Interactive Music Science Collaborative Activities

Team Teaching for STEAM Education

Deliverable 4.11

Final Version of Music Visualization and Analysis Tools

Date:	17/07/2018
Author(s):	Jean-Marie Laborde (CABRI), Aggelos Gkiokas (ATHENA), Maximos Kaliakatsos-Papakostas (ATHENA), Manuel Bouillon (UNIFRI), Robert Piechaud (IRCAM), Quentin Lamérand (IRCAM), Marcus Liwicki (UNIFRI)
Contributor(s):	
Quality Assuror(s):	Robert Piechaud (IRCAM), Carlos Acosta (LEOPOLY)
Dissemination level:	PU
Work package	WP4 – Core enabling technologies modules of iMuSciCA
Version:	1.0
Keywords:	music analysis tools, music visualization tools
Description:	Final version of the music analysis and visualization tools.

Horizon 2020
European Union Funding
for Research & Innovation

H2020-ICT-22-2016 Technologies for Learning and Skills
iMuSciCA (Interactive Music Science Collaborative Activities)
Project No. 731861
Project Runtime: January 2017 – June 2019

Executive Summary

In this deliverable we present the final version of the music analysis and visualization tools with a list of their updates. For tools description one may refer to the first version in deliverable D4.5.

The demonstrators of these tools are given as standalone web browser applications, loaded from the servers of partners who are mainly responsible for their development (links are given below). These tools can be grouped into the following categories:

- **Basic Core-enabling Musical Technologies:** this category encompasses tools that enable basic musical actions like counting time (Metronome), tuning an instrument (Tuner or Reference Pitch) and record audio (Audio Recorder).
- **Music, Sound and Basic Sound Visualisation:** tools in this category allow students to explore musical ideas by drawing using different timbres (MuSoVis), watch basic visualisations (including spectrogram, and frequency response) and experiment with additive synthesis and visualise the waveform plot along with also the mathematical expressions that produce the sound.
- **Advanced Visualisation tools:** this category includes i) the Snail tool which features a musically-informed audio visualisation ii) the 3D Spectrogram tools which employ advanced signal processing for presenting visualisations relevant to rhythm and timbre.
- **Mathematics and Geometry Visualisation Tools**: the set of tools included in Cabri Express that allows numerical calculation, graph visualisations and geometry constructions using various mathematical modules.

List of available online tools

Music Tools

Reference Pitch:

http://tesla.ilsp.gr:1994/tools/v3/iFrames/pitcha.html

Metronome:

http://tesla.ilsp.gr:1994/tools/v3/iFrames/metronome.html

All tools (including the Tuner and the Audio Recorder):

http://tesla.ilsp.gr:1994/tools/v3/

Basic Visualization Tools

UNIFRI Sound Analyzer:

https://unifri.imuscica.eu/drawme/analyser.html

Sinusoids:

http://tesla.ilsp.gr:1994/sinusoids.1/

Advanced Visualization Tools

3D Spectrogram:

https://athena.imuscica.eu/visualizations/spectrogram/stable/demo.html

Snail:

http://devtest.leopoly.com/3d-interaction-repository-V1/modalys/public/snailtest.html

Mathematics and Geometry

CabriExpress:

https://cabri.imuscica.eu/cabriexpress4imuscica

Version Log			
Date	**Version No.**	**Author**	**Change**
25-06-2018	0.1	Laurent Varin (CABRI)	Initial content
03-07-2018	0.2	Other authors	Chapters contributions
10-07-2018	0.3	Laurent Varin (CABRI)	Small updates and submit it to reviewers
13-07-2018	1.0	Laurent Varin (CABRI), Manuel Bouillon (UNIFRI), Aggelos Gkiokas (ATHENA)	Final revision
17-07-2018	1.0	Vassilis Katsouros (ATHENA)	Submission to the EU

Disclaimer

This document contains description of the iMuSciCA project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of iMuSciCA consortium and can in no way be taken to reflect the views of the European Union.

iMuSciCA is an H2020 project funded by the European Union.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Description
MuSoVis	Music and Sound Visualisation
FFT	Fast Fourier Transform
WebGL	Web Graphic Library
CSS	Cascading Style Sheet
STEAM	Science, Technology, Engineering and Maths combined with Arts
WP	Work Package
PU	Public document
ATHENA	ATHENA RESEARCH AND INNOVATION CENTER IN INFORMATION COMMUNICATION & KNOWLEDGE TECHNOLOGIES
UCLL	UC LIMBURG
EA	ELLINOGERMANIKI AGOGI SCHOLI PANAGEA SAVVA AE
IRCAM	INSTITUT DE RECHERCHE ET DE COORDINATION ACOUSTIQUE MUSIQUE
LEOPOLY	3D FOR ALL SZAMITASTECHNIKAI FEJLESZTO KFT
CABRI	Cabrilog SAS
WIRIS	MATHS FOR MORE SL
UNIFRI	UNIVERSITE DE FRIBOURG

1. Introduction

The iMuSciCA learning environments should allow easy and intuitive access to core-enabling technologies around music and visualisations. To this end, music/audio and mathematical visualisation tools have been developed along with basic music-related tools that will allow students to obtain intuitive information about sound, mathematical equations and geometric shapes through visualisations, while basic tools will facilitate music training (e.g. metronome) and ear training (e.g. pitch reference tool). These core-enabling technologies are in accordance with the purposes of STEAM education and follow a simple and intuitive design approach, as demonstrated by the urls provided in the following section.

2. Basic Core-Enabling Musical Technologies: Metronome, Tuner, Audio Recording and Reference Pitch

2.1 Installation and technical requirements

The core enabling technologies that are incorporated in the basic musical tools are demonstrated in the following link (except from the Audio Recording Tool which requires audio input from the workbench and can be therefore demonstrated only on the workbench):

http://tesla.ilsp.gr:1994/tools/v3/

The link given above can be opened by any web browser in any operating system. Since those tools allow the generation and capturing of audio, speakers and a microphone are required for using/testing them. For demonstration purposes, this web page includes all the tools pertaining to the basic musical core enabling technologies. The tools will be available in separate iframes embedded in the workbench and they can be made available to use upon user request. After the description of each tool, a dedicated URL is given for each tool for opening each tool individually.

2.2 Description of changes in the final version

Shortly, (and since the description is similar to the first version of the tools in Deliverable 4.5) the core-enabling technologies in these tools allow students to perform the following tasks:

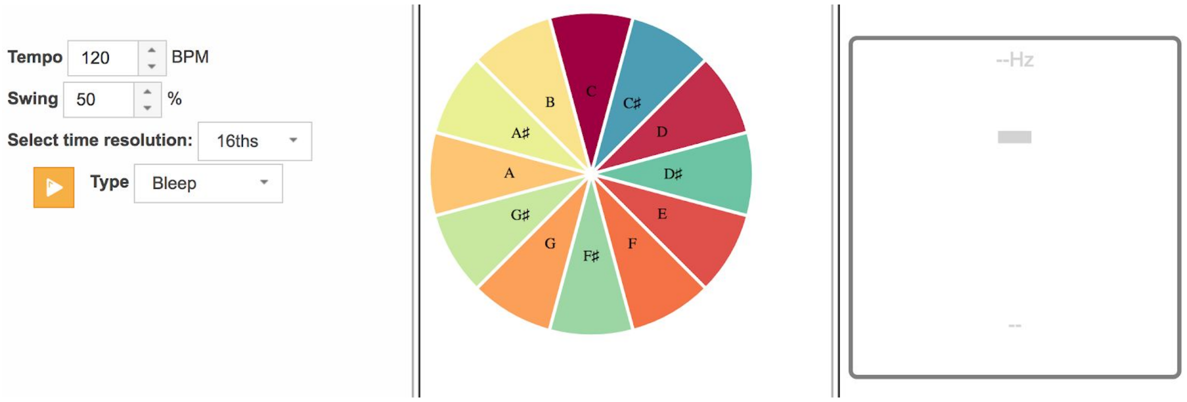
- Use the metronome for keeping a steady time during musical performance (i.e. during free or gesture-based interaction with the instrument),
- Use the reference pitch tool to listen to a reference pitch for, e.g., tuning the instrument during the instrument design phase.
- Use the tuner to tune the instrument during instrument design in cent-level detail.

- Record the produced audio from all sound-producing tools in the iMuSciCA workbench, except from the metronome, and save it either to audio file or on the cloud.

It should be noted that the audio file playback tool has been removed since it overlapped with the Performance Sample Sequencer tool.

Figure 2.2.1 illustrates the final version the tools pertaining to the basic musical core-enabling technologies (this figure is a screenshot of the web page given in the link in the previous section). Even though the appearance of the tools is subject to changes, the functionalities are in final version. A short description of those tools is given in Deliverable D4.5; the following points focus on the changes in the final version:

1. Metronome: The user can now select between two audio schemes for the metronome: "Bleep", which produces short pure sine wave sounds (as in the first version) and "Tick" which produces wooden stick sounds. Dedicated URL: http://tesla.ilsp.gr:1994/tools/v3/iFrames/metronome.html
2. Reference Pitch: In this version the tool has been circularly shifted counter clockwise by some degrees to become aligned with the Snail visualisation. Additionally, the clicking distance from centre defines the octave of the note to be produced. Dedicated URL: http://tesla.ilsp.gr:1994/tools/v3/iFrames/pitcha.html
3. Tuner: Similar with the first version but with decreased sensitivity. Please use the URL demonstrating all the tools for examining the tuner: http://tesla.ilsp.gr:1994/tools/v3/
4. Audio recording: In comparison with the first version, this tool now offers the possibility not only to record audio, but also to upload the recording to clipboard, save the recording to audio file, import and playback recording from clipboard and load audio clips from the user files. Please use the URL of the entire workbench to test the audio recording tool.



Tempo 120 BPM

Swing 50 %

Select time resolution: 16ths

Type Bleep

Figure 2.2.1: Final version of the basic musical core-enabling technologies.

3. UNIFRI MuSoVis (Music and Sound Visualization)

UniFri MuSoVis is a simple and yet powerful music and sound visualization tool.

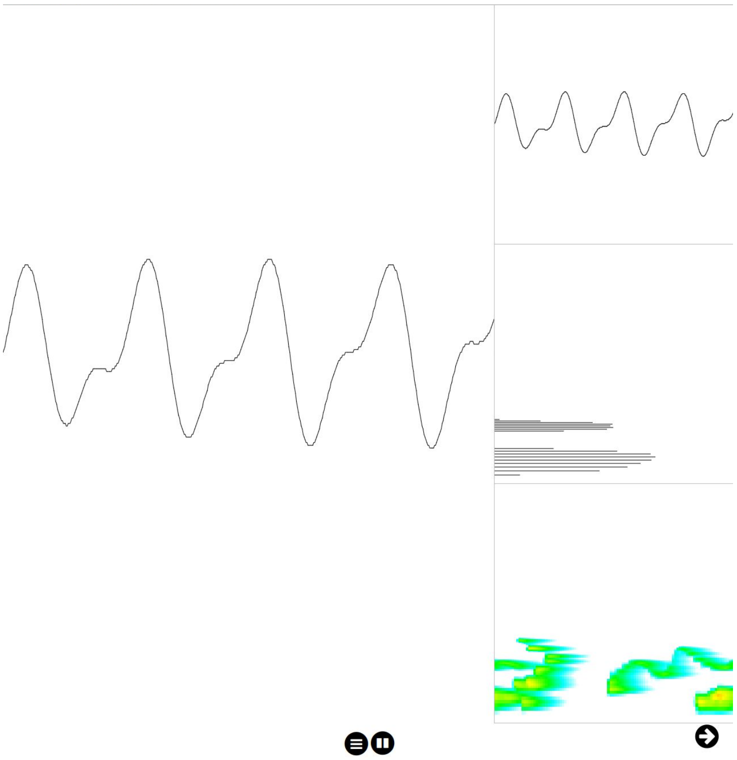


Figure 3.1. *UniFri MuSoVis with the main view in the middle, the control buttons at the bottom and the three possible visualization views on the right side.*

3.1 Installation and technical requirements

UniFri MuSoVis is based on JavaScript, so it runs on any modern browser (Chrome, Firefox, Edge, etc.) and operating system (Windows, GNU/Linux, macOS) and needs a microphone. It is accessible at the following URL: https://unifri.imuscica.eu/drawme/analyser.html. You can insert UniFri MuSoVis in an iframe with the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang=en>
<head>
```

```
    <meta charset=utf-8>
    <title>UniFri MuSoVis</title>
</head>
<body style="margin:0px;padding:0px;overflow:hidden">
       <!-- DAME iframe -->
       <iframe     id="tool"     src="https://unifri.imuscica.eu/drawme/v1/analyser.html"
frameborder="0" height="100%" width="100%"></iframe>
</body>
</html>
```

3.2 Description of demonstrator and user manual

UniFri MuSoVis is composed of three parts, the drawing canvas in the middle, the control buttons at the bottom and the visualization views on the right.

3.2.1 Main view

The user can visualize sound and music in the main view, which can be either of:

- A waveform view
- A Fourier transform view
- A spectrogram view

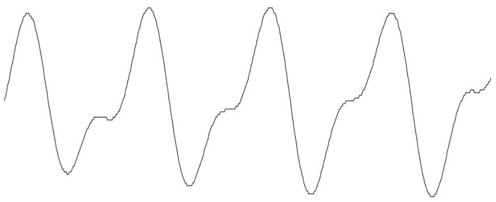


Figure 3.2.1.1. *The main view displaying the waveform view.*

3.2.2 Control buttons and settings

Below the main view, the control buttons enable to select the sound source between:

- UniFri DrAwME (See Deliverable 4.2)
    - For this one, just open DrAwME and click on the right panel
    - Works best open the two side by side
- The user microphone
    - Click on the microphone button and the sound of the microphone will be visualized

The pause/play button enable to freeze the current view and then export it in JSON format.

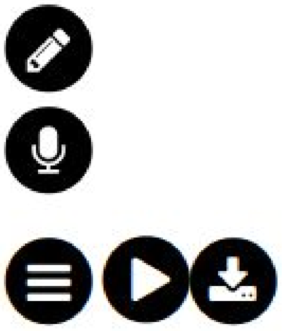


Figure 3.2.2.1. *The control buttons with the source selection on the left, the pause/play button in the middle and the export option on the right.*

3.2.1 Miniature panel

The miniature panel on the right display small previews of the three different views:

- The waveform view;
- The Fourier transform view;
- The spectrogram view.

Each of the three view can be display in the main view by clicking on it. The miniature panel can be hidden by clicking on the arrow at the bottom right.

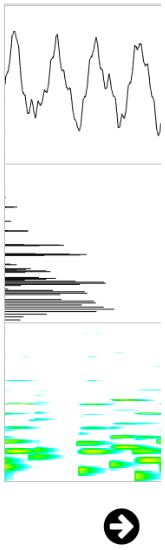


Figure 3.2.1.1. *The visualization panel include three different views: a waveform, a Fourier transform and a spectrogram. The visualization panel can be hidden by clicking on the arrow on the bottom right.*

4. A Core-Enabling Technology for Studying Timbre: the Additive Synthesis Tool

4.1 Installation and technical requirements

The Additive Synthesis tool constitutes a core-enabling technology that allows students to explore timbres as an addition of multiple sinusoidal elements. The main description of this tool is given in Deliverable 4.5 and it can be found in the following link:

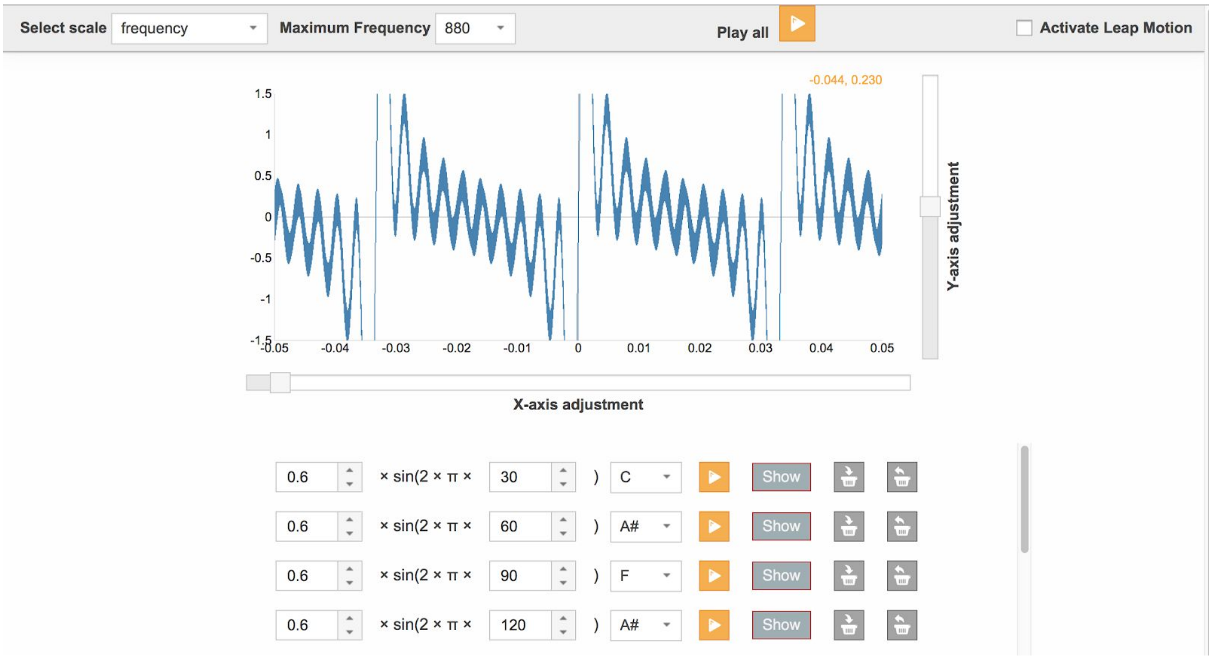
http://tesla.ilsp.gr:1994/sinusoids/v3/

The link given above can be opened by any web browser in any operating system. Since this tool generates audio, speakers are required for using/testing it.

4.2 Description of changes in the final version

The elements that have changed from the first version of the tool (described in Deliverable 4.5) are the following:

1. Users are given the option to reproduce frequencies that correspond to given scales (no scale or ‘frequency’, chromatic, C major, minor, minor pentatonic and symmetric).
2. Users can control the frequencies and amplitudes of up to 10 sinusoids using their respective extended fingers, as captured by the Leap Motion sensor.
3. The users can copy to the clipboard the setup of any sinusoid element and paste it either to any other tool that receives sinusoid objects (e.g. the sonification tool), or import from clipboard a sinusoid element.
4. The user can select to play or pause all elements.



Select scale frequency

Maximum Frequency 880

☐ Activate Leap Motion

0.6 × sin(2 × π × 30) C

0.6 × sin(2 × π × 60) A#

0.6 × sin(2 × π × 90) F

0.6 × sin(2 × π × 120) A#

Figure 4.2.1: Final version of the Additive Synthesis core-enabling technology.

5. The Snail: an advanced music visualization tool based on spectral analysis

Snail.js is the browser version of the standalone application "TheSnail" developed by Ircam[1].
The key concept of The Snail, unlike classical sound representations such as spectrogram or sonogram, is to display the spectrum of a sound in a very musical fashion, that is, on a spiral featuring notes (equally tempered) and octaves. Each component of a sound - or *partial* according to the Fourier theory[2], is displayed as a "blob" which size and color depend on the energy and phase shift in the Fourier decomposition. With this representation, which evolves very dynamically over time, anyone with a little or no scientific background can immediately "see" the sound and its spectral structure, as it is being produced.

5.1 Installation and technical requirements

The Snail is a delivered as a javascript library - **snail.js + snail.wasm** - that can be instantiated anywhere inside a HTML5 browser page. Very much like modalys.js, snail.js is compiled from C\C++ using Emscripten[3] using heavy optimizations. The FFT used behind the hood is pffft[4], an highly optimized open source fft library written by Julien Pommier. Snail.js will run in Chrome, Firefox, Edge or Opera. snail.js has dependency to multirange.js for one the settings control.
It comes with a CSS (**snail.css**) and font to be displayed correctly. This can of course be customized.

[1] http://www.ircamlab.com/products/p2242-The-Snail/
[2] https://en.wikipedia.org/wiki/Fourier_transform
[3] http://s3amdev.ircam.fr/
[4] https://bitbucket.org/jpommier/pffft

As the WebAssembly is loaded asynchronously, a promise is used to know when it's possible to instantiate a Snail.

A snail object is typically created passing these parameters:

- The div that will embed the snail
- The audio context
- An option object for the settings

Finally, an audio source (modalys.js, the microphone, an existing sound file etc.) can be connected to the snail for analysis and visualization and an output node if you want the sound to be heard after the Snail. By default, the microphone and an audio player are connected to the Snail, but it can be configured.

Here is a minimal example of instantiation:

```
<!DOCTYPE html>
<html lang="en">
<head>
  [...]
  <link rel="stylesheet" type="text/css" href="css/multirange.css">
  <link rel="stylesheet" type="text/css" href="css/snail.css">

  <script src="script/multirange.js"></script>
  <script src="script/snail.js"></script>
</head>

<body>
  <div id="snail"></div>
  <script>
    var audioContext = new (window.AudioContext || window.webkitAudioContext)();
    Snail.isready.then(function () {
      Var snail = new Snail(document.getElementById("snail"), audioContext);
      snail.output.connect(audioContext.destination);
    });
  </script>
</body>
</html>
```

5.2 Description of the prototype demonstrator

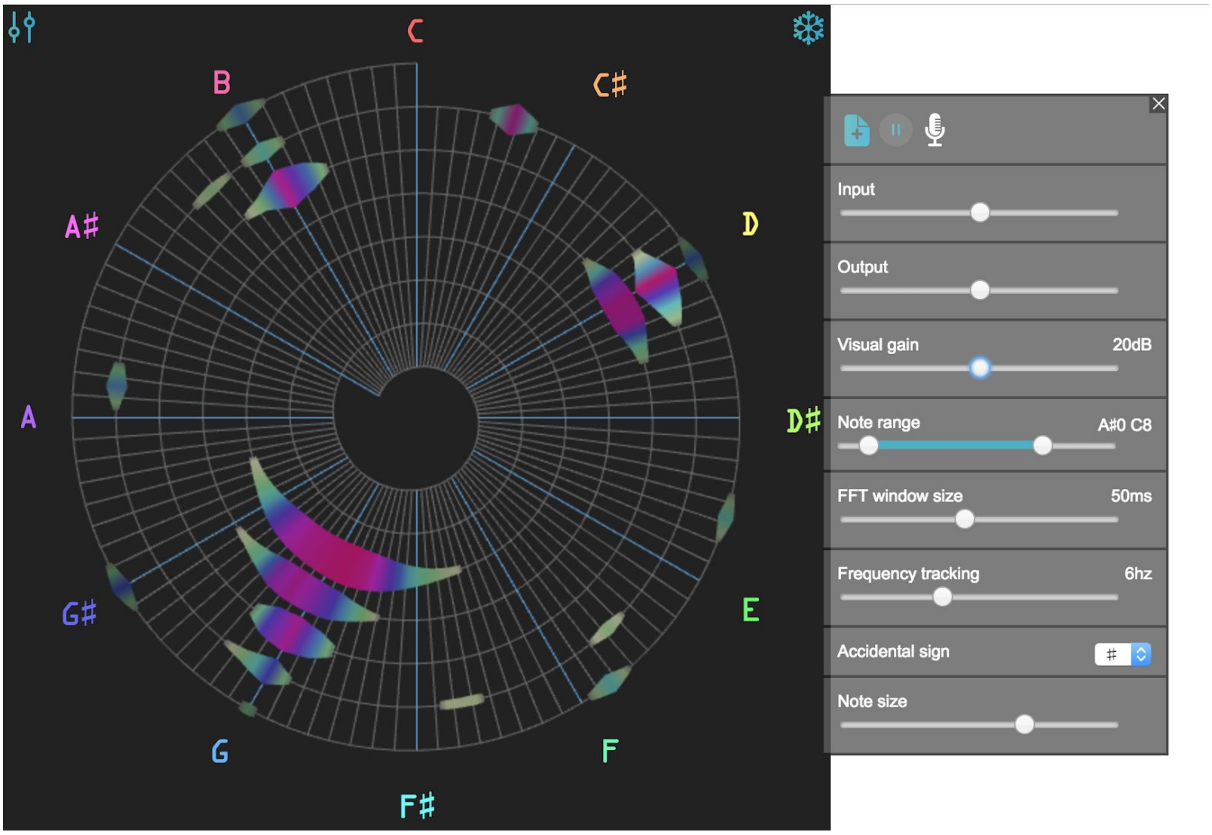
The standalone prototype is hosted at this url:
https://s3amdev.ircam.fr/snailtest.html

The Snail demonstration page features the following:

- The Snail area per se, drawn in pure WebGL
- A button for opening an existing sound file[5]
- The Note Range slider pair
- The FFT window size slider (default is 50 ms)

[5] Beware that certain browsers accept certain formats only. For instance, Firefox will accept .ogg or .wav only.

- The frequency tracking slide (default is 6 hz).
- The note name popup menu (English #/b, French #/b, Greek #/b).
- The Snail global size (in pixels). Everything is sized according to this value.
- The note font size, as in CSS code.



Input

Output

Visual gain 20dB

Note range A#0 C8

FFT window size 50ms

Frequency tracking 6hz

Accidental sign #

Note size

Figure 5.2.1: Interface of the web browser version of the Snail visualisation tool.

5.2.1 Advanced parameters

All of the parameters (like the FFT window size or the frequency tracking bandwidth) default values can be configured via the settings object. They can be changed at run time via the API or UI controls (that use the same setters). All or part of the settings can be configured or not as there is a fallback value for each parameter. Here are the default values, if none is overridden.

```
{
  "bufferSize": 1024,
  "snailSize": 601,
  "noteSize": 0.68,
  "noteMinMidi": 10,
  "noteMaxMidi": 127,
  "fftWindowSize": 50,
  "frequencyTracking": 6,
  "showControls": false,
  "disableControls": false,
  "disableFreeze": false,
  "controls": [
    "inputs",
    "inputGain",
    "outputGain",
    "visualGain",
    "noteRange",
    "windowSize",
    "frequencyTracking",
    "flat/sharp",
    "snailSize",
    "noteSize"
  ],
  "averageProcessTimes": 16,
  "displayedProcessTimes": 8,
  "averageGLTimes": 16,
  "displayedGLTimes": 8,
  "inputs": [
    "player",
    "microphone"
  ],
  "inputGain": 1,
  "outputGain": 0,
  "visualGain": 0,
  "controlsDraggable": false,
  "accidentalSign": "flat", // other possible value is "sharp"
  "language": "en"
}
```

5.2.2 Language selection

UI language can be set using the settings object ("language") or the corresponding setter (setLanguage()). It can also be overridden by adding the *lang* request parameter to the url.
For example, https://s3amdev.ircam.fr/snailtest.html?lang=fr will display the Snail example in french.

5.3 Performances

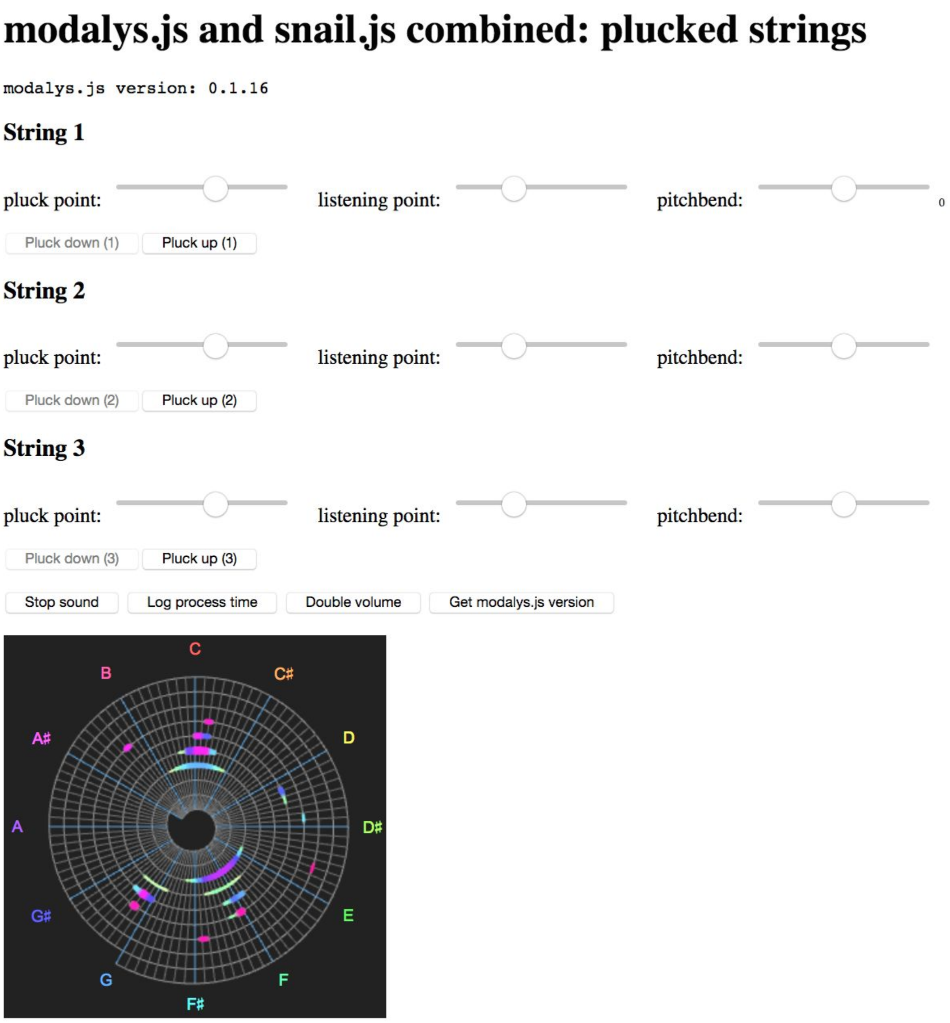
Snail.js features pretty decent performances. The average audio processing time for a buffer of 1024 is around 1ms and 0.5ms for the WebGL process, when running Google Chrome on a 2014 Macbook Pro.

5.4 Example of a Snail connection: with Modalys

In the https://s3amdev.ircam.fr/pluckedstrings_with_snail.html example, we connect an instance of the Snail to Modalys:

```
Modalys.isready.then(function() {
  var audioContext = new (window.AudioContext || window.webkitAudioContext)();
  Modalys.init(audioContext);
  Snail.isready.then(function () {
    var snail = new Snail(document.getElementById("snail"), audioContext, {
      "controls": ['inputGain', 'outputGain', 'visualGain', 'noteRange',
'windowSize', 'frequencyTracking', 'flat/sharp', 'snailSize', 'noteSize'],
      "outputGain": 1
    });
    Modalys.output.connect(snail.input);
    snail.output.connect(audioContext.destination);
  });
});
```

This example demonstrates how easily the Snail can be connected to an audio node for a deeper understanding of the musical structure of sound.



modalys.js and snail.js combined: plucked strings

modalys.js version: 0.1.16

Figure 5.4.1: Demonstration of using Snail for visualising the audio produced by another tool, in this case Modalys, featuring three plucked strings: C, F and and low G.

6. Advanced Music Visualisation Core-Enabling Technologies: The 3D Spectrogram

The Advanced Music Visualization Technologies described in the Deliverable 4.5 - First Version of Music Visualization and Analysis Tools, namely the Envelope, the Tempogram and the 3D Spectrogram were merged into a single advanced visualization, the 3D Spectrogram. The envelope is now represented as a projection of the 3D spectrogram to the time-amplitude plane. The same stands for the amplitude-frequency plane, where the accumulated spectrogram across time is shown. The Tempogram proved to be not educationally useful, and is removed from the final version of the visualization tools.

6.1 Installation and technical requirements

To be integrated within the Workbench, the 3D Spectrogram is imported as an iFrame.

```
<body style="margin:0px;padding:0px;overflow:hidden">
<!-- iframe -->
<iframe id="3dSpectrogram"
src="https://athena.imuscica.eu/visualizations/spectrogram/stable"
frameborder="0" height="100%" width="100%"></iframe>
</body>
</html>
```

6.2 Description of demonstrator and user manual

The 3D Spectrogram works directly on the current stable version of the Workbench in the following URL:
https://workbench.imuscica.eu/
A standalone web page for demonstrating the 3D spectrogram is on the following url:
https://athena.imuscica.eu/visualizations/spectrogram/stable/demo.html

A screenshot of the page is shown in Figure 6.2.1.

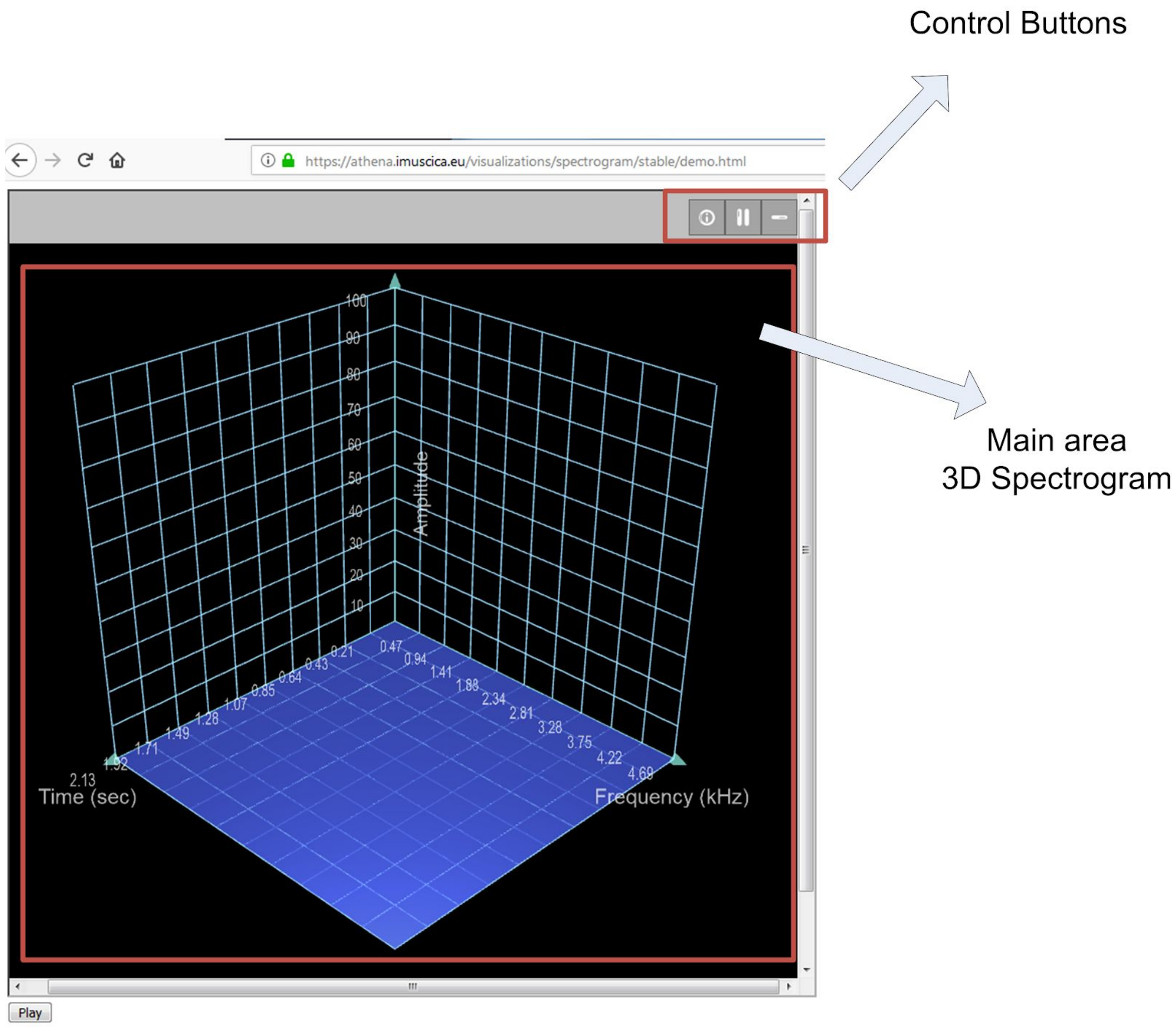


Figure 6.2.1: Screenshot of the demonstration web page that hosts the 3D Spectrogram visualisation tools, where the "Play" starts the reproduction of an audio file.

On the bottom left corner of the window the user can double click on the *Play* button to start playing an audio file. The 3D Spectrogram is displayed in real-time. The user can navigate in the 3D spectrogram by rotating (holding left mouse button), moving (holding right mouse button) and zoom in and out (mouse scroll wheel).
On the top right corner there are three control buttons:

- The info button: It displays the peaks of the 3D Spectrogram.
- Pause/Play button: It pauses/resumes the 3D Spectrogram.
- Show/Hide grids: It shows/hides the grinds on the planes.

Rotate, zoom and move 3D Spectrogram

The user can rotate, move and zoom in and out the 3D spectrogram using her mouse, as shown in Figure 6.2.2.

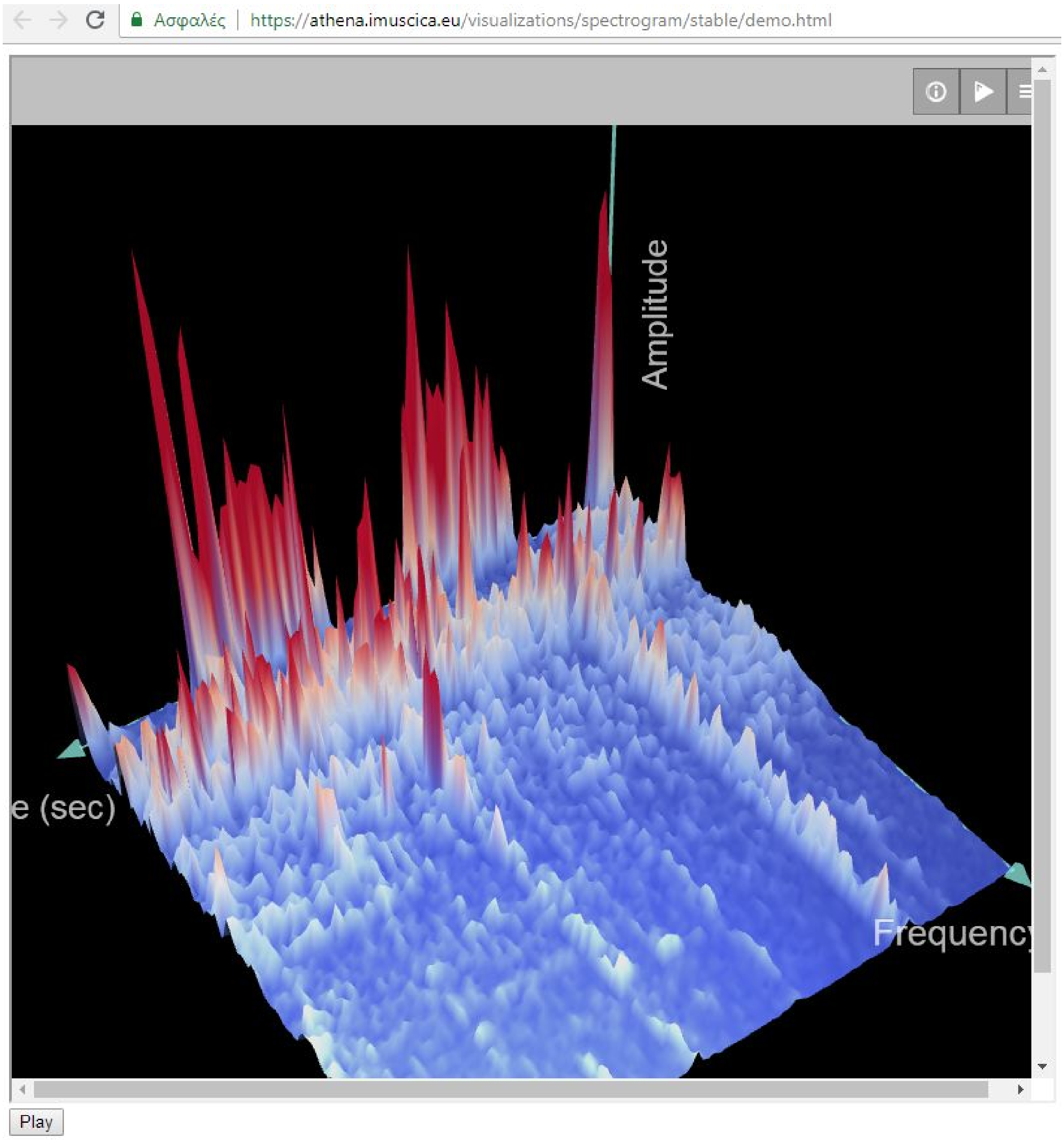

Ασφαλές | https://athena.imuscica.eu/visualizations/spectrogram/stable/demo.html
Amplitude
e (sec)
Frequency
Play

Figure 6.2.2: Screenshot of the 3D Spectrogram that is zoomed and rotated in order to better inspect the spectrogram.

Show/Hide Grids

User can show/hide grids in the three surfaces (time-frequency, time-amplitude, amplitude-frequency) by pressing the show/hide grids button.

Show Spectrogram Peaks

By clicking the info button the user can inspect the peaks of the spectrogram in the time-frequency domain as small spheres, as illustrated in Figure 6.2.3. By moving the mouse over a peak, the user can see information about his peak, namely the time, the frequency and the amplitude.

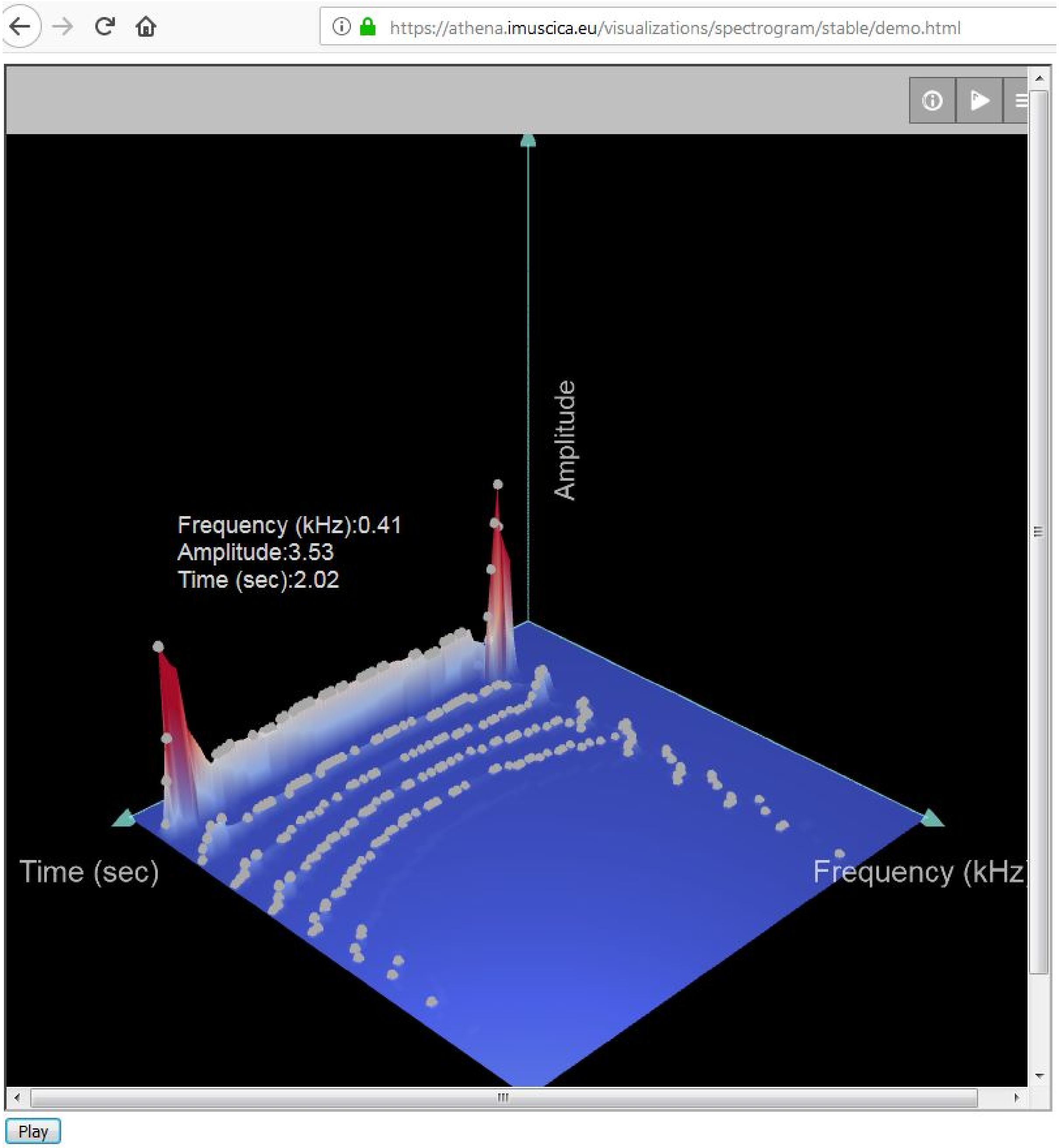


Figure 6.2.3. 3D Spectrogram showing peak information.

Projections of the 3D Spectrogram

The 3D Spectrogram can be projected in any of the three planes:

I. Time-Frequency plane
II. Amplitude-Frequency plane
III. Amplitude-Time plane

If the user double-clicks on any of the three planes, then this plane is displayed only. For example, if someone double clicks on the Time-Frequency plane (case I), the 3D Spectrogram is reduced to the 2D Spectrogram. Regarding case (II) it is transformed to an accumulated across time spectrogram, as shown in Figure 6.2.4. The peaks in the case correspond to peaks of the accumulated spectrogram.

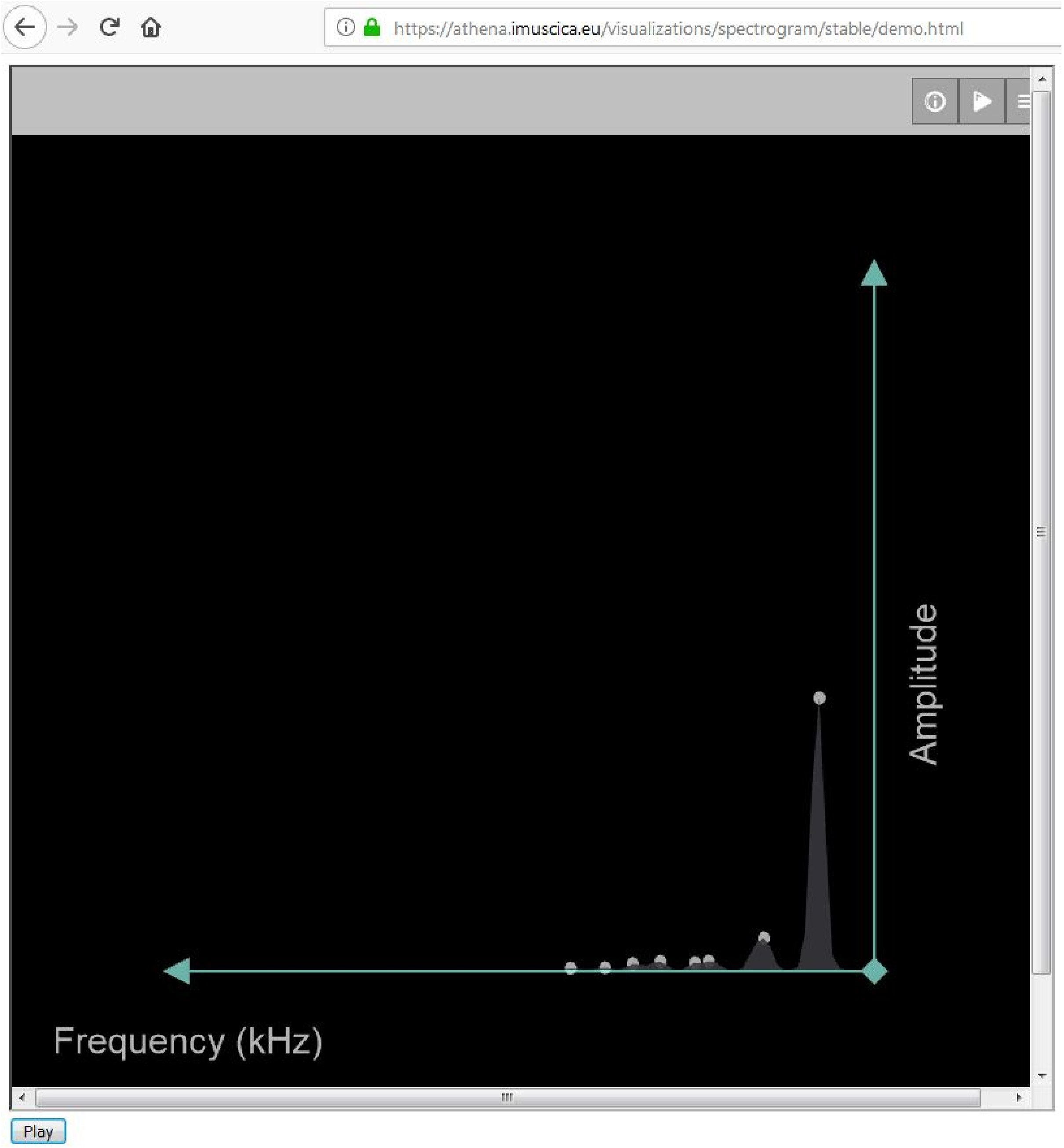

https://athena.imuscica.eu/visualizations/spectrogram/stable/demo.html
Amplitude
Frequency (kHz)
Play

Figure 6.2.4. Accumulated spectrogram with peak information.

Similar to (II), if the user double clicks on the Time-Amplitude plane, the view is transformed to envelope of the signal across time.

7. Cabri Express, all purpose math exploratory tool

7.1 Installation and technical requirements

The mathematics and geometry tools are available in a software called Cabri Express.
For the integration as an iFrame inside the iMuSciCA Workbench, a specific version is accessible using a subdomain URL of imuscica.eu:
https://cabri.imuscica.eu/cabriexpress4imuscica

This version includes initialisations and features specific to iMuSciCA and shared among the tools (loading and saving, copy/paste). Of course, if used as a standalone application (using the URL given above), those features are not available (for a full iMuSciCA experience, one need to test it through the iMuSciCA Workbench).

7.2 Description of demonstrator and user manual

All the descriptions and usages of geometry and numeric tools as well as the calculator, available in D4.5, are still valid in this final version.

The next chapter "7.3 Updates" describes the improvements and new features available in the final version, and their usage as an addendum to the user manual present in D4.5.

7.3 Updates

7.3.1 Improvements

- Graphing speed (particularly useful for graphing sine waves in the iMuSciCA context).
- Pile icon (tools on the left in the UI) now have an arrow to inform the user that a "Pile" can also be opened to select one of the tools in the pile (this improvement is a result of the usability testing). See Figure 7.3.1-1
- General improvements in expressions (functions) display and evaluation.

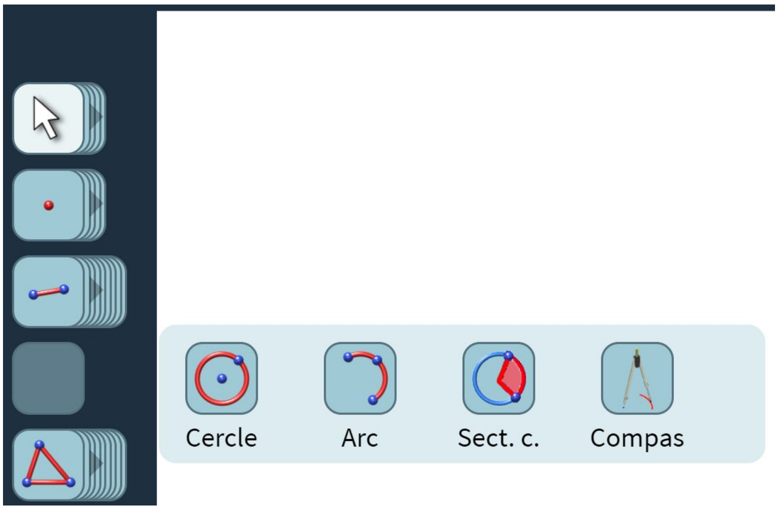


Figure 7.3.1-1: New Piles look - an arrow indicates that you can either select the top pile tool, or click on the arrow to expand the pile and select another tool (e.g. 4th pile).

7.3.2 New features

By double-clicking on an object, you can access its parameters, giving access to 3 new features:

- Set the fill and pen color of geometric objects (see Figure 7.3.2-1).
- Set the number of decimals displayed for a numeric value (see Figure 7.3.2-2).
- Set the slider (numeric value defined by a range cursor) start value, end value, step, and continuous or discrete behavior (see Figure 7.3.2-3).

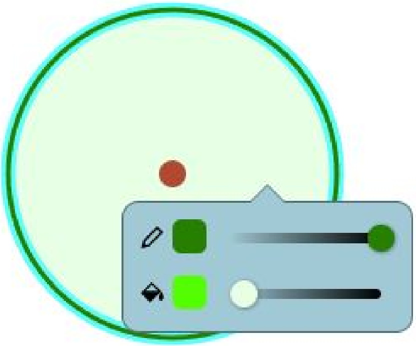


Figure 7.3.2-1: Pen and fill color can be set

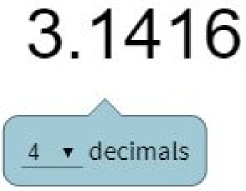
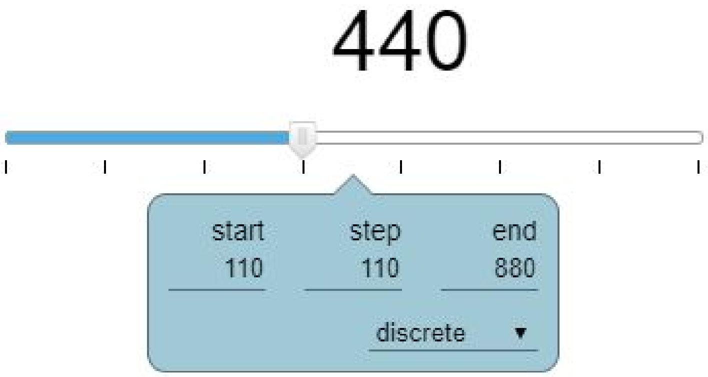


Figure 7.3.2-2: Number of decimals can be set


440

start 110

step 110

end 880

discrete ▾

Figure 7.3.2-3: Slider properties can be set

The iMuSciCA version of Cabri Express also includes the copy/paste feature linked to the iMuSciCA Clipboard. Two usages are possible:

1) Inside the page: with the pointer tool, select a number, and copy (or paste) the value to (or from) the Clipboard. To be able to paste a value, a number must be selected in the Clipboard. You can also copy and paste functions following the same principle (not available at the time of writing).
2) Inside the calculator: if the Cabri Calculator is opened, you can copy the current result to the Clipboard (if it is currently calculable), or you can paste a number (if currently selected in the Clipboard) as a new input in the computed expression.

8. Correction of Rhythm Inconsistencies of Music Performances.

The aim of this tool is to assist the students during their performances, by we employing an onset quantization algorithm based on the metrical grid method [1]. The method enables the reconstruction of a recorded performance in a psychologically plausible way by rectifying the performer's rhythmic inconsistencies. Moreover, a visualization tool is developed so that it depicts both the performed and the transposed onset locations on a metrical grid, thus juxtaposing the two results and enabling some extra interaction features.

When recording a music performance, the triggered Modalys events are captured and stored in a composite data type format (Music Performance Type, see Deliverable 3.6 - Final Version of iMuSciCA workbench design and overall architecture). Henceforth, the created object can be stored and retrieved with the respective developed functionalities.

The onset quantization algorithm relies on finding a fixed metrical grid that suits best to the performed recording. This grid is found with a brute-force approach on all tuples of time-offsets and periods from the corresponding two-dimensional space. For the assessment of each two-dimensional vector we employed a simple heuristic cost function:

$$\sum_{i}^{N} \|x_i - s(x_i))\|^2 ,$$

where x_i , $i = 1, \ldots, N$ represents the i's onset time location and $s(\cdot)$ is a function that returns the location of the nearest available grid's time slot for a given x_i. Suitable meter is considered to be the one with the lowest cost among our choices. After choosing our metrical grid, we simply transpose the recorded onsets to the nearest available time locations as defined by our *s*(·) function. The *s*(·) function can be such that avoids overlapping of identical event types or not, according to the user's preference.

In order to allow the user to see, edit and correct her performance, we created an environment that provides functionalities of sound reproduction and onsets' graphical representation (with the use of canvas and svg tools) for both the recorded performance and its corrected quantized version. The user can also listen to the result of his new version of the recording. In Figure 8.1, we can observe the depicted recorded onsets with yellow color and the quantized reconstructed onsets with red. Monitoring is enabled through a sliding bar moving along the grid and over the event locations in accordance to the excerpt's running-time while the music is playing. The user can also tamper with the performed recording by dragging with his mouse the onsets along the time axis (thus transposing the onset locations).

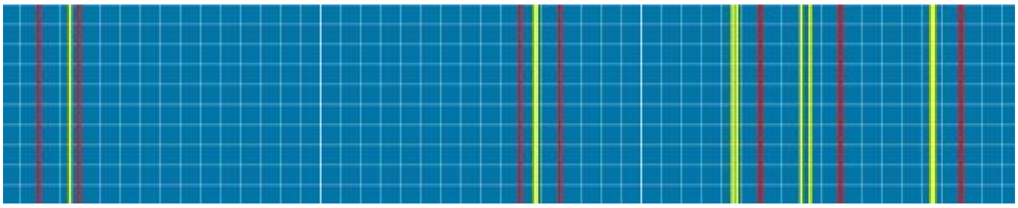


Figure 8.1: Onset visualization

References

[1] Desain, P., & Honing, H. (1989). The quantization of musical time: A connectionist approach. *Computer Music Journal*, *13*(3), 56-66.