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 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] and 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.

1 Heroku https://www.heroku.com/
2 Node.js https://nodejs.org/en/
3 Socket.IO library website https://socket.io
4 WebAudio API https://www.w3.org/TR/webaudio/
5 WebGL API https://www.khronos.org/webgl/

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: portfinder10 - a port scanner on the current machine; osc-js11 package - to use the Open Sound Control

1 http://www.bragamediaarts.com/pt/
2 https://github.com/luisArandas/akson
3 NPM https://www.npmjs.com/
4 Express https://expressjs.com/
5 portfinder https://www.npmjs.com/package/portfinder
6 osc-js https://www.npmjs.com/package/osc-js
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 WebMIDI API\(^\text{17}\) 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\(^\text{18}\) library does an abstraction over the input navigator.requestMIDIAccess and we integrated it alongside WebMidi.js\(^\text{19}\), 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\(^\text{10}\) (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\(^\text{20}\) 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.

```javascript
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(); cAdonian(); cHungarianMajor(); cHirajoshi(); cIonian() and cDorian(). 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.

---

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\(^\text{32}\). 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: jQuery\(^\text{13}\) - for the use of its functions for document processing; The NexusUI\(^\text{14}\) project [2] - an open-source collection of HTML5 interfaces and JS helper functions to assist with building web audio instruments in the browser; the WUI\(^\text{15}\) 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 Bootstrap\(^\text{16}\) collection. A responsive and popular component library for quick prototyping.

---

\(^{12}\) winston https://www.npmjs.com/package/winston
\(^{13}\) jQuery https://jquery.com
\(^{14}\) Nexus https://nexus-js.github.io/ui/
\(^{15}\) WUI https://github.com/grz0zrg/wui
\(^{16}\) Bootstrap https://getbootstrap.com

\(^{17}\) WebMIDI https://webaudio.github.io/web-midi-api/
\(^{18}\) SimpleMidiInput.js https://github.com/kchapelier/SimpleMidiInput.js
\(^{19}\) WebMidi.js https://github.com/djipco/webmidi
\(^{20}\) Tone.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 package is a recent example of that mathematical implementation, and also the foswig.js 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 ([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.

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 AWS; Microsoft Azure) 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

\[\text{js-markov} \text{ https://www.npmjs.com/package/js-markov} \]
\[\text{foswig.js} \text{ https://github.com/mrsharpoblunto/foswig.js} \]
\[\text{three.js} \text{ https://threejs.org/}\]

---

21 js-markov https://www.npmjs.com/package/js-markov
22 forwig.js https://github.com/mrsharpoblunto/foswig.js
23 Three.js https://threejs.org/

24 Amazon AWS https://aws.amazon.com
25 Microsoft 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.](image)

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 its context, part of UNESCO’s Creative Cities Network (UCCN)27.

11. REFERENCES


26 UNESCO’s European Website https://en.unesco.org
27 UCCN https://en.unesco.org/creative-cities/home


