones and sonic spaces within a concert or music performance. The gestural space encompasses the sphere of influence of the individual and is characterized by the physical action and movement of the instrumentalists. This describes also a very close and intimate sonic space, where the sound is very close

to the ear. The Ensemble Space is the sphere of influence of the entire ensemble and encompasses the entire acoustic field of the ensemble and thus also the stage space. The Arena Space finally includes the entire performance space with the ensemble and the spectators.

3.2 Sound zones with distributed loudspeakers

These divisions are more or less clearly separated, and the dynamic relationships of the individual sound spaces vary with the division of the concert space and the circumstances of the concert. These descriptions fit for an instrumental performance with a classic arrangement of ensemble, stage, and audience. But which acoustic spaces arise in a concert with electroacoustic music in which the loudspeakers are arranged around the audience, the performers are perhaps in the middle of the room, or the smartphones are used as sound sources and are distributed throughout the room? In this arrangement, there are sound zones that overlap and are interlocked. The sound space of the arena results from a multitude of action zones, which can be configured and changed depending on the listener's point of view and the nature of the room. This results in a series of interesting questions on how such a constellation can be described, and how it can be further developed: Which sound zones are created in such a concert situation, what characterizes the individual sound zones, and what are their dynamic relationships? What are the connections between the physical, the musical, the social and the acoustic space? Is the moving sound synonymous with the gestural space of an ensemble?

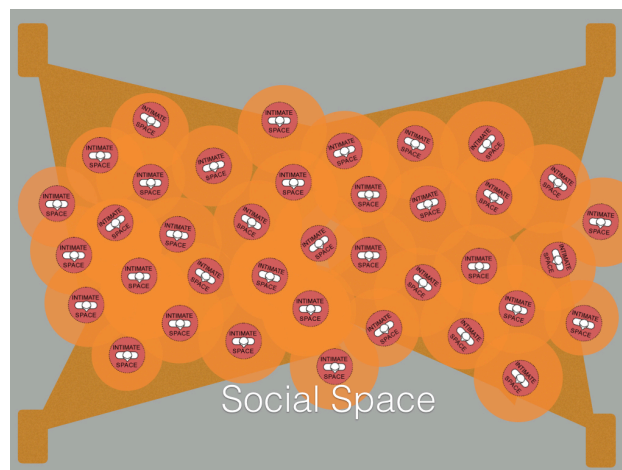


Figure 3. Sound zones within a performance space using mobile devices of the audience together with a loudspeaker system

3.3 Dislocation and Relocation

With technology it is possible to record sounds (and also images) and remove it from the original source and relocate it. Pierre Schaeffer’s original definition of acousmatic music and the ‘acousmatic dislocation’ derives from the consequences of the use of new technologies like telephone, radio or recordings. With

technology it was possible to separate the sound from its source in space and time. To the dislocation of space and time Emmerson added the dislocation of mechanical causality. He describes sound synthesis as a radical rupture with the past, where no longer a mechanical object produces the sound. In acousmatic music we sometimes do not know what causes the sound we hear in a loudspeaker. Instead of listening to the quality and musicality of the sound we tend to think about the question, what causes the sound. But Emmerson asks “(...) but need we know?” [5] New media was developed to overcome the distance of time and space and bridge the gap. Emmerson also emphasizes the interactive and responsive dimensions of new technologies. It is possible to communicate over far distances. Emmerson argues that the dislocation gives a way to relocation, but perhaps only when our anxieties over the process have been reduced. Times are changing and people get used to things. We have learned to accept, that any kind of mediatised sound is part of our world. There is no need to make distinctions between real or virtual sound and images any more. There is a new generation of listeners which has extended the memory of the mechanical world of sounds including electronic sounds which are part of our world.

3.4 Integration and recombination of media

In the “Responsive Space” also visuals are integrated. The intention was to create a setup to examine the relationship between different media layers like audio, visuals and the space.

As we have experienced surprising effects with the use of mobile devices when we distributed audio through their loudspeakers, it is a logical step also to use the mobile devices for visual output. It is interesting to see, how the visuals influence the artistic artwork and how it is playing together with sound and space. The visuals are generated and distributed on the screens of the mobile devices.

The visual content can also be generated with computers or various kinds of analogue techniques completely independently from audio sources. A commercial film tries to recombine the different media sources to create an illusion of a fluent narrative work. Artistic audiovisual performances more freely recombine different media types. These works have more in common with abstract paintings and experimental films of the early days (like Oskar Fischinger and others) [9]

However, in audiovisual performances we tend to see audio and visual events related to each other. According to Chion the “audiovisual contract” [10] let us search for synchronized moments or let us think of references and connection between audio and visuals. Chion explains this with the word “Synchrèse”, where image and sound can melt to a new unit. According to Cook’s Thesis the best way to combine different media types is to find complementation. The other possibilities, like being in contrast or similarity would lead to more uninteresting artistic works. [11] For my setup it would be interesting to see, how the different media types can be recombined and how the dimension of the space comes into the game. Obviously the “audiovisual contract” also works with distributed screens. On one hand, the audience is still

searching for synchronized events between audio and visuals. In the case of the distributed audio sources, which are more delayed than the visual output, these relationships become more complicated. Then it is not so clear anymore, which visual belongs to which audio source. Still it was possible to find these synchronized moments, but it seems to be more interchangeable in this distributed form. The audience has also the choice to perceive only the very personal space the output of the personal device or to perceive the surrounding space. Ideally, the viewer feels personally addressed, as well as part of a group and part of a public space. This special listening situation and the awareness of space leads to a sympathy which I would call “Liveness”. [12]

However, it is a difficult task to create these performances that appeal to individuals and groups without overstraining attention.

3.5 From reduced listening to extended listening

The same phenomena could be observed in the visuals. Concerning the acousmatic music Pierre Schaeffer suggested to find a way to exclude the distraction of the sound source and only try to listen to the sound itself. This kind of ‘reduced listening’ was for sure helpful in the early days of electronic and acousmatic music, but how should we listen to and perceive multimedia and intermedia performances?

Further, I would argue that there is also no need to exclude the material world, like the ‘reduced listening’ [19] suggested. It is indeed possible to listen to quality of sounds, and extend our ears (and eyes) to the concert space, the social space and the atmosphere around us.

3.6 The use of mobile devices

The relationship of people to their mobile devices is shaped by daily interactions. It is a very personal device, and many private and business communicative work is done with it. It ranges from phoning, writing e-mails, text messages or notes to a collection of personal photos and video recordings. In addition, smartphones are also multimedia machines. Videos or music are played and the personal collection of apps can perform all sorts of tasks for business or distraction. The smartphones and other mobile devices can also be seen as a kind of extended human sensory organ [13], with which humans perceive and interact with their digitized environment. The device thus works in a very personal and individual area, and in the context of the “Responsive Space” it is something surprising and unusual for the audience that these devices are addressed in the performance. After our experience with the performances, even during the concert, even if the mobile devices are used only for playback, an interaction with the devices and a communicative exchange between the audience with each other arises. Viewers begin to play with their devices, they have control over the volume and orientation of the devices, and start to communicate with each other. Not always verbally, but also through gestures and the control of their devices. Similar observations are described in the project Fields [14] in which similar systems were developed. The electro-acoustic music of the performers was emitted on a quadrophonic speaker system and the

mobile devices of the spectators, who were in the middle of the performance area. A framework has been developed that allows possibilities of sound synthesis on the mobile device and also offers modules for interactive use [15][16].

3.7 From social space to collective space

As described in the project, the smartphones, which use a very private context, will be expanded to a collective experience in these concerts. The single control with his personal device to a private composition, personal space occurs in communication to a public, collective space. This leads to the emergence of a social space in concert. This can also lead to a series of further developments, opening up a series of interesting questions: what conditions must be met for such a collective experience to be achieved? What is the relationship between personal space and the collective space? Which compositions and performances are possible with such constellations? Which compositions are possible, or how can they be arranged? Is it possible to perform a collective composition with the participation of the audience? In what relation do the performers stand to the collective?

3.8 Liveness in the multimedia performance

In this setup certainly people can be reached in their personal or even intimate space. However, the mobile devices are not just sound sources, they also provide a visual interface that can be used for artistic performances. It seems to be clear, that the personal mobile device can address the listener very directly, because the listener already has a kind of relationship to her/his personal device. The distance of the visual output of the mobile devices to the listener does not have the same impact. The visual output works through the staging and integration of the visuals in the room. But the generated image in the mobile devices can also be seen as a part of a bigger image, which comes to existence through the interaction of many individual devices. It increases the effect, that the audience feels personally addressed and part of a community at the same time.

4 CONCLUSION

The sound space of the performances was characterized by a special liveliness, which for me arose from the peculiarities of the spatial situation as well as from the possibility to address the listeners with their personal devices and to integrate them. The use of sound sources or screens in a heterogeneous distribution and the interplay of different sound and impact areas could be a way to integrate the viewer/listener into a performance. It offers great opportunities to experiment with different spaces and the spatialization of sound. It is a challenge to make performances which work smoothly, but also very interesting to explore its possibilities.

6 ACKNOWLEDGMENTS

Thanks to ZHdK (Zürcher Hochschule der Künste) and the ICST (Institute for Computer Music and Sound Technology) as well as

Art University of Graz (KUG) and the Doctoral school for the support of the project.

7 REFERENCES

- [1] Visser, Jeroen; Vogtenhuber, Raimund: Die Neukoms. Local streamed live-performance with mobile devices. Proceedings of the Audio Mostly Thessaloniki, Greece, 2015
- [2] Visser, Jeroen; Vogtenhuber, Raimund: Cloudspeakers – a mobile performance network. Web Audio Conference WAC-2017, August 21-23, 2017, London, UK.
- [3] <https://github.com/sebpiq/rhizome>
- [4] <https://github.com/responsivespace/>
- [5] Emmerson, Simon. Location – Dislocation – Relocation (‘Where is live electronic music?’). In: IV Seminário Música Ciência Tecnologia: Fronteiras e Rupturas. São Paulo, 2012
- [6] Smalley, Denis. Space-form and the acoustic image. *Organised Sound* 12(1): 35-58. 2007. Cambridge University Press, UK.
- [7] Hall, Eduard. *The hidden dimension*. Garden City, New York 1966
- [8] Schaeffer, P. *La musique concrete*. Paris: Presses Universitaires de France, 1973.
- [9] Abbado, Adriano: *Visual Music Masters, Abstract Explorations: History and Contemporary Research*. Skira Editore S.p.A., Milano, Italy 2017.
- [10] Chion, Michel: *Audio-Vision: Sound on Screen*. Columbia University Press, New York, 1994. Translated by Claudia Gorbman von L’Audio-Vision. Editions Nathan, Paris 1990.
- [11] Cook, Nicholas: *analysing musical multimedia*. Oxford University Press, New York 1998, reprinted 2004
- [12] Croft, John. *Thesis of Liveness*. *Organised Sound* 12(1): 59-66. 2007. Cambridge University Press, UK.
- [13] McLuhan, Marshall. *Understanding Media: The Extensions of Man*. McGraw-Hill, New York 1964; critical edition: Gingko Press, Corte Madera 2003.
- [14] Shaw, Tim; Piquemal, S’ebastien; Bowers, John. *Fields: An Exploration into the use of Mobile Devices as a Medium for Sound Diffusion*. Proceedings of the International Conference on New Interfaces for Musical Expression, Baton Rouge, LA, USA, 2015
- [15] Lambert, Jean-Philippe; Sébastien, Robaszkiewicz; Schnell, Norbert. *Synchronisation for Distributed Audio Rendering over Heterogeneous Devices, in HTML5*. Web Audio Conference WAC, 2016, Atlanta, USA.
- [16] Schnell, Norbert; Matuszewsk, Benjamin: *Playing (with) Mobile Devices and Web Standards ... Together*. Web Audio Conference, London, 2017, Atlanta, USA.
- [17] Fischer-Lichte, E. *Ästhetik des Performativen*. 2004. Suhrkamp, Frankfurt am Main, Germany.
- [18] Auslander, P. *Liveness. Performance in a mediatized culture*: 11. 2008. Routledge, New York, USA.
- [19] Kane, B. *Sound Unseen: Acousmatic Sound in Theory and Practice*. Oxford University Press, Oxford 2014.



Talks

Creating a DJ-ready Web Player for Interactive Music

Attila Haraszti
Independent Author

attila@haywirez.com

ABSTRACT

The talk will explore the motivation, challenges and solutions devised in a year-long quest to build a DJ-ready web player for songsling.studio, a tool I'm building for publishing and presenting interactive music on the Internet. The player itself utilizes an HLS-like solution as a streaming mechanism, and allows arbitrary changes to the playback speed, direction and content. I will dive into the history and legacy of computerized music playback systems, the dangers of leaving music presentation in the hands of a few dominant platforms and highlight the unrealized potential of our everyday computing devices.

1. INTRODUCTION

In a short introduction story chronicling my background in electronic music production and DJing, I will illustrate how the format of interaction influences the content of music and the process of creation. I will point out that by 2020, an entire generation will have grown up experiencing music overwhelmingly through dominant web streaming platforms such as YouTube, Spotify or Soundcloud.

2. THE POWER AND LEGACY OF PLAYBACK SYSTEMS

By a combination of earlier technical limitations and conventions related to digital playback, the web players provided by these platforms promote an extremely limited way of interacting with music, even when compared to previous analog systems of inferior quality. The end result is an encouragement of passive listening in order to steer the listeners towards the monetary goals of the platforms, instead of nurturing what should be a sacrosanct, intimate relationship between people and music [1].

2.1 Examination of Current Playback Systems

The first surprising feature of typical present-day media players is how little of the visual user interface is dedicated to any kind of meaningful control over the sonics themselves. Control elements such as a seekbar, playback and volume control occupy typically below 3-5% of the visual field. They are also limited in their essence – the seekbar quickly jumps to the correct position in a recording, but limited auditory information is conveyed when

compared to cueing music using a fast-forward or rewind action on a classical turntable or tape machine.

2.2 Desired design

In my approach to designing a player from scratch, I set a hard design constraint of allowing the user full control and explorability of the sound played. That means that at any time, listeners should have the ability to near-instantly reverse playback, speed up or slow down. Provided that the given portion of a composition has been fetched from the network, they should also be able to near-instantly jump to that arbitrary point in the timeline.

3. THE TRIALS & TRIBULATIONS OF BUILDING A PLAYER USING THE WEB AUDIO API

Most players on the web make use of the `<audio>` HTML element, even if it might be piped into a Web Audio API graph via a `MediaElementSourceNode` to provide more pleasant fade-ins and fade-outs. This approach is unsuitable for our use case given the inability to control playback speed in a satisfactory manner. Besides the limited playback rate range of 0.5-4x, most browsers implement pitch-preserving and time-stretching algorithms that were likely designed with one use case in mind — listening to human speech at faster or slower speeds. The proposed `preservesPitch` attribute that would allow developers to switch off this processing step is not implemented in any browser except Mozilla Firefox. Backwards playback is also not possible, therefore I had to explore a more low-level solution using the Web Audio API.

3.1 The Naïve Approach

An `AudioBufferSourceNode` has a k-rate `playbackRate` parameter that has a sufficient resolution for controlling playback speed for our needs. However, compositions are typically much longer than a single buffer is designed to hold, and it also would not be efficient to fetch them from a network resource using a single, large request. Therefore, the use of streaming segmentation techniques such as HLS¹ is well-advised. The Web Audio API provides sufficient precision for scheduling so that individual segments of audio can be played one after another in a seamless, gap-free manner — colloquially known as “buffer stitching”. We can therefore use this technique to pre-schedule a few segments in front of the playhead as needed. The problem arises from the fact



Licensed under a Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹ HTTP Live Streaming — a protocol for transferring unbounded streams of multimedia data

that the `AudioBufferSourceNodes` are fire-and-forget, single use only — they can be stopped before they start, but not rescheduled. As the playback speed should be allowed to change at any point and in an unpredictable manner, all scheduled nodes that have not yet started playback have to be canceled. Even if the same `AudioBuffer` can be reused and referenced in several `AudioBufferSourceNodes`, creating the new nodes takes too long, which leads to problems when our current playhead position is close to the tail edge of the node that is currently playing.

3.2 Buffers of Buffers

As described above, it is not possible to create and reschedule the `AudioBuffers` sufficiently fast. Luckily, this can be circumvented by taking inspiration from multiple buffering techniques used in computer graphics rendering for preventing visual tearing or stuttering. Thus, we can simply create a sufficiently large cyclic buffer of `AudioBufferSourceNodes` in advance and backfill them as needed. For a use case involving audio, we need to have significantly more nodes ready than the double or triple buffered approach typically used for graphics. I have found that an acceptable perceptual responsiveness can be achieved by approximating the number of common frames per second refresh rates.

3.3 Controlling state

The implementation of the necessary scheduling mechanism quickly leads to complex state management issues that proved cumbersome to debug. In order to make it easier to reason about the code and improve code maintainability, I have opted for formalizing the business logic using statecharts, a well-known application modeling technique [2]. This allowed the decoupling of the implementation from the description of behavior, with several substates representing particular components of the code.

3.4 Custom resampling via `AudioWorklet`

Unfortunately, the buffer-stitching approach has another problem that materializes even in the best-controlled scenarios: Interpolation between sample frames at certain sample and playback rates might lead to inconsistent frame boundaries between neighboring buffers. Most of the times, this is not disturbingly perceptible, but it does become apparent if we test the player using a perfectly segmented sine wave. As a workaround with more added complexity, overlapping buffers can be created with precisely scheduled volume switchovers [3]. An alternative, theoretically perfect solution can be achieved by implementing a custom resampling algorithm and playhead tracking mechanism inside an `AudioWorklet` process. However, as popular browser support is still lagging, we have to weigh the benefits and tradeoffs.

4. A VISION OF THE FUTURE: PRESENTING MUSIC VIA WEB APPS INSTEAD OF RECORDINGS

My goal was to preserve, or at least imitate, the tactile feedback and interaction allowed by analog playback mechanisms.

However, it is just as important to examine what future types of musical interaction could come as a result of a networked, distributed online playback system.

In my vision, artists should be concentrating on interactive music that supersedes the current focus on simple, static recordings. This means that each web player becomes a networked interaction portal, which can influence what other listeners hear and see. Using the latest iteration of the player engine, I will demonstrate an implementation of horizontal re-sequencing [4, 5] as a composition method of interactive music and what possibilities this might offer to the wider community of music producers.

5. ACKNOWLEDGMENTS

I would like to give my thanks to Stephan Hesse for the suggestion on researching computer graphics processing approaches, Chris Wilson for open-sourcing his DJ deck implementation using the Web Audio API [6] and Christoph Guttandin for creating and maintaining the standardized-audio-context library [7]. I would also like to express my gratitude for all current and previous organizers of the Web Audio Conference for inspiration and support, as well as electronic music pioneers such as Laurie Spiegel and Jeff Mills for making me continually rethink my approaches to music.

6. REFERENCES

- [1] Spiegel, L. 1992. Music: Who Makes it? Who just takes it? *Electronic Musician* #8, 1 (January 1992), 114. Retrieved October 6, 2019 from http://retiary.org/ls/writings/em_back_page_slashed.html
- [2] Harel, D. 1987. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.* 8, 3 (June 1987), 231-274. DOI= [https://doi.org/10.1016/0167-6423\(87\)90035-9](https://doi.org/10.1016/0167-6423(87)90035-9).
- [3] Harris, R. 2016. *Phonograph.js: Tolerable mobile web audio*. (August 2016). Retrieved October 6, 2019 from https://medium.com/@Rich_Harris/phonograph-js-tolerable-mobile-web-audio-55286bd5e567
- [4] Sweet, M. 2015. *Writing interactive music for video games: a composers guide*, Upper Saddle River, NJ: Addison-Wesley. (September 2014), 278-279
- [5] Phillips, W. 2017. *A Composer's guide to game music*, Mit Press Ltd. (2017), 188-193
- [6] Wilson, C. *wubwubwub* (2014), GitHub repository, <https://github.com/cwilso/wubwubwub>
- [7] Guttandin, C. *standardized-audio-context* (2019), GitHub repository, <https://github.com/chrisguttandin/standardized-audio-context>

Building an intuitive web-based musical instrument

Ashish Dubey
84H, Sector 18
ashish.dubey91@gmail.com

ABSTRACT

In this paper, I will describe the design of Pastelloops, which is a web-based musical instrument and an interface that can be used for composition and live improvisation using intuitive free-hand drawing on a canvas using a mouse or touch-based interaction.

1. INTRODUCTION

Sketching has been explored extensively as a natural interface for musical expression. Such sketching interfaces provide a simple way of synaesthetic expression of visual art to music or vice versa. When not used in artistic expression, they can be used in an easy and playful way to explore music by people with varying levels of understanding of music.

This application, Pastelloops, has been inspired by similar ideas. A user can interact with a canvas to draw notes freely along a musical scale with different sounds generated using machine learning for interesting timbral variations. A slightly different application called Music Mouse [1], conceptualized by Laurie Spiegel and recently adapted on web [2] by Tero Parviainen was an inspiration behind some pre-defined harmonic constructs in the application so that users who do not have a strong understanding of music can still use the tool and create something musical. These constructs can however be modified by experienced musicians who want more control over their creation. The real-time sonic feedback and the playback mode provides a way to the user for monitoring the musical result of their sketches. This makes Pastelloops a helpful tool which can be used as a musical instrument for learning music, improvising live and composition as well.

While the concept of making music through sketching isn't new in itself and there have been quite a bit of research and development on similar interfaces, Pastelloops aims bring many of these ideas together, and using Web APIs especially Web Audio, it does so in a distribution format which is very easy to adopt and extend. Pastelloops has been made with open-source libraries like Tone.js for handling most of the audio capabilities and P5.js for visuals. In this paper, we will go through the design of Pastelloops and later discuss possibilities that can be explored by tweaking its individual components and how it compares with similar tools that exist.

2. INSTRUMENT DESIGN

Thor Magnusson in his book [3], describes a digital music instrument mapping model which I've taken as a design guideline for the application. It has helped in approaching the application in

terms of different components of a digital instrument which work in conjunction for the overall experience and results of the instrument. In this section, we'll talk about those components in Pastelloops.

2.1 User Interface

The interface to create sonic output in Pastelloops, as shown in Figure 1, is a drawing interface consisting of a 2D canvas on which the user can draw freely using a mouse or touch wherever enabled. The interface is web-based, which means it can be loaded on any device where a modern browser, supporting Web Audio APIs is available.

There are controls available which can be used for selecting texture of the brush which changes the color and thickness. Different shades of brushes are associated with different properties of the sonic output which are described in sound engine and mapping engine sections. Specific drawn sections on the canvas can be cleared with an erase tool like on other free-hand drawing interfaces or an entire canvas can be cleared. A few other controls are provided to change other musical properties, including a slider control for changing tempo and a slider for changing output volume of individual sounds.

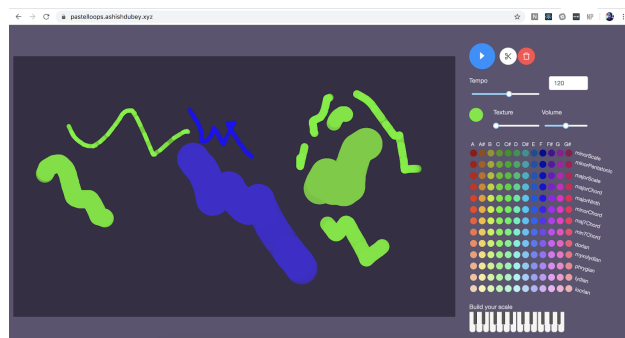


Figure 1: Pastelloops UI with all the controls: playback button, tempo, texture, volume, scale palette and custom scale creator

2.2 Mapping Engine

Most of the mappings are connections between the visual and interactive elements on drawing on a canvas and the sonic properties of the output that is produced.

Each point on the Y axis is associated with one of the pitches arranged in a descending order. These pitches are chosen based on which musical scale the user wants to play. This scale can either be chosen from a predefined palette where each color corresponds to a scale or can be constructed by the user themselves. The selection of a musical scale lets the user focus on

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the author(s).

drawing freely and still create an effect of the kind of musical harmony they intend. Of course, musicians who need more control can create their own sequence of notes that are played along the canvas.

While being able to choose a scale and playing through the notes along the canvas, the user can also control the timbral properties of the sonic output by changing the visual properties of what they draw on the canvas. Such a mapping is established between the texture of the brush, and the different sound samples that are mentioned in the sound engine.

2.3 Sound Engine

In Pastelloops, to allow the user to explore different and new possibilities of what they can create, several sounds can be chosen from. This gives the user to experiment with the timbral properties of the music that they produce with the application.

While the instrument allows giving the control to the user for being able to tweak sounds, some attention has been paid to keep the process simple for the user. Instead of something like giving the control over the properties of a synthesizer, the user is given an easy way to switch through sounds that smoothly morph between two sounds of different textures by changing the texture of the canvas strokes. These sounds have been generated by GANSynth [4], which is a deep learning model released by Google's Magenta team. These interpolated sounds do not just allow the user to easily vary the sound texture in a continuum, but also makes the experience a bit more interesting because of the unique acoustic characteristics of the sounds generated.

Since the process of generating the sounds is compute heavy, the sound samples have been pre-generated, fetched from the server, and loaded into a Tone.js instrument

2.4 Sonic Feedback

There are two interaction modes phases on the canvas - one is drawing freely on the canvas and the other is playback. When the user draws on the canvas, it's as if they are playing the instrument with free-hand brush strokes. The notes and sound corresponding to the shade of the brush and the position in the canvas can be heard immediately which aids the user in exploring the instrument. With the playback mode on the other hand, the user can listen to all the combined sonic artifact of their drawing as a playhead scans across the canvas from left to right at the default 120BPM unless changed by the user. This allows the user to use this tool to compose music from multiple structures and patterns they draw on the canvas.

3. RELATED WORK

Pastelloops is an amalgamation of different ideas, aiming to offer a versatile interface for musical exploration to different kinds of audiences for different purposes, which differentiates it from other tools.

Some well-researched and developed tools are Griddy [5] and Hyperscore [6]. With Hyperscore, the user can pen musical ideas as strokes and lines, store them for later use, and create new pieces—all in one expansive canvas. The graphical elements in the drawing are mapped to musical structures, allowing the user to interweave and shape musical voices and define harmonic progressions visually.

Another application is Griddy, which uses a background image for style-related features and applying an image sonification based method to probe the image to allow users to use the result as a base for their musical exploration.

Pastelloops differentiates itself from these tools in a way that

while the above tools are mainly for composition use-case, Pastelloops can be used as a live musical instrument, making it useful for improvisation. As a musical instrument it also offers its users, an easy way to explore sounds with interesting acoustic properties generated through GANSynth.

4. CONCLUSION AND FUTURE WORK

Pastelloops as described in this paper is live¹ and the source code can be found on Github². The project is under active development and there are several experience fixes and enhancements that are planned. Experience fixes mostly in terms of its usability on touch screen devices. As for known enhancements, one of them is that, instead of limiting the playback to an XY-scrolling playhead, allowing the strokes to be played as they were drawn over time. This would add a dimension of time while the user is sketching on the canvas, and allow them to play it back in the same order, and not necessarily from left to right in space. One of the limitations right now is that there is only one canvas which can be used for composition. With a capability to arrange multiple canvases in serial or parallel, an artist can maintain multiple musical structures which they can use for their compositions.

Because Pastelloops interface is designed for the web, it's also easy to build adapt the application to involve different users in the musical experience. Web APIs like WebRTC can be used to build a collaborative sketching interface towards an experience of music jamming using sketches for friends located in different geographies. Application of machine learning to melody generation would be another interesting experiment worth trying. Latent Loops [7] is an example of using MusicVAE for generation of interpolated melodies based on melodies input in a very similar sketching interface.

Besides some known changes as mentioned above, there is ample time to be spent doing user studies with the target users, including artists and musicians. That, I hope, will help generate ideas to enhance the musical aesthetics and interface in order to allow users to explore more musical possibilities as an instrument and a composition tool.

5. REFERENCES

- [1] Music Mouse by Laurie Spiegel
<http://retiary.org/ls/programs.html>
- [2] Music Mouse - An Intelligent Instrument - An emulation
<https://teropa.info/musicmouse/>
- [3] Sonic Writing by Thor Magnusson
<https://www.bloomsbury.com/uk/sonic-writing-9781501313868/>
- [4] GANSynth: Making music with GANs
<https://magenta.tensorflow.org/gansynth>
- [5] Kim, Luke K. and Woon Seung Yeo. "Griddy: a Drawing Based Music Composition System with Multi-layered Structure." ICMC (2014).
- [6] Hyperscore
<https://opera-beta.media.mit.edu/portfolio/hyperscore>
- [7] Latent Loops
<https://magenta.tensorflow.org/composing-palettes>

¹ <https://pastelloops.ashishdubey.xyz/>

² <https://github.com/dash1291/pastelloops>

The New WAC Website: Towards a Sustainable Conference

Jørgen Varpe
Department of Music
Norwegian University of Science and Technology
(NTNU)
Trondheim, Norway
varpe1992@gmail.com

Eigil Aandahl
Department of Music
Norwegian University of Science and Technology
(NTNU)
Trondheim, Norway
eigilaa@stud.ntnu.no

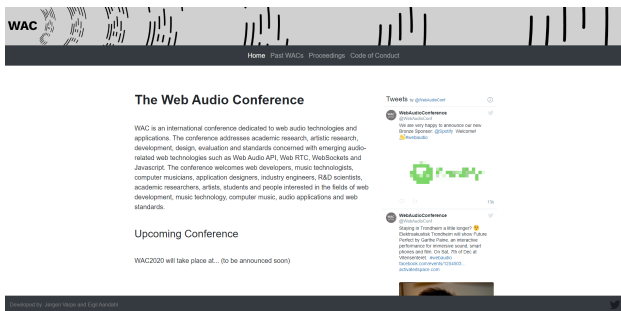


Figure 1: Screenshot of the website design.

ABSTRACT

Started in 2015, WAC is a young conference, but has also reached a maturity state with regular 1-1.5 year intervals between editions. There is however a lack of consistency between the conference proceedings which makes them difficult to index and find. Also, publications often include associated audiovisual media. We have created a centralized website with the proceedings indexed consistently combined with other relevant information. We have also developed a collection of guidelines and designed the website in a format that can be used by the forthcoming conference organizers.

In this talk, we will present the new website, the project workflow, and the tools and libraries used for development. We will also reflect on the lessons learned and a set of recommendations to keep the conference sustainable.

1. WEBSITE

The website contains metadata and PDFs from proceedings up to current date, which has been catalogued and prepared for indexing. We have used Eleventy (11ty¹), a static site generator, for building the website. Using several Node.js packages and JavaScript, the system automatically

¹<https://www.11ty.io/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

generates sites containing the data of each proceeding. From 2015 until 2018 all sites are generated from collections of BibTeX files. These collections are made using the reference manager Mendeley². Since Mendeley uses special notation for special characters in BibTeX, we found another reference manager called JabRef³, which we used as a tool to remove all special notations. For the current and future editions, the system also supports input from excel spreadsheets generated by the conference management website.

As part of the sustainability plan, the domain used in 2018 will now become the permanent home for the Web Audio Conference:

<http://webaudioconf.com>

We have also documented how to maintain and upload new content here: <https://github.com/web-audio-conference/Website/blob/master/README.md>

2. ACKNOWLEDGMENTS

This work has been done under the technical supervision of Gerard Roma and the usability supervision and project coordination of Anna Xambó. We thank all the Web Audio Conference organizers who have helped us to compile all the information for this project: Norbert Schnell, Guillaume Pelerin, Jason Freeman, Alessia Milo, Florian Thalmann, Sebastian Ewert, Ken O'Hanlon, Christoph Guttandin, and Jan Monschke. This work has been done with the support of the Audio Communication Group at Technische Universität Berlin, with special thanks to Athanasios Lykartsis and Stefan Weinzierl.

²https://www.mendeley.com/?interaction_required=true

³<http://www.jabref.org/>

Essentia in the browser

Luis Joglar-Ongay
SonoSuite
Music Technology Group
Universitat Pompeu Fabra
luis@sonosuite.com

Albin Andrew Correira
Music Technology Group
Universitat Pompeu Fabra
albin.correira@upf.edu

Pablo Alonso-Jiménez
Music Technology Group
Universitat Pompeu Fabra
pablo.alonso@upf.edu

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
xavier.serra@upf.edu

Dmitry Bogdanov
Music Technology Group
Universitat Pompeu Fabra
dmitry.bogdanov@upf.edu

ABSTRACT

Web technologies are evolving every day providing higher capabilities and enabling all kinds of software. We believe that audio signal processing and other MIR related tasks should not be an exception, and there is a clear interest and need of such web tools in this community. Ideally, these tools should be developed to be as powerful as the ones already available on other languages such as C/C++ and Python.

1. ESSENTIA AS JS LIBRARY

This motivation drove our exploration on how to use Essentia [2], an open source C++ library for music and audio analysis, description and synthesis developed by the Music Technology Group of the Universitat Pompeu Fabra, in the web client as a JavaScript library using Emscripten [6] and WebAssembly [3].

The biggest strength we see in using the same code both in the browser and in native applications is the robustness of our development process. This way all our data and efforts can be put into improving the algorithm available in Essentia instead of having to maintain two implementations in two different programming languages.

1.1 Compilation

In this talk, we will present the steps to follow to use Essentia in the client, as well as some use cases. We will also discuss difficulties we encountered during the implementation, such as problems compiling Essentia for Emscripten. Furthermore, we will analyze some decisions we took along the way and other questions that are still open.

1.2 Examples

As our main example, we will show the use of Essentia in the context of SonoSuite's digital music distribution platform, where users submit their audio tracks online to be immediately analysed in the client to detect possible audio quality problems. The system gives feedback directly back

to the users, allowing them to make decisions on the quality of their tracks.

The employed algorithms, now used in the browser, have been previously developed and integrated into the core C++ Essentia library in a collaboration R&D project [1]. This project aimed to develop algorithms to automatically detect the most common audio problems SonoSuite quality control team encounters on client's audio files.

2. AVALUATION AND RESULTS

Although this is still a work in process a preliminar avalluation and comparison to other JavaScript libraries, such as Meyda [5] and JS-Xtract [4] will be presented. Similar tasks will be implemented and executed using all three libraries in the browser to compare their results and efficiency.

3. FURTHER WORK

At SonoSuite, we plan to keep using Essentia to improve quality control, as well as to keep innovating to help our users for a better experience. For instance, this will include implementing automatic metadata annotations by genre and language. Finally, in this talk we will expose our ideas for further work.

4. REFERENCES

- [1] P. Alonso-Jiménez, L. Joglar-Ongay, X. Serra, and D. Bogdanov. Automatic detection of audio problems for quality control in digital music distribution. In *AES 146th Convention*, Dublin, 20/03/2019 2019.
- [2] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. Essentia: an audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR'13)*, pages 493–498, Curitiba, Brazil, November 2013.
- [3] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien. Bringing the web up to speed with webassembly. *SIGPLAN Not.*, 52(6):185–200, June 2017.
- [4] N. Jillings, J. Bullock, and R. Stables. JS-XTRACT: A realtime audio feature extraction library for the web, 8



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

2016. Online. Available from <http://dmtlab.bcu.ac.uk/ryanstables/JSXtractISMIR2016.pdf>.

- [5] H. Rawlinson, J. Fiala, and N. Segal. Meyda: an audio feature extraction library for the web audio api. In *1st Web Audio Conference*, Paris, France, January 2015.
- [6] A. Zakai. Emscripten: An llvm-to-javascript compiler. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, OOPSLA '11, pages 301–312, New York, NY, USA, 2011. ACM.

Comparing apples to oranges: A comparison of AudioWorklet polyfills

Christoph Guttandin
Media Codings
Berlin
chrisguttandin@media-
codings.com

ABSTRACT

The specification of the Web Audio API is nearing the V1 milestone but there is still only one implementation of the AudioWorklet so far in Chromium based browsers. Firefox is expected to catch up soon but there is uncertainty whether or not the AudioWorklet will be implemented in Safari. For a foreseeable future Web Audio developers can't rely on the AudioWorklet to be available in all browsers.

But as many other new browser technologies the AudioWorklet can be used in browsers which do not support it natively by utilizing a polyfill as fallback strategy. Three polyfills are currently available which do all provide the same API as the native implementation but do of course have other performance characteristics. Every polyfill follows a slightly different approach.

This talk will incorporate a comparison of the native AudioWorklet and all available polyfills. They do for example differ in the way in which they emulate the AudioWorkletGlobalScope. One uses a Web Worker, the other uses an iframe and the third one is evaluating the AudioWorklet code on the main thread. This has not only implications on the

performance but can also be a security risk when running third party AudioWorklets from untrusted sources. On the other hand, executing the code on the main thread will save some time because it avoids passing around costly messages. In some scenarios it might be possible to achieve a lower latency using that technique. But this assumption falls short in case the main thread is very busy and can't invoke the `process()` function in a timely manner. In such a situation it is better to offload the work to an iframe or a Web Worker to execute the code in parallel.

For some web apps it might be possible to avoid a polyfill at all. If for example Web MIDI is a hard requirement it can be assumed that the AudioWorklet is available as well. For other web apps it could be beneficial to keep using the now deprecated `ScriptProcessorNode` even when the AudioWorklet is available. At least when running on underpowered low-end mobile devices this could make sense.

As with many other decisions choosing the right AudioWorklet polyfill is a decision depending on many factors. This talk will provide an opinionated checklist which is supposed to help to choose the most appropriate polyfill for any use case.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Opening the Dragon's Den: Interactive Welcome to Baton Rouge Entrepreneurship Week

Jesse Allison
LSU School of Music &
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA, USA
jtallison@lsu.edu

Derick Ostrenko
LSU School of Art &
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA, USA
dostrenko@lsu.edu

ABSTRACT

An interactive system was developed to invite participants to the Baton Rouge Entrepreneurship Week opening event. The system consisted of a reactive entryway with an imposing projection on a curved wall, a video tracking and projection system covering the floor of the space, an array of iPad stations throughout the area, and participants using their cellphones to participate in the event. Through these cell phones, web audio was used to create a uniquely personal connection with sound effects and spoken phrases that could not be achieved by the shared public display alone.

This talk will cover the technical aspects of the installation including the novelty of pairing video tracking with distributed performance systems to allow for the identification of individual participants and enable targeting them with interactions both on their mobile device and in the public space beyond.

1. BACKGROUND

Baton Rouge Entrepreneurship Week¹ is an event to help foster entrepreneurial activities within the state of Louisiana. As a series of business events, it kicks off with an opening night gala each year to create excitement about the upcoming talks, panels, workshops, and pitch events. Our installation was created for the entrance to the opening event for the week. Its purpose was to be visually striking, reactive or interactive, and inspiring.

To achieve this, we gathered and drew from 114 quotes that showcased the entrepreneurial spirit such as: "Every once in a while, a new technology, an old problem, and a big idea turn into an innovation." - Dean Kamen; and "I can't understand why people are frightened of new ideas. I'm frightened of the old ones." - John Cage

These quotes set the tone for the opening event and were projected on the Innovation Wall, displayed on tablets in stands around the room, displayed on individuals mobile devices, and integrated into the BREW website. When the

installation targeted an individual, it was spoken on the device and then across other mobile devices in the space as well. The large number of things moving, speaking and displaying on their own helped to develop a dynamic sense of energy to support the intended networking at the event.²



Figure 1: The Innovation Wall entry display

2. TECHNICAL DETAILS

As outlined above, the system consisted of a reactive entryway with an imposing projection on a curved wall, a video tracking and projection system covering the floor of the space, an array of iPad stations throughout the area, and participants using their cellphones to participate in the event. Through these cell phones, web audio was used to create a uniquely personal connection with sound effects and spoken phrases which could not be achieved through the public display alone. Once participants pulled up the site on their cell phones in the hall the system engaged them via:

- Electro-glitch synth music distributed all across the mobile devices.

²<https://brew2017.emdm.io/>

¹<https://www.celebratebrew.com/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

- Spoken and displayed inspirational entrepreneurial words and phrases.
- Displayed a rotating artistic interpretation of the Baton Rouge Entrepreneurship Week logo.
- Responded in a simple fashion to touch (color change, ripple, sound effect, and sharing your message)
- Rapid fire spotlighting events across participants illuminating their phone, projecting onto the individual, speech, and exciting electrical crackling sounds.

The effect of the spotlighting event struck across multiple phones, each person experiencing: their phone screen color swell and brighten, they were spotlighted from above, their phone either said a phrase to them specifically or spoke a portion of a phrase loudly along with some electrical noise. This happened rapidly across the audience cell phones to say phrases like [Welcome] [to] [Baton Rouge] [Entrepreneurship] [Week]

To achieve this technically, we tied together the giant projection wall, a grove of tablets in stands, a second projection on the floor, and participants with their own mobile devices through a single web application. Each display was created using web animations and web audio, and connected with websockets via nexusHub.[1] To achieve individual identification, we used live video color tracking from above to detect individuals and circle them with the BREW Logo as we activated their mobile device.

An array of 10 iPads in stands were installed around the area, each running a standalone version of the web page. This way, if no one was using the site on their phones, the entry was still being activated with illumination and sound.



Figure 2: Quotes and interactive display on an array of tablets

Web audio was engaged specifically to create a distributed field of sound juxtaposed with individually targeted audio. An underlying humming electrical sound and rhythmic pulse was played through the entire space, however individual mobile devices and the tablet array played higher pitched musical sounds, electric sizzle effects, and spoke the quotes when they were triggered. By using Web Audio, each participants device spoke specifically to them. Combined with the projected spotlight, screen flashing color, and electrifying sound on their device, it instilled a remarkable awareness that the system was speaking directly to you, anonymity was no longer an option.

The ability to traverse from communal, public space to a personal experience was entirely enabled by utilizing web

audio playback on the participant's device. If it had been only an external projection and audio, however well tied to the individual, it would not have had the power of reaching you through your own personal mobile phone. As this was a fun and lively event, it was simply an exciting feature, however the creepiness of an invasive public/personal system like this did not go unnoticed by the artists and leaves quite an opening for interesting public space engagement.

3. CONCLUSIONS & FUTURE

The reception was well attended with the installation incorporated into the business-party-like atmosphere. Most attendees commented on the giant projections, and read a few of the entrepreneurial aphorisms as they headed into the event. The ones intrepid enough to try it out on their own phones generally grabbed a couple of friends showing them the targeting event before moving into the full reception.

The installation was created as a welcome experience for the 2017 BREW festivities. Unless commissioned for another event, it will not be further developed in its current form. However, components of the installation are in active development for artistic purposes.

The original installation was intended to use fog and lasers to target the screens of users, lighting them up, and triggering sounds and effects on their screen. In the US, this requires FDA approval through a many month approval process and had to be abandoned in favor of the projection approach. The laser targeting has proven to be valid and is being reserved for a future artistic performance.

The novelty of sending a color and subsequently color-tracking the devices is being further refined for a number of other artworks that engage public spaces.

4. ACKNOWLEDGMENTS

We would like to thank the Baton Rouge Entrepreneurship Week, the LSU Office of Innovation & Technology Commercialization, and the Center for Computation & Technology for their support of the project.

5. REFERENCES

- [1] J. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *Proceedings of the New Interfaces for Musical Expression conference*, 2013.
- [2] G. Essl and A. Müller. Designing mobile musical instruments and environments with urmus. In *New Interfaces for Musical Expression*, pages 76–81, 2010.
- [3] L. Gaye, L. E. Holmquist, F. Behrendt, and A. Tanaka. Mobile music technology: report on an emerging community. In *Proceedings of the 2006 conference on New interfaces for musical expression*, NIME '06, pages 22–25, Paris, France, 2006. IRCAM - Centre Pompidou.
- [4] T. Shaw, S. Piquemal, and J. Bowers. Fields: An exploration into the use of mobile devices as a medium for sound diffusion. In *Proceedings of the international conference on New Interfaces for Musical Expression*, pages 281–284. The School of Music and the Center for Computation and Technology (CCT), Louisiana State University, 2015.

Connecting Web Audio to Cyber-Hacked Instruments in Performance

Anthony T. Marasco
Experimental Music & Digital Media
School of Music and CCT
Louisiana State University
Baton Rouge, LA, USA
amarasco@lsu.edu

Jesse Allison
Experimental Music & Digital Media
School of Music and CCT
Louisiana State University
Baton Rouge, LA, USA
jtallison@lsu.edu

ABSTRACT

Artists and composers of distributed and networked music tend towards mobile devices that contain web browsers (smartphones, tablets, and laptops) as their primary mechanism for creating interactive sonic works. This integrates well with technologies explored at this conference such as web audio, web sockets, web midi, device sensors and click/touch interactions in conjunction with the computational devices themselves. As audiences are increasingly exposed to distributed & networked musical performances (D/NMP) presented through their personal, do-it-all devices, the opportunity arises to move beyond this novelty and expand the palette of networked devices while retaining musical comprehensibility.

To expand the pool of possible hardware devices capable of controlling, reacting to, generating sound and processing sound in collaboration with web audio-based applications, we present artistic endeavors in creating and composing with the recent frameworks Bendit.I/O¹ and NexusHUB². Together they allow nearly any commercially-available electronic device to be cyber-hacked and performed collaboratively along with web audio applications in typical D/NMP settings. Interrogation of select historical precedents and an explanation of the Bendit.I/O and NexusHUB frameworks, precede discussion of the *gravity|density* performance showcasing the potential for expanding the pool of web audio art into the realm of re-purposed technology and the Orchestra of Things (OOT/IoMusT).

1. INTRODUCTION

The proliferation of compact, powerful, and accessible mobile devices across the planet provide artists of distributed and networked music with a hardware-based outlet for disseminating web-based graphics and audio in performance. A 2019 study by the Pew Research Center found that an estimated 5 billion people globally own mobile devices, with half

of those devices being smart phones and the strongest concentration of those phones located in nations with advanced economies[17]. With their near global ubiquity, plethora of on-board sensors, and integration of multi-point click/touch interactions through high-quality touchscreens, it is not surprising to see these devices become fully integrated into the world of web audio.



Figure 1: gravity|density spinning discs

1.1 Incorporating Cyber-Hacked Devices

While smartphones and other mobile devices will no doubt continue to play a prominent role in the performance of interactive Web Art, new research in the realm of the Internet of Musical Things (IoMusT)[18] and the Orchestra of Things[4] demonstrates a creative potential in utilizing custom-made web-enabled instruments that, in conjunction with mobile devices, can be seamlessly merged with web audio systems in D/NMP performance settings. At the same time, the proliferation of IOT and smart home devices (ranging from automated security systems to remote-controlled cookware) in the worldwide consumer market may allow for audience members to become more familiar with the activity of remotely interacting with and influencing the actions of hardware through their mobile devices.

As the traditional web audio toolsets become more popular and advanced, the time is right to expand the pool of possible hardware devices capable of controlling, reacting to, generating sound and processing sound in collaboration with web audio-based applications. Our interest was peaked when we considered the possibilities of merging the artistic practice of circuit bending/hardware hacking (which centers on the act of creating avant-garde instruments out of existing consumer toys and hardware through modification of the device's internal circuitry[15][6]) with the same approach to networked performance topologies practiced by web audio

¹<https://www.benditio.com/>

²<https://github.com/nexus-js/nexusHub>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

web audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

artists. Performing with hacked hardware draws connections to the Upcycling movement and highlights a nostalgic connection between the audience, performer, and the device. It is our hope that interacting with repurposed technology through the use of mobile devices will serve as a natural extension of web interactivity with online media. For example, the same touch and drag interactions across a graphical, web page UI often used for modifying playback of a sound sample could now extend to the skipping and seeking functions of a hardware CD player or tape machine.

This research led to the creation of two new frameworks: Bendit_I/O[9] by the first author and NexusHUB[3] by the second author. Together, these hybrid hardware/software frameworks allow almost any commercially-available electronic device to be cyber-hacked and performed collaboratively along with web audio applications in typical D/NMP settings.

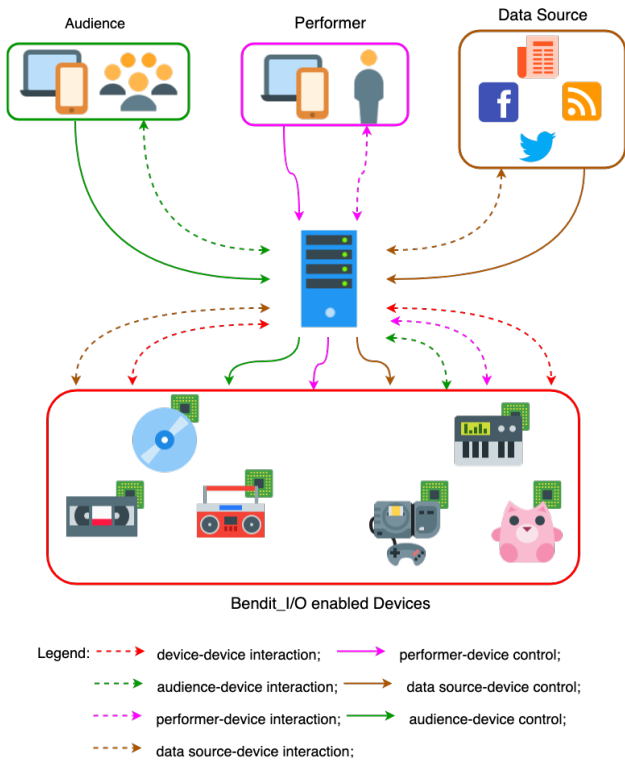


Figure 2: Possibilities for interaction and control between connected users and hacked devices using the Bendit_I/O and NexusHUB frameworks.

2. HISTORICAL PRECEDENT

Performance using interconnected electronic devices has been explored since the advent of electronic devices. The development of modular synthesizers, explorations by David Tudor[5], explorations at the San Francisco Tape Music Center, and such groups as the HUB all provided prescient work at connecting computer and electrical devices for making music. Our exploration is specifically rooted in manipulating the internet, the standardized global computational communication platform, to connect with electronics that were not designed to be able to communicate over this network. To

reach this state, these prior explorations happened:

1) *Traversal*[7], a series of pieces from 2007 to 2012 by John Fillwalk and Jesse Allison using a web server to connect audience cell phones; touch screen interfaces; virtual objects, structures and avatars in Second Life; and musically performable physical objects such as a bell tower, carillon, and midi-controlled organ together for live performance.

2) control of networked musical robotic devices exemplified by the work of Eric Singer and the League of Electronic Musical Urban Robots[14] [10] [13], the work of Ajay Kapur and collaborators [19] [8], and the robotic musical works of Trimpin[8].

3) hacking of electronics to make musical and artistic sonic art such as the CD skipping artwork of Nicolas Collins[16] the techniques of which were codified and disseminated in the original *Hardware Hacking* book and the subsequent book *Handmade Electronic Music*[6]

4) Web Audio and distributed performance systems enabling interconnected, browser-based music making such as NexusHUB, Sebastian Piquemal’s Rhizome project [12], and the Soundworks framework from IRCAM [11]. These helped bridge the gap between musical distributed performance and performance over an Orchestra of Things or Internet of Things.

Each of these developments coalesced into a number of prior artworks by the artists including Anthony Marasco’s *The Spinning Earth will Spread Before You* (2019) for two web-controlled circuit-bent CD players and the works *Causeway* and *Diamonds in Dystopia* by Jesse Allison, Derick Ostrenko, and Vincent Cellucci for poet and audience mobile devices[2] [1].

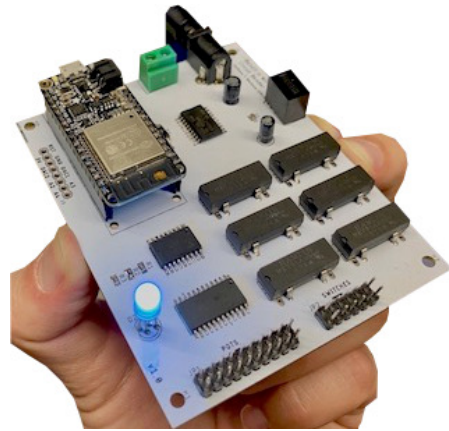


Figure 3: The Bendit_I/O board.

3. OVERVIEW OF HARDWARE & SOFTWARE DESIGN

The Bendit_I/O system consists of two components: a Wi-Fi-enabled I/O board (which users connect to a hacked device) and an accompanying software. The Bendit_I/O board contains an ESP32 microcontroller, an eight-channel digital potentiometer IC, eight reed relay switches, and a dual-H bridge motor driver. Output signals from the board terminate in multi-pin headers or screw terminal connectors, allowing users to replace physical switches or dials typically

used for connecting bend points on a hacked device with the cabling attached to a Bendit board[9].

Connecting the Bendit-hacked devices is a NexusHUB server, allowing them to communicate both amongst themselves and with any other web-enabled devices (mobile devices, laptops, etc.).

The NexusHUB framework is a tool for managing distributed performance across browsers and OOT/IOMusT arrangements utilizing web-sockets. It offers a relatively standard setup of various pages for audience members, performers/controller pages, web-socket connections to external audio or media applications, and a page for projection/display. In conjunction with Bendit_I/O it provides a centralized server that can sample recordings of sounds live at the server, can serve client web pages to audience members which can sample and upload recordings from their device, and can distribute the live recordings to any client browser for playback and/or manipulation. As the recordings are also on the server, they can be controlled and played back directly on the central server over house sound system, or into further processing devices.

4. COMPOSING WITH THE SYSTEM

Working within this combined system, we have composed *gravity|density*, an interactive work for cyber-hacked devices and web audio applications.

In *gravity|density*, we begin by manipulating fixed-audio sources through the performance of hacked CD players. The sonic results of this mangled audio is sampled by audience members and can be played back alternately by the audience, and then shifting into passive mode and performed dynamically across the field of audience mobile devices. This passive distribution of recordings allow us to create intricately-spatialized rhythmic interplay between the glitching CD players and the blanket of overlapping samples dispersed throughout the networked audience. Active distributions allow the audience to join in our performance; by choosing small portions of the audio sampled and sending these selected samples back to us, we string this audio together and feed it into a cyber-controlled distortion pedal before feeding it back to the audience for more sampling and manipulation. This results in overlapping cycles of control and audio generation between performer, audience, network, and machine.

The work received its world premiere in Trondheim, Norway at the fifth Web Audio Conference in December of 2019. Figures 3-4 demonstrate performance elements of *gravity|density* including the cyber-hacked CD players and distortion pedal, and the audience-participation web interface.

5. FUTURE DIRECTIONS

The Bendit_I/O and NexusHUB systems are almost infinitely expandable. As new electronic objects are hacked and able to be controlled, it expands not only the sound palette that is available, but also folds in the social meanings inherent in the physical interactions with each object. This meaning can then be used to make connections between the audience and the disturbed technology. Looking forward, electronics such as coffee grinders, flashlights, alarm clocks, toothbrushes, and other day-in-the-life type objects can be incorporated for pieces that abstract, or toy with the sounds

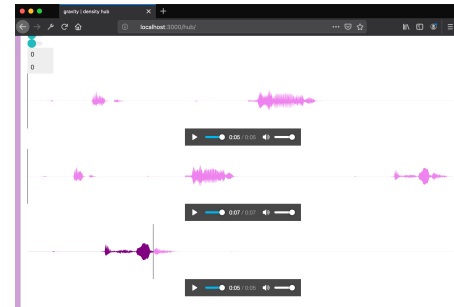


Figure 4: *gravity|density* hub containing samples captured by the audience



Figure 5: *gravity|density* performed with Bendit_I/O cyber hacking

of our 21st century lives. Drones are also a natural hacking target as they are becoming increasingly inexpensive, noisy, and have a developing identity in unregulated intrusiveness via surveillance and warfare that offers substantial areas for artistic exploration and societal critique.

On the systemic side, further development of the control structures for interacting with hacked electronics are being explored. Many of these are moving away from direct control of the device and into mapped or process based control that may have more meaning, or at the very least allow for performers to incorporate more devices in sophisticated arrangements. These include using data sources to sequence and trigger events, procedural control that can create musical textures, aggregated audience input, and events generated by devices which are mapped to other devices creating a cyber-rube-goldberg machine.

6. DOCUMENTATION

Further information about our work composing with these frameworks can be found at <https://gravity.emdm.io/>.

7. REFERENCES

- [1] J. Allison, V. Cellucci, and D. Ostrenko. Creative data mining diamonds in dystopia. *Media-N*, 13(1), 2017.
- [2] J. Allison, D. Ostrenko, and V. A. Cellucci. Causeway. Georgia Institute of Technology, 2016.
- [3] J. T. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *NIME*, pages 1–6, 2013.
- [4] S. D. Beck and C. Branton. The orchestra of things: a new approach to managing, controlling, and composing for laptop and mobile device ensembles.

Presented at the 30th Annual SEAMUS conference, Virginia Tech, Blacksburg, VA, 2015.

- [5] N. Collins. Composers inside electronics: Music after david tudor. *Leonardo Music Journal*, 14(1):iv–1, 2004.
- [6] N. Collins. *Handmade electronic music: the art of hardware hacking*. Routledge, 2014.
- [7] J. Fillwalk and J. Allison. Between two worlds: Virtuality in arts and teaching. *MG 2009 Proceedings*, page 6, 2009.
- [8] A. Kapur, M. Darling, J. Murphy, J. Hochenbaum, D. Diakopoulos, and T. Trimpin. The karmetik notomoton : A new breed of musical robot for teaching and performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 228–231, Oslo, Norway, 2011.
- [9] A. T. Marasco, E. Berdahl, and J. Allison. Bendit_I/O: A system for networked performance of circuit-bent devices. In M. Queiroz and A. X. Sedó, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 331–334, Porto Alegre, Brazil, June 2019. UFRGS.
- [10] B. Neill and E. Singer. Ben neill and lemur. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 331–331, Pittsburgh, PA, United States, 2009.
- [11] S. G. B. M. Norbert Schnell, Jean-Philippe Lambert. Soundworks, 2015.
- [12] S. Piquemal. Rhizome, 2014.
- [13] E. Singer, J. Feddersen, and B. Bowen. A large-scale networked robotic musical instrument installation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 50–55, Vancouver, BC, Canada, 2005.
- [14] E. Singer, J. Feddersen, C. Redmon, and B. Bowen. Lemur’s musical robots. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 181–184, Hamamatsu, Japan, 2004.
- [15] J. Skjulstad. Circuit bending as an aesthetic phenomenon. Master’s thesis, University of Oslo, 2016.
- [16] C. Stuart. Damaged sound: Glitching and skipping compact discs in the audio of yasunao tone, nicolas collins and oval. *Leonardo Music Journal*, pages 47–52, 2003.
- [17] K. Taylor, L. Silver, K. Taylor, and L. Silver. Smartphone ownership is growing rapidly around the world, but not always equally. *Pew Research Center’s Global Attitudes Project*, Feb 2019.
- [18] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. Internet of Musical Things: Vision and Challenges. *IEEE Access*, 6:61994–62017, 2018.
- [19] N. D. Villicaña-Shaw, S. Salazar, and A. Kapur. Mechatronic performance in computer music compositions. In T. M. Luke Dahl, Douglas Bowman, editor, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 413–418, Blacksburg, Virginia, USA, June 2018. Virginia Tech.

Flexible visualization of sound collections

Gerard Roma
CeReNeM
University of Huddersfield
groma@hud.ac.uk

ABSTRACT

This submission proposes a talk on content-based visualization of sound collections. Previous works have covered visualization of sounds using content-based descriptors, and visualization of collections using dimensionality reduction algorithms. Combining both provides a flexible framework for devising novel web audio interfaces.

1. INTRODUCTION

The availability of the digital sampler had a huge impact in the history of electronic music, and audio sampling is a basic practice of many musical genres. A sampler is in itself a computer with a specialized interface for music production. Hence, as computers became ubiquitous, many musicians switched to general purpose computers for sampling. Some advantages of general purpose computers are easier file management via the desktop metaphor, and a large screen. In this sense it is a bit strange that so many software samplers (e.g; the EXS24 sampler that ships with Logic) emulate vintage hardware. Alternative approaches for playing samples on general-purpose computers have evolved from the traditions of granular and concatenative sound synthesis [5]. Within this context, content-based descriptors developed in domains such as automatic speech recognition or music information retrieval can be leveraged for indexing, browsing and visualizing sounds.

2. MAPPING SOUND COLLECTIONS

This talk will summarize recent research by the author and collaborators on visualization of sound collections, with the aim of enabling novel interfaces for creative applications on general-purpose computers, phones and tablets. There are two main aspects to visualizing a sound collection. One is the location of each sound in a 2D (sometimes 3D) display. This has been demonstrated by several contributions to the web audio conference (WAC) [4, 1]. Typically, a dimensionality reduction algorithm is used to project audio feature statistics extracted from an audio segment to the display. In a recent work, we compared several algorithms and demonstrated their application to web-based multitouch

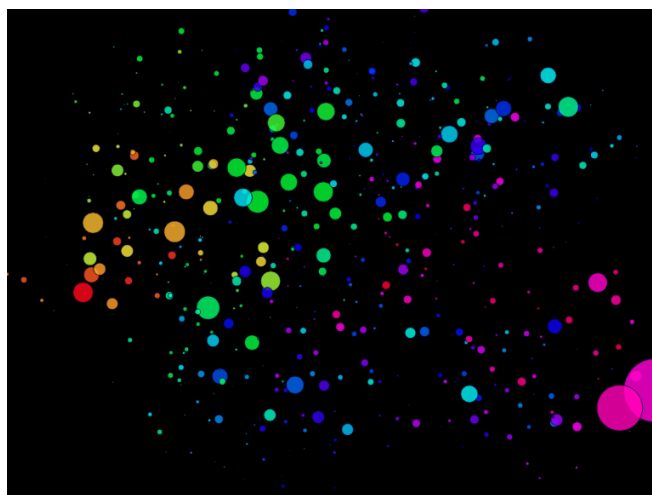


Figure 1: Sound collection scatterplot.

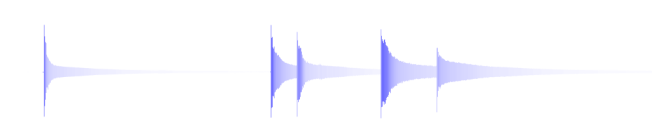


Figure 2: Waveform visualization.

interfaces [2]. An example is shown in Figure 1.

Another issue is how to represent each sound. For this task, a library for audio visualization of descriptors was presented at WAC 2018 [3]. Multiple combinations of plot type and descriptors are possible. Figure 2 shows an example using a wave plot where luminance is controlled by RMS amplitude.

As a follow-up, this talk will describe further work on a flexible framework for combining dimensionality reduction and sound visualization. An example using spectral shape descriptors is shown in Figure 3.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

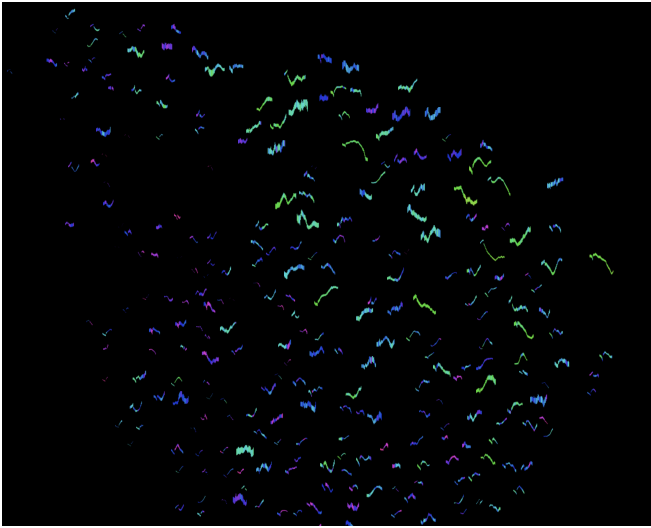


Figure 3: An example sound map.

3. ACKNOWLEDGMENTS

This research was part of the Fluid Corpus Manipulation project (FluCoMa), which has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 725899).

4. REFERENCES

- [1] F. Font Corbera. Freesound explorer: make music while discovering freesound! 2017.
- [2] G. Roma, O. Green, and P. Tremblay. Adaptive mapping of sound collections for data-driven musical interfaces. In *NIME – The International Conference on New Interfaces for Musical Expression*, Proceedings of the conference on New Interface for Musical Expression (NIME), 3 2019.
- [3] G. Roma, O. Green, A. Xambó, and P. Tremblay. A javascript library for flexible visualization of audio descriptors. In *Proceedings of the 4th Web Audio Conference, WAC-2018*, 9 2018.
- [4] G. Roma and X. Serra. Music performance by discovering community loops. In *Proceedings of the Web Audio Conference (WAC)*, 2015.
- [5] D. Schwarz. Concatenative Sound Synthesis: The Early Years. *Journal of New Music Research*, 35(1):3–22, 2006. cote interne IRCAM: Schwarz06b.



Posters

Ongaq JS: An elementary library for programming music

Hiroyuki Takakura
CodeNinth Ltd, Tokyo
708 9-20 Higashi-Kishicho, Urawa, Saitama
takakura@codeninth.com

ABSTRACT

In this paper, I describe concepts and usages of Ongaq JS, a new elementary JavaScript library for programming music. This library makes it possible for many people, especially beginners or young students, to program music using various sound resources.

We, CodeNinth Ltd., are going to make this library accessible as OSS [1] and open a website [2] to read documents, view samples and manage licenses in 2019.

1. CONCEPTS

The main concept of Ongaq JS is to make it easier and more pleasant to program music. These days, many people want to acquire skills of programming and create what can keep motivating them to study. Therefore, we have developed Ongaq JS for this purpose.

2. USAGES

In this section, I describe usages of Ongaq JS in order of appearance in general program.

2.1 Create An Account

Before starting to write code, we have to create an account of Ongaq JS and obtain an API key. This API key is required for fetching sound resources from our server. The details of the sound resources are described in my previous papers [3].

2.2 Prepare A Context

Prepare a context of Ongaq JS by initializing an Ongaq object (See Figure 1). This object imports Part objects which constitute music compositions. We can play and pause them by methods on the Ongaq object.

```
const context = new Ongaq({  
  api_key: "MY_API_KEY",  
  volume: 50,  
  bpm: 130,  
  onReady: ()=>{  
    // ready to play  
  }  
})
```

Figure 1. Initializing an Ongaq JS context

2.3 Add Part Objects

Create Part objects to form our music compositions. A Part object can be compared to a member of a band. We assign each Part object a sound resource and attach them to the Ongaq object. (See Figure 2)



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

```
const context = new Ongaq({  
  api_key: "MY_API_KEY",  
  volume: 50  
})  
  
const guitar_part = new Part({  
  sound: "jazz_guitar"  
})  
  
context.add( guitar_part )
```

Figure 2. Attaching a Part object to the context

2.4 Define Behaviors with Filter Objects

We can add Filter objects to the Part objects to define their behaviors. There are various types of Filter objects which define different behaviors. For example, a “note” type Filter defines when and which musical scales are played and a “pan” type Filter defines when and from which direction sounds come. Note that for some instruments, names like “hihat” or “kick” may be used instead.

In Figure 3, the Part object is defined to play “C2”, “D2#”, “G2” at beat 0 and beat 8 for 4 beat length. The function which is assigned to “active” property is called right before each beat while the context is playing, receiving the beat index as argument.

```
guitar_part.add(  
  new Filter({  
    key: ["C2", "D2#", "G2"],  
    length: 4,  
    active: (beat, measure) => {  
      return beat === 0 || beat === 8  
    }  
  })  
)
```

Figure 3. Adding a Filter object to the Part object

By default there are 16 beats in a measure. This can be overwritten by settings of Part objects.

2.5 Chord: Helper Object

Chord is a helper object which provides a list of musical scales consisting of a chord. It also has some methods to shift or rearrange them. This makes it easier for developers to obtain harmony.

Here is an example of utilizing a Chord object in Figure 4.

```
guitar_part.add(  
  new Filter({  
    key: new Chord("Cm7"),  
    length: 4,  
    active: (beat, measure) => {  
      return beat === 0 || beat === 8  
    }  
  })  
)
```

Figure 4. Utilizing a Chord object

3. NOTATION

3.1 Musical Scales

We can write musical scales like “[scale][octave][#(optional)]”. Available scales are from “C1” to “B4” for 4 octaves.

Instead of this notation, we can also use calculable one as “[index of scale][octave]”. “1\$1” corresponds to “C1” and “4\$12” corresponds to “B4”.

3.2 Chords

We can use general notation of chords as shown in Table 1. However, write directly “b” instead of “b”.

Table 1. Some examples of chord names

Chord Name	Root	Scheme
Cm7	C	m7
GbM9	Gb	M9
Am7(11)	A	m7(11)

4. TECHNICAL POINTS

4.1 Scheduling Sounds

The most basic function of precisely scheduling sounds is based on the idea described in the article of HTML5 Rocks [4]. While the context is playing, the context continuously collects chains of AudioNode objects from Part objects.

4.2 Considering Memory

Ongaq JS potentially uses a lot of memory, and we have worked hard to minimize its memory footprint. After repeated use, the program would sometimes crash, and we found that this was related to the number of allocated objects. By reducing object allocation to a minimum the program no longer experiences fatal performance problems.

5. FUTURE WORK

5.1 Enhancement

We plan to keep upgrading Ongaq JS. We would like to make it rich enough to use for live performance or interactive musical art. Furthermore, in order to make the program more accessible for beginners, we plan to provide enough documents, samples and guidance for them to study by themselves.

5.2 Tools for STEM Education

We plan to develop visual programming tools for STEM education with Ongaq JS. In Japan, the government has decided to start lessons to cultivate basic skills of programming at all elementary schools in 2020. We want to provide suitable tools which can be applied to those needs.

As we can find at preceding projects, such as *Music Blocks* [5], music is really suitable for students to make friends with numbers, calculation or abstract concepts like variables or iteration. Therefore, we also plan to provide tools which encourage students learn programming and mathematics inclusively.

6. ACKNOWLEDGEMENTS

I would like to thank Torkel Berli from Wovn Technologies, Inc. for reviewing English usage, Hikaru Takakura, co-founder and engineer of CodeNinth Ltd., for taking part in discussions and code reviewing.

7. REFERENCES

- [1] CodeNinth Ltd., 2019. The GitHub repository of Ongaq JS. Retrieved from <https://github.com/codeninth/ongaq-js>
- [2] CodeNinth Ltd., 2019. The portal site of Ongaq JS. Retrieved from <https://www.ongaqjs.com/>
- [3] Takakura, Hiroyuki. 2017. WebbyJam, a Web Tune Editor to Find Enjoyment. In *Proceedings of 3rd Web Audio Conference* (London, U.K., August 21 – 23, 2017). <https://qmro.qmul.ac.uk/xmlui/handle/123456789/28066>
- [4] Wilson, Chris. 2013. A Tale of Two Clocks - Scheduling Web Audio with Precision. Retrieved from <https://www.html5rocks.com/en/tutorials/audio/scheduling/>
- [5] Sugar Labs. 2019. The GitHub repository of Music Blocks. Retrieved from <https://github.com/sugarlabs/musicblocks>

8. APPENDIX

8.1 Sample Code

This is a sample code that corresponds to a looping music consisting of drums and keyboard.

```
const context = new Ongaq({
  api_key: "MY_API_KEY",
  volume: 50,
  bpm: 130,
  onReady: ()=>{
    // ready to play
  }
})

const keyboard = new Part({
  sound: "plain_keyboard",
  measure: 4
})

keyboard.add( new Filter({
  key: new Chord("GbM9"),
  length: 16,
  active: (beat,measure)=>{
    return beat === 0 && measure === 8
  }
}) )
context.add( keyboard )

const drums = new Part({
  sound: "small_cube_drums"
})

drums.add( new Filter({
  key: beat => beat % 8 === 0 ?
    "kick" : "hihat",
  active: beat => beat % 4 === 0
}) )
drums.add( new Filter({
  type: "pan",
  active: beat => beat % 8 === 4,
  x: (beat,measure)=> measure % 2 === 0 ?
    45 : -45
}) )
context.add( drums )
```

A 2 Million Commercial Song Interactive Navigator

Michel Buffa
Jerome Lebrun
Université Côte d'Azur
CNRS, INRIA
{buffa, lebrun}@i3s.unice.fr

Johan Pauwels
Queen Mary University of
London
j.pauwels@qmul.ac.uk

Guillaume Pellerin
IRCAM
guillaume.pellerin@ircam.fr

ABSTRACT

In this paper, we present a web-based interactive tool for exploring a collection of two million commercially released songs. It gathers song information from a large number of heterogeneous sources, web-based and audio-based, and integrates work from multiple research groups. The resulting tool can be used to request information about a specific song such as lyrics, metadata and chords; to navigate further on to linked external resources such as Discogs, AllMusic, MusicBrainz or a number of streaming providers; or to browse the collection by artist's discographies or band membership. Several Web Audio applications are integrated and use the dataset to enrich the experience.

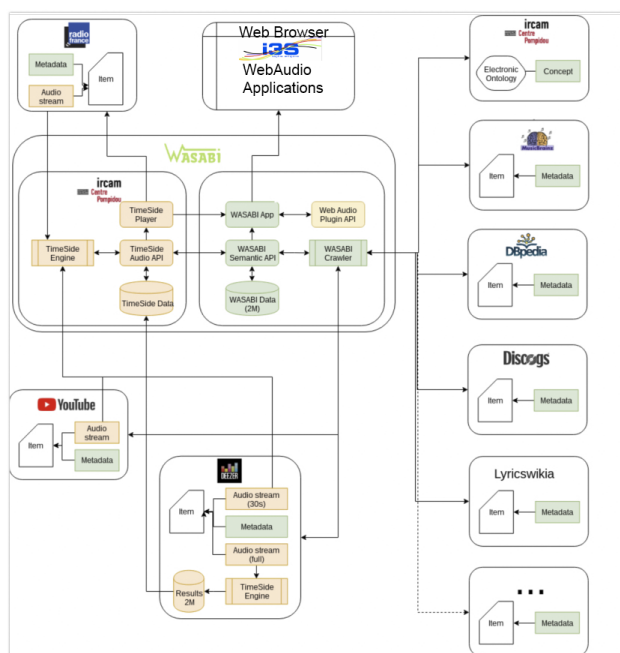


Figure 1: a schema of the WASABI network architecture and workflow.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

1. INTRODUCTION

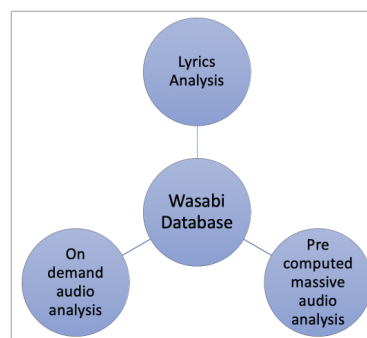


Figure 2: the metadata of the WASABI database is complemented by computational analysis of lyrics and audio, both on-demand and precomputed.

Since 2017, a 2M song database consisting of metadata collected from the Web of Data [4] has been constructed in the context of the WASABI research project¹. The heterogeneous sources that have been consulted are displayed on the right of Figure 1.

These metadata include the identifiers of the corresponding audio on a variety of audio platforms, which allowed to enrich the database with computational analyses of the lyrics and audio data (see Figure 2). Several research groups have contributed to the analysis and have built interactive Web Audio applications on top of the output. For example, IRCAM linked their TimeSide analysis and annotation tool to make on-demand audio analysis possible. Queen Mary University of London, through the FAST project², performed an offline chord analysis of 442k songs, and built an online enhanced audio player [15] and chord search engine [16] around it. The I3S laboratory, responsible for the design and implementation of the database, extracted song structure based on lyrics analysis [10, 12]. Furthermore, they have developed a rich set of Web Audio applications and plugins [6, 7] that allow, for example, songs to be played along with sounds similar to those used by artists.

These metadata, computational analyses and Web Audio applications have now been gathered in one easy-to-use web interface, the WASABI Interactive Navigator, which is presented in this paper.

¹<http://wasabihome.i3s.unice.fr/>

²<http://www.semanticaudio.ac.uk>

2. RELATED WORKS

Other research projects aimed to collect metadata on a large set of commercial songs, such as The Million Song Dataset project from 2011, which is mainly based on audio data [3] and did not exploit the availability of large structured data sources from the Web of Data to the full extent.

MusicWeb and its successor MusicLynx [1] link music artists within a Web-based application for discovering connections between them and provides a browsing experience using extra-musical relations. The project shares some ideas with WASABI, but works on the artist level, and does not perform analyses on the audio and lyrics content itself. It reuses, for example, MIR metadata from AcousticBrainz.

The WASABI project has been built on a broader scope than these projects and mixes a wider set of metadata, including ones from audio and natural language processing of lyrics. In addition, as presented in this paper, it comes with a large set of Web Audio enhanced applications.

3. THE WASABI INTERACTIVE NAVIGATOR HOMEPAGE

The primary starting point for the interactive navigator is the home screen as seen in Figure 3. From here, a textual search can be performed for one of the 2 million songs, 200k albums or 77k artists included in the database [4]. The resulting page displays information aggregated from a huge set of online sources, as shown in Figure 4.

A set of tabs at the top link directly to the tools based on the computational analysis of audio and lyrics, which will be discussed next.

4. SONG STRUCTURE DERIVED FROM LYRICS AND AUDIO

On the pages for individual songs, the musical structure is indicated by grouping the lyrics in blocks, as shown in Figure 5. Song lyrics contain repeated patterns that facilitate automated segmentation, with the detection of constitutive elements of a song text (e.g., intro, chorus) as goal.

We proposed to segment lyrics by applying a convolutional neural network to a synchronized audio-text representation of a song. First, we created a corpus projecting the segmentation of the lyrics of the WASABI corpus onto a synchronized lyric-audio corpus (DALI corpus³). We have shown that the information in the text enriched with the characteristics of the audio signal allows our segmentation model to surpass the state of the art method, which is based solely on textual characteristics [10, 12].

5. CHORD PLAYER AND SEARCH

Chord transcriptions can be requested from the song pages, and are presented as an interactive player showing the chords in sync with the music, such that musicians can play along with them (see Figure 6, top). As an alternative entrypoint to the WASABI database, a chord searching interface is available to find specific songs that contain a certain set of chords (bottom of Figure 6). This type of search interface has been previously tested with the Jamendo music collection [16].

³<https://github.com/gabolsgabs/DALI>

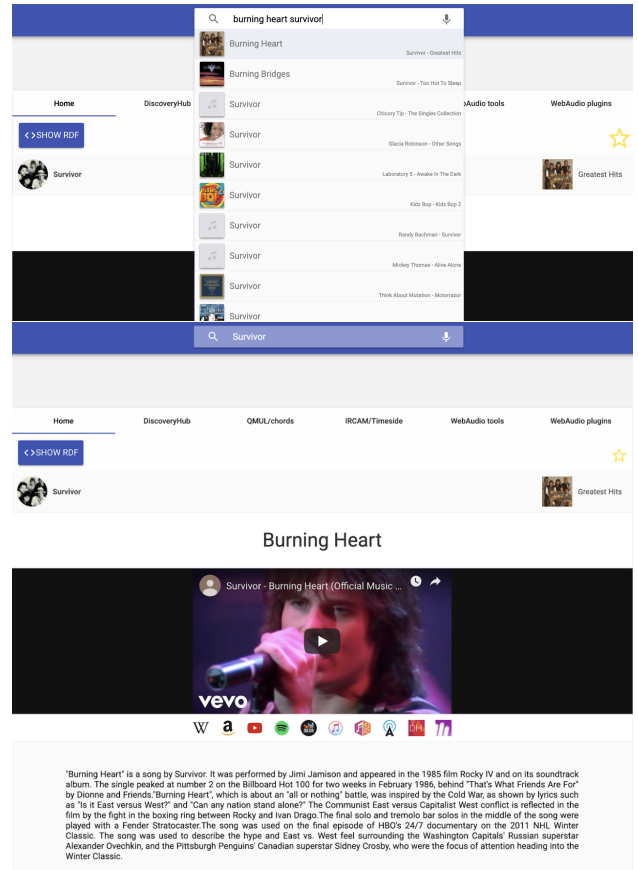


Figure 3: main page of the WASABI navigator interface.

For the search interface to work, the songs need to be indexed offline. So far, 442k files have been analysed with the chord analysis algorithm proposed in [14]. In order to comply with copyright requirements, a remote processing toolchain has been set-up in which algorithms packaged as a Docker container are sent to Deezer for processing on their servers. The output is then returned to us and stored in the WASABI database. Calculation of further music descriptors such as tempo and key is ongoing.

6. ON-DEMAND AUDIO ANALYSIS

On-demand audio analysis is complementary to pre-computed analysis in the sense that it avoids large, up-front computational costs and scales easily to changes in data and algorithms. It comes with its own challenges, however. Building a web platform that depends on various external services requires that the underlying software architecture and data model must be robust against disappearing sources. For example, if a YouTube video gets removed, we still want its computational analysis to remain available in case any of the web services built on top refer to the track and its related metadata. The TimeSide framework⁴ has been designed to provide a RESTful API as well as plugin based core library dedicated to audio processing that can

⁴<https://github.com/Parisson/TimeSide/>



Figure 4: a large set of online databases has been exploited for building the metadata part of the WASABI database.

provide this resilience.

In the context of the WASABI project, we demonstrate how the service (see Figure 1) can be used on demand by a master semantical application that will feed TimeSide with URLs coming from various providers (YouTube, Deezer, etc.) and then dynamically return the analysis data to the client application to feed Web Audio based tools. A number of audio-based characteristics can be requested from a song's page in this way.

7. MULTITRACK PLAYER AND EFFECTS

In order to assist music schools, which is one of the target audiences of the WASABI project, some Web Audio tools



Figure 5: song lyric structure detection [10].

have been integrated into the Navigator. One possible scenario is that music teachers pick a song from the database for their students to learn. An enhanced multi-track player (Figure 7) is presented to the users, which displays the song's sheet music and allows to play back or selectively mute the different instruments in the song. Furthermore, a simplified DAW is available in the browser for recording and playing back student or teacher performances. It includes real-time audio effects, with ready to use presets, to attain a realistic and attractive studio sound.

The in-browser audio effects have been implemented using a Web Audio plugin standard [8,9] (including SDK, on-line validation tools and examples of host applications), for which a large set of plugins⁵ already exists. An online IDE for designing these plugins [11,13] and a guitar tube amp simulator designer [6,7] are also available for developers.

8. FUTURE ENHANCEMENTS

Fine-tuning the parameters in many of our Web Audio applications (tube amplifiers, pedals, etc...) has been quite tedious and time-consuming with a high level of expertise

⁵<https://github.com/micbuffa/WebAudioPlugins>

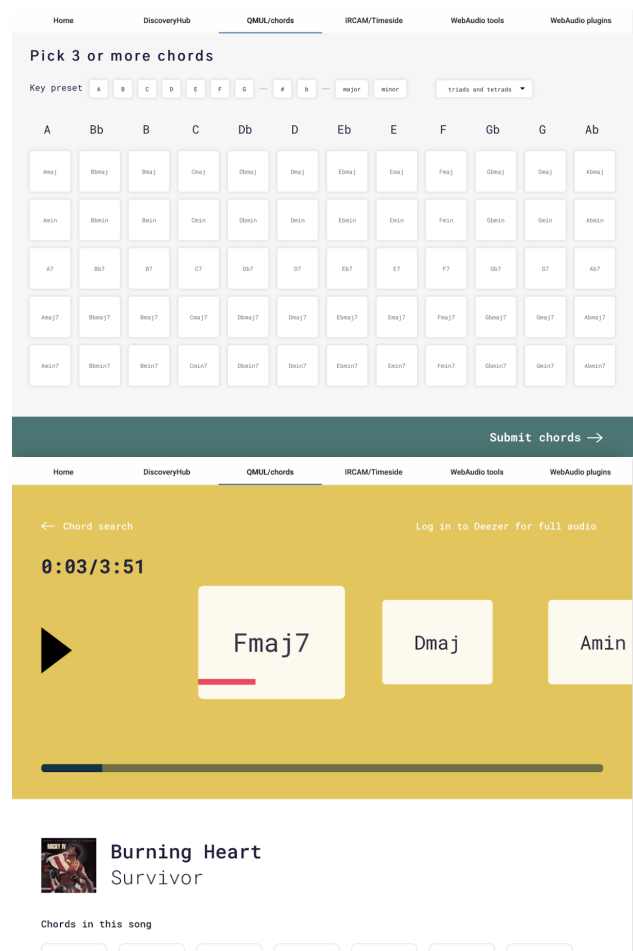


Figure 6: augmented player and search engine, integrated in the navigator interface.

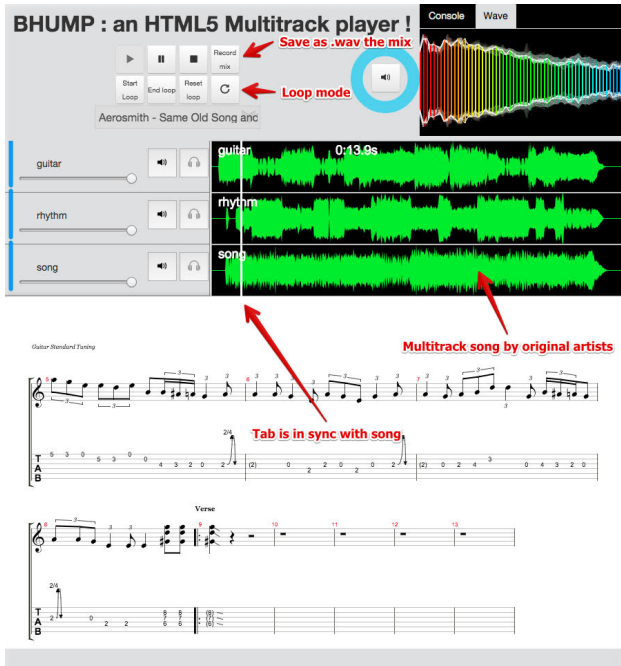


Figure 7: a multitrack player capable of displaying Guitar Pro music tabs in sync with the audio [5].

required [6, 7]. To ease this process of presetting for music school students, we are currently exploring machine-learning based approaches (with scattering wavelet based convolutional neural networks [2]) to learn and extract the relevant features from large datasets of songs and to match these with presets leading to similar subjective timbre in simulated instruments and also to automatically classify songs in the WASABI database.

Furthermore, offline indexation of more audio-based characteristics is ongoing.

9. ACKNOWLEDGMENTS

This work was supported by the French Research National Agency (ANR) and the WASABI team (contract ANR-16-CE23-0017-01). Part of this work has been funded by UK EPSRC grant EP/L019981/1.

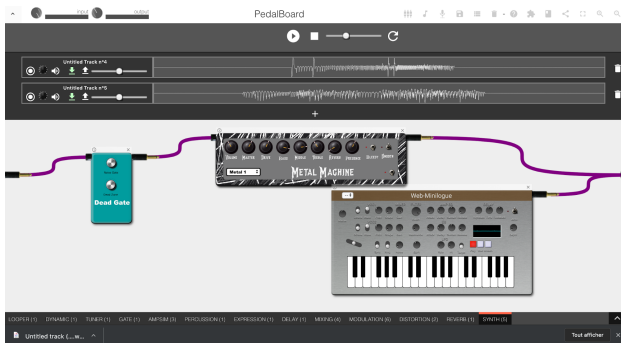


Figure 8: Web Audio embedded DAW and effect chain for recording and playback.

10. REFERENCES

- [1] A. Allik, F. Thalmann, and M. Sandler. MusicLynx: Exploring music through artist similarity graphs. In *Companion Proc. (Dev. Track) The Web Conf. (WWW 2018)*, 2018.
- [2] J. Anden and S. Mallat. Deep scattering spectrum. *IEEE Trans. Sig. Proc.*, 62(16), 2014.
- [3] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset. In *Proc. 12th Int. Soc. Music Information Retrieval (ISMIR 2011)*, 2011.
- [4] M. Buffa et al. WASABI: a two million song database project with audio and cultural metadata plus WebAudio enhanced client applications. In *Proc. 3rd Web Audio Conf. (WAC 2017)*, 2017.
- [5] M. Buffa, A. Hallili, and P. Renevier. MT5: a HTML5 multitrack player for musicians. In *Proc. 1st Web Audio Conf. (WAC2015)*, 2015.
- [6] M. Buffa and J. Lebrun. Real time tube guitar amplifier simulation using webaudio. In *Proc. 3rd Web Audio Conference (WAC 2017)*, 2017.
- [7] M. Buffa and J. Lebrun. Web audio guitar tube amplifier vs native simulations. In *Proc. 3rd Web Audio Conf. (WAC 2017)*, 2017.
- [8] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, and S. Letz. Towards an open web audio plugin standard. In *Companion Proc. (Dev. Track) The Web Conf. (WWW 2018)*, 2018.
- [9] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, S. Letz, and G. Pellerin. WAP: Ideas for a Web Audio plug-in standard. In *Proc. 4th Web Audio Conf. (WAC 2018)*, 2018.
- [10] M. Fell, Y. Nechaev, E. Cabrio, and F. Gandon. Lyrics segmentation: Textual macrostructure detection using convolutions. In *Proc. 27th Int. Conf. on Computational Linguistics (COLING2018)*, 2018.
- [11] S. Letz, Y. Orlarey, and D. Fober. Compiling Faust audio DSP code to WebAssembly. In *Proc. 3rd Web Audio Conf. (WAC 2017)*, 2017.
- [12] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters. DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *Proc. 19th Int. Soc. of Music Information Retrieval (ISMIR 2018)*, 2018.
- [13] Y. Orlarey, D. Fober, and S. Letz. Syntactical and semantical aspects of Faust. *Soft Computing*, 8(9), 2004.
- [14] J. Pauwels, K. O'Hanlon, G. Fazekas, and M. Sandler. Confidence measures and their applications in music labelling systems based on hidden Markov models. In *Proc. 18th Int. Soc. Music Information Retrieval (ISMIR 2017)*, 2017.
- [15] J. Pauwels and M. Sandler. A web-based system for suggesting new practice material to music learners based on chord content. In *Joint Proc. 24th ACM IUI Workshops (IUI2019)*, 2019.
- [16] J. Pauwels, A. Xambó, G. Roma, M. Barthet, and G. Fazekas. Exploring real-time visualisations to support chord learning with a large music collection. In *Proc. 4th Web Audio Conf. (WAC 2018)*, 2018.

Cooperation experiments in web-based audiovisual works

Balázs Kovács, dr. habil.

University of Pécs, Faculty of Arts, Department of Electronic Music and Media

H-7630 Pécs, Zsolnay Vilmos u. 16.

kovacs.balazs@pte.hu

ABSTRACT

Network possibilities are inherently ready for real-time cooperation among the users of the same website, in order to participate in running a multimedia installation, webaudio instrument, or audio-game. In this paper I summarize my works in this field, starting from web-only projects (Webcards, Forgattató, WebSynth), finishing at communication between locative installations and the web (Ring the Web!). Most of them are radically cooperative, which means that they aren't running stand-alone or even off-line because of their multi-user approach. With this initiative I argue that networked audiovisual artworks and instruments realize a communicative way of performing.

1. INTRODUCTION

Web and networked cooperation use the same technology: anonymous, asynchronous communication between server and client, UDP protocol for quick access. But the peer-to-peer communication among users of the web has been limited for decades for file sharing. With the spread of node.js's websocket and socket.io [1], these boundaries have been overridden, starting to making the web really decentralized and transparently communicative. In this paper, I focus on my audio-visual projects, while mentioning, that it's only a first sign of opening the client-(server)-client hierarchy toward a community-based, self-regulated (or regulation-free) network.

2. WEB-BASED COOPERATIVE WORKS

Networked collaboration in media art has a long history: with the beginning of the internet, we meet with several ideas starting from the telepresence-gardening of Ken Goldberg [2], through decentralized network orchestras [3] and cooperative composition solutions like Playsound.space [4] or EarSketch [5] to distributed audio performances.

2.1 Webcards (2016)

In my definition webcards are contemporary postcards: they're multimedia/network-based ones, including various visual, textual and auditory data. My first experiment with webaudio and node.js was created during a scholarship at ICEM-Folkwang Hochschule, Essen, Germany, where I used the website to present audio-visual

collages controlled by multiple users. The visual and audio materials were collected on-site Essen and mostly in Kettwig, and the materials are processed and manipulated real-time indirectly and anonymously by the users of the webpage¹ while they explore the content.

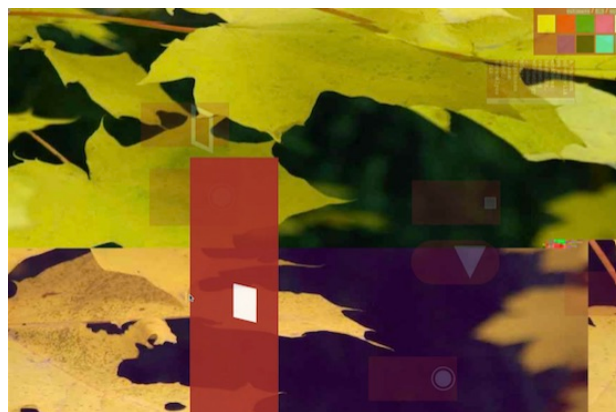


Figure 1. User interface of the Kettwig Webcards.

2.2 Pattogtattató and Forgattató (2018)

Inspired by Plink [6] and similar web-based cooperative audio games, I focused on the field between cooperative instruments and games for the Web. Since 2016 I'm working with interactive web projects. I present two of the latests. Both of them use browser-native Webaudio API for audio, CSS and Javascript for animation, and they experiment with physics, degradation and data-loss in the virtual space.

The first one is the **Pattogtattató**² (bouncerer). The user unleashes new balls by clicking somewhere, and they bounce by the amount of grativation etc., until they fade. Meanwhile they generate sounds, depending their position, speed, fade-position.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹ <http://kbalazs.periskopradio.hu/works/webcards/>

² <http://kbalazs.periskopradio.hu/egyb/pattogtattato.html>

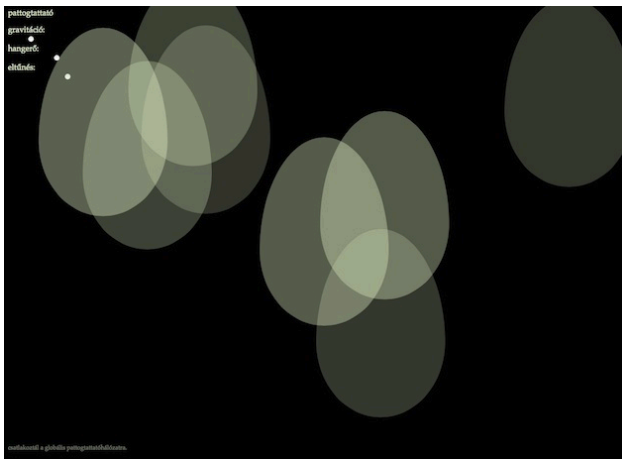


Figure 2. User interface of the Pattogattató.

The second one is coming from the first, but it simulates rotation, with several upgrades. It's called **Forgattató**³ (rotater), where You can rotate forms, making sounds. The fade-out depends on You also: it's possible to restart the rotation by hovering the objects. The sound is also depending by the position and other factors, resulting a rich toned fm sound.

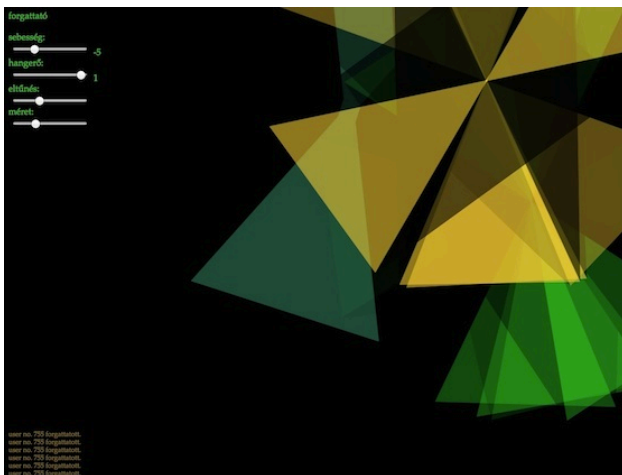


Figure 3. Multiple users on Forgattató.

Forgattató is natively multi-user: transparently connects to the socket.io network, sharing informations among other users. The rotating objects with different colors show different users' activity.

2.3 Websynth and his Webseq (2018)

While looking for network-specific audio instruments instead of adopting an existing synth for a website, I decided to split sound creation among different users, to reach the same audio output on different locations. Websynth⁴ and the sequencer created for it is a first experiment in this field. Any user can join in, and able to control one common synthesizer. They hear the same sound, and see one shared interface. Controllable parameters: freq + length; modFreq + ratio; modDep + ratio. User-specific ratio can be

³ <http://kbalazs.periszkopradio.hu/egyebe/forgattato.html>

⁴ <http://kbalazs.periszkopradio.hu/egyebe/websynth/>

adjusted by clicking on the top left corner. The two-dimensional sequencer works the same way, extending the process with an eroding option, where the data is going to be quiet and quick step-by-step. Both of them are mobile-friendly applications.

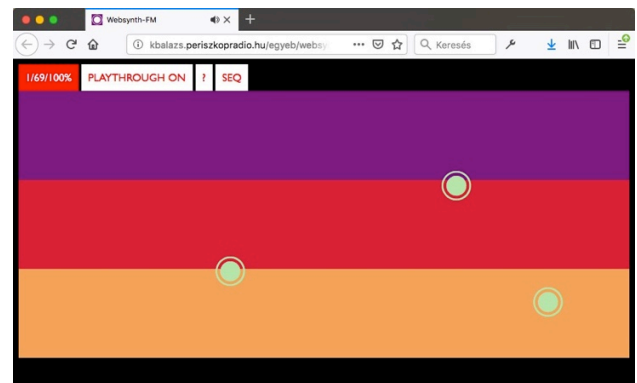


Figure 4. The Interface of WebSynth.

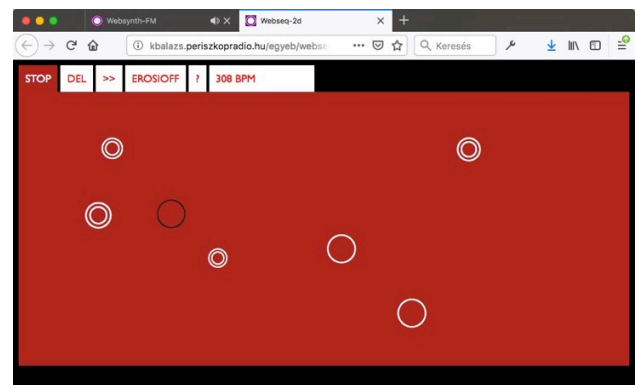


Figure 5. The Interface of WebSynth's sequencer.

2.4 Cooperation in Makkeróni live coding system (2018-)

Makkeróni⁵ is a web-based live coding system, presented and described in detail previously [7]. It's inspired by web-based live coding application Gibber [8], but it simulates a linux shell and their shortcuts for efficient use. Makkeróni use full range of the Web Audio API: waveform and FM synthesizers, sampling, low-level effects, extended by command argument parsing, batch running etc., as possible in BASH. The application is natively collaborative, with its "connect", "disconnect" and "wall" commands (see "help" or "wall --help" for detailed documentation). "Wall" command makes it possible to send realtime messages to other connected users, and it's possible as well to run commands on different users' host with the -c argument. For example the "wall -c play" command plays a randomly chosen soundfile on every connected users' computer, except the local computer. It's easy to distribute the playback on multiple connected devices targeting the information with the "-min" and "-max" arguments.

⁵ <http://makker.hu/makkeroni/>

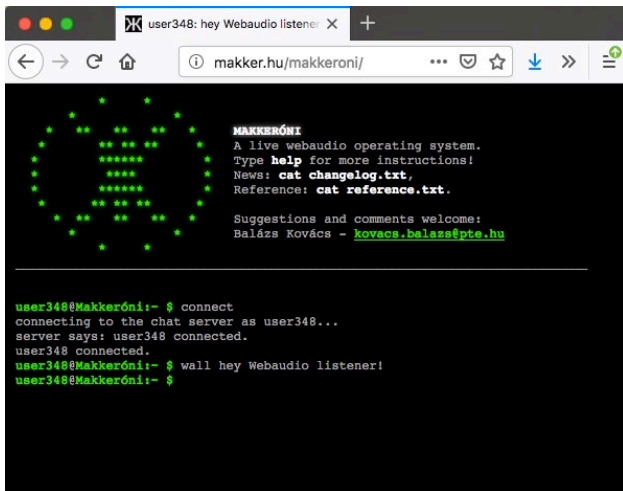


Figure 6. Makkeróni connecting to the chat server.

3. DISTRIBUTED AUDIO EXPERIMENTS

Peer-to-peer network control of audio events makes it easily possible to share audio playback on different (mobile) devices, creating a special acoustic environment and let the audience be a creative part of the process as well. Andrey Bundin's Concert for Smartphones utilizes the smartphones of the audience as speaker of the whole piece, addressing mass or unique devices for playback the piece 'conducted' by the author.⁶ SoundSling [9] implements audio distribution in an installation, where the different devices makes possible spatial movements of the sound, without any loudspeaker array used by ambisonic or wave front synthesis solutions. I realize my distributed works in a different way: they are participatory and 'democratic' meaning that every participant has the same role in creating audio events for other users, because there is hierarchy among participants, and there is no pre-existing work to perform. All of these example built for and tested on mobile devices.

Számháború (war of numbers)⁷ is a first example of them: a network edition of a real-world game, where the participants should say the enemies' numbers in order to exclude them from the combat. In this realization the participants' numbers should be clicked to disable them, while it's easy for everybody to re-enable him-/herself by clicking the lost number's place.



Figure 7. Screenshot and topology of Számháború.

Similar technology but cooperative approach is used in the **Pingpong** network dataflow game.⁸ Every connected participant of the network is able to start a data packet, which jumps around on every client; every clients's device can stop or forward it, and also start a new stream of data. The result is a slow erosion or development of high-pitched sounds' cloud.

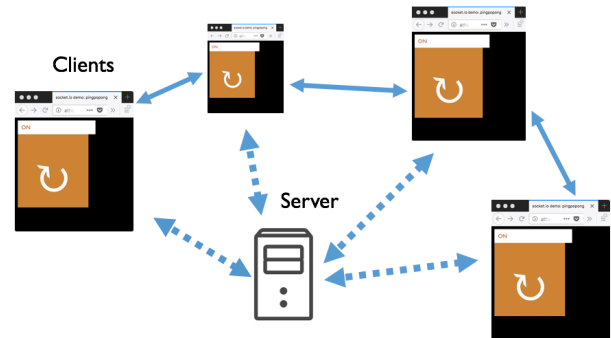


Figure 8. Topology of Pingpong.

As a last experiment with distributed audio, **Netpan** has been created for directly controlling sound playback on separate devices. Its interface is very basic: a number which shows a position on a virtual axis, and a slider what controls playback centered for a target position. For example, if the user has number 5 as position, and the slider moves from 5 to 1, it can listen a sound 'moving' from that device to devices with lower numbers. The closer the numbers, the highest amplitude the devices are playing back. It can be utilized to create a choir of sounds, controlled by every participants.

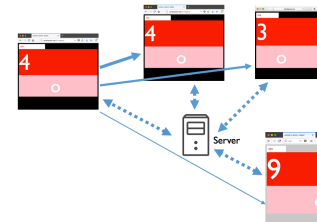


Figure 9. Topology of Netpan.

4. LOCATIVE TO WEB COMMUNICATION

Since 2011 I work on anonymous, one-way, web-to-real world communication projects (web-controlled hardware and light organ etc). In 2018 I turned back the direction of the communication, and created the Ring the Web instrument, a publically placed real ringbell, which can be used to ring a virtual place, usually a website. The idea behind it is that the virtual space is a delocalized interpretation of reality. While we're full of IoT (internet-controlled real) objects, there's a need for Really Controllable Virtual Entities, or: Reality of Codes. Therefore came Ring the Web project real. Ring the Bell is a public art (ever in real and virtual means of public space) project is a place for listen to existing web objects ('bells') placed somewhere in the world. They can interact with this or with any other websites. In

⁶ <https://www.youtube.com/watch?v=jtf4I1yB46c>

⁷ <http://mumia.art.pt/pt/webaudio/athallasok/4/>

⁸ <http://mumia.art.pt/pt/webaudio/athallasok/3/>

this case, the test object changes the style of this page while give a sound as well.



Figure 10. The bell of the Ring the Web! project, using an RPi Zero.

The project is stopped now due to closing the hosting institution Makker, but the documentary can be seen on the Ring the Web page.⁹ And also there are temporary installations, for example on the "DAN u Radničkom domu" exhibition in Novi Sad, where the ring manipulated the website of the whole festival,¹⁰ and it was extended with an independent web→real-world gate for controlling LED lights on place.

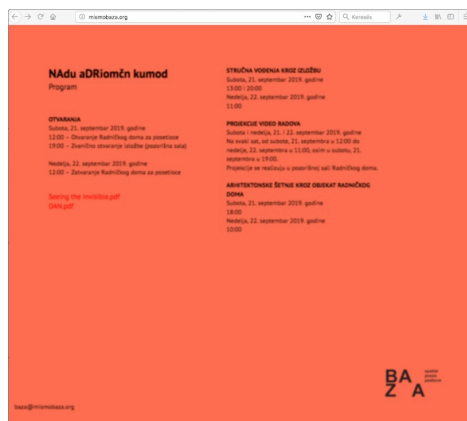


Figure 11. <http://mismobaza.org/> manipulated by Ring the Web!

While looking for cooperative project, I found the way to the real-world to virtual world communication projects more and more important. They express my opinion: the virtual place is the current public space.

5. DISCUSSION

Networked instruments (even audio-visual projects or simulated ones) open up the way for thinking about performing in a cooperative and dislocative fashion. Cooperation can be realized

⁹ <http://makker.hu/RingTheWeb/>

¹⁰ Mismobaza.org manipulation's graphical effect has been realized by Ivan Mesaroš.

between real and virtual places as well, extending webaudio into the realms of web-mediated acoustic or kinetic events. We're on the way to extend the definition of interaction towards a location-free peer-to-peer meaning. Here we have new, open questions:

1/ How we can differentiate audio instruments, games and artworks?

2/ What to do with anonymity among participants? Is it necessary to have a nickname/number/real name/avatar instead of other informations that in the world of the application are relevant (eg. physical distance from each other)?

3/ Could we interpret distributed sessions as a network-connected locative performance which has no sense for users out of that place? How we could implement non-local participants' contribution?

4/ In one-way Web→real world interactions, how we could deal with the lack of feed-back? Why users prefer to *not* have a feed-back about their activity?

While looking for answers to these questions, we'll found inspirations for new artworks upon new constellations of the above.

6. ACKNOWLEDGMENTS

Thanks to my university for supporting the travel to present this paper. And also thanks to my wife for letting Santa Nikolaus in at home during my presentation in Trondheim. And to Máté Labus for hosting me there.

7. REFERENCES

- [1] <https://socket.io/>
- [2] See more examples on telepresence at Wilson, Stephen. 2002. Information Arts - Intersections of Art, Science, and Technology, Cambridge: The MIT Press. Section 6.3. Teleconferencing, Videoconferencing, Satellites, the Internet, and Telepresence.
- [3] Knotts, Shelley, Collins, Nick. 2014. The Politics of Laptop Ensembles: A Survey of 160 Laptop Ensembles and their Organisational Structures. Proceedings of NIME'14, Goldsmith, London.
- [4] Stolfi, Ariane, Ceriani, Miguel, Milo, Alessia, Barthe, Mathieu. 2018. Participatory musical improvisations with Playsound.space. WAC-2018, Berlin.
- [5] Sarwate, Avneesh, Tsuchiya, Takahiko, Freeman, Jason. 2018. Collaborative Coding with Music: Two Case Studies with EarSketch. WAC-2018, Berlin.
- [6] <http://labs.dinahmoe.com/plink/>
- [7] Kovács, Balázs. 2019. Introducing Makkeróni, a web-based audio operating system. International Conference on Live Coding, Madrid.
- [8] Roberts, Charles, Wakefield, Graham, Wright, Matthew. The Web Browser As Synthesizer And Interface. 2013. NIME'13, KAI ST, Daejeon, Korea.
- [9] Marasco, Anthony T., Allison, Jesse. SoundSling: A Framework for Using Creative Motion Dada to Pan Audio Across a Mobile Device Speaker Array. WAC-2018, Berlin.



Demos

Tweakable

Julian Woodward
Visual Systems Ltd
42 Veda road London
jw@vsys.co.uk

ABSTRACT

In this paper, I describe some of the core features of *Tweakable*, a new interactive algorithmic music system. Live examples can be found on the web at <https://tweakable.org/>.

1. INTRODUCTION

Tweakable is a web-based visual programming system (VPS) designed to lower the barrier of entry for creating algorithmic music, while still offering vast possibilities for experimentation.

From a palette of components, users can quickly design an algorithmic system, and expose parameters through a user interface that enable the algorithm to be 'tweaked' in real time.

2. INCLUSIVITY

A key goal of *Tweakable* is put into anyone's hands the tools to experiment, play, discover and share ideas about how mathematics, sound and music can interrelate, acting as a catalyst for learning. Web-based, it is easy for users to share projects without worrying about missing dependencies. Much of the software already available for making algorithmic music requires specialist knowledge and sometimes complex software / hardware setup. Hugely successful, *Max MSP*[1] "provides an open and experimental environment for artistic expression, (however) it also restricts access to many users through a steep learning curve and requirement for low-level DSP knowledge". [2]

As a painter/programmer I aim to enhance engagement and user experience by infusing the user interface with a strong visual quality.

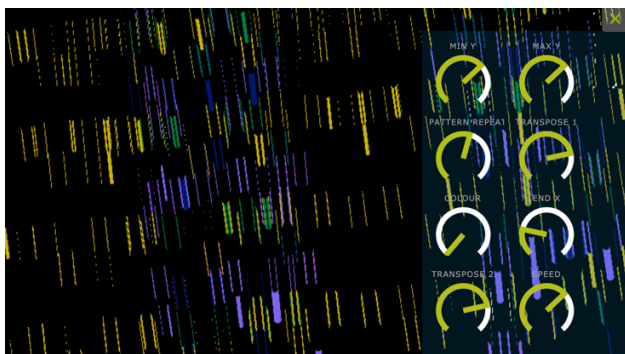


Figure 1. Visualization example - Tweakable Musical Tapestry

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.
© 2019 Copyright held by the owner/author(s).

3. COMPONENTS

Available components fall into three categories: Data input / flow, Sequencing, and Audio. In addition, it is possible to create custom modules to encapsulate and reuse behavior.

3.1 Data input / flow

Handling user input and controlling the way information flows through the system.

3.1.1 Data input

Control input components (Slider, Knob etc) enable user interaction and can be connected to parameters to enable the algorithm to be tweaked.

Some components automatically detect touch, enabling the x and y coordinates of the touch to be wired to parameters.

Midi input & output allows midi to be received from and sent to other applications.

3.1.2 Data flow

Data flow components include state machine, switch, counter, splitter, and filter, providing ways to control the way data flows through the system. State Machine can be connected to multiple components' parameters to create cross-component presets which can be selected or performed as a set by transitioning between each setting at specified intervals.

3.2 Sequencing

3.2.1 Sequence source

A sequence is a list of instructions that can be scheduled to generate events or control automation.

Graph components generate sequences from a mathematical function, or instead receive user input from touch or mouse.

Grid components generate sequences from user interaction with a piano-roll style grid.

Script components generate sequences using a custom notation language. Letters of the alphabet correspond to notes of the scale, lower / uppercase changes corresponding to lowering / raising the pitch / octave.

Rhythm components generate sequences from note / step parameters using the Euclidean algorithm [3].

3.2.2 Transformation, modulation, composition

Sequences can be passed through various transformations: reverse, invert, crop, transpose, scale, stretch, constrain etc. which can be applied conditionally or periodically, parameters changing in real time. Users can also code their own transformations in JavaScript.

Complex patterns can be generated by modulating one sequence with another. Sequences can be combined in series using letter notation (for example ABCBA describes a Rondo form). A

sequence can be exploded by the Harmonize component, which generates multiple contrapuntal parts taking a chord progression as input.

3.3 Audio

Sequences are rendered into audible sound by connecting them to a scheduler, and from the scheduler to a pre-built instrument (sampler or synth), or combinations of lower-level components such as oscillators, envelopes and LFO's, whose parameters can also be automated using sequences. Audio effects are implemented as objects that can be connected to audio outputs.

3.3.1 Automation

Individual parameters of audio components can be automated by sequences either by connecting the scheduled sequence directly to the parameter input, or by using Controller components which listen for events on specified controller numbers on the instrument.

3.4 Custom modules

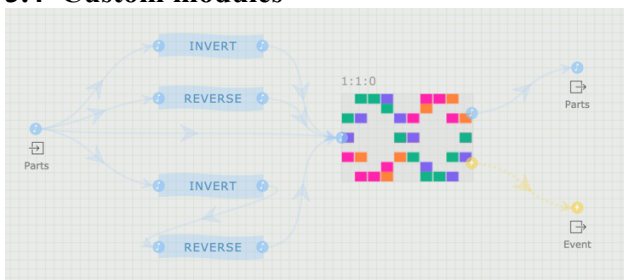


Figure 2. An example of a module that takes a sequence as input and outputs four sequences: the original, inverse, reverse, and inverse-reverse

Behavior can be encapsulated by creating modules, which act as containers for other components and can expose input and output parameters. See Figure 2 for an example of a module.

Custom modules can be exported and imported into other projects, and shared between users.

4. APPLICATION ARCHITECTURE

A key design concern was to ensure a clean separation between the User Interface ('UI' / 'front end') and the conceptual 'back end' functionality so that music and sound can continue to play when switching between views, and to minimize dependency on the front-end framework. Maintaining independence from the UI, audio output can be prioritized where necessary, and animation disabled where it becomes a problem for low performance devices.

Components are implemented as *TypeScript* classes that talk to each other using the Reactive Extensions for JavaScript library (*RxJS*). Subscriptions are created between them when connections are made. Component classes are data-bound to a corresponding UI component (implemented with *Angular*).

4.1 Aesthetic choices

The UI has been designed to minimize clutter and be as clear as possible. In the node-graph view, components each have a strong visual identity which hopefully helps with the understanding of their function.

A built-in palette system enables users to change the overall color palette. Rather than choosing colors separately for each element,

palette indices are assigned, helping to ensure an overall visual consistency by limiting the palette of colors in use.

4.2 Mobile devices

Users can design their UI to be responsive by defining both narrow screen and wide screen views. *Tweakable* automatically selects the version which matches the end-user's device width. All components work with touch devices.

4.3 Visualization

Visualization is employed to aid user understanding of the patterns controlling the audio output. Each UI component implements an 'animate' function. Rather than running multiple `requestAnimationFrame` (RAF) loops, a single RAF loop runs, calling each component's 'animate' function in turn. Visualizations use HTML5 canvas. Audio visualizations are provided for waveform 'oscilloscope' and audio envelope & partials. Sequences are visualized by the Piano Roll component. Custom visualizations can be coded in JavaScript.

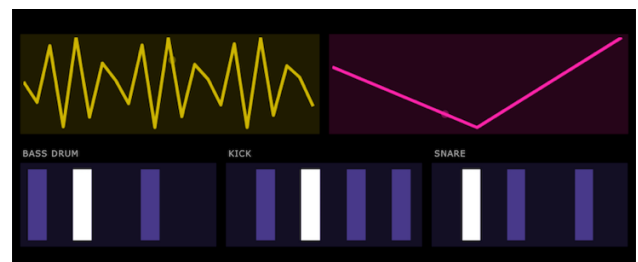


Figure 3. Tweakable algorithmic music example.

Melody and bassline and can be controlled by touch-dragging up/down (pitch) and left/right (shape), rhythm can be modified and rotated by dragging horizontally / vertically

5. ACKNOWLEDGMENTS

Acknowledgements are due to many open source software libraries, notably:

- RxJS (<https://rxjs.dev>) handles all data flow between components
- Tone.js (<https://tonejs.github.io/>) for audio instruments and effects (see 3.3), as well as envelope, filter and LFO are implemented as components
- jsPlumb (<https://github.com/jsplumb/jsplumb>) - the ability to drag connections between nodes in the 'node graph' GUI
- Codemirror (<https://codemirror.net/>) - a coding window enabling live coding in JavaScript or the in-built custom music notation language
- Angular (<https://angular.io/>) - the front-end framework that provides data-binding capability

6. REFERENCES

- [1] Cycling '74. *Max MSP*. [Software] Cycling '74. Available from: <https://cycling74.com>. 2019
- [2] Bullock, Jamie. 2018. *Designing interfaces for musical algorithms*. The Oxford Handbook of Algorithmic Music, The Oxford University Press, 2018
- [3] Toussaint, Godfried. *The Euclidean Algorithm Generates Traditional Musical Rhythms*. School of Computer Science, McGill University Montreal

Sounds Aware

Tate Carson

1. DESCRIPTION

Sounds Aware is a web application that runs on a smartphone and uses machine learning to detect human-made sound (anthrophony) and masks it with ambient music as a user walks around their environment. A study was completed to determine if this app is an effective means of shifting a user's attention away from anthrophony and to biological (biophony) and geophysical (geophony) sounds while walking and encouraging environmental awareness. Though the model is pre-trained with the author's local environmental sounds, the user can train the model further on their unique soundscape so that each user gets a personalized experience. After the training process, the user can listen to ambient music based on traits of the surrounding anthrophony. If the app senses less anthrophony and more biophony or geophony, then the music fades away, bringing the user's attention to the anthrophony.

The goal of *Sounds Aware* is to bring the user's awareness to the geophonic and biophonic soundscape, which is often so masked by noise pollution that it has fallen out of awareness for many of us. *Sounds Aware* seeks to shift the user's concept of nature to something that has no starting or ending point; it is all around us. The app brings awareness by focusing attention on the environment. Because of the predominance of eye culture [2], our reliance on seeing rather than listening as a primary means of sensing the world, it is a lot to ask of a person who might be uninterested in acoustic ecology to "just listen" to their environment. But, if you give them a tool that urges listening in the quieter places, where the natural world will be more audible, there is a better chance of them engaging with those sounds because the app focused their perception. *Sounds Aware* is a means of technologically mediated "ear cleaning," as described by R. Murray Schafer in *Ear Cleaning: Notes for an Experimental Music Course* [8].

A 2011 World Health Organization (WHO) report found that "there is overwhelming evidence that exposure to environmental noise has adverse effects on the health of the

population [6]." *Sounds Aware* shifts a user's attention away from noise pollution and to nature, which may help mitigate adverse health effects caused by noise pollution. Psychologist Stephen Kaplan found that stress reduction can be aided by the experience of the natural environment by providing a 'restorative environment' that reduces the fatigue caused by directed attention [4]. Kaplan did not mention sound directly, but a recent study by Eleanor Ratcliffe *et al* [7] has extended his research to show that certain bird sounds may provide restorative benefits. While a reduction in environmental noise at the source would be the best way to solve noise pollution, masking the noise is a stopgap solution. A masking solution has been implemented by several projects [5, 9] but not yet with a mobile device. *Sounds Aware* implements a similar idea but with a mobile phone.

Sounds Aware is accessed by going to <https://walking.netlify.com> in a web browser on a smartphone. It requires Internet access to download the default data set. After that, the app will work offline, so it is appropriate for various network conditions. Headphones are required so that the microphone on the phone does not pick up the music playback. Headphones with a microphone are preferred so that if the user wants, they can put their phone in their pocket while walking.

When a user first opens the application, they will see it guess the surrounding sounds based on a pre-trained data set. When assured that the microphone is working, the user can then start the music by clicking the toggle switch (see Figure 1a). The music now responds to what the surrounding sounds. The user can adjust the listening sensitivity of the microphone to their liking to match the acoustic environment if it is particularly quiet or noisy. After testing the success of the system in interpreting the user's environment, the user can now add their own training data (see Figure 1b). For this, the user will select a sound category such as a car. Then the user will wait for a car to drive by and then record it by clicking the record toggle. This will make the system more accurate in listening to the user's specific environment. Users are not currently able to add their own sound category tags.

The musical composition of *Sounds Aware* is influenced by ambient music. Synthesized sounds were used that would not be too jarring to jump in and out of and did not have an obvious beginning or ending. This type of synthesized sound blends in with the surrounding acoustic environment as a composition. The synthesized sounds are simple frequency



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

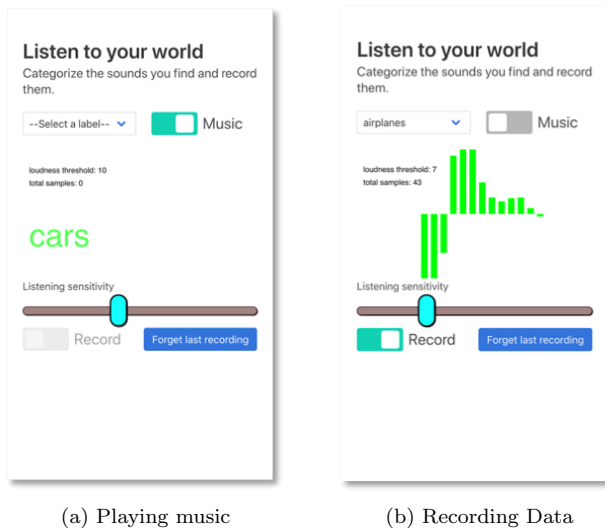


Figure 1: Two states of the application







Category	Effect	
Geophony 	amplitude mapped 	amplitude modindex harmonicity
Anthrophony 		amplitude -3
Biophony 		amplitude -60

Figure 2: Mapping of tag category to music

modulation synthesis with reverb and delay effects. They are tuned to just intervals so they are more likely to coincide with tunings in nature, influenced by Aeolian practices [1] and La Monte Young [3].

The app maps the loudness¹ of the acoustic environment to the amplitude of the ambient composition (see Figure 2). The previous 200 loudness values are averaged and the amplitude ramps to a given value over one second for signal smoothing. If *Sounds Aware* hears an anthrophonic sound, the amplitude is faded up to -3 dB. If it hears a geophonic sound, the amplitude of the geophonic sound is mapped onto the synthesized sounds amplitude, creating a wind chime effect. Geophonic sounds also affect the modulation index and harmonicity of the frequency modulation synthesis, creating a variety of timbres depending on the character of the current external soundscape. If a biophonic sound is recog-

¹A perceptual feature from <https://meyda.js.org/audio-features.html>

nized, the amplitude of the ambient wash is faded down to -59 dB, which is perceptibly silent when listened to in an urban environment.

2. LINK

<https://walking.netlify.com>

3. AUTHOR BIO

3.1 Tate Carson

Tate Carson is a composer from New Orleans, Louisiana. He studied jazz composition and performance at both Loyola University New Orleans and the University of New Orleans. Carson was active in the New Orleans jazz improvisation scene from 2009 until 2015 when he moved to Oakland, California to attend Mills College where he earned an MFA in Electronic Music and Recording Media. He is currently pursuing a PhD in Experimental Music and Digital Media at Louisiana State University. More information about his work can be found at <http://www.tatecarson.com>

4. REFERENCES

- [1] R. Bandt. Taming the wind: Aeolian sound practices in australasia. *Organised Sound*, 8(2).
- [2] J.-E. Berendt. *The Third Ear: On Listening to the World*. H. Holt, New York, 1992.
- [3] K. Gann. La monte young's the well-tuned piano. *Perspectives of New Music*, 31(1):134–162, 1993.
- [4] S. Kaplan. The restorative benefits of nature: toward an integrative framework. In: *Journal of Environmental Psychology*, 15:169–182, 1995.
- [5] G. Licitra, M. Cobiainchi, and L. Brusci. Artificial soundscape approach to noise pollution in urban areas. *Proceedings of the Proceedings of INTER-NOISE*, page 11, 2010.
- [6] W. H. Organization. *Burden of disease from environmental noise: quantification of healthy life years lost in Europe*. World Health Organization, 2011.
- [7] E. Ratcliffe, B. Gatersleben, and P. T. Sowden. Bird sounds and their contributions to perceived attention restoration and stress recovery. *Journal of Environmental Psychology*, 36:221–228, Dec 2013. 00146.
- [8] R. Schafer. *Ear Cleaning: Notes for an Experimental Music Course*. Clark & Cruickshank, 1969.
- [9] D. Steele, E. Bild, C. Tarlao, and C. Guastavino. Soundtracking the public space: Outcomes of the musikiosk soundscape intervention. *International journal of environmental research and public health*, 16(10), 2019.

Cassettes – An Online Audio Editor

Xingxing Yang
CCRMA
digdongaa@gmail.com

Jatin Chowdhury
CCRMA
Jatinchowdhury18@gmail.com

ABSTRACT

Cassettes is a project that started in 2018 summer. The motivation is to make an online audio editor in the browser, like an online version of Audacity. Through several months' development, now we can use it to record, edit, play and download audio. It is currently working well in Chrome.

1. INTRODUCTION

In this demo, we present Cassettes, which is an online audio editor. Compared to other online DAWs, it is mainly for audio editing. Users can easily adjust gain (fade in/out/buffer gain), drag and drop audio files cross tracks, cut buffers, glue buffers, adding audio effects, choose area to download, etc.

Its interface is as follows:

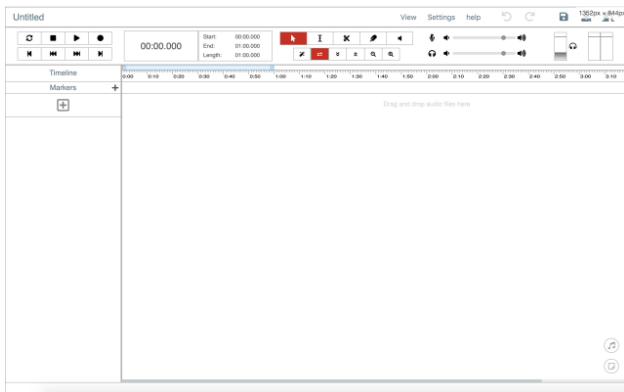


FIGURE 1: The Cassettes Interface

2. RELATED WORK

There have already been many online DAWs.

The open-sourced ones include [opendaw](#), [waveform-playlist](#), etc.

The commercial ones include Soundtrap, Bandlab, Soundation, etc. Most of them are mainly used for music production, while they may also be used for podcast/audio production.

The project is originally inspired by [waveform-playlist](#). The limitation of waveform-playlist is that the interactivity is not so mature.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

3. IMPLEMENTATION

3.1 Software design and architecture

The architecture mainly consists of 2 parts, the user interface and the audio engine. To make them communicate, some essential data is stored.

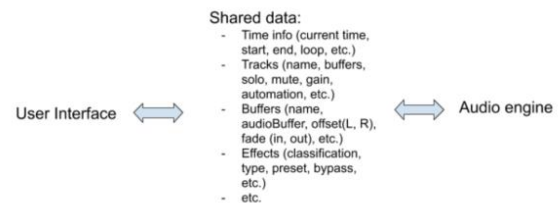


FIGURE 2: Architecture

The audio engine responds to user interaction by shared data. (We use React, so our shared data is mainly stored in a Redux store.)

3.2 Parts

The audio editor can be divided into several parts, including timeline, waveform, tracks, buffers, audio effects, audio libraries and download, etc. In the following sections, we will describe the implementation of waveform, audio effects and download.

3.2.1 Waveform

Previous work for browser-based audio waveform visualization includes [waves-ui](#) by IRCAM and [peaks.js](#) by bbc R&D. Our implementation of waveform visualization partially refers to their work.

For drawing the waveform, the goal is to display the waveform instantly as the audio buffer is ready and to respond quickly as the user zooms in and out while keeping the waveform looking comfortable to users. To achieve this, we used technologies including canvas, web workers and html img.

As users initially import the audio file, a canvas is drawn by extracting the max and min values for every finite length audio samples depending on the zoom level (e.g. 1024 samples for zoom level 3). In the meantime, a web worker is working to compute the canvas data and image for other zoom levels (11 is used in our case). For browsers supporting OffscreenCanvas (e.g. Chrome), img is used for further display as user zooms in and out, which is much faster than canvas when loading.

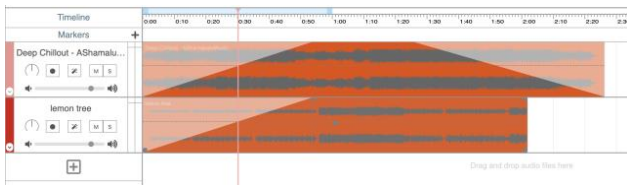


FIGURE 3: Waveform Overview

3.2.2 Audio Effect Plugins

In our design, each track has a port to link audio effect plugins. The master track also has a port to link audio effect plugins. This design is very common in desktop DAWs.

The main process of the audio engine for handling audio effects when the user hits play is as follows: creating effect => adding effect => setting effect params and bypass => setting effect automation. When the effect state changes during playing, the audio engine correspondingly responds.

The web audio API, and in particular the new Audio Worklet have made possible a whole world of web-based audio effects. Along with using built-in web audio effects, the low-level nature of the Audio Worklet have allowed us to fine-tune our own DSP algorithms to run in the browser, as well as providing outstanding performance even for the most complex of DSP algorithms.

Along with basic EQ, compressor, and channel strip effects that can be found in most DAWs, we have developed, a limiter, distortion, delay, chorus, flanger, and more. We also offer several reverb effects, including convolution and algorithmic reverb, plus a physical model of reverb inside a tube. Each parameter of these effects can be fully automated inside the audio editor.

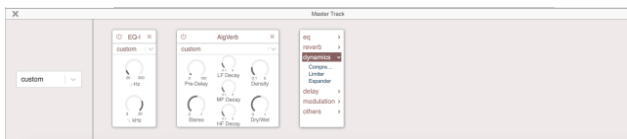


FIGURE 4: Audio Effect Plugins

Following is one of our plugins, which is written using web Audioworklet:



FIGURE 5: Algorithmic Reverb (Audio Effect Plugins)

For future advancements, we would like to continue expanding our effects library, adding effects including noise suppression, as well as potentially supporting third party plugins, allowing anyone to developing web audio effects to be able to use their effects inside our editor.

3.2.3 Download

In our design, users should choose the area they want to download first, then choose which format they want to use.

The download part is mainly achieved by OfflineAudioContext, which allows us to render the buffer quickly and finally to be available for encoding to other audio formats.

The audio encoding libraries we used include lamejs, libvorbis and fdkaac.

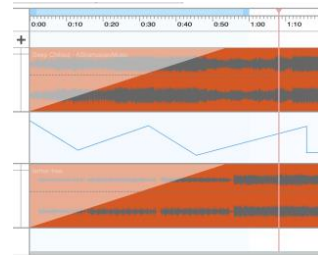


FIGURE 7: Choose The Area first



Figure 8: Click To Download

4. Future work

During our development, we mainly do test manually by listening to it. In the future, we'd like to add automated tests to save time.

We also want to add more features to it, such as video integration, third-party web audio plugins, text to speech generator, etc.

5. Links

The website can be accessible at <https://cassettes.herokuapp.com>.

6. REFERENCES

- [1] Naomiaro - Waveform Playlist:
<https://github.com/naomiaro/waveform-playlist>
- [2] Peaks.js
<https://waveform.prototyping.bbc.co.uk/>
- [3] Web Audio API:
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

Demo Proposal: Synth Kitchen

Spencer Rudnick
Ableton AG, DE
Schönhauser Allee 6-7
spencer@synth.kitchen

ABSTRACT

Synth Kitchen leverages the Web Audio API and Web MIDI API to bring interactive modular synthesis to the web. The goal of this project is to make modular synthesis cheap and accessible to anyone with a modern browser.

1. INTRODUCTION

Synth Kitchen began as a set of JavaScript tools for defining Web Audio Node graphs, and grew into a React-based UI for defining and manipulating said graphs. The addition of support for the Web MIDI API enables the use of Synth Kitchen alongside external hardware and software.

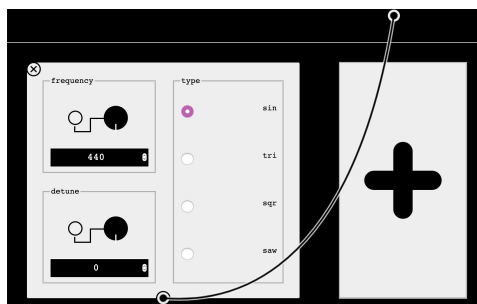
Synth Kitchen was created by Spencer Rudnick, and is supported with contributions from Olha Danylchenko for design and CSS, Stefan Osorio for React, and Ika Podola for Project Management.

2. PROJECT GOALS

The goal is twofold: to invite newcomers to explore and learn about synthesis, and to provide a free internet-accessible alternative to hardware and software.

2.1 Accessibility

Being based on the web platform, Synth Kitchen is intended to be usable by anyone who can use a keyboard or pointer device. Relying on HTML5 and JavaScript, we strive to create a meaningful accessible experience, making this project unique in its approach to the design of a software synthesis system.



2.2 Education

There are plans to add tutorials to the site. These will be geared toward absolute beginners in the synthesis world, and will cover various aspects of modular synthesis with the goal of empowering anyone to build their own patches.

2.3 Sharing

As a publicly available website, Synth Kitchen will eventually support the ability to save and share patches. All patches will be licensed under Creative Commons and will be free for anyone to use and change.

2.4 MIDI Integration

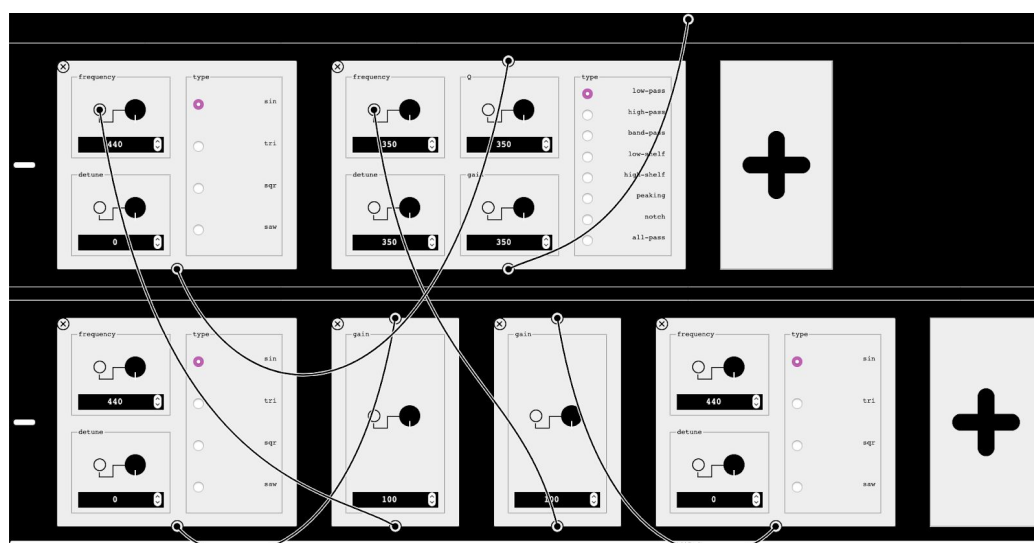
By supporting Web MIDI, Synth Kitchen integrates with any external MIDI sequencer. This allows Synth Kitchen to transcend the web platform and be used as a tool for musicians and artists.

3. TECHNICAL REQUIREMENTS

To properly demonstrate Synth Kitchen requires a laptop with headphones. A headphone splitter could enhance the experience by allowing pairs to learn together. Depending on the available space, a projector and speaker could be used to demonstrate more complicated patches.

4. MEDIA

Included are screen-shots of the prototype, a version of which is public at <https://synth.kitchen>. Also included are links to recordings made with the prototype: [synth-kitchen-beta-0](#), [synth-kitchen-beta-1](#), [synth-kitchen-beta-2](#).



moodplay.github.io: an online collaborative music player

Alo Allik, Florian Thalmann, Cornelia Metzger, Mark Sandler
Centre for Digital Music, Queen Mary University of London
{a.allik, f.thalmann, c.metzig, mark.sandler}@qmul.ac.uk

ABSTRACT

In this demo, we present an online music streaming system that allows users to collaboratively choose music by mood, but also personalise the experience by creating private parties where users can control who to invite to join the party and in which tracks can be shared with other participants. The music is automatically mixed by an auto-DJ module that models various DJ-ing styles using content-based audio features that represent musical parameters such as tempo, beats, bars, keys, instrumentation and volume.

DESCRIPTION

moodplay.github.io is an online music streaming platform that allows users to collaboratively choose music by mood using browsers on mobile devices. Users can participate in the global Moodplay party where everyone is added upon arrival, but they can also create personal parties and control who are invited. There is functionality that allows participants to share their favourite tracks with other invited participants. The system analyses the uploaded tracks by audio features to find their corresponding mood coordinates, so that the automatic DJ module can incorporate the new additions to the continuous mix. The interface is based on 2-dimensional scattering of tracks in a mood space derived from user tags in <https://www.last.fm/>. Each track has been assigned coordinates that range between negative and positive on the horizontal axis and calm to excited on the vertical from bottom to top. The users can explore the mood space and vote what kind of music they want to listen to while also being shown the preferences of other participants. Since each vote lasts for a limited time, the system keeps updating the average mood selection of all participants of the party. This means that the player cursor that keeps track of the average moves continuously around the space and a new track is selected after a certain time. The music is automatically mixed by an automatic DJ module that models various DJ-ing styles using content-based audio features that represent musical parameters such as tempo, beats, bars, keys, instrumentation and volume to find the best match for next tracks in the continuous mix by tempo,

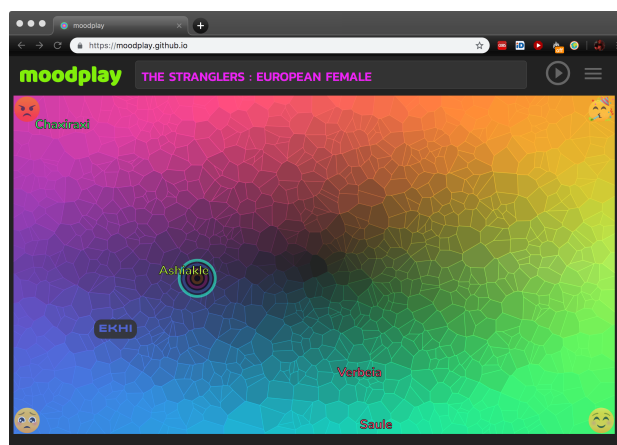


Figure 1: moodplay.github.io user interface

tonality and timbre.

The system consists of an Angular¹ front-end accessible at <https://moodplay.github.io> and Express.js² server application, <https://moodplay-data.herokuapp.com/> that stores track metadata, mood coordinates and audio features for the auto-dj³ module.

TECHNICAL REQUIREMENTS

moodplay.github.io can be demonstrated with the simplest of setups on a laptop, but can be enhanced, if facilities permit, by a large screen and an audio system (if it is not disruptive to other demonstrators). Alternatively an audio interface with multiple headphone outputs could be helpful.

ACKNOWLEDGMENTS

This work was supported by EPSRC Grant EP/ L019981/1, “Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption”. We would like to acknowledge the contribution of Mathieu Barthet and György Fazekas, who created the original installation version of Moodplay.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹<https://angular.io>

²<http://expressjs.com>

³<https://www.npmjs.com/package/auto-dj>

Develop WebAudio Plugins in a Web Browser

Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Fober
GRAME, 11 cours de Verdun LYON
renshihong@hotmail.com
(letz, orlarey, michon, fober)@grame.fr

Michel Buffa, ElMehdi Ammari, Jerome Lebrun
Université Côte d'Azur
CNRS, INRIA
(buffa, lebrun)@i3s.unice.fr,
ammarielmehdi@gmail.com

ABSTRACT

We propose to demo an online IDE based around the FAUST DSP audio language [1], that includes a source code editor, embedded compiler and GUI editor allowing to directly test, generate and deploy WebAudio Plugins (WAP). The tool is available online¹.

1. INTRODUCTION

When audio effects or audio/MIDI instruments have to be shared between several DAWs or audio environments, a plugin model is usually preferred.

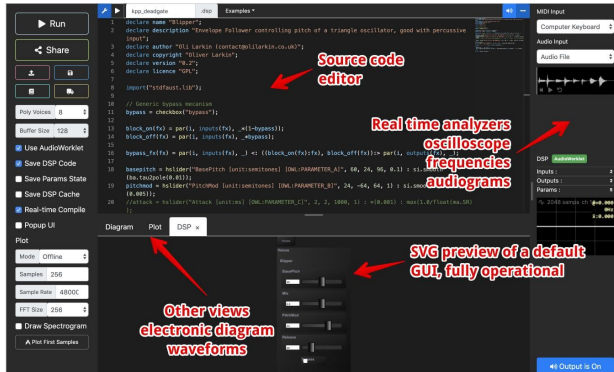


Figure 1: the FAUST IDE provides many embedded tools: oscilloscopes, spectroscope and spectrogram, functional default GUI, schema preview, etc.

Several native audio plugin formats are now popular, including Steinberg's VST format (Virtual Studio Technology, created in 1997 by Cubase creators), Apple's Audio Units format (Logic Audio, GarageBand), Avid's AAX format (ProTools creators) and the LV2 format from the Linux audio community. In the much newer WebAudio API (2011), there was no standard format for high-level audio plugins. With the emergence of Web-based audio software such as digital audio workstations (DAWs) developed by

companies such as SoundTrap, BandLab or AmpedStudio, it is desirable to have a standard to make WebAudio instruments and effects interoperable as plugins compatible with these DAWs and more generally with any compatible host software.

Such a plugin standard needs to be flexible enough to support these different approaches, including the use of a variety of programming languages. New features made possible by the very nature of the Web platform (e.g., plugins can be remote or local and identified by URIs) should also be available for plugins written in different ways. To this end, some initiatives have been proposed [2, 3] and with other groups of researchers and developers we proposed [4] a standard for WebAudio plugins called WAP (WebAudio Plugins), which includes an API specification, an SDK, online plugin validation tools, and a series of plugin examples written in JavaScript but also with other languages². These examples serve as proof of concept for developers and also illustrate the power of the Web platform: plugins can be discovered from remote repositories, dynamically uploaded to a host WebApp and instantiated, connected together etc. The reader can get a "multimedia" idea of this work by watching online videos that present the results of this work³. Since the last year, WAP now includes support for pure MIDI plugins (a GM midi synthesizer, virtual midi keyboards, a MIDI event monitoring plugin, etc). We propose to demo a new online IDE, that is well suited for coding, testing, publishing WAP plugins written in FAUST, directly in a Web browser. The IDE includes a GUI editor that allows developers to fine-tune the look and feel of the plugins. Once complete (DSP + GUI) the plugins are packaged in the form of standard W3C WebComponents and published on remote WAP plugin servers. The plugins will then be directly usable by any compatible host software, using their URIs.

2. THE ONLINE IDE

We embedded a WebAssembly version of the FAUST compiler in the IDE (Fig 1.) created by the Emscripten transpiler, to dynamically generate WebAudio nodes from FAUST DSP codes.

¹ <https://faust.grame.fr/ide/>, experimental version with the WAP GUI Builder available at <https://mainline.i3s.unice.fr/idewap>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

² <https://github.com/micbuffa/WebAudioPlugins>
³ <https://www.youtube.com/watch?v=pe8zg8O-BFs>

The node can be an *AudioWorkletProcessor* or a *ScriptProcessor* to be connected to audio I/O devices or other DSPs.

Based on this main feature, we built a code editor with full IDE user experience that could provide more information and details of a DSP through graphical representation in a Web page. A DSP developer probably not only needs to hear how the DSP sounds, but also to test it with other audio inputs, or to precisely display the time domain and frequency domain data of outputs. We have added several testing, visualisation and debugging tools into a basic code editor with following UI layout:

- All options related to FAUST code compilation are situated using controllers from the left sidebar panel, including a virtual file system manager that can be used by the compiler.
- All options and displays related to DSP runtime, such as MIDI, audio inputs, a recorder, and quick signal probing are placed in right sidebar panel.
- The remaining central region of the page is divided into two parts with configurable heights: a source code editor on the top and a multi-tab display panel which can display the logs from the compiler, a FAUST block diagram corresponding to the DSP code, a larger signal scope, a running GUI of the plugin being developed, and finally a GUI Builder / exporter for designing the user interface a WAP plugin version of the code, usable in external host applications.

In the signal scope panel, we designed four modes of signal visualisations: data table, oscilloscope (stacked and interleaved by channels), spectroscope and spectrogram to help FAUST users to debug their DSPs. After a FAUST DSP is tested in the editor, users can export the DSP to different architectures including WebAudio Plugins (WAPs). A dedicated GUI builder is integrated in the online IDE that receives FAUST DSP's GUI definitions while it is compiled (Fig. 2).

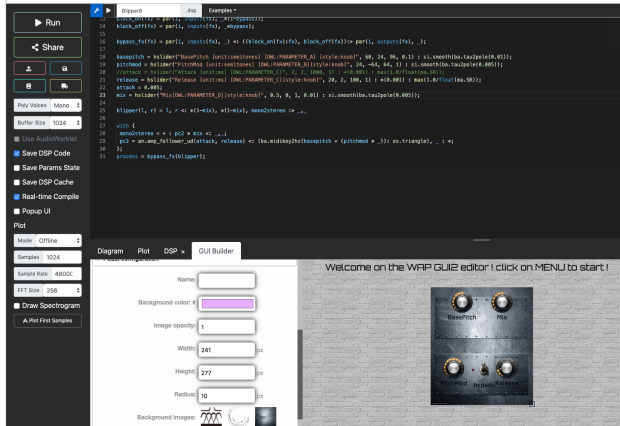


Figure 2: The default GUI can be customized: change textures, knobs, sliders, switches positions size, look and feel and labels etc.

At any time the plugin (DSP + GUI) can be tested from within the IDE, without the need to download it on a local disk. It is then possible to refine the GUI, adjust the layout, adjust the look and feel of the controllers among a rich set of knobs, sliders, switches (Fig. 4 shows different looks and feels that can be created from the same DSP code). At any time, the plugin can be published on

a remote plugin server, using standard Web services (Fig. 3). The plugins will then be directly usable by any compatible host software such as PedalBoard⁴.

3. ACKNOWLEDGMENTS

This work was supported by the French Research National Agency (ANR) and the WASABI team (contract ANR-16-CE23-0017-01).

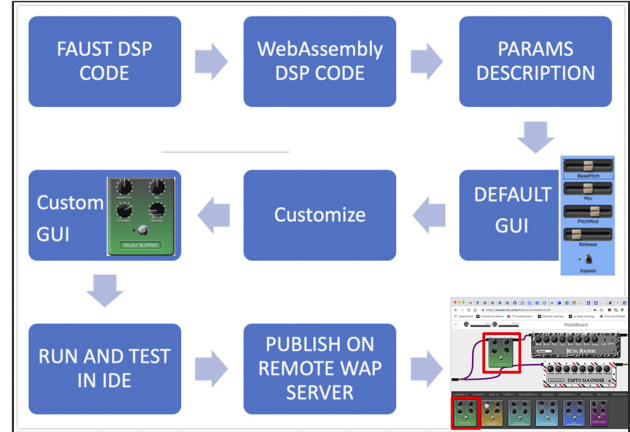


Figure 3: workflow of the end-to-end design and implementation of a WebAudio plugin

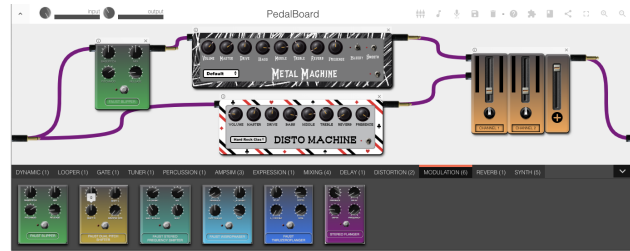


Figure 4: the virtual pedalboard host application scans multiple remote WAP plugin servers.

4. REFERENCES

- [1] Letz, S., Orlarey, Y., and Fober, D.. 2018. "Faust Domain Specific Audio DSP Language Compiler to WebAssembly". In *Companion Proc of the Web Conference, International World Wide Web Conferences Steering Committee, Lyon France 2018*.
- [2] Jillings N. and al. 2017. "Intelligent audio plug-in framework for the Web Audio API". In *Proc. 3rd Web Audio Conference (WAC 2017)*. London, UK.
- [3] Kleimola, J. and Larkin, O. 2015. "Web Audio modules". In *Proc. 12th Sound and Music Computing Conference (SMC 2015)*, Maynooth, Ireland.
- [4] Buffa, M., Lebrun, J., Kleimola, J., Larkin, O. and Letz, S. "Towards an open Web Audio plugin standard". In *Companion Proceedings (Developer's track) of the The Web Conference 2018 (WWW 2018)*, Mar 2018, Lyon, France. <hal-01721483>

⁴ <https://wasabi.i3s.unice.fr/dynamicPedalboard/>



Artworks

Generative.fm – Generative Music in the Browser

Alex Bainter
alex@alexbainter.com

ABSTRACT

Generative.fm is an open-source platform for playing generative music in the browser that features a collection of browser-based generative music systems. The Web Audio API is an essential component of this project.

1. LINK TO ARTWORK

The artwork is accessible at <https://generative.fm>, and the code is available at <https://github.com/generative-music/generative.fm>.

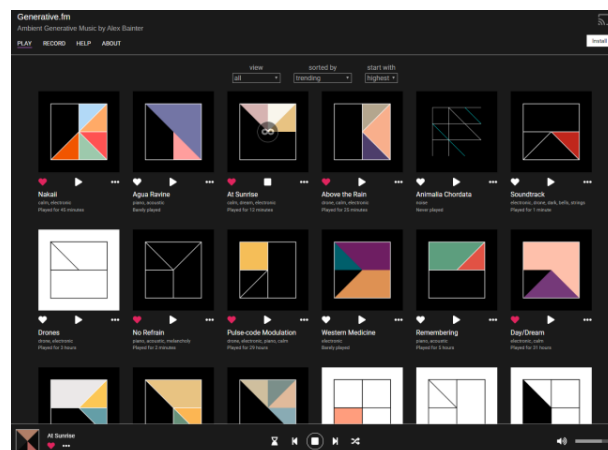
2. DESCRIPTION

Generative.fm is a platform for generative music in the browser. It currently features a variety of ambient music pieces that neither end nor repeat. Each performance is unique and lasts as long as a user chooses to listen. In addition, users can generate recordings of the music on the site, and these recordings are typically created faster than real time. Recordings from Generative.fm are licensed under a Creative Commons Attribution 4.0 License (CC BY 4.0).

Generative.fm uses the Web Audio API and Tone.js for all sound production and scheduling. A majority of the sounds are created by loading and manipulating audio sample files client-side. Generative.fm is also a progressive web app with offline capabilities, and supports installation on mobile and desktop devices.

“Generative music” is a term popularized by Brian Eno to describe music that changes continuously and is created by a system. Eno has experimented with generative music beginning as early as the 1970s through today with the albums *Discreet Music*, *Music for Airports*, *Neroli*, *Reflection*, and several others. For decades, Eno merely recorded the output of generative music systems to create these albums, and in a 1993 essay on the subject, he wrote how he wished he could “sell the system itself, so that a listener would know that the music was always unique.”

Today, the Web Audio API makes the production and distribution of generative music systems more feasible than ever. Generative.fm is a celebration of the accessibility of generative music systems finally afforded by the Web Audio API.



3. ABOUT THE AUTHOR

Alex Bainter is a web developer and musician from the United States who enjoys creating audio/visual experiences both digital and not. His other work is available at alexbainter.com.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Alex Bainter.

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Permutable

Julian Woodward
Visual Systems Ltd
42 Veda road London
jw@vsys.co.uk

ABSTRACT

In this paper, I describe the interactive musical artwork *Permutable*. This artwork was created with the algorithmic software system *Tweakable* by the same author and can be found on the web at <https://tweakable.org/jwvsys/permutable>.

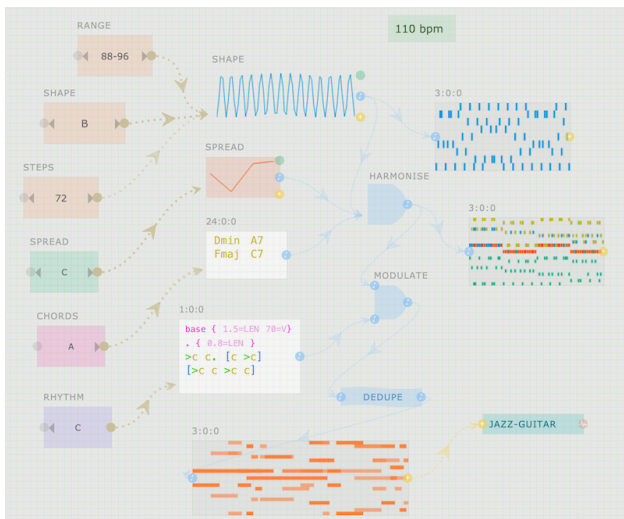


Figure 1. The components used in *Permutable*, as seen in the edit user interface of *Tweakable*

1. INTRODUCTION

As Leigh Landy discussed in *What's the Matter with Today's Experimental Music? Organized Sound Too Rarely Heard* [1], parametric approaches to making music stretch back to Cage, Stockhausen, Babbitt, Messiaen, and Xenakis, in their attempts to take serialism further, exploring how parameters such as dynamics, pitch, and rhythm could be split out and recombined.

Since then many artists (Robert Rowe, Clarence Barlow and David Cope et al) have further explored the possibilities of parametric music creation, details of which are beyond the scope of this paper. Software systems such as the astounding *Opusmodus*[2] specialize in providing parametric composition environments, but they are designed for the specialist user and are not accessible to the non-musician/programmer.

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.
© 2019 Copyright held by the owner/author(s).

2. INCLUSIVITY

Permutable aims to bring this approach to composition to the public in the form of an interactive musical artwork, enabling users to create sections of a composition by changing the parameters of a pre-defined structure.

Users of the artwork can make their own parameter selections to form a customized section of the composition. Once submitted, the system will examine the user's choices and musical relationships between existing sections, and append or insert the new section into the continually evolving composition while it plays in real time.

While the permutations are virtually limitless due to the number of parameters and values that can be set, the possible outputs are all constrained by the same underlying rules of the system. Because of this, each section will have some harmonic, melodic or rhythmic relationship with one or more of the other sections.

Permutable aims to bring a sense of exploration and improvisation to people who are not necessarily skilled musicians, generating curiosity amongst users of the artwork, to encourage an enquiring spirit to think and wonder about music and its ingredients.

3. PARAMETERS

The following parameters can be changed to customize a section of the composition.

3.1 Shape

The music generated by *Permutable* originates from a simple mathematical function. The shape parameter controls the selected function, options for which are variations of $\sin(x)$, and the start and end (x) values from which the output (y) value will be read to define the 'shape' of the initial melody.

3.1.1 Steps

The output melody is further defined by the number of (x) points sampled from the function, which is called the shape steps parameter.

3.1.2 Range

The shape range parameter governs the upper and lower limit of the output pitch of the melody

3.2 Harmonize

The Harmonize component has three inputs: shape, spread and chords. It outputs multiple parts that express a harmonized version of the input shape.

3.2.1 Spread

The spread parameter controls the density (the number of voices/parts) of the resulting harmony, and the upper and lower

pitch range of the parts. The starting note of each part of the harmony is spread across this range with slightly greater density in the upper register.

3.2.2 Chords

The chords input provides information about the harmonic progression, expressed in standard chord notation. The Harmonize component uses this information to constrain the parts generated by spreading the input shape over the spread pitch range to fit into the notes of the specified chord progression.

3.3 Modulate

The next step is to take the output from the Harmonize component and modulate it with a rhythm, as selected by the rhythm parameter.

The rhythm is supplied to *Tweakable* in script form, a custom notation which allows for some expression by defining accents (>) and note lengths (. and _). Square brackets delineate successive subdivisions of the overall defined phrase length (borrowed from *Tidal Cycles* [3] notation).

3.4 Dedupe

Finally, the output from Modulate passes through a process to remove repeated notes create sustains instead.

The dedupe process actively effects the rhythmical output of the music.

4. USER INTERACTION

As a web application, the piece can be experienced in a web browser, with either headphones or speakers connected to provide the audio output.

When installed in an exhibition, the visuals should be projected onto a screen and the audio output via speakers so that observers can experience the piece while someone is interacting with it and creating content.

To create content, users tweak the available settings to create a bank of presets, while the system plays the current section in a loop. When satisfied, the user can save their settings to the database, providing a name to identify them.

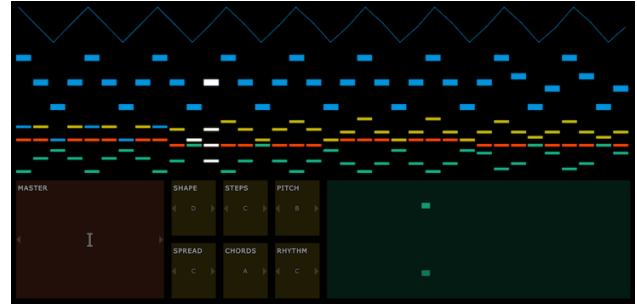


Figure 2. Part of the viewers' user interface to Permutable

The user's settings are then added to the continuously evolving musical composition which the system returns to playing, displaying the name of the section currently being played.

The artwork can be found on the web at <https://tweakable.org/jwvsys/permutable>.

5. REFERENCES

- [1] Landy, Leigh. *What's the Matter with Today's Experimental Music? Organized Sound Too Rarely Heard*. Taylor & Francis; 1991
- [2] Opusmodus *Opusmodus* [Software] Opusmodus Ltd. Available from: <https://opusmodus.com>; 2019
- [3] McLean, Alex and others. *Tidal Cycles*. [Software] Available from: <https://tidalcycles.org>; 2009

Wireplex, an Extended Flow of Data as Distributed Sound Sculpture

Luis Arandas
Faculdade de Engenharia e
Universidade do Porto
Braga Media Arts
luis.arandas@gmail.com

José Alberto Gomes
Portuguese Catholic University
CITAR School of Arts
Braga Media Arts
jagomes@porto.ucp.pt

Rui Penha
Escola Superior de Música e
Artes do Espectáculo
INESC-TEC Porto
ruipenha@esmae.ipp.pt

ABSTRACT

This article presents a networked sound sculpture made using web technologies. The intervention described explores the generation of content in large numbers of distributed interfaces with potentially large distances across the globe. Using Akson audio-visual (AV) environment, a distributed connection is maintained to all machines that remain publicly linked. We explore and present *Wireplex* as an artistic use of the Internet creating an irregular platform to reproduce the artwork.

1. SOUND AND SCULPTURE

Referring to Marshall McLuhan's (1964) quote that the human being has already extended his senses and nerves by the various media [7], we present *Wireplex*.

This artwork explores an automatic and pre-structured version of Akson AV environment, in order to generate and collect constant streams of data over the internet. Exploring detailed information related to the geolocation of the machine connected, it aims to create a complex structure of both resonant devices linked by the cloud. This text also reflects on digital networked systems as means of sound diffusion, reiterating some common practices in sound sculpture using contemporary technology.

Using the practice present in the twentieth century as a base reference for the development of this project, we start by mentioning the great change that existed in the practice of sculpture as art [2, 3, 5, 9]. As said by Rosalind E. Krauss (1979), surprising artworks have come to be called sculpture. "*narrow corridors with TV monitors at the ends; large photographs documenting country hikes; mirrors placed at strange angles in ordinary rooms; temporary lines cut into the floor of the desert*". As she says, nothing can be an empirical motive that allows anyone to call sculpture to anything. Unless, that is, the category can be made to become almost infinitely malleable [9].

When the practice includes sound as material for artistic expression, we underline two different positions that the author can take in the development of the artwork. The materialisation of sound into a three-dimensional physical

object as a sculpture derived from the analysis of the acoustic phenomenon, and the development of three-dimensional objects as sound producers in space [4, 6].

We have built this system in a way that respects both practices by creating analogies in the technical development of the platform and its reproduction between various devices. We acquire data from each connection and use it to produce sound, treating those devices as objects of sound reproduction in space.

2. REMOTE INTERACTION

Referring to the development and historical importance of networked collaboration using digital connections in the artistic practice [1], we start to define the system that supports the proposed artwork, *Wireplex*.

During the installation it will be possible to access the <http://www.akson.xyz> domain from anywhere in the world as long as it is done by a device with Internet. If the device allows, and can participate in the intended website will then be one of the public links that define the sound sculpture. This volatile feature regarding the change of shape of the artwork, allowing its existence in a distributed way will be a lever for the exploration of different playback setups.

As said before, this system is a generative exploration of the Akson AV environment. It has a purpose-built graphical system populated with geometries that allow interaction via click on the screen and an audio system that includes both a synthesizer and a low background. The way the audience interacts with their device is through the touchscreen, this way notes will be played on all connected devices.

The way both the notes that are played and the attributes of the two systems that make *Wireplex* are defined is by the geolocation of the device [8]. A two-dimensional matrix has been created that divides the position of the connections into predefined areas using latitudinal and longitudinal degrees. Static characteristics are thus assigned to the various points on the planet, making a machine in Japan have different sound and visual properties from California.

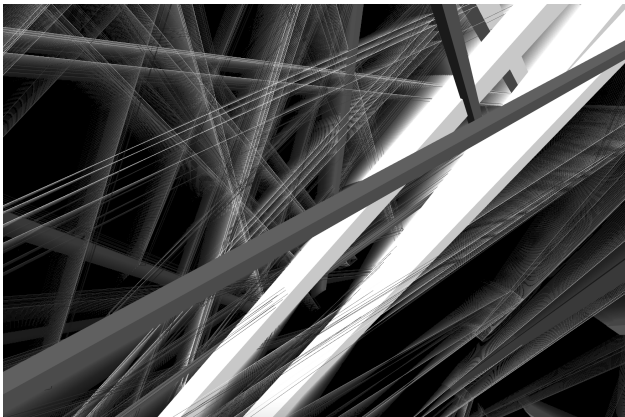
The development of this project is directly related to the aesthetic AV result and the intrinsic indeterminism present in the means of reproduction. In this way, a metaphor is made to the artistic phenomenon, as an event that exists only from the relationship established by the public or the author, in this case with a digital interface.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).



3. ACKNOWLEDGMENTS

The artwork *Wireplex* reflects and exposes part of an investigation conducted under Braga Media Arts¹ on collaborative AV environments and web technologies. All related contributions are in it's context, part of UNESCO's² Creative Cities Network (UCCN)³.

4. REFERENCES

- [1] A. Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. 2003.
- [2] F. Baschet and B. Baschet. Sound sculpture: sounds, shapes, public participation, education. *Leonardo*, vol. 20 no. 2, 1987.
- [3] J. Burnham. *Beyond Modern Sculture: the effects of science and technology on the sculpture of this century*. 1968.
- [4] J. V. Carvalho and L. G. Martins. Caixa de música - o espaço tecnológico e a arte pública. *Public Art in the Digital Creativity Era - International Conference Proceedings*, 2017.
- [5] B. Fontana. The relocation of ambient sound: Urban sound sculpture. *Leonardo*, vol. 20 no. 2, 1987.
- [6] A. Licht. Sound art: Origins, development and ambiguities. *Organised Sound*, Vol. 14, 2009.
- [7] A. McLuhan. Understanding media: The extensions of man. *MIT Press*, 1964.
- [8] A. P. Pejić, Bojan and Z. Čović. Uses of w3c's geolocation api. *11th International Symposium on Computational Intelligence and Informatics (CINTI)*. *IEEE*, 2010.
- [9] K. Rosalind. Sculpture in the expanded field. *October*, Vol. 8. (Spring, 1979), pp. 30-44. *MIT PRESS*. United States, 1979.

¹Braga Media Arts <http://www.bragamediaarts.com/pt/>

²UNESCO <https://en.unesco.org>

³UCCN <https://en.unesco.org/creative-cities/home>

5. BIOGRAPHIES

Luis Arandas

Media artist, and sound designer based in Porto. Currently a researcher at Braga Media Arts, Luis holds both an M.S. in Sound Design and Interactive Music by University of Porto and a B.A. in Sound and Image by Catholic University. He focuses his work on artistic performance, new media in art and musical technology. Luis has installed and/or performed in venues such as eINTERFACE, xCoAx, Orbits festival and Semibreve festival. More info at <http://www.luisarandas.org/>

José Alberto Gomes

José Alberto Gomes is a musician, sound artist and curator from Porto, Portugal. Graduate in Music Composition, he created strong bonds with new technological possibilities and in the role of music in theatre, film, installations and electronic improvisation, taking particular interest in seeking new ways and new musical 'places'. Has completed his PhD in Computer Music and he is a professor at Aberta University and School of Arts - UCP, teaching Digital Art, Sound and Computer Music. Since 2018, he is the educational project director of Braga Media Arts (UNESCO's UCCN). He performs regularly in public projects both in solo and group (BlacKoyote, Digitópia Collective, Srosh Ensemble, Hans-Joachim Roedelius, Jon Rose). He is creator in music and sound design for theater plays and videos (From Peter Handke's Essay about the Successful Day - FITEI/Rivoli, City Domingo - Oficina Theater, Prometeu - Theater of Animated Forms, Ínsua - Silent Rupture); creation and programming interactive sound installations; and composing for electronic and instrumental ensembles (Remix Ensemble, Drumming, João Dias, Henrique Portovedo, FactorE, Studio Orchestra). More info at <http://jasg.net>

Rui Penha

Composer, media artist, and performer of electroacoustic music, Rui Penha was born in Porto in 1981. He completed his PhD in Music (Composition) at the University of Aveiro. His music is regularly recorded and played in festivals and concert halls around Europe and North America, by musicians such as Arditti Quartet, Peter Evans, Remix Ensemble, or the Gulbenkian Orchestra. He was a founder and curator of Digitópia (Casa da Música, Porto) and has a deep interest on the relationship between music and its technology. His recent production includes interfaces for musical expression, sound spatialisation software, interactive installations, musical robots, autonomous improvisers, and educational software. More recently, Rui has focused his attention on the problems of defining and guiding artistic research. He taught at several Portuguese institutions, in both music, art and engineering faculties, and is currently an assistant professor at ESMAE and researcher at INESC TEC. More info at <http://ruipenha.pt>

Acoustic Atlas Artwork for WAC 2019

Cobi van Tonder
Artist/Creator
Potsdamer Str 100, 10785 Berlin
vantondc@tcd.ie

Guergana Tzatchkova
Programmer
Berlin
pestanias@yahoo.com

ABSTRACT

Acoustic Atlas aims to digitally preserve the acoustics and soundscapes of natural and cultural world heritage sites. The project's innovative approach will make the acoustical heritage of endangered heritage sites widely accessible. It will bring environmental, educational, conservation and artistic benefits by promoting and enriching heritage research and connecting international researchers and sound artists in the field of heritage acoustics. The online web-audio platform enables Acoustic Atlas to run on most computers or mobile devices and utilizes the built-in microphone and headphone output of a device to transport a visitor to the selected heritage site via headphones and live microphone feed. Any participant can thus interact with the acoustic simulation (termed auralization) of each site from a first-person point of hearing. The user can click on an image of a location, control various aspects of the sound whilst singing or projecting any sound in real-time, into a simulation of the selected site, to hear the reverberations, resonances and echoes thus experience it in a direct sensory way. Acoustic Atlas is suitable for presentation on a computer kiosk with headphones as long as there is a built-in or plug-in microphone available on the computer.

1. INTRODUCTION

Acoustic Atlas is a virtual acoustic map, for the cultivation of the capacity to listen to and connect with, remote heritage sites. Acoustic Atlas invites people to sing and emit sound into virtual acoustic environments and experience how their voices, as human sonar signals, reveal the hidden interiors, forms and textures of these heritage sites. Such listening experience allows for a phenomenological connection with the remote site, which becomes particularly relevant for the preservation of heritage sites and for sonic exploration. In the context of acoustical and environmental intangible heritage, virtual reconstructions of world heritage sites are becoming increasingly useful to allow for multi-sensory immersive access, research and conservation [1-3]. The sonic component of the virtual reconstruction of a site is termed 'auralisation' which means rendering a space audible [4]. Examples of uses of auralisations include simulations of ancient and historic sites, to determine the likelihood and nature of rituals and historical actions that may have happened in these sites [5][6], or to monitor how sites may have changed over time [7].



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author

To date, however, while there are vast amounts of digital visual data such as maps, photogrammetry, 3D models, Google street-view and photos of heritage sites, the number of available and accessible auralisations is sparse. Browse Google Earth to discover spectacular visual details anywhere on this planet. Try to listen to them and it is astonishingly quiet. Should climate change, corporate greed or unforeseen catastrophe alter or destroy important natural and cultural heritage sites, we can remember such sites with masses of already existing visual documentation, however, without any acoustic data, these same sites are under risk to be silenced forever.

Preservation is done by collecting room impulse responses (RIRs) of each location, to enable the creation of auralisations. Field recordings of environmental sounds belonging to each site are also incorporated into the immersive auditory experience. To collect the room impulse responses (RIR), a sine-sweep test signal is played through a loudspeaker and recorded through an Ambisonics microphone. This allows the researcher to capture the spatial characteristics of each location. For a full study of each space, the researcher will choose a set of source and receiver positions. The source positions are where the loudspeaker will be positioned and correspond to parts of the space in which natural or artificial sources are present. The receiver positions correspond to listener positions. A room impulse response is captured for every source-receiver combination, allowing for a full acoustic mapping of each space. These RIRs allow the researcher to derive the acoustic characteristics of the space through the analysis of acoustic parameters such as those determined by ISO 3382-1. To conduct auralisations, dry sound signals are convolved with the RIRs to make them sound as if they were speaking/singing in the selected site.

2. WEBAUDIO

The online Acoustic Atlas platform is created with open source web-audio technology [8], to allow for a browser-based audio application that renders real-time auralisations from a mic input. In parallel, the Acoustic Atlas collection aims to be an inclusive platform: researchers active in the field of recording acoustic measurements of cultural and natural heritage sites are invited to participate and upload their acoustic measurement (RIR) data including additional meta-tagging information to the platform, thus giving their work more 'audibility' whilst allowing for the number of sites available in Acoustic Atlas to expand.

Acoustic Atlas aims to work on any smart mobile device or computer running a web browser such as Chrome or Safari. It utilises the built-in microphone and headphone output of the device to transport a visitor to the selected heritage site via headphones and live microphone feed. The user can talk, sing, hum or project any sound in real-time, into a simulation to hear the reverberation,

resonances and echoes of each site and thus experience it in a direct sensory way.

The auralisation is realized with tone.convolver. The simplified flow of audio is as follows: microphone input – gain – microphone equalization – convolver – master mixer. When the user clicks on an image of a location, the corresponding RIR is loaded.

Further development of this project will include features such as parallel convolution streams that are cross faded at the master mixer to allow for multiple source-receiver combinations. This would create more engaging virtual acoustic movement. It adds another dynamic layer to the overall spatial realism – by crossfading from one source-receiver position to the next, the overall sound effect is more convincing. As a dynamic musical parameter, such movement can be accelerated, slowed down and used as an added textural quality. Additionally, it is also possible to select and listen to different sound layers of environmental sounds from each site, as the collection of field recordings submitted by the community grows this will be included. Music composed for sites will also be added for listening. Another desired feature will be the ability to record through the Acoustic Atlas website, allowing a user to save their own auralised sounds. This feature will enable participants to upload their recordings to share them or use their audio in whichever way they like, such as their own creative practice or even for the creation of educational resources on the use of caves. Various other interface controls will allow for smart EQ of incoming signal, giving the user the ability to choose to filter out for example, unwanted low frequencies from the mic input.

Acoustic Atlas takes inspiration from archival environmental sound projects with map interfaces [9-11] as well as acoustic heritage auralisation projects that are partly similar to Acoustic Atlas e.g. the Soundgate App [12], or the live auralisation concert of Cappella Romana at Stanford [13].

3. LINK TO WORK

The site under development can be viewed here:

<https://acousticatlas.de>

4. BIOGRAPHIES

Cobi van Tonder is a practice-led researcher and artist from South Africa/Germany interested in ways of achieving abstraction in music, installation, video and immersive contexts. She experiments with the reduction of phenomena in order to expose human sensuality. This has led her to work with drone, microtonal music, nature field recordings, mathematical patterns as musical material and spatial audio-physical elements of sound. The experience of sound in and as space with attention to artificial (or real) acoustics is a prominent part of the material and overall texture.

Guergana Tzatchkova is an IT Professional. Her interests in the creative uses of media and technology led her to complete a Master in Design of Multimedia and Interactive Systems and a Master in Communication in Barcelona. She has experience in projects combining video, design and web development as well as projects related to gender and politics in Chiapas (Mexico) and Mexico City. She is currently based in Berlin working as Front-End / Creative developer.

5. ACKNOWLEDGEMENTS

Thanks to Guergana Tzatchkova for user interface programming and overall implementation. Thanks to Carlo Cattano for his contributions to the sections of node.js implementation. Thanks to Dr. Lidia Alvarez for providing room impulse responses and Dr. Mariana Lopez for overall academic input.

6. REFERENCES

- [1] Bogdanovych, A., Rodriguez-Aguilar, J., Simoff, S., & Cohen, A. 2010. Authentic Interactive Reenactment of Cultural Heritage with 3D Virtual Worlds and Artificial Intelligence. *Applied Artificial Intelligence*, 24(6), 617–647.
- [2] Lercari, N. 2016. Simulating History in Virtual Worlds. In *Simulating History in Virtual Worlds*. Retrieved from https://doi.org/10.1007/978-3-319-22041-3_13.
- [3] Pujol, L., & Champion, E. 2012. Evaluating Presence in Cultural Heritage Projects. *International Journal of Heritage Studies*, 18(1), 83–102. Retrieved from <https://doi.org/10.1080/13527258.2011.577796>;
- [4] Kleiner, M., Dalenbäck, B.-I., and Svensson, P., “Auralization—An overview,” *J. Audio Eng. Soc.* 41, 861–874 (1993).
- [5] Abel, J. S., Rick, J. W., Huang, P. P., Kolar, M. A., Smith, J. O., & Chowning, J. M. 2008. On the Acoustics of the Underground Galleries of Ancient Chavín de Huántar, Peru. In *Proceedings of Acoustics '08*. Paris. Retrieved from <https://ccrma.stanford.edu/groups/chavin/publications/Acoustics08.pdf>;
- [6] Weinzierl, S., & Lepa, S. 2017. On the Epistemic Potential of Virtual Realities for the Historical Sciences. A Methodological Framework. In *Augmented Reality* (pp. 61–80). Berlin: De Gruyter.
- [7] Alvarez-Morales, Lidia., Cathedral Acoustics. <https://www.cathedralacoustics.com/>, accessed on 21 June 2019.
- [8] Adenot, P., & Choi, H. 2018. Web-Audio-API. Retrieved 2018 from <https://webaudio.github.io/web-audio-api/>. Accessed on 1 June 2019.
- [9] Krause, B. The Great Animal Orchestra. <http://www.legrandorchestredesanimaux.com>. Accessed on 28 July 2019; Anderson, M., Nature Sound Map. <http://www.naturesoundmap.com>. Accessed on 30 June 2019;
- [10] Aporee Radio. <https://aporee.org/maps>. Accessed on 14 July 2019.
- [11] Barclay, L., et.al. 2015. The Biosphere Soundscapes. <http://www.biospheresoundscapes.org>. Accessed on 14 July 2019;
- [12] Till, R., 2016. EMAP Interactive Soundgate, 2016, Digital or Visual Products.
- [13] Abel, J. S., & Werner, K., 2017. Live Auralization of Cappella Romana at the Bing Concert Hall, Stanford University. In Pentcheva, B. V., *Aural Architecture in Byzantium: Music, Acoustics, and Ritual* (pp. 198–223). Routledge.



Performances

Performing with QuaverSeries Live Coding Environment

Qichao Lan
RITMO
University of Oslo
qichao.lan@imv.uio.no

Çağrı Erdem
RITMO
University of Oslo
cagri.erdem@imv.uio.no

Alexander Refsum
Jensenius
RITMO
University of Oslo
a.r.jensenius@imv.uio.no

ABSTRACT

This performance will use QuaverSeries, a collaborative live coding environment. The three performers will collaborate on the website of QuaverSeries, writing programs to produce ambient techno music with its domain-specific language. The first two performers will perform at the conference scene in Trondheim, while the third performer will join the performance online in Oslo.

1. ENVIRONMENT INTRODUCTION

QuaverSeries is designed and developed by the first performer, under the supervision of the third performer. It consists of a domain-specific language and a single-page web application for collaborative live coding in music performances. Its domain-specific language borrows principles from both programming and digital interface design in its syntax rules, and hence adopts the paradigm of functional programming. The collaborative environment features the concept of ‘virtual rooms’, in which performers can collaborate from different locations, and the audience can watch the collaboration at the same time. Not only is the code synchronised among all the performers and online audience connected to the server, but the code executing command is also broadcast. This communication strategy, achieved by the integration of the language design and the environment design, provides a new form of interaction for web-based live coding performances.

2. TECHNICAL DETAILS

The three performers will use individual laptops, and enter the same ‘virtual room’ on the QuaverSeries website¹. Another laptop will be used to connect to the same room, serving as an ‘audience’. Its screen will be projected to the main screen on the scene. The code will be synchronised among these four laptops, together with the code executing actions.

There are two reasons to use the fourth laptop. First, the figure of the third performer in Oslo can be broadcast to the

¹<https://quaverseries.web.app>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

```
1 // for WAC 2019
2
3 -bd: loop 20 20 20 20 >> membrane >> amp 0.2
4
5 -drone: loop 30 _ _33 _ >> sawtooth
6 >> adrs 0.1 0.3 0 _
7 >> lpf -lfo_a 1 >> hpf 100 1
8 >> reverb 0.7 0.7
9 >> amp 0.04
10
11 -lfo_a: lfo 1 100 400
12
13 -hh: loop _1 _1 _1 >> white >> hpf 8000 1 >> amp 0.03
14
15 -texttone: loop 16
16 >> sawtooth
17 >> lpf -textlfo 1
18 >> adrs 0.9 0.7 0.5 0.8
19 >> reverb 0.7 0.7
20 >> amp 0.01
21
22 -textlfo: lfo 0.5 200 1200
23
24 -texttwo: loop 84 86 84 86, 84 86 84 86, 84 86 84 86, _ _ _ _ _86_87_68_90_50
25 >> pluck
26 >> lpf 1500 1
27 >> pingpong 2 5
28 >> amp 0.06
```

Figure 1: A screenshot of the code written in a rehearsal.

conference scene through a web camera, shown in a small window on the main screen. Second, having a laptop as an ‘audience’ can demonstrate our design of room hierarchy, i.e. how the performers can control the browser of an audience using web technologies².

The actions of the performers can be identified with the cursor sharing function implemented in the environment. The performers will each make a one-line self-introduction, to specify the cursor colours (see Figure 2). Also, the body movement of the first two performers can be watched at the scene, while the motion of the third performer will can be seen through the web camera.

Hence, the requirements of the performance include:

- Good network connection
- A main screen for projection
- Basic audio playback system with two speakers

3. RISK CONTROL

The risk of performance lies in the code errors. Each performer will have the browser console open on the side, to receive and handle any coming error message. We have carried out several rehearsals, and have formed a practice for

²<https://github.com/chaosprint/QuaverSeries>

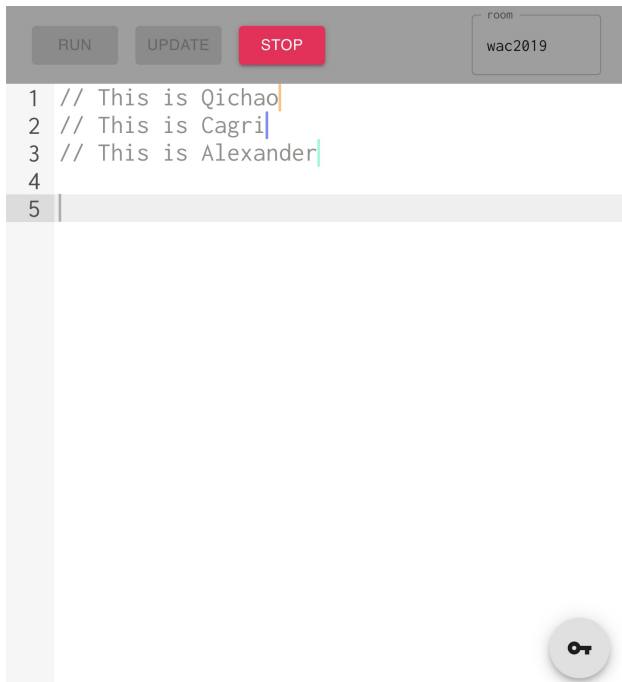


Figure 2: The performance interface. Entering as an audience, one cannot execute the *run* or *update* commands, but can watch new code appearing to generate new sound. The three performers will make one-line self-introduction and specify individual cursor colour.

performing collaboratively that each performer should comment out the code when editing it³. We will do more rehearsals to lower the risk of code errors to the minimum. Also, we are improving the code with `if` and `try-catch` statements, to ensure that if the errors should appear, the music can continue.

4. BIOGRAPHIES OF ALL PERFORMERS

• Qichao Lan

Qichao Lan is a researcher specialising music programming, with a background of computational linguistics, electroacoustic music composition and live coding. In 2018, he gained his Master’s degree in Sonic Arts at the University of Sheffield, during which he frequently participated in activities of the Algorave community. Now, as a doctoral research fellow at the RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion at the University of Oslo, he is researching on how new technologies and new forms of music interfaces can influence the embodied music interaction.

• Çağrı Erdem

Çağrı Erdem is a performer and researcher specializing in improvised electroacoustic music and body-machine instruments. Following a training in composition and guitar performance, he earned a Master’s degree in Sonic Arts, which

³One of the rehearsal videos can be found on YouTube: <https://youtu.be/1tMeSe8lShE>

focused on the development of a new musical interface for the extraction of body movements in the form of biosignals to be used to expand the player’s control space. As a PhD fellow at the RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion at the University of Oslo, he is expanding his research on the relationship between the dynamics of motion and sound in the context of both traditional and new musical instruments.

• Alexander Refsum Jensenius

Alexander Refsum Jensenius is a music researcher and research musician. His research focuses on why music makes us move, which he explores through empirical studies using different types of motion sensing technologies. He also uses the analytic knowledge and tools in the creation of new music, with both traditional and very untraditional instruments. As chair of the NIME steering committee, he is a leading figure in the international computer music community. From 2017 he co-directs RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion, an interdisciplinary centre of excellence at the University of Oslo. As a member of the Young Academy of Norway and the EUA Open Science Committee, he is also involved in pushing for modernizing the way research is conceived and conducted.

5. ACKNOWLEDGMENTS

This work was partially supported by the Research Council of Norway through its Centres of Excellence scheme, project number 262762 and by NordForsk’s Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

Three Pidgins: Live Coding Performance

Francisco Bernardo
Emute Lab, Department of Music
University of Sussex
f.bernardo@sussex.ac.uk

Chris Kiefer
Emute Lab, Department of Music
University of Sussex
c.kiefer@sussex.ac.uk

Thor Magnusson
Emute Lab, Department of Music
University of Sussex
t.magnusson@sussex.ac.uk

ABSTRACT

We present *Three Pidgins*, a live coding musical performance as an example of the live coding systems created with *Sema*, a web audio-based technology. Using three bespoke mini-languages, Kiefer, Bernardo and Magnusson will collaborate on a co-located networked musical performance where music is live coded in real-time. Each of the languages serves as an instrument in the ensemble of three.

1. PERFORMANCE PROPOSAL

Live coding practitioners typically engage simultaneously in programming with a domain specific language (DSL) and other modalities, including audio and visual synthesis, instrument design, algorithmic creation, composition and performance [1].

Three Pidgins is a 10-minute performance that combines elements of improvisation and machine agency. This performance brings together and connects three live coding languages developed with our new system, *Sema*, which we will present in the main track of Web Audio Conference 2019 [2]. *Sema* enables users to define their own mini-languages and perform with them in a modern Web-based environment.

For *Three Pidgins*, we have created three distinct mini-languages that serve as instruments of a musical ensemble. We explore the extent to which these mini-languages enable an expressive live coding performance. In this performance and in the live coding style, the code and screens will be projected onto the wall, enabling the audience to follow the performance as it is played with the pidgin languages.

Magnusson, Kiefer and Bernardo are experienced improvisers, who will be performing as a trio for the first time with this performance. Magnusson and Kiefer are veterans of the live coding and algorave movement and have both previously developed their own live coding performance systems. Bernardo is a multi-instrumentalist and improviser, experienced in different musical genres and audiovisual media languages.

2. DOCUMENTATION

Three Pidgins will be publicly performed for the first time at the Web Audio Conference 2019. As such, there is no documentation of this performance. However, there is video-documentation of the idiosyncratic systems and languages that the participants of our first MIMIC Artist Summer Workshop developed with *Sema*, and performed with in a live coding event in Brighton.

3. PERFORMER BIOGRAPHIES.

Thor Magnusson is a worker in rhythm, frequencies and intensities. His research interests include musical improvisation, new technologies for musical expression, live coding, musical notation and digital scores, artificial intelligence and computational creativity, programming education, and the philosophy of technology. These topics have come together in the *ixiQuarks*, *ixi lang*, and the *Threnoscope* live coding systems he has developed.



Chris Kiefer is a computer-musician and musical instrument designer, specialising in musician-computer interaction, physical computing, and machine learning. He performs with custom-made instruments including malleable interfaces, touch screen software, interactive sculptures and a modified self-resonating cello. Chris is an experienced live-coder, performing under the name 'Luuma'. He performs with Feedback Cell and Brain Dead Ensemble, and has released music with ChordPunch, Confront Recordings and Emute.



Francisco Bernardo is a computer scientist, an interactive media artist, and a multi-instrumentalist. His research is focused on human-computer interaction approaches to toolkits that broaden and accelerate user innovation with interactive machine learning. Francisco has been working in applied research in projects at the intersection of art and innovation, human-centred machine learning, front-end software engineering, interaction design and greenfield product management. In his artistic practice, Francisco has been performing with different acts (e.g. FRANTICØ, :papercutz), and with his most recent solo project, MNISTREL.



Live Code Performances with Sema at MIMIC Artist Summer Workshop (<http://www.emutelab.org/blog/summerworkshop>):
https://youtu.be/3PKRDkaL_Nk
<https://youtu.be/Wt63531sWP0>
<https://youtu.be/-vU9Ka0RiLk>

4. TECHNICAL REQUIREMENTS

- 3 stereo DIs or an onstage mixer with 6 channels
- Ideally 3 projectors and screens or walls to be projected on – the cables need to reach the stage. If this is not possible, we'll find a workaround with one or two.
- 3 tables and chairs on stage

5. REFERENCES

- [1] T. Magnusson. Herding Cats: Observing Live Coding in the Wild. *Computer Music Journal*, 38 (1):91–101, 2014.
- [2] F. Bernardo, C. Kiefer, T. Magnusson. An AudioWorklet-based Signal Engine for a Live Coding Language Ecosystem. In *Web Audio Conference*, forthcoming.

6. ACKNOWLEDGMENTS

Our work is funded by the AHRC through the MIMIC project, ref: AH/R002657/1 (<https://gtr.ukri.org/projects?ref=AH/R002657/1>)

7. LINKS

Sema: <https://github.com/mimic-sussex/sema>

MIMIC Project website: <https://mimicproject.com>

Thor Magnusson:
<http://www.sonicwriting.org>, <http://www.ixi-audio.net>

Chris Kiefer: <https://luuma.net/>

Francisco Bernardo:
<https://frantic0.com>, <https://mnistrel.com>

Cyclic Gibbering

Charles Roberts

Interactive Media & Game Development Program
Department of Computer Science
Worcester Polytechnic Institute
charlie@charlie-roberts.com

ABSTRACT

This live coding performance uses Gibber, a browser-based environment for live coding. It is the first performance to feature a browser-based implementation of the TidalCycles mini-notation, enabling new rhythmic and sequencing possibilities in Gibber.

1. PERFORMANCE NOTES

The community around TidalCycles [2] has nurtured it into one of the world's most successful live coding languages, used internationally to both perform and teach. I've been attracted to the expressiveness of its mini-notation for describing rhythmic patterns, and often wished that I could bring the same rhythmic expressiveness to sequencing in Gibber, a browser-based live coding system I design and develop [3]. With the help of Mariana Pachon-Puentes, I recently developed a browser-based implementation of the TidalCycles mini-notation, and incorporated it into Gibber, adding experimental annotations to help reveal the state and progression of running musical patterns as shown in Fig. 1.

This performance will be the first public performance to use the TidalCycles mini-notation in Gibber, and will feature sounds synthesized in the browser mixed with samples drawn from the Freesound online repository [1].

2. PERFORMER BIO

I am an Assistant Professor in the Department of Computer Science at the Worcester Polytechnic Institute, with an affiliation in the Interactive Media and Game Development program. I research human-centered computing in digital arts practice, and am the lead designer and developer of Gibber, an open-source, creative coding environment for the browser. I have given live coding performance using Gibber throughout North America, Europe, and Asia.

3. REFERENCES

- [1] V. Akkermans, F. Font Corbera, J. Funollet, B. De Jong, G. Roma Trepas, S. Togias, and X. Serra. Freesound 2: An improved platform for sharing audio clips. In *Klapuri A, Leider C, editors. ISMIR 2011*:

```
verb = Bus2('spaceverb')
delay= Bus2('delay,1/6')
Tidal.cps = 140/120/2
drums = EDrums().connect( verb, .05 ).connect( delay, .1)

drums.tidal(
  < kd kd*2 >
  [ sd kd ]
  ~
  < [ kd ~ sd kd ]
  [ kd sd ]
  [ kd sd sd ]>
  < ch ch*2 >
  < cp cp ~ cp*2 ~ kd >
  ` )

bass = Monosynth('bass')
bass.note.tidal( `[<0 7 -7>] 3 ~ 4 ~ [<7 0>]]*2` )

pad = PolySynth('bleep').connect( verb, 1 )
pad.chord.tidal( `[0,2,4,5]` )
```

Figure 1: Multiple annotated sequences running concurrently. One kick drum token and two notes in the bass line are highlighted to indicate activity.

Proceedings of the 12th International Society for Music Information Retrieval Conference; 2011 October 24-28; Miami, Florida (USA). Miami: University of Miami; 2011. International Society for Music Information Retrieval (ISMIR), 2011.

- [2] A. McLean and G. Wiggins. Tidal-pattern language for the live coding of music. In *Proceedings of the 7th sound and music computing conference*, 2010.
- [3] C. Roberts, M. Wright, and J. Kuchera-Morin. Music Programming in Gibber. In *Proceedings of the International Computer Music Conference*, pages 50–57, 2015.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Trondheim EMP Repository processing

Øyvind Brandtsegg
Norwegian University of
Science and Technology,
Trondheim
oyvind.brandtsegg@ntnu.no

Anna Xambó
Norwegian University of
Science and Technology,
Trondheim
anna.xambo@ntnu.no

Trond Engum
Norwegian University of
Science and Technology,
Trondheim
trond.engum@ntnu.no

Andreas Bergsland
Norwegian University of
Science and Technology,
Trondheim
andreas.bergsland@ntnu.no

Carl Haakon Waadeland
Norwegian University of
Science and Technology,
Trondheim
carl.haakon.waadeland@ntnu.no

ABSTRACT

The ensemble Trondheim Electroacoustic Music Performance (EMP) investigates new modes of communication in an ensemble when new technology is introduced as part of the ensemble repertoire. In the recent three years, the focus has been on crossadaptive processing as a musical intervention in the interplay. More info on the crossadaptive project can be seen at the project blog [3] and also the album *Poke It With A Stick / Joining The Bots* released in 2019. For more background on aspects of crossadaptive processing and performance, see [1, 2, 4, 5, 6]. For this performance we work with a web-based repository of sounds, explored via a live coding interface [7, 8]. This allows access to a massive archive of sounds from Freesound.org, and the selection of sounds is done via sound descriptors. The integration of this instrument in an ensemble setting is interesting, as traditional and nontraditional modes of musical interaction are activated in dialogue. Using web-based access to the repository allows a generality of instrument design, and this is combined with a strategic mode of performance in this instrument. With "strategic" we mean here, that most sounds are not performed directly by physical action, but cued up in patterns via live coding. These sounds are then live processed by other members of the ensemble, responding more directly with physical and gestural instrumental action on the signal coming from the web instrument. The signal is also live processed in a crossadaptive fashion, so that audio features extracted from other performers' actions will modulate the parameters of processing for the web audio instrument. The ensemble also utilize gesturally based interfaces from interactive dance, here used to control elements of audio synthesis and processing.

0.1 Audio documentation

The proposed performance is improvised, and as such we can not provide a recording of the actual piece to be performed. Rather we submit examples from this type of

performances with the ensemble, from a session in June 2019. The recordings can be found here:
<http://folk.ntnu.no/oyvinbra/wac2019/TEMP-14.06.19-track5-mix1.1.wav>

0.2 Technical requirements

Øyvind (electronics)

- 2 output lines, balanced jack or XLR
- 2 solid music sheet stands for equipment (!)
- Small table (approx. 70x100 cm) for equipment
- 2 return lines (mono drums mix, dry vocal) for processing

Anna (electronics)

- 2 output lines, balanced jack or XLR
- Small table (approx. 70x100 cm) for equipment

Trond (guitar, electronics)

- 4 output lines, balanced jack or XLR
- Small table for equipment
- Guitar stand

Andreas (electronics)

- 2 output lines, balanced jack or XLR
- Small table (approx. 70x100 cm) for equipment

Carl Haakon (drums)

- 1 snare drum (on snare drum stand)
- 1 floor tom (preferably 14")
- 1 mounted small tom (preferably 10")
- 2 cymbal stands
- 1 drum chair
- 1 small table (for various percussion)
- Miking: 2 Overhead, 2 close mics

Monitoring

- Stereo monitoring with the same mix as the front of house sound. Approx 8 monitor speakers distributed between the musicians on stage.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

0.3 Bios

Øyvind Brandtsegg is a composer and performer working in the fields of algorithmic improvisation and sound installations. Recent writings include *Csound: A Sound and Music Computing System* (Springer, 2016, with J. Fitch, S. Yi, J. Heintz, O. Brandtsegg, and I. McCurdy). His main instruments as a musician are the Hadron Particle Synthesizer, ImproSculpt and Marimba Lumina. Hadron is an extremely flexible realtime granular synthesizer, widely used within experimental sound design with over 200.000 downloads of the VST/AU version. Brandtsegg uses it for live processing of the acoustic sound from other musicians. As musician and composer he has collaborated with a number of excellent artists, e.g. Motorpsycho, Maja Ratkje, and he runs the ensemble Trondheim Electroacoustic Performance (T-EMP).

Anna Xambó is an experimental electronic music producer and researcher. Her musical practice includes live coding, multichannel spatialization, tangible music, collaborative interfaces, audience participation with mobile devices, and real-time music information retrieval. To date, she has released three solo recordings: “init” (2010, Carpal Tunnel), “On the Go” (2013, Carpal Tunnel) and “H2RI” (2018, pan y rosas). Her solo and group performances have been presented internationally in Denmark, Germany, Norway, Spain, Sweden, UK and USA, including “Hyperconnected Action Painting” (WAC 2017) and “Imaginary Berlin” (WAC 2018).

Trond Engum is a guitarist, composer and music-technologist. Engum has an international career as a composer and performer since mid 90’s in bands such as “The Soundbyte”, “The 3rd and The Mortal” and “T-EMP”. His works are published through numerous recordings, tours and concerts on stages and festivals around the world. Engum has also composed music for several theatrical performances and television programs.

Andreas Bergsland has composed several pieces for interactive dance that have been presented in Norway, Sweden, Denmark, Germany, Austria, Greece, Italy, Canada and the US in collaboration with choreographer Robert Wechsler and others. He has also been involved in composition and sound design for exhibitions, installations, large-scale multi-media event, in addition to doing live-electronics performances and working with computer instrument design for motion capture systems. Together with the MotionComposer team he received a special recognition award in the 2016 Guthman Musical Instrument Competition.

Carl Haakon Waadeland is a drummer and researcher within empirical rhythm research and models of rhythm performance. He has participated on a large variety of recordings and tours with artists and bands like Dadafon, Anne-Lise

Berntsen, Siri’s Svale Band, Åge Aleksandersen & Sambandet, Warne Marsh, Kenny Wheeler, Mikis Theodorakis & Arja Saijonmaa. Waadeland has moreover published several articles in international journals, e.g.: Journal of New Music Research, 2001; Experimental Brain Research, 2009; NeuroImage, 2011; Journal of the Acoustical Society of America, 2015; Human Movement Science, 2017; Springer Lecture Notes in Computer Science, 2018.

1. ACKNOWLEDGMENTS

The recording session for the submission was made possible with the technical support and engineering assistance of Thomas Henriksen at the Institute of Music’s studio in Olavshallen, Trondheim.

2. REFERENCES

- [1] M. Baalman, S. Emmerson, and Ø. Brandtsegg. Instrumentality, Perception and Listening in Crossadaptive Performance. In *Proceedings of the 2018 Conference on Live Interfaces*, 2018.
- [2] Ø. Brandtsegg. A Toolkit for Experimentation with Signal Interaction. In *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, pages 42–48, 2015.
- [3] Ø. Brandtsegg. Cross Adaptive Processing as Musical Intervention, 2019. <http://crossadaptive.hf.ntnu.no/>.
- [4] Ø. Brandtsegg, T. Engum, and B. I. Wærstad. Working Methods and Instrument Design for Cross-Adaptive Sessions. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 1–6, Blacksburg, Virginia, USA, June 2018. Virginia Tech.
- [5] Ø. Brandtsegg, S. Saue, and V. Lazzarini. Live Convolution with Time-Varying Filters. *Applied Sciences*, 8(1), 2018.
- [6] J. D. Reiss and Ø. Brandtsegg. Applications of Cross-Adaptive Audio Effects: Automatic Mixing, Live Performance and Everything in Between. *Frontiers in Digital Humanities*, 5:17, 2018.
- [7] A. Xambó, A. Lerch, and J. Freeman. Music Information Retrieval in Live Coding: A Theoretical Framework. *Computer Music Journal*, 42:9–25, 2019.
- [8] A. Xambó, G. Roma, A. Lerch, M. Barthet, and G. Fazekas. Live repurposing of sounds: Mir explorations with personal and crowdsourced databases. In T. M. Luke Dahl, Douglas Bowman, editor, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 364–369, Blacksburg, Virginia, USA, 2018. Virginia Tech.

Responsive Space

Raimund Vogtenhuber
Institute for Computer Music and Sound Technology
Zurich University of the Arts
raimund.vogtenhuber@zhdk.ch

ABSTRACT

The performance "Open Form III" is created with a framework called "Responsive Space". The framework allows a flexible way of working with distributed sound and image projections. The system consists of a multichannel speaker-system, 2-3 video projections a local Wifi network with a connected webserver and mobile devices. The audience is invited to log into the local network with their mobile devices. In the browser of the mobile devices sound and image is generated or streamed.

As we experienced in previous projects with live audio-streaming and mobile devices the spatial gesture of music as a musical parameter comes to the fore. In the performances the audio is streamed with an icecast-server. The visual output is generated with javascript in the client's browser. Shortcomings, like the various time delay of the audio-stream of different mobile devices or the variation of spatial and visual output, lead to interesting artistic effects. The system offers great opportunities to experiment with different spaces and the spatialization of sound and image.

1 INTRODUCTION

The project deals with the question of how the audience can be addressed and integrated in multimedia performances. It is about the development of a performance system that focuses on the use of mobile devices (smartphones) of the audience and explores the possibilities in connection with a loudspeaker setup and projections.

This mobile setup should enable a flexible distribution of sound and image projections. The aim is to experiment with it in different rooms and situations. The viewers can log into a local network with their mobile devices. The browsers of the smartphones or tablets will then generate audiovisual content and offer opportunities for interaction.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

1.1 Description of the piece

The piece "Open Form III" generates variations with five different sound modules for each playback. Simple basic sounds, such as a sinewave, noise and granulators generate tonal structures which are formed into musical sequences according to contrapuntal criteria and stochastic processes. The superposition and layering of these simple tonal gestures results in a complex musical structure. The individual sounds are translated into visual forms and determined by the parameters of the sound and an independent process. The sequence of the piece is created generatively and is not predetermined. It experiments with the interplay of sound and image and their distribution in space through various reproduction media.

1.2 Setup

The Setup includes a multi-channel speaker system, up to three visual projections distributed throughout the room, and a local network for the audience's mobile devices.

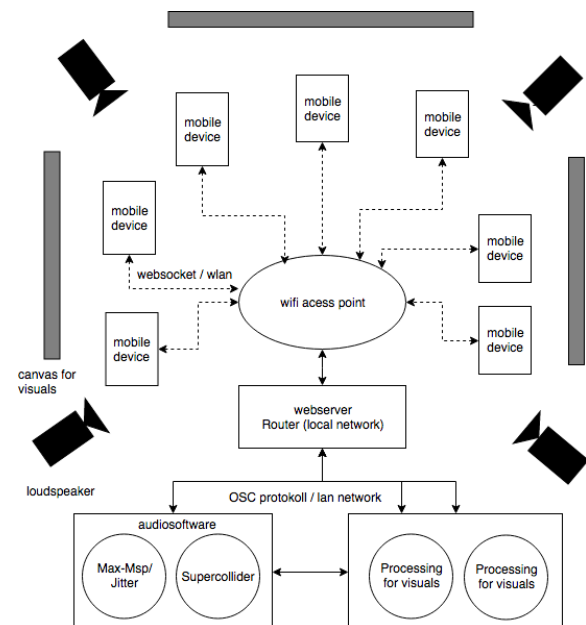


Figure 2. Communication diagram of the setup

gravity|density

Anthony T. Marasco

Experimental Music & Digital Media
School of Music and CCT
Louisiana State University
Baton Rouge, LA, USA
amarasco@lsu.edu

Jesse Allison

Experimental Music & Digital Media
School of Music and CCT
Louisiana State University
Baton Rouge, LA, USA
jtallison@lsu.edu

ABSTRACT

gravity|density is a work for cyber-hacked devices and Web Audio applications. Our goal is to develop systems that merge repurposed and hacked pieces of hardware into the networked world of web art. While the electronic sophistication of mobile devices and the flexibility of web applications allow artists to create immerse audiovisual environments without the use of traditional music hardware, we believe that digital artists should not cast aside the tools of the past, but rather find new and creative ways of modifying them so that they can inform the ways in which we explore and create with new digital, web-based tools. Through these new hybrid systems, we can both embrace the limitations and push the boundaries of any hardware we use for the purpose of creating collaborative sonic environments.

In *gravity|density*, we begin by manipulating fixed-audio sources through the performance of hacked CD players. The sonic results of this mangled audio is sampled and then distributed to the audience's mobile devices in both passive and interactive manners. Passive distributions allow us to create intricately-spatialized rhythmic interplay between the glitching CD players and the blanket of overlapping samples dispersed throughout the networked audience. Active distributions allow the audience to join in our performance; by choosing small portions of the audio sent to them and sending these selected samples back to us, we string this audio together and feed it into a cyber-controlled distortion pedal before sending it back to the audience for more manipulation. This results in overlapping cycles of control and audio generation between performer, audience, network, and machine.

1. TECHNICAL REQUIREMENTS

The performance duration is approximately 9 minutes. We will require the following tech from the venue:

- 5 channels audio output: 5 mono signals (two per CD player, one from the distortion pedal), provided to the house from the stage.
- Hall distribution pattern should be as follows: CD



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).



Figure 1: gravity|density spinning discs

player 1 stereo (front left, front right), CD player 2 stereo (rear left, rear right), distortion pedal (center channel). Audio will also be distributed to audience mobile devices in various configurations throughout the performance.

- Two tables: one for the performers and one to place the CD players and distortion pedal.
- Two chairs.
- Video projection: a camera feed of the CD players will be provided to the house so that it may be projected for the audience to see.
- A robust WiFi network for connecting audience members to the networked web performance interface

All other tech (cyber-hacked CD players, distortion pedal, performer laptops, video camera) will be provided by the performers.

2. DOCUMENTATION

Further information and a video demonstration can be found at <https://gravity.emdm.io/>

3. ARTIST BIOGRAPHIES

Anthony T. Marasco is a composer and sound artist who takes influence from the aesthetics of today's Digimodernist culture, exploring the relationships between the eccentric and the every-day, the strict and the indeterminate, and the retro and the contemporary. These explorations result in a wide variety of works written for electro-acoustic ensembles, interactive computer performance systems, and multimedia installations. An internationally recognized composer, he has received commissions from performers and

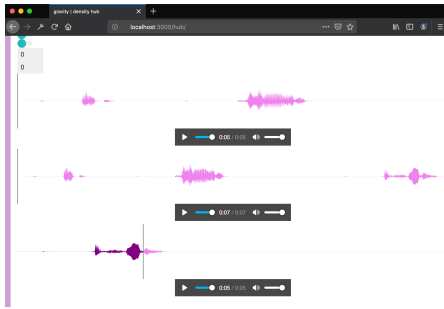


Figure 2: gravity|density hub containing samples captured by the audience

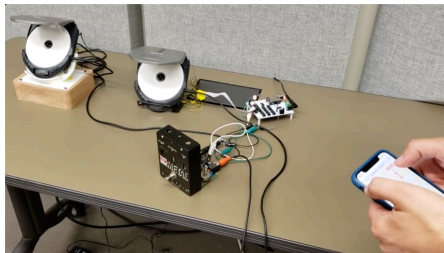


Figure 3: gravity|density performed with Ben-dit.I/O cyber hacking

institutions such as WIRED Magazine, Phyllis Chen, the American Composers Forum Philadelphia, Quince Contemporary Vocal Ensemble, Toy Piano Composers, the Rhymes With Opera New Chamber Music Workshop, Data Garden, andPLAY Duo, and the soundSCAPE International Composition Exchange. Marasco was the grand-prize winner of the UnCaged Toy Piano Festival's 2013 Call for Scores, a resident artist at Signal Culture Experimental Media Labs, and a grant winner for the American Composers Forum's "If You Could Hear These Walls" project. His works and research have been featured at festivals across the globe, such as NIME, the Web Audio Conference, the Toronto International Electroacoustic Symposium, SEAMUS, Electroacoustic Barn Dance, New York City Electroacoustic Music Festival, ICMC, Montreal Contemporary Music Lab, and Omaha Under the Radar. Marasco is currently a Ph.D. candidate in Experimental Music & Digital Media at Louisiana State University.

Jesse Allison is a leader in sonic art technology, thought, and practice. Dr. Allison holds the position of Assistant Professor of Experiment Music & Digital Media at Louisiana State University. As part of the Cultural Computing focus of the LSU Center for Computation & Technology, he performs research into ways that technology can expand what is possible in the sonic arts. Prior to coming to LSU, he helped to found the Institute for Digital Intermedia Art at Ball State University and Electrotap, an innovative media arts firm.

Research and invention interests include computer interactivity in performance, distributed music systems, mobile music, interactive sonic art installations, hybrid worlds, and multi-modal artworks, those that can be experienced through a variety of means. As such, he manages the Media Interaction Laboratory and Library (MILL), co-directs the

Laptop Orchestra of Louisiana (LOLs), and heads up the Mobile [App | Art | Action] Group (MAG) for the CCT.

As an artist, Allison has disseminated his work around the globe through live performance art, interactive installations, virtual and hybrid worlds interventions, and presentations. Recent performances/exhibits include the Pixelations Festival, New Instruments for Musical Expression (NIME), Siggraph, Techfest Bombay, International Computer Music Conference (ICMC), the IUPUI Intermedia Festival, Boston Cyberarts Festival, and the Society for Electro Acoustic Music in the United States. Allison received his doctor of musical arts in composition from the University of Missouri at Kansas City.

Map mop

Gerard Roma
CeReNeM
University of Huddersfield
g.roma@hud.ac.uk

ABSTRACT

This submission proposes a music performance to be delivered by the author on a multi-touch tablet. The piece is based on a collection of sounds generated using various analog devices. The collection is presented as an interactive visualization. The interface is projected, so both the collection and the performer's actions are readable to the audience.

1. PROJECT DESCRIPTION

This piece explores the use of "sound maps": visualizations of sound collections generated automatically from audio analysis. An example is shown in Figure 1. The maps use content-based audio analysis for visualizing each sound and also for arranging the layout. The technology will be described in detail in the associated talk at the conference. A multi-touch tablet is used to play a map. Touch actions trigger sounds. Recording of these gestures result in autonomous agents that navigate the space. The interface is projected, making the performance readable in the tradition of live coding. For this performance, the material is created from various DIY analog devices focusing on circuit bending and chaotic behavior. Such devices create very interesting sounds, but they are, by design, very difficult to control. So for this piece, a large quantity of sounds is produced in advance, then the map is used a control interface.

2. TECHNICAL REQUIREMENTS

The performance is executed on an iPad tablet, which needs to be connected to both a stereo PA and a projector. The proposed setup is to use an HDMI adapter and HDMI audio extractor provided by the author. This will result on separated HDMI video and unbalanced audio signals. The later would be connected to a DI provided by the venue. Ideally the audience should be sat and the light should be dim. There should be a screen or projection surface, preferably on stage above the performer. A table and chair are required on stage. The expected duration of the set is 10 minutes.

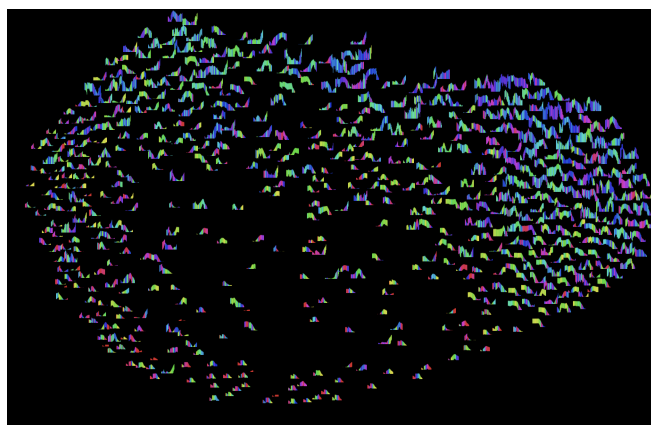


Figure 1: An example sound map.

3. DOCUMENTATION

The piece and associated technology are still a work in progress. Some recordings of the current state can be obtained from google drive: <https://drive.google.com/open?id=1wI2WXr70we0Q5bHE4KrrqM9QtEG29RHq>

4. ACKNOWLEDGMENTS

This research was part of the Fluid Corpus Manipulation project (FluCoMa), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725899).



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).



Keynotes

Becoming latency-native

Rebekah Wilson
Amsterdam, The Netherlands
rebekah@loopcontrol.io

ABSTRACT

The potential richness of audio technology on the internet springs from advancements integral to developments driven by the primary concerns of commerce and science, giving rise to operable and affordable bandwidth in newly-accessible geographical areas as well as the growing sophistication of codecs, browser technology and audio frameworks. Yet use of these tools remains unexplored for music performance, with the primary cause of disruption to performance flows being transmission latency. Through the employment of sophisticated tools and processes, musicians may, however, learn to navigate Networked Music systems as a native performance platform.

1. A TOOL AND A HUMAN ACTIVITY

In 1969, 50 years ago as of the writing of this text, UCLA graduate student Charley Kline sent the first digital message to Bill Duvall in Stanford, the other side of California. That communication link was how ARPANET began, leading to today's modern internet. In these last 50 years we have seen digital communications give rise to an expanse of services built on the internet, such the webRTC framework that has prompted the development of an expanse of high-quality services for transmitting real-time audio suitable for performance. This has been made possible thanks to a few key developments: the provision of the OPUS codec with music-quality encoding, the ability to disable echo cancellation, and the ability to process and manipulate transmitted audio using audio APIs. As Heidegger observed, technology is both a means to an end as well as being a human activity [1]. When a technology affords us an expressive outlet, we cannot help but to explore this technology through a creative lens. Armed with this knowledge, any developer may create a platform for long-distance music performance, designed specifically for her purpose.

2. NON-LINEAR PERFORMANCE

What was once limited to complex systems—high-quality, low-latency audio over long distances—now becomes ubiquitous. Networked Music Performance, a real-time synchronous audio system for remotely interacting performances over geographic distances [2], is however subject to limitations specific to its nature: latency, technical uncertainty and the loss of full sensory feedback such as acoustic resonance. Technical uncertainty refers to the oft-complex systems that musicians need to put in

place in order to perform together remotely; this particular limitation can be solved by way of improvements in interface design. The lack of full sensory feedback—for example, when we cannot sense a musician's subtle bodily gestures that may otherwise be providing significant information to us in an ensemble performance—can be accommodated for with increasingly sophisticated technology as video streaming improves in speed and quality; VR/AR advances ensure a future rich with immersive remote experiences. Latency however is an endemic property of digital networks and will remain so while data can travel no faster than the speed of light.

Since sound transmission has been possible, extensive research has been conducted on the effects of latency in music. In Networked Music Performance the “overall delay experienced by the players includes multiple contributes due to the different stages of the audio signal transmission” [2], most significantly network packetization and transmission latency, and hardware and software processing latency. Alexander Carôt states that “if the latency exceeds a certain value, a realistic musical interplay becomes impossible” [3] an agreement shared by most literature that musicality is dependent on latency being as low as possible.

And yet, we live in a non-linear world.

In Daniel Chua's poetic text on the operatic echo, he says “music takes time. The echo, by measuring the distance between subject and object, simply stretches the point” [4] In Monteverdi's *Vespers 1609*, an antiphonal echo sounds reverberantly from behind the audience. Echoes are “messy; they don't synchronize or return to their place of origin; they travel unexpected distances to connect unrelated times” (ibid). Similarly, latency in music creates a kind of messiness. As in Monteverdi, the effects can be spectacular.

3. LATENCY AND FLOW

When performing together in composite space—where musicians share both location and time—a great deal of information is exchanged that fosters Csikszentmihalyi's concept of ‘flow’, a desirable psychological state defined as a “collective state of mind” that occurs when “members develop a feeling of mutual trust and empathy, in which individual intentions harmonise with those of the group” [5]. Flow is experienced when musicians play in time together, and when they resonate with each other in acoustic space. When performing together remotely over a network, not only do musicians lose the experience of acoustic resonance but their interactions, subject to latency, lead to interrupted rhythms and difficulties with synchronisation. Yet, human reaction time is on the order of hundreds of milliseconds and quite variable [6], opening the question that it is not transmission latency that



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

disrupts musicians as a rule in networked music, but our perception of latency.

Music data is traditionally relational while language is transactional [7]. Transactional data can be packetised, encoded and decoded, while relational data relies on cues and forms of data which may not be able to be encoded in an internet transmission, such as the glance of one musician to another to indicate the start of musical section or a change in improvised direction. Latency disrupts relational data; it breaks us into parts and “nobody can tell us whether what we have spilt up can be put together again or not” (ibid). When we lose sufficient information to intuitively interpret data that has been split up, data becomes transactional rather than relational. Any interface that “seeks to engage with our personal act of knowing needs to be able to afford us our relational dimension in balance with our transactional” (ibid); unless we take care in the design of our interfaces to accommodate for this need for balance, we will experience tension when we perform together.

4. LATENCY, US AND MACHINES

Between continents, latency is too high even on dedicated networks for traditional performance. Consider efforts to colonise space—even to the moon the latency is around 1.25 seconds. So, what kind of music will we make together when we are on Mars where, depending on the position of the planets, we will experience a latency of 3-20 minutes? Fortunately, when working within the realm of the technology system itself we may make use of the means of transmission and employ machines to work on our behalf not only as transmitter but as interpreter of data.

Nevejan says that where ‘latency breaks us, it makes us stronger’ [Nevejan, personal communication]; consequently, technology gives us opportunities to create new realities. However, to make new realities “you have to be bilingual” (ibid). Becoming bilingual with networked music means becoming latency-native and adapting ourselves as performers and adapting our idea of what is considered musical. Contemporary musicians are well-suited to this experience; as pianist Reinier van Houdt notes, musicians have been “disciplined enough to be free and expect the unexpected” [van Houdt, personal communication] and are already challenged by dangerous body movements, fatigue, masks, cramps, extreme temperatures, live processing; the “experimental mind is not worried about the conditions but sees them as a challenge, a medium for research for new experiences.” (ibid).

A number of Networked Music explorations have experimented with strategies to explore performative issues specific to the platform, such as Ethan Cayko’s toporhythmic patterns [8], Jacktrip’s wavetable mode [9], Andrew MacMillan’s system for remote parameter control [10], and contextual analysis through machine learning [11]. Machine learning tools shows promise in detecting musical information and is a growing source of great innovation. In a Networked Music Performance context, achieving flow may be reached through the awareness and interaction with latency’s effect on time. Taking a cue from Nevejan’s concept of “finding the first beat” [12], where moving together in time occurs when recognisable rhythmic patterns begin with clear starting point, we can take advantage of the machine’s ability to detect and

analyse the patterns of how we move together as musicians, aiding us to strengthen and reinforce rhythmic cohesion while encouraging new aesthetic perspectives and sonic realities. By more closely interfacing our performances and our rhythms with the Networked Music machine-system itself, we become closer to becoming bilingual and latency-native.

Yet we must remember that the machine’s purpose is determined by the engineer at the time of execution, while meaning is created at the time of observation by a human. Machines are only ideal for data mining and processing—not for creating meaningful observations; any meaning that is created is tangential to the machine’s purpose. A machine has no tacit knowledge from which to create meaning—it can only expose explicit information to be acted on. Can these machines and algorithms be designed to act in ways humans cannot? What can we encode, construct, transform and transmit with these machines? How will these machines co-exist with multiple authors? Will it give rise to something beautiful, when we experience the technological illusion of being present in a place other than our physical selves and actions?

5. REFERENCES

- [1] Heidegger, M., 1977. *The Question Concerning Technology and Other Essays*. Harper & Row, New York, NY
- [2] Rottondi, C., Chafe, C., Allocchio, C. and Sarti, A. 2016. “An Overview on Networked Music Performance Technologies,” *IEEE Access*, vol. 4, pp. 8823–8843
- [3] Carôt, A., 2009. “Towards a comprehensive cognitive analysis of delay-influenced rhythmical interaction”. *Proceedings of the 2009 International Computer Music Conference, ICMC*
- [4] Chua, D. 2005., “Untimely Reflections on Operatic Echoes: How Sound Travels in Monteverdi’s L’Orfeo”, *The Opera Quarterly*, Volume 21, Issue 4, Pages 573–596
- [5] Sawyer, R., 2003. *Group Creativity: Music, Theater, Collaboration*. Taylor & Francis.
- [6] Chafe, C., 2011. “Living with net lag,” in *Audio Engineering Society Conference: 43rd International Conference: Audio for Wirelessly Networked Personal Devices*
- [7] Gill, S., 2015. *Tacit Engagement: Betwixt and Inbetween* Springer International Publishing
- [8] Cayko, E., 2016. *Rhythmic Topologies and the Manifold Nature of Network Music Performance*. Dissertation, University of Calgary.
- [9] Cáceres, J.-P. and Chafe, C. 2010. “JackTrip: Under the Hood of an Engine for Network Audio,” *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187.
- [10] MacMillan, A. and Wilson, R., 2019. “Being less un-together”. *JONMA*, vol 1.
- [11] Wilson, R., 2019. “Towards Responsive Scoring Techniques in Networked Music Performance”. *Proceedings of the 2019 TENOR Conference*, Melbourne.
- [12] Nevejan C., Sefkatli P., Cunningham S., 2018. *City Rhythm*. Delft University of Technology

On our Past, Present, and Future with Web Audio Technologies – a participatory keynote address

Norbert Schnell
Furtwangen University
sor@hs-furtwangen.de

Preamble

- (1) Since I am infected, I often have a hard time to enjoy concerts. Most of them feel like ancient rituals representing the idea that we are unable to listen to and adequately express ourselves towards each other when we are more than a few. These rituals seem to remind us that if we express ourselves we have to make the others listen and that if ever we are more than a few we have to follow a leader and/or a detailed script. Meaningful relationships are often pictured as requiring elaborated action plans and persevering repetition. I don't really remember how this relates to my life.
- (2) Since I am infected, I don't understand anymore why so many creators spend so much time on producing works which exclude their audience from the relationships they represent. At best, the audience is allowed to observe meticulously elaborated symbolic enactments from afar.
- (3) However, I appreciate the collective silence before the show, when all participants are united in mutual attention.
- (4) Like other communication technologies, web technologies extend our possibilities to create, maintain, and control our relationships to others.
- (5) Music consists in its essence of metaphors for the relationships our lives are made of and for the way we create, maintain, and control them.
- (6) What I always liked about digital music technologies is the possibility to constantly build new instruments and environments that play with these metaphors to embrace very different practices, people and communities.
- (7) What I particularly like about web technologies is how they allow for effortlessly creating networks of mutual attention and multilateral interaction that may stand as lively metaphors for open, diverse, and pluralistic societies.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

Conclusions

Evidently, since I am infected with the idea that our collective practices could be far more based on mutual listening and multilateral interaction than they are, I cannot conceive a keynote address as another of these rituals where an audience listens to somebody on stage.

Consequently, my contribution to WAC 2019 is the attempt to create a *participatory keynote address* where the WAC attendees are invited to talk about their experiences, projects and ideas.

Preparing this keynote, I am in touch with the difficulties of creating an experience where the participants are invited to express themselves. Despite my beliefs outlined above, I have strong doubts: Will the people actually be interested in *expressing themselves*? Don't they expect me to deliver a performance? – I guess, the WAC committee didn't invite me to let the others do the work. Are they able to listen to *each other*? How can I guarantee that this will be interesting and pleasant? – After all, *that* is my ultimate responsibility.

Despite the lack of understanding I displayed above, I do understand very well *why so many creators spend so much time on producing works which exclude their audience*. Over the years, I have learned to perform presentations, to lead the audience through ideas, to tell stories, to be funny, to provoke, impress, and please. But how to organize the spontaneous expression and exchange of others? Shall I guide them? Shall I let them free?

Over the past years, I have experimented with web (audio) technologies to create situations where participants would use their smartphones to spontaneously express themselves through sound. The participants were invited to connect to networks, collaboratively play on musical instruments, and collectively generate complex soundscapes. None of these musical experiences was as elaborate as the performance of a symphony or the free improvisation of a skilled musician. But similar to those, they created moments of profound collective attention and rich metaphoricity. Just that the attention of the invited participants was not directed towards a common object of interest, but towards each other.

An experience of this kind is what I would like to contribute to WAC 2019. Hope that works...

Recent and future evolution of the Web Audio API

[Extended Abstract]

Paul Adenot
Mozilla Corporation
padenot@mozilla.com

ABSTRACT

There have been 358 commits in 125 pull requests to the Web Audio API specification repository since the Web Audio Conference 2018 in Berlin, with a number of important changes.

This yearly talk at the Web Audio Conference aims to demystify the world of web specifications, as well as informing authors of the current specification status, new features, bug fixes, and what the group has planned for the future.

In consequence, this talk is structured in three parts:

- What's new in the spec since WAC'18?
- Where are we in the standardization process?
- What's next?

New in the spec since WAC'18.

As usual, a taxonomy of those changes can be established, with four general categories:

- Features
- Bug fixes
- Specification holes
- CR process compliance

This time around, no real features were introduced (with the possible exception of the various new bits of normative behaviour about auto-play of `AudioContext`).

Most of the time was spent finding and fixing specification bugs, as the specification is now very deep in the Candidate Recommendation process. Because there only is a single implementation of `AudioWorklet`, we can't really proceed further. However this is only temporary, and Firefox's implementation is coming.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

A lot of care was taken to avoid making breaking changes this year, and only stabilize the prose for a V1 release. However, changes are always necessary, sometimes to solve a real problem, sometimes because of mistakes made by the group in the past. Sometimes, those specification changes didn't make it into implementation, so the specification change can go through as-is. Other times, some data is required, or engagement with authors, to understand whether or not the change will break existing content.

Another set of issues was the aforementioned *specification holes*. Most of them have been resolved, especially when it comes to the text related to `AudioWorklet`, that was brand new. Additionally, this new bit of specification interacts a lot with script. Interacting with script and execution contexts in a specification requires lots of additional text, very often at the intersection of `ECMAScript` and `WebIDL`.

In the same vein, nailing constructors and converting a number of concepts from using English prose to using an algorithm in pseudo-code with clear steps was a clear win, for specifying edge cases: consider a JavaScript statement using a Web Audio API function, that contains two errors that would result in throwing an exception: which exception should be thrown?

Concepts that were too handwavy for a proper specification document did get clearer. In particular, the normative behaviour for the getter for the `value` attribute of an `AudioParam` finally got specified. Cycle handling, and cycles containing `AudioParam` also got specified. The relationship between the `AudioListener` and the `PannerNode` was also clarified.

Various other interactions between parts of the API also got defined: the text for the garbage collection observability issue was finally laid down, including a rewritten section about `AudioNode` lifetime.

The decision was made to punt on a few edge cases or features, and leave things for V2, so that the current document could be finished and stable.

Where are we in the standardization process.

The group has extended its charter (3 years originally, ex-

tended for 6 months), to allow finishing some work, and also to wait for Firefox's implementation of the **AudioWorklet**. We're still waiting for all PRs to merge, and all problems to be fixed, before continuing. At time of writing, there are 19 issues opened, all of them with resolution (waiting for the prose to be written). Another round of security and privacy review will take place before becoming a recommendation (the last step of the W3C process).

The Audio Working Group has moved all the issues that were not in scope for V1 to another repository called **web-audio-api-v2**, preserving all comments and links. The group has triaged the different issues using the **Project** tab, and is currently seeking input, both for features that should be included, and for discussing the details and specifics of features already included in this new version.

The Audio Community Group is a good way to engage with the standard body, with a low barrier to entry, and a monthly conference call to discuss ideas and issues.

The group expects to be done with the current issue this year, and then change the process to put more emphasis on the future development of the API.

What's next.

Multiple efforts have started that are linked to the Web Audio API specification:

- Web Codecs
- Web Audio API V2
- Audio Device Client

The Web Codecs effort aims at providing low-level primitives to have a more flexible way to decode and encode media data on the Web. It will compose well with **MediaStreams** and **Streams**, and will allow real-time and non-real-time processing of media data.

For the Web Audio API, it will be an improvement over **decodeAudioData**, that always has been the source of problems (no progress information, no chunk decoding, no encoding support, etc.). All the issues related to encoding and decoding of media files have been subsequently closed, as they are best addressed there.

The new repository Web Audio API V2 will be used for all new feature. It is still an unknown if the current text will be copied over and forked, or if the new features will be developed referencing the previous version of the standard.

Audio Device Client is a different proposal, that is related to the Web Audio API, but aims at being lower-level, while being capable of interfacing to the existing APIs on the Web platform, such as **MediaStreams**, **AudioContexts** and **HTMLMediaElements**. It is in the early stages, and its usefulness is still debated, compared to just adding the few missing features to the **AudioContext**, that already has the capabilities to interface with the Web Platform.

In any case, input from the greater community of Web Audio API users is going to be a critical component of the

success of a new iteration of the specification. The first iteration direction was in large part decided by the existing implementation, on top of which a few necessary features were added. The next version will address the problems found during the long period of time authors have experimented with the API.

Conclusion.

The Audio Working Group is currently at a pivotal point. After a number of years putting all new features in the backlog, a coherent text is being published, and new feature request will be considered (along with all the features and modification that were proposed in the past, and have been triaged).

An essential part of this new effort will be in other working groups, and non-members are encouraged to participate, via the newly formed Audio Community Group.



Workshops

Designing and Performing with Live Coding Languages for Signal Processing and Machine Intelligence on the Web

Francisco Bernardo
Emute Lab, Department of Music
University of Sussex
f.bernardo@sussex.ac.uk

Chris Kiefer
Emute Lab, Department of Music
University of Sussex
c.kiefer@sussex.ac.uk

Thor Magnusson
Emute Lab, Department of Music
University of Sussex
t.magnusson@sussex.ac.uk

ABSTRACT

This workshop will give participants the opportunity to learn how to design their own live coding mini-languages and perform with them. Participants will get familiar with *Sema* [1], our live coding language design system, a user-friendly, web-based toolkit that provides an accessible entry point to signal processing, machine learning and machine listening. Participants will engage hands-on with our system and learn how these techniques can be used in the exciting domain of live coding musical performance. Additionally, the workshop will include a reflection and group discussion about participants' experience with the system and techniques, new features and improvements.

1. INTRODUCTION

Recent advances in machine learning and machine listening show great potential to support musical composition and performance. Such emerging technologies may disrupt the paradigm for music making by shifting it from instrumental to conversational and collaborative. However, despite the formidable advances that are continuously succeeding, machine learning and machine listening represent complex computational techniques which can be difficult to understand by non-computer science creative coders.

This workshop is part of the MIMIC research project, which explores how machine learning and machine listening can be designed in a user-friendly way and provide for live coding, rapid prototyping and fast development cycles of musical applications. We build upon recent user research in Web-based technologies for creative audio processing and interactive machine learning [2], deep and transfer learning [3], [4], and live coding language design [5][6].

2. DESCRIPTION

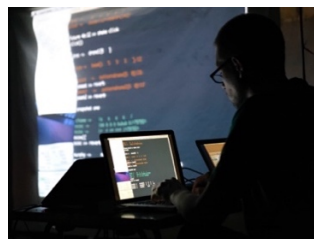
This workshop will introduce the role that machine learning and machine listening technologies can have in live coding performance [6] through language design. Participants will get familiar with our live coding language design environment and language design techniques, preparing them to work in the creation of their own live coding languages. Participants will learn how to express sound and music concepts using signal processing

expressions. They will experiment with selected machine learning models and machine listening JavaScript classes and explore some of their direct benefits, including beat detection, pattern detection and generation.

Participants will gain understanding about workflows in our system that will allow them to develop their own live code language using language design techniques. For instance, they will learn how to create, inspect and change language specifications to generate language parsers. Participants will also explore practical techniques for performing with the live code mini languages they have constructed or customised, including live coding with interactive machine learning and pre-trained generative models.

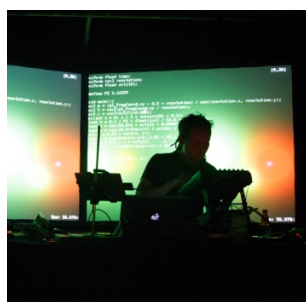
Additionally, the workshop will include a group discussion about participants' experience learning and using our system. As a result of this workshop, we believe participants will gain a better understanding of what signal processing, machine learning machine listening are in the context of live coding language design and performance.

3. BIOGRAPHY



Thor Magnusson is a worker in rhythm, frequencies and intensities. His research interests include musical improvisation, new technologies for musical expression, live coding, musical notation and digital scores, artificial intelligence and computational creativity,

programming education, and the philosophy of technology. These topics have come together in the *ixiQuarks*, *ixi lang*, and the *Threnoscope* live coding systems he has developed. As well as performing and writing about music, he lectures in music at the University of Sussex, Brighton.



Chris Kiefer is a computer-musician and musical instrument designer, specialising in musician-computer interaction, physical computing, and machine learning. He performs with custom-made instruments including malleable foam interfaces, touch screen software, interactive sculptures



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

and a modified self-resonating cello. Chris' work also concentrates on machine learning and signal processing for audio and interaction, with a particular emphasis on nonlinear and dynamical systems.



Francisco Bernardo is a computer scientist, an interactive media artist and a multi-instrumentalist. His research is focused on human-computer interaction approaches to toolkits that broaden and accelerate user innovation with interactive machine learning. Francisco has been working in applied research projects at the

intersection of art and innovation, front-end software engineering, interaction design and greenfield product management.

4. ACKNOWLEDGMENTS

The research leading to these results has received funding from AHRC through the MIMIC project, ref: AH/R002657/1 (<https://gtr.ukri.org/projects?ref=AH/R002657/1>)

5. REFERENCES

- [1] F. Bernardo, C. Kiefer, and T. Magnusson, "Sema" *github.com*, 2019. [Online]. Available: <https://github.com/mimic-sussex/sema>. [Accessed: 03-Dec-2019].
- [2] F. Bernardo, M. Zbyszyński, R. Fiebrink, and M. Grierson, "Interactive Machine Learning for End-User Innovation," in *Proceedings of the Association for Advancement of Artificial Intelligence Symposium Series: Designing the User Experience of Machine Learning Systems*, 2017, pp. 369–375.
- [3] A. Roberts, C. Hawthorne, and I. Simon, "Magenta.js: A JavaScript API for Augmenting Creativity with Deep Learning," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 2–4.
- [4] D. Smilkov *et al.*, "TensorFlow.js: Machine Learning for the Web and Beyond," in *Proceedings of the 2nd SysML Conference*, 2019.
- [5] G. Wakefield and C. Roberts, "A Virtual Machine for Live Coding Language Design," *Proc. New Interfaces Music. Expr.* 2017, pp. 275–278, 2017.
- [6] C. Kiefer and T. Magnusson, "Live Coding Machine Learning and Machine Listening: A Survey on the Design of Languages and Environments for Live Coding," in *Proceedings of the International Conference on Live Coding.*, 2019.

Live coding with Makkeróni - workshop

Balázs Kovács, dr. habil.

University of Pécs, Faculty of Arts, Department of Electronic Music and Media

H-7630 Pécs, Zsolnay Vilmos u. 16.

kovacs.balazs@pte.hu

ABSTRACT

Makkeróni is a web-based live coding system which simulates a linux shell. The user operates with commands known from the BASH (ls, help, cat etc.), and also there are some audio-specific commands (play, freq, tone etc.). In this workshop we're going to learn how to use it, while possibly focusing on efficiency (pipes, joker characters, command history tricks) and network cooperation. If the participants are open for it, we could discuss the targets of further development as well.

It's recommended for both new and experienced Linux shell users.

Optimal length: 45-70min

Link to the application: <http://makker.hu/makkeroni/>

1. INTRODUCTION

The idea of Makkeróni came from two directions: on one side, I've been working on linux system modifications for creating the running processes audible (it's called Proc filesystem music¹), inspired by the SonicFinder project by William W. Gaver (1989). In this case my interest continuously focused on low-level approaches for sound synthesis or processing, trying to use system-level processes instead of auditory icons. On the other side I began to create inter-website or real-world ↔ web communication projects, for example a webpage-controlled mechanical device or public light or-gan projects on the chimneys of the Zsolnay Cultural Quartier, Pécs. These projects opened up the way for starting cross-website multimedia interaction.

Collecting something from all of the above projects, the idea of a web-based sonic operating system was born: a web application that simulates a linux shell - the operating system which is run-ning the webserver itself - while the user is controlling it with commands and their arguments on the client's side. The result is Makkeróni, a text-based audio "operating" system.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

¹ Proc filesystem music, as documented at the following URL: <http://kbalazs.periszkopradio.hu/index.php?file=works/2005-lad>

While designing Makkeróni, the following principles were used:

1: create a simple-to-use, easy-to-run live coding system with zero or minimal external libraries; create an useful and practical interface, which makes easy to seamlessly learn the use of the Linux shell;

2: create an audio system which utilizes the efficiency of linux: not only with the command and file structure, but with the low-level approach to processes, data etc. as well. It could be a goal also to create a system, which could be a virtual terminal too for a real, locative server;

3: utilize the native possibility of the web: two-way communication between client and server or client and other clients, data sharing etc.;

I tried to achieve these principles with the following methods:

- Makkeróni is much simpler to use than other live coding systems. It comes from the shell behaviour: one line + enter key = one command. Everything is self-documented in a practical way. The user can be quickly introduced into the application while simply using and interacting with it: help and other command works in order to explain the use. Changelog, reference and other documents are also in-system.

- using existing shell commands (ls, cat, ps etc.) include the operating system commands into the live coding process - which means also that they produce sounds also by running, creating sounds which are reflecting to the result of the command itself;

- using joker characters to randomize parameters;

- using command history to quickly recall the previous events;

- using autocomplete for efficiently typing of the commands;

- the system should be capable to be expanded with user-added sounds and presets, available for the other users of Makkeróni;

- the users can save the running processes' list on the server, letting them possible for others to use their results.

Makkeróni has been realized with adapting the JQuery Terminal Emulator by Jakub Jankiewicz² which offers many possibilities to create a shell-like interpreter, parsing commands and arguments, provides mobile-friendly interface etc. As a

² <https://terminal.jcubic.pl/>

starting point, I expanded this wonderful framework with extra functions that could perform the audio processing and live controlling. In the workshop, I'll going to present how it works.

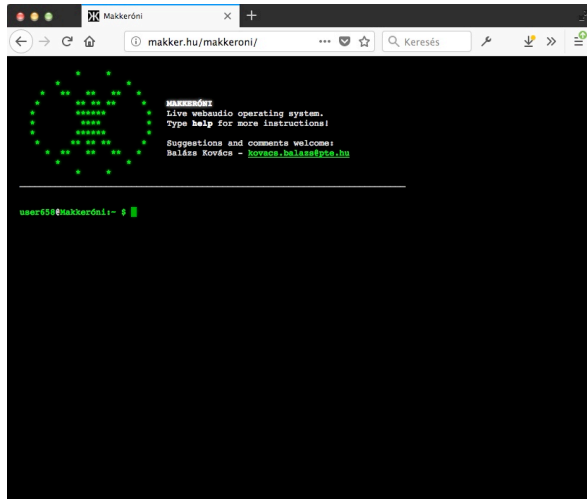


Figure 1. Makkeróni start-up screen. Type 'help' for getting started!

2. SHORT REFERENCE

In the followings I summarize the main commands of Makkeróni in the current phase of development:

Table 1. Short reference

Command	Description
freq	play a sinewave tone
fmfreq	play an fm-modulated sinewave tone
tone	play a simple tone with filter
makkeróni	fm-modulated synth with audio rate mathematical pair
ls	list contents of the home, soundfiles & saved presets folders
play	play a soundfile
loopplay	play a soundfile looped
fadeplay	play a soundfile looped, with linear fade-out
watch	repeatedly start play, freq, fmfreq or any other commands
batch	start a command in multiple instances
seq	store a number of sequences into one of the 4 sequence slots
seqlist	simple stepsequencer for lists of numbers stored by the "seq" command
stop	stop one or more loop-play thread or watch process

sleep	sleep (mute) one, more or all processes
resume	resume (unmute) a process
remakker	restart a loop-play or watch thread. Provides new process id
replace	replace a thread with a new command
ps	list of running loopplay and watch processes
degrade, bitshift	sample-level audio manipulation (LOUD!)
upload	upload a sample (wav,mp3 or ogg) into the soundfile folder
help	short description of commands
fontsize	set fontsize (default: 12)
statuslength	set number of lines in status bar (default: 7)
clear	clear window
cat	print the contents of a textfile
connect, disconnect	connects or disconnects to/from the chat server
wall	send a command or message to the other users

In addition of the above, there are some general syntax to be applied for most of the commands:

- *: random number (randomize parameters on play, loopplay, fadeplay, freq, and fmfreq)
- ↑ and ↓ arrows: browse command history
- something and TAB key: autocomplete command (for. ex 'lo' + TAB gives 'loopplay' back)

3. ADDITIONAL INFORMATIONS

The workshop is open for anybody interested in using linux shell for creating audio or multimedia processes. The active participants need a computer, I'll need a projector and PA. And of course we all need network access:) If You're interested, please visit Makkeróni's website: <http://makker.hu/makkeroni/>. If You have recommendations for a special topic to cover, please drop me a line in advance, to the email address above. And, if You have ideas to implement, and/or time to help in further development, You're mostly welcome!

4. BIO

[Balázs Kovács](#), PhD habil., philosopher, holds a PhD in aesthetics of interactive sonification. Head of the [Electronic Music and Media Arts](#) programme at University of Pécs, Faculty of Arts. Founder of the [Hangfarm](#) (Soundfarm) open-air media art exhibition place in Ellend, Hungary.

5. REFERENCE

- [1] Kovács, B. 2019. Introducing Makkeróni. *International Conference on Live Coding*. <http://iclc.livecodenetwork.org/2019/papers/paper62.pdf>

Web Audio API vs. native, closing the gap, take 2

[Extended Abstract]

Paul Adenot
Mozilla Corporation
padenot@mozilla.com

1. TOPICS

My keynote at the first Web Audio Conference at IRCAM in Paris attempted to answer this question, but it was clearly too early in the lifetime of the Web Audio API to have a definitive answer.

The situation has evolved quite a lot, with the addition of various features, not necessarily related to audio.

Shared memory, atomics, SIMD, Audio Worklets, WASM, Web MIDI are some of the features that allow building programs that were impossible to write a few years ago.

This workshop will take a bottom up approach to writing high-performance applications with the Web Audio API, with a definitive focus towards writing real-time audio code in the context of a web application.

In doing this, the reasoning will follow a bottom-up approach: understanding the properties needed for a specific system, and try to them map web platform constructs, with a definitive focus on high-performance and white-box analysis. Links to implementations themselves and the primitives chosen will have an impact on the final quality of the result, in terms of rendering speed, memory footprint, robustness, and extensibility.

The limitations of the web platform will be discussed, with possible workarounds, along with the multiple efforts are in the works to remove those limitations.

1.1 Audio Worklets

AudioWorklet is the back-bone of the real-time audio processing on the web, outside of the pre-defined **AudioNode** that have been available for a long time. It provides a way to execute script as part of the rendering of the audio, with a corresponding main thread object that has custom **AudioParam** exposed, along with a **MessagePort**.

The rendering thread side has the other end of the **MessagePort**, and must have a method called **process** with

a number of parameters, where all the signal computations happen. This method received the input of the node (if applicable), has a buffer where the output signal can be written, and has a third buffer with the value of the **AudioParam**.

1.2 WASM

WASM (Web Assembly) plays a very important role in moving towards native-like performances, because it allows using a language that does not use a garbage collector. Because the audio latencies on modern platforms are very low, blocking the audio rendering thread, even what seems to be short periods of time, is problematic.

Additionally, it executes most of the time much faster than JavaScript, and allows authoring the signal processing algorithm in languages that are better suited to the task at hand (C++, Rust, Faust, etc.).

1.3 Shared memory

Shared memory is now available on the web, with the object **SharedArrayBuffer**. It can be transferred via a **MessagePort**, and then its content can be mutated rapidly on multiple threads. Shared memory is a powerful tool, but authors should have a certain discipline when using it. Special care must be taken when using it, from a software engineering but also performance standpoint.

This construct has quite a few security implications, that will be quickly discussed, and good practices for deployment of programs using shared memory will be provided.

1.4 Web Workers

Web Worker opens the door to multi-thread computing and concurrency. This construct has been available for a long time on the web platform, but it has been made more powerful recently thanks to the availability of shared memory.

This construct will be looked at from an operating system perspective, to understand in which circumstances it is possible and feasible to offload some processing to a Web Worker, and why.

1.5 Atomics

Atomics, along with ‘**SharedArrayBuffer**’, allow implementing lock and wait-free algorithms on the web platform, that are essential for audio. In particular, the wait-free single-producer single-consumer ring-buffer (the bread and butter of audio programming), will be investigated.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

1.6 SIMD

SIMD (Single Instruction Multiple Data) is a way to optimize computation, on a single thread, via operating on multiple scalar values at once. Often, algorithms need to be tweaked to make efficient use of SIMD. A couple approaches useful to deal with the specifics of audio algorithm will be explained.

1.7 Web MIDI

Web MIDI is essential to access external hardware: control surfaces, keyboard for input, analog synthesizers and other drum machine on the output. Special care must be taken when integrating Web MIDI and the Web Audio API, in particular when it comes to timing.

2. SYNOPSIS

A straightforward signal processing algorithm has been chosen for illustration purposes, and a number of signal processing primitives are provided in languages that are suited to the task at hand. The first task is to have a working effect, by implementing the glue code, compiling the result to WASM, and to write the JavaScript code to load the WASM module in an `AudioWorklet`.

The effect is now working but no controls are provided. Control of the effect via either `AudioParams` or shared memory and atomics is implemented, along with potential optimizations via SIMD, and performance measurements performed. A discussion starts on what contributes to high-performance algorithms and on how to measure performance of a real-time algorithm on the web platform and in general, based on the findings of the group.

Control via Web MIDI or other means (OSC using WebSockets or WebRTC) to integrate with other systems will be implemented. Packaging into a standard format (Web Audio Modules) is discussed.

With a sort of gap analysis, the participants that have prior knowledge developing native audio processing algorithms are reflect on what the web platform provides, and what is missing. Possible solutions are explored, with pointers to existing web platform efforts (discussions, events, issues opened on specifications, other specifications, etc.).

Diversity Workshop – WAC 2019

Andrea Hegdahl Tiltne
Girl Geek Dinner Trondheim
Trondheim
Andrea.tiltne@ntnu.no

Miranda Moen
NTNU alumni
Oslo
mirandamoen@icloud.com

ABSTRACT

Two Norwegian organizations working with equality and diversity in technology and the arts sector cooperate with this workshop on diversity. We will present our experiences and work as well as offer some tools to stimulate and achieve diversity in the workplace and in communities such as Web Audio. The workshop will thus have two modules; (1) one first module by The Art of Balance and (2) one second module by Girls Geek Dinner. The workshops will be held in English. Previous knowledge of diversity and other equality concepts will not be necessary.

1. WORKSHOP MODULES

1.1 The Art of Balance: Challenge the norms!

The Art of Balance (*Balansekunst*) is a Norwegian association of 80 organizations, companies, festivals and other arts and culture institutions that engage in promoting gender equality and diversity in the arts. In this part of the workshop, Victoria Øverby Steinland from Balansekunst will talk about how *norms* is a keyword to diversity work.

In order to create room for diversity, it is necessary to identify the barriers that may prevent some people from participating. One way to address this issue is by the means of *norm-critical perspectives*. With a norm critical approach, we bring attention to the norms that shape society and our ways of thinking.

In this workshop we will talk about how norms operate in society. We will touch on topics such as gender, sexuality, ethnicity and (dis)ability and investigate whether implicit norms hinder diversity in the field of arts and culture. If so, what do we do about it? The workshop will vary between presentation, exercises and discussion, with the aim of an interesting conversation.

The everyday norms such as ways of greeting or organizing a line at the grocery store are all fine and well, but what about society's norms relating to our identities? Norms determine what and who society deems normal, and consequently what and who is considered less normal, different or strange. Being labeled as different or 'not normal' is unfair in itself. Moreover, when being perceived as different, one is more vulnerable to discrimination. A norm-critical approach seeks to tackle issues of discrimination and exclusion, by identifying and challenging the underlying norms.

An important point is that no person is ordinary or different in its

own right. Rather, it is the norms in society that draw lines which exclude people. On the upside, norms are produced and negotiated by people in society, and thus, we can do something about them. The first step is to identify and talk about them.

Norm-critical perspectives, as a concept and an approach, originates from Sweden, drawing on influences from international feminist and queer studies and critical pedagogy [1][2][3].

1.2 Girls Geek Dinner Trondheim

The second part of the workshop will be conducted by Girls Geek Dinner (GGD) Trondheim. Girl Geek Dinners Trondheim is a place to meet for women who are interested in technology. We have had events on a broad field of different technologies hosted by very different companies and organizations. We started up in finance, energy, music, health, sports and construction to mention a few.

Girl Geek Dinners was started in 2005 by Sarah Lamb in London and now have chapters in almost 100 different places around the world. All our events are free and open for everyone.

Kari Dahn works as a user experience consultant in Bouvet and has been active in Girl Geek Dinners since the network started up in Trondheim. In March 2019 she suffered a heavy viral ear infection, that resulted in a mild/moderate permanent hearing loss, tinnitus, and a distorted pitch perception. Kari will share with us how getting such a sudden and complex hearing disability affected her; in daily communication, and in her interest in music. Her aim is to foster more understanding about how life with hearing impairments can be, and the need for universal design, both from a personal and professional perspective.

1.3 Speakers' bios

Victoria Øverby Steinland (The Art of Balance) is developing Balansekunst's project against sexual harassment within culture and the arts. She holds an MSc in sociology and has previously worked with projects on gender and sexual diversity, as well as teaching adolescents about sex, boundaries and consent.

Kari Dahn (Girl Geek Dinner Trondheim) works as a user experience consultant in Bouvet and has been active in Girl Geek Dinners since the network started up in Trondheim. Dahn holds a master's degree in Information design

1.4 Workshop organizers

Miranda Moen: Miranda Moen is educated in aesthetics (University of Bergen), arts management (OsloMet), and completed a master's degree in Equality and Diversity at NTNU June 2019. Affiliated with KOSO, the Art of Balance and AKKS.

Andrea Hegdahl Tiltne: Andrea Hegdahl Tiltne is educated journalist (Nord University) and worked as a news reporter for some years after finishing her studies. Since 2014 she has worked as Communications Advisor at NTNU.



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

2. REFERENCES

- [1] Bromseth, Janne and Renita Sörensdotter. 2012. Norm-Critical pedagogy. An opportunity for a change in teaching, in *Gender Studies Education and Pedagogy*. Swedish Secretariat for Gender Research.
- [2] Kumashiro, Kevin. 2002. *Troubling Education. "Queer" Activism and Anti-Oppressive Pedagogy*. Routledge.
- [3] The Living History Forum and RFSL Ungdom. 2009. *Break the norm! Methods for studying norms in general and the heteronorm in particular*

Audiovisual Programming in Live Performance

Charles Roberts

Interactive Media & Game Development Program
Department of Computer Science
Worcester Polytechnic Institute
charlie@charlie-roberts.com

ABSTRACT

In this workshop, participants will explore a variety of live coding technologies (including Gibber, Hydra, and the Force, among others) and discuss different approaches to multimodal programming / audiovisual synchronization with web technologies.

1. WORKSHOP NOTES

In many ways, the browser is an ideal vehicle for audiovisual performance. In addition to affordances for low-level audio programming, it also features a wide variety of approaches for graphics programming, including its `canvas` element, the Scalable Vector Graphics (SVG) format, and WebGL. These are all conveniently united under a single programming language, JavaScript, that is also easily targetable by domain-specific languages for multimedia performances.

Given these powerful features, it is natural to think about combining these technologies to create rich audiovisual experiences. This workshop will explore techniques for doing so, ranging from how to manage the analysis node of the Web Audio API to drive graphical parameters, to examining different techniques that various visual/audiovisual environments use to synchronize and map between the audio and visual domains. We'll specifically be looking at browser-based systems for live coding performance, including (but not limited to) The Force [1], `marching.js` [2], Hydra¹, and Gibber [3].

Along the way we'll also discuss the aesthetics of audiovisual compositions and watch examples of how audiovisual artists have dealt with the topic of multimodal mappings, ranging from impressionistic approaches to more literal techniques such as audification. Ideally, workshop participants will leave with a new set of tools to explore, some background knowledge on the techniques used to create expressive mappings, and having discussed interesting works exploring audiovisual composition and performance.

2. FACILITATOR BIO

I am an Assistant Professor in the Department of Computer Science at the Worcester Polytechnic Institute, with an affiliation in the Interactive Media and Game Development program. I research human-centered computing in digital arts practice, and am the lead designer and developer of Gibber, an open-source, creative coding environment for the browser. I have given live coding performance using Gibber throughout North America, Europe, and Asia.

3. REFERENCES

- [1] S. Lawson and R. R. Smith. The dark side. In *Centro Mexicano para la Música y las Artes Sonoras (Mexico): Proceedings of the Third International Conference on Live Coding*, 2017.
- [2] C. Roberts. Live Coding Ray Marchers with `Marching.js`. In *Proceedings of the Fourth International Conference on Live Coding*, 2019.
- [3] C. Roberts, M. Wright, J. Kuchera-Morin, and T. Höllerer. Gibber: Abstractions for creative multimedia programming. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 67–76. ACM, 2014.

¹<http://hydra-editor.glitch.me/>



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

© 2019 Copyright held by the owner/author(s).

