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 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.

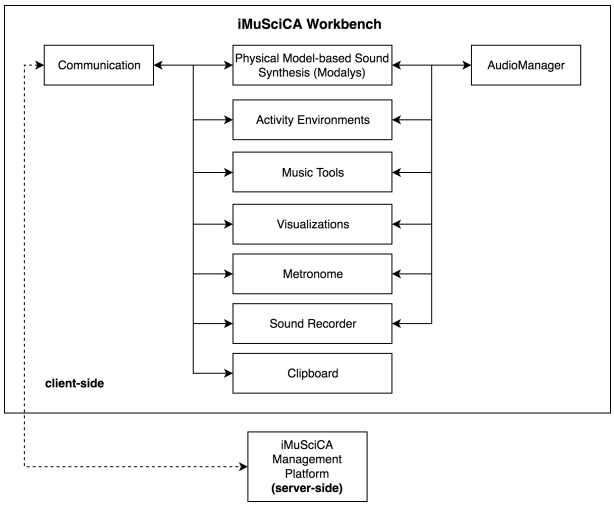


Figure 1: General overview of the system architecture.

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 Workbench[1] 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/

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

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 Manger of the iMuSciCA Workbench, create a transmitting node, and finally share it with the central Audio Manger by calling the following function.

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

[2]https://github.com/postaljs/postal.js

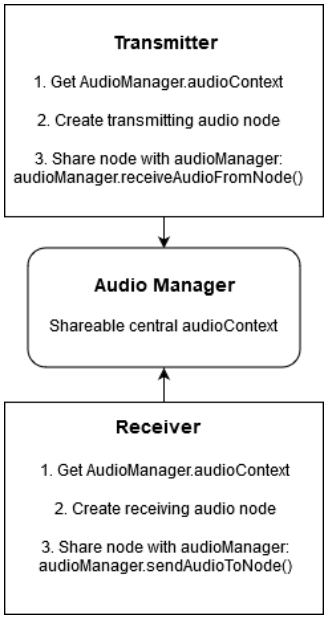


Figure 2: The Audio Manager framework.

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.

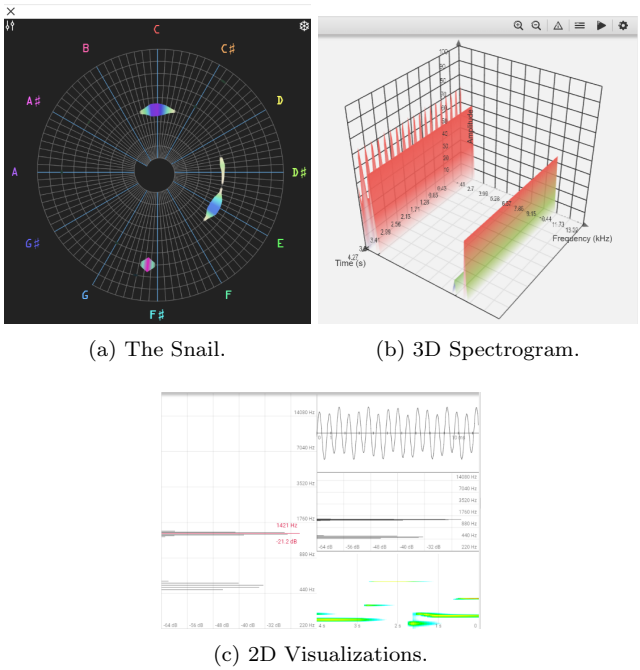
2.6.1 Snail

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/



(a) The Snail. (b) 3D Spectrogram.

(c) 2D Visualizations.

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

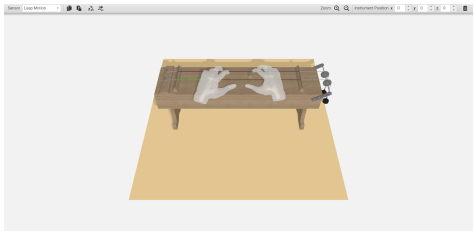
The iMuSciCA workbench provides nine AEs in total. However in this section we focus only on those environments that support musical activities, including the Musical Whiteboard, Performance Sampler, Tone Synthesizer, 3D Instrument Design and Interaction environments as well as the Acouscope.

2.7.1 The Musical Whiteboard

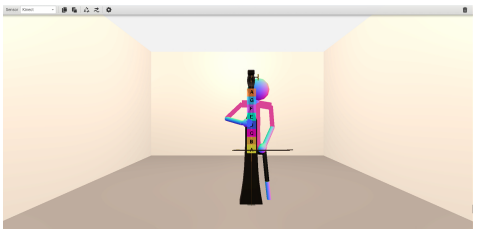
The Musical Whiteboard (MW) is a web-based environment that enables the free drawing of music on touch-enabled computers [2]. The x-axis represents time and the y-axis is mapped to the frequency domain, that is displayed on the right in Hertz with a correspondence in notes on the left (see Figure 4f). 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

(a) Performance Sampler.

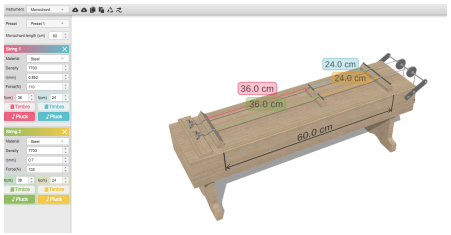
(b) Tone Synthesizer.



(c) Leap Motion Interaction Environment.



(d) Kinect Interaction Environment.



(e) 3D Virtual Instrument Design.

(f) The Musical Whiteboard.

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-enable web environment. The server executable is available online[10]. 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

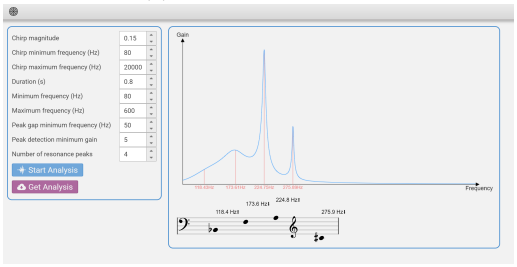
The Acouscope Environment employs a hardware device called HyVibe[11], that uses state-of-the-art actuation technology for identifying the frequency response of any surface. As it is presented in Figure 5a, the hardware comprises of two transducers: a) an electric actuator that is used for applying force 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) Acouscope Web Interface.

Figure 5: The Acouscope 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 lstms 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.