

MT5: a HTML5 multitrack player for musicians

Michel Buffa, Amine Hallili, Philippe Renevier

▶ **To cite this version:**

Michel Buffa, Amine Hallili, Philippe Renevier. MT5: a HTML5 multitrack player for musicians. WAC 1st Web Audio Conference January 26-28, 2015 - IRCAM & Mozilla Paris, France, IRCAM & Mozilla Paris, May 2015, Paris, France. hal-01150455

HAL Id: hal-01150455
https://inria.hal.science/hal-01150455

Submitted on 27 Aug 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MT5: a HTML5 multitrack player for musicians

Michel Buffa
University of Nice-Sophia Antipolis
WIMMICS research team
I3S Laboratory
Sophia Antipolis, France
buffa@i3s.unice.fr

Amine Hallili
WIMMICS research team
I3S Laboratory
Sophia Antipolis, France
Amine.halili@i3s.unice.fr

Philippe Renevier Gonin
University of Nice-Sophia Antipolis
I3S Laboratory
Sophia Antipolis, France
philippe.renevier@i3s.unice.fr

ABSTRACT
MT5 is a multitrack player based on the Web Audio API [1]. It's open source, runs on multiple devices (all recent desktop browsers except IE, all IOS browsers based on Safari Mobile, Android Chrome, Opera and Firefox mobile). It has been entirely developed in a web browser using the Cloud9 online JavaScript IDE. A demo version can be tried online at http://mt5demo.gexsoft.com while another version that proposes rock classics song by original artists has a restricted access (http://mt5.gexsoft.com).

Notes
The HTML5 Web Audio domain is still new and we found very few academic research work published, thus most of the references will be URLs of applications, libraries, web sites, etc.

Categories and Subject Descriptors
K.3.1 [**Computer Uses in Education**]: Distance learning.

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Web Audio, Web based IDE, JavaScript, e-learning.

1. INTRODUCTION
Context: authors of this paper are involved in several teaching activities related to Web Audio, and two of them are amateur musicians –rock guitarists who play in bands, and have been taking guitar lessons in public music schools for years. In 2012, The W3C proposed a number of online training courses on key technologies[1]. One of the authors of this paper is in charge of the HTML5 course[2], and had to cover the different APIs that come with this standard, including the Web Audio API [1]. The writing of MT5 – a multitrack player in a web browser- started as a small hack to learn how this technology worked. Several multitrack datasets where available on the net, such as MedleyDB [3] that was primarily developed to support research on melody extraction (and included 122 multitrack songs) or The 'Mixing Secrets' Free Multitrack Download Library [2] made for practicing mixing. Writing a web application that could play multiple tracks in sync was a sort of challenge back in early 2013. A proof of concept quickly worked (Figure 1) and we used it as a theme for a master student course at the Polytech Nice engineer school[3] in France. In addition to the free, legal, multitrack datasets, our students rapidly found multiple bittorrent/mega files[4] packed with hundreds of rock classics songs played by original artists such as the Rolling Stones, Metallica, Guns'n'Roses, etc. We found out that these songs came from different sources: most have been ripped from the video game series Rock Band and Guitar Hero (that included songs with 3-16 tracks depending on the version of the games), some came from sites like hdtracks.com that propose for sale high quality, master multitrack songs (Queen, David Bowie), some have been released officially on collector compact disc editions - this is the case of a few Led Zeppelin songs- and finally some had tracks separated from the original stereo mix using audio editing software (this is the case of some songs by "The Police" where we can hear the other instruments on each separate tracks, if we listen carefully). Being able to play and mix in a browser these classic rock songs by original artists led MT5 to become popular in private circles such as students from our web technology courses, musician friends or musician students from the rock class in our public music school.

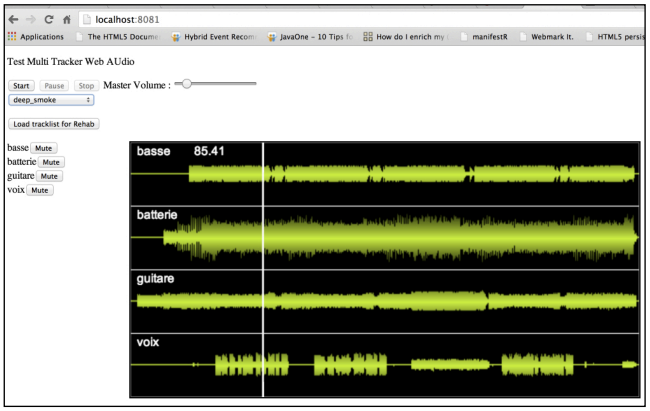


Figure 1 Proof of concept of a multitrack player, 2013

The rock band of the music school played rock classics and most songs members had to learn were available in a multitrack format. Along with Guitar Pro and YouTube, MT5 found its place. You want to learn Aerosmith, ok, but what if Steven Tyler sang along with you instead of a midi generated melody? You think this guitar tab is not faithful, you think this is not how Angus Young from AC/DC *really* plays its guitar solo in Back in Black? Isolate his track and listen carefully! Then mute the track and try yourself. Just with your web browser. The current version shown in Figure 2 is rather stable and runs on any recent browser except Internet Explorer that will support Web Audio in version 12. It also runs on Safari, Chrome and Firefox mobile. It is open source and available on Github[5].

[1] W3C online training: http://w3devcampus.com

[2] http://www.w3devcampus.com/html5-w3c-training/

[3] http://tinyurl.com/m338ggm

[4] http://multitrackdownloads.blogspot.fr/

[5] https://github.com/squallooo/MT5

Copyright © 2015 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editor..
1st Web Audio Conference (WAC), January 2015, Paris, France.

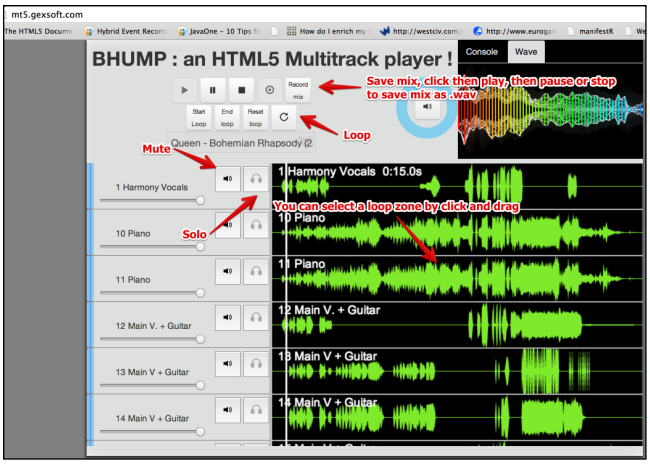


Figure 2 Current version of the multitrack player, 2014.

2. Related works

There are several multitrack web audio applications available on the web now. We can find "GarageBand on the web" like commercial applications such as SoundTrap [4] or CLAW (CLoud Audio Workstation) by Gabriel Cardoso [5] that offers collaborative features for recording or mixing a multitrack song project. SoundTrap includes support for midi controllers, WebRTC video conferencing and companion mobile applications, while CLAW proposes features for managing dependencies on parts of a musical project. OpenDAW [6] is a non-commercial alternative that was a source of inspiration and useful for learning how to code with the web audio API.

There are also JavaScript libraries for writing multitrack Web Audio applications, that come with example demos, such as Helios Audio Mixer [8], a library based on the Google Chrome experiment Mix.js [9].

We also find NoteFlight [13], a commercial web application/web site for teachers and students that proposes impressive music score rendering and playing (they describe themselves as "Finale in a web browser"). This application uses midi events and a bank of sound samples to play the songs. This site is more dedicated to classical music scores and reminds a lot the Guitar Pro software in terms of sound rendering. A Metallica score will for sure not sound really good, as with all midi rendering engines.

3. MT5 features

We interviewed several musician students who learned how to play rock songs. Most of them took guitar, drum, bass or piano courses at the music school we use to go. The main software they use for working at home is Guitar Pro, as this is the tool used by their teachers for writing guitar tabs or music sheets. There are thousands of musical scores / tablatures available on the web in the Guitar Pro (GP) format, so we also guess that teachers download GP files, correct some mistakes and then use them with their students. These latter almost never write any music score, they just use the multitrack play mode of GP. For them GP is a multitrack player (while it is also a music score editor), they can loop on certain parts, mute or solo some tracks, lower the speed, change the pitch, and as most beginners are not fluent in reading musical scores / tabs, they also associate the rhythm patterns to what they hear. In addition, they look for guitar lessons on YouTube or on dedicated free sites like VanderBilly.com. There are numerous videos of variable quality on these sites where people show how to play classic rock songs, how to place the fingers on the guitar's neck, etc. And there are specialists of some Rock Bands that even got recognition from the bands themselves. SoloDallas (his nickname on YoutTube), for example is a known expert for explaining how to play any AC/DC song, June626 for playing Led Zeppelin and the Beatles, etc.

So, in our player, we reproduced the features used by students with Guitar Pro: load a song, play, stop, pause, mute or solo some tracks, change the volumes of tracks, loop on a parts of a song (**Figure 3**), etc.

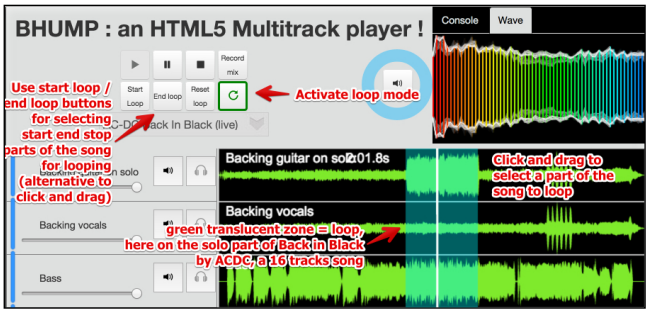


Figure 3 Selecting a loop area and activating loop mode in MT5, here on a 16 tracks song.

In Guitar Pro, we can see parts of the songs as we have bars, and markers that identify the structure of the song. With audio files we can either use markers too (song metadata) or a more visual approach that consists in visualizing the sound samples. The sound wave shapes help identifying what is going on in a particular track, **Figure 3** shows in the first track –Angus Young guitar solo track of Back in Black- the noticeable wave shape of the guitar solo (we selected this part for looping in the presented example). At the end of this paper we will discuss the next version of MT5 that will make heavy use of metadata for navigating along a track, and add support for Guitar Pro score rendering in sync with multitrack audio.

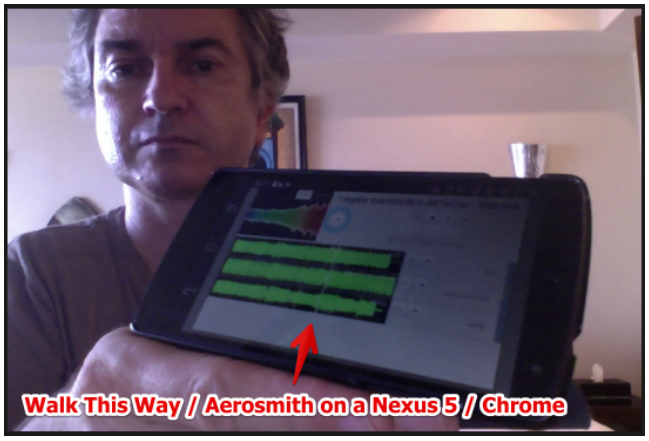


Figure 4 MT5 on an Android Smartphone.

We also added the possibility to save client-side a mix of the current song as a .wav file. This works only with browsers that implement the FileSystem and FileWriter HTML5 APIs (Chrome and Chrome mobile on android). Just mute the solo guitar track, click on the "save mix button", play the song, stop it and it will write a .wav file on your hard disk. You can listen to it directly with any audio player on your desktop or phone. This feature has been written as an experiment.

We also kept the GUI simple and reduced the "technological risk" by limiting the number of features of this first version, so that we

can focus on stability, and on making it usable on mobile devices, as shown in Figure 4 and Figure 5. There is however room for improvements, in particular for adapting the GUI to smartphones (we will address this in a next version).

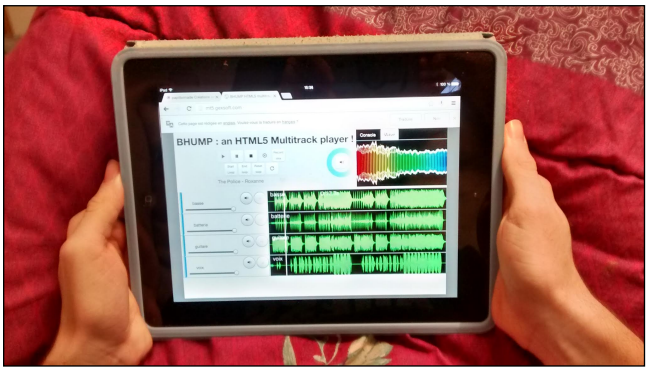


Figure 5 MT5 on an iPad / Safari

In the current version it is not possible to change the speed of a song without changing the pitch, or changing the pitch without changing the speed. This is called time stretching and doing that in real time on a multitrack audio song, in JavaScript, is a real challenge. There are pieces of JavaScript source code that do that, like the implementation proposed by the author of the musical score rendering JavaScript library VexFlow [11][10], that does not work in real time, or the one in the Keyboard-Singer application [12] in which your voiced is pitched in real time depending on a note you play on a virtual keyboard on the screen. These experiments only work with a single audio stream.

4. Implementation

MT5 was developed as a proof of concept and is in its first stable iteration. The server side code is really simple, about 50 lines of JavaScript code that runs on a NodeJS server. Each song, server side, is a directory whose name is the name of the song, and whose files inside are the tracks, in mp3, ogg or wav format.

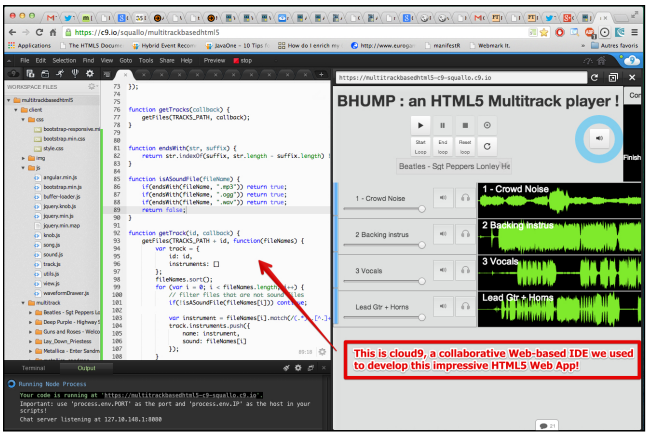


Figure 6 MT5 has been developed 100% in a web based IDE.

In our research team we developed an online JavaScript web based IDE for writing applications that exploits the web of data and semantic data, called WikiNext [15]. In 2012 we were making a state of the art and started studying other existing IDEs so we decided to use the Cloud9 IDE for MT5, in order to see if web based IDEs were valuable tools. MT5 has been developed 100% in a web based IDE, it's not a huge application but the experience was a success and we kept developing it in our web browser. Cloud 9 includes a NodeJS server for the back end of projects, supports collaborative editing (à la Google doc), includes a Unix terminal and sync project files with source code repositories like GitHub, BitBucket, etc. We were able to push the source code to our hosting server directly from the terminal in Cloud9 web page of the MT5 project, as shown in Figure 6.

We implemented two web services using the "express" module of NodeJS: one for getting the list of tracks, the other one for getting the list of tracks in a particular song.

http://mt5demo.gexosoft.com/track will answer a JSON array like that:

```
[
"Admiral Crumple - Keeps Flowing - Demo",
"Big Stone Culture - Fragile Thoughts",
"How To Kill A Conversation -
HeartOnMyThumb",
"John McKay - Daisy Daisy",
"Londres Appelle",
"Street Noise - Revelations",
"Tarte a la cerise",
"Wesley Morgan - Flesh And Bone"
]
```

And
http://mt5demo.gexsoft.com/track/Admiral%20Crumple%20-%20Keeps%20Flowing%20-%20Demo will return the list of tracks of the first song:

```
{
"id":"Admiral Crumple - Keeps Flowing -
Demo",
"instruments":[
 {"name":"01_Kick1","sound":"01_Kick1.mp3"},
 {"name":"02_Kick2","sound":"02_Kick2.mp3"},
 {"name":"03_Snare","sound":"03_Snare.mp3"},
 {"name":"04_Hat1","sound":"04_Hat1.mp3"},
 {"name":"05_Hat2","sound":"05_Hat2.mp3"},
…
]}
```

Client side we had to face several difficulties and we learned the hard way how to work with the Web Audio API.

Understanding the web audio graph concepts: Play/Stop/Pause/Jump in an unnatural way

There are some strange things to learn when you just want to implement a simple play/pause/stop/jump operation with a web audio graph. In MT5 we have a very simple audio graph. Nothing complicated. Let's have a look at it with a 3-track song:

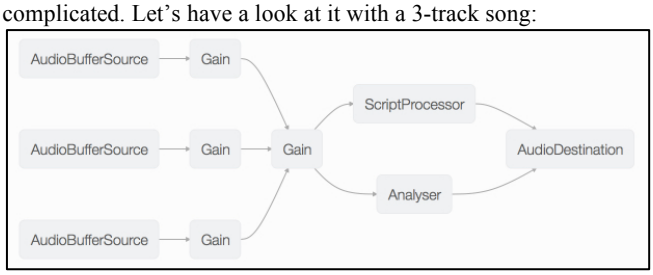


Figure 7 Web Audio Graph of a 3-track song in MT5

Each sound sample is represented as an AudioBufferSource node. This corresponds to each track. Then each of these nodes is connected to a gain node (volume of the track), then all volumes are connected to another gain node (the master volume of the song), then to an analyzer node that is used to display frequencies that dance in real time when the music is played, and into a

ScriptProcessor node (a custom node) used for saving the mix as a .wav file if requested. Finally the audio stream comes to an AudioDestination (the speakers).

The things we learned the hard way are:

- We can call the start() method on an AudioBufferSource only once,
- If we call the stop() method on an AudioBufferSource, the node cannot be used anymore.
- There is no pause() method on an AudioBufferSource,
- There is no easy way to jump to another part of the song, in the middle of a sound sample, if we already called the start() method (if it's already playing).

So, once a song is stopped by calling the stop() method on each source nodes, the nodes are meant to be thrown away. We need to recreate new ones and re-connect them to the audio graph if we want to play the song again.

As pausing a song or jumping to another part of a song is equivalent to stopping it and playing it again, we've got to re-create the AudioBufferSource nodes of each track and connect them to the rest of the audio graph before calling the start() method on each node. This seemed very unnatural to us but apparently the web audio API is optimized for this pattern.

Also, we thought that calling stop() on these nodes kind of destroyed them and that the JavaScript garbage collector would release the corresponding memory. We were wrong. The web audio debugger available in the last versions of Firefox showed that these nodes were not garbage collected unless we deleted them explicitly. Without this precaution, looping or jumping a lot on a song with many tracks quickly slowed down the whole user experience.

Figure 8 presents pseudo JavaScript code that shows how to implement the play/stop/pause operations:

```
play = function(startTime) {
  buildGraph();

  sampleNodes.forEach(function(s) {
    // First parameter is the delay before playing the sample
    // second one is the offset in the song, in seconds, can be 2.3456
    // very high precision !
    s.start(0, startTime);
  });
  paused = false;
}

stop = function() {
  if(paused == true) return; // cannot stop more than once.

  sampleNodes.forEach(function(s) {
    // destroy the nodes
    s.stop(0);
    delete s; // mandatory otherwise not Garbage Collected
  });
  paused = true;
}

pause = function() {
  if (!paused) {
    // if we were not paused, then we stop
    this.stop();
  } else {
    // else we start again from the previous position
    play(elapsedTimeSinceStart);
  }
}
```

Figure 8 Pseudo code for play / pause / stop

Dealing with time

The AudioContext object at the heart of the WebAudio API has a property named currentTime, a high precision timer. We thought we could use it to display the currentTime when we play a song, for jumping to any part of a song, etc. The problem is that this variable is a double representing an ever-increasing hardware time in seconds used for scheduling. It starts at 0 at a time that differs from one browser to another and cannot be stopped, paused or reset. In order to measure a period of time you've got to measure deltas between different values of the currentTime variable, at different times. Chris Wilson wrote a very good article about the different approaches one could use when dealing with time in a web audio application [14], and we chose to use the intervals generated each 60th of a second by an animation loop scheduled by the requestAnimationFrame HTML5 API. A function like that:

```
function animateTime() {
    // Draw something only if a song has been loaded
    if(currentSong !== undefined) {
        if (!currentSong.paused) {
            // Draw the time on the front canvas
            currentTime = audioContext.currentTime;
            var delta = currentTime - lastTime;

            // DO SOMETHING
        }
    }
    // Call at 60 frames/s the animateTime function
    requestAnimFrame(animateTime);
}
```

Figure 9 using the requestAnimationFrame API to measure time deltas

Dealing with changes in the API

As stated on an article posted on the Mozilla Developer Network web site: "*The Web Audio standard was first implemented in WebKit, and the implementation was built in parallel with the work on the specification of the API. As the specification evolved and changes were made to the spec, some of the old implementation pieces were not removed from the WebKit (and Blink) implementations due to backwards compatibility reasons. New engines implementing the Web Audio spec (such as Gecko) will only implement the official, final version of the specification, which means that code using webkitAudioContext or old naming conventions in the Web Audio specification may not immediately work out of the box in a compliant Web Audio implementation.*"

When we started working on MT5, there was only the webkitAudioContext implementation available. Then Firefox supported the official AudioContext version and we had to choose between them. Fortunately a library named "webkitAudioContext monkeypatch"[6] was released soon after, that makes most code targeting webkitAudioContext to work on the standards based AudioContext out of the box. So we included this library, adapted all our code to the official API and it could run on nearly any recent browser, mobile or desktop.

Drawing sound sample shapes

We chose to have an accurate representation of the sound samples, with an oversampling approach: iterate through the sample values and draw the corresponding frequencies in a HTML5 canvas. As there are many values in a sound samples (can be several millions…), this is a much longer process that we can have with an under sampling approach (iterate through the columns of a canvas, and pick the sample value). This leads to very different results as shown in Figure 10 and Figure 11. In MT5 we used an adapted version of the code from the WaveSurfer.js[7] library.

[6] https://github.com/cwilso/webkitAudioContext-MonkeyPatch

[7] https://github.com/katspaugh/wavesurfer.js

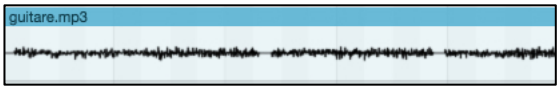


Figure 10 Under sampling

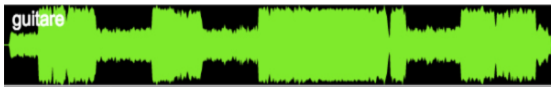


Figure 11 Oversampling

5. Evaluation and perspectives

MT5 is used by some students from two public music schools, and two music teachers, one is the electric guitar teacher from the music school of Biot, a village in the French Riviera, the other one teaches multiple instruments and is in charge of the OnOff association[8] that proposes teenagers to play in rock bands. These students have to learn regularly classic rock songs, entirely or just parts of them. Or they just learn some songs by themselves. The MT5 version packed with hundreds of original rock classics really had its success. Play with Mick Jagger while you do the Keith Richard's clone in your bedroom!

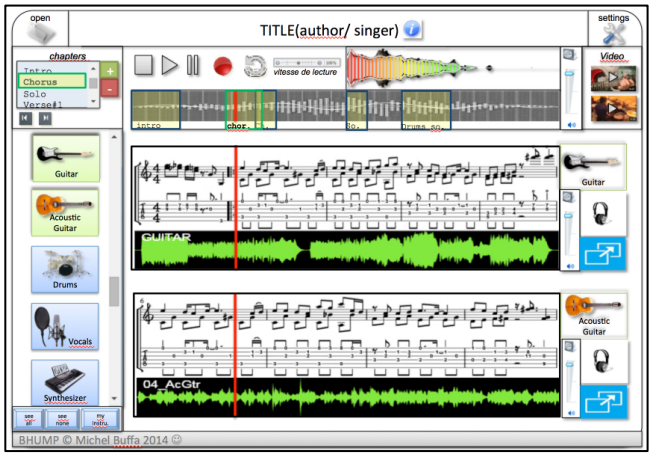


Figure 12 Mockup of MT5 version 2. Notice the chapters on the left and a logical timeline at the top. Scores can be showed/hidden

We also started an evaluation program with students who follow the Human Machine Interface master 2 at the Polytech Nice engineer school. They started to conduct interviews, and ask MT5 users how they used the tool and what they would like to see in future versions. The study is still on its way but it appeared that one thing everyone did was to look at sheet of papers with the musical score or tablature, or looked at a Guitar Pro rendering of the musical score from time to time, and then tried to play along with the audio multitrack songs. They also used to watch some lessons on video sharing sites like YouTube. They complained that MT5 on smartphones was really not adapted: the GUI was too small and they had to zoom in a lot, and it could take very long to decode the song tracks once downloaded. Also, the phone was becoming very hot during this operation.

The teachers proposed to include some videos also, synced with the musical score (we filmed a teacher explaining how to play the different parts of a song and will include it as a sample in the next version). They also expressed the need to have "chapters" or a kind of predefined structures in songs, to make easier working on a given part: solo, intro, verse1, bridge, chorus 2, etc.

So we started to make some mockup prototypes of a new version, and also looked for technical solutions in order to display a musical score or tablature in sync with the multitrack audio or with a video.

Here are some mockups we made. We will validate them with the students and teachers:

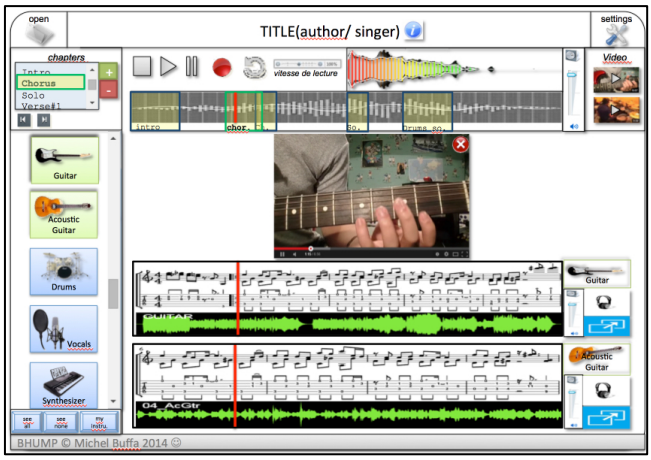


Figure 13 Mockup with a pedagogical video.

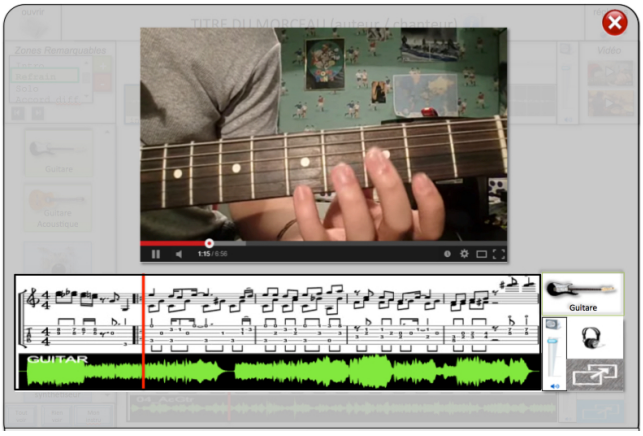


Figure 14 Mockup of a zoomed view of a track with video and score.

Rendering Guitar Pro musical scores / tablatures in real time

We integrated the alphatab.net library [16] on the work version of MT5. This library renders Guitar Pro files in real time in an HTML5 canvas, and while its documentation is rather poor, it works pretty well. We designed an API layer that allows us to render in real time any bar interval of any track of a song, as shown in Figure 15.

6. Conclusion

We presented MT5, a web audio open source multitrack player, used as a learning tool by musicians, mainly involved with rock music. The current version is stable and runs on most recent browsers, desktop or mobile, except IE that announced support for web audio in its version 12. The tool is stable, available online, and a version 2 is on the work, that should propose a much richer experience. As a first mid-size experience with web audio

[8] http://associationonoff.free.fr/

development, MT5 is already a usable tool that found its place in some music schools.

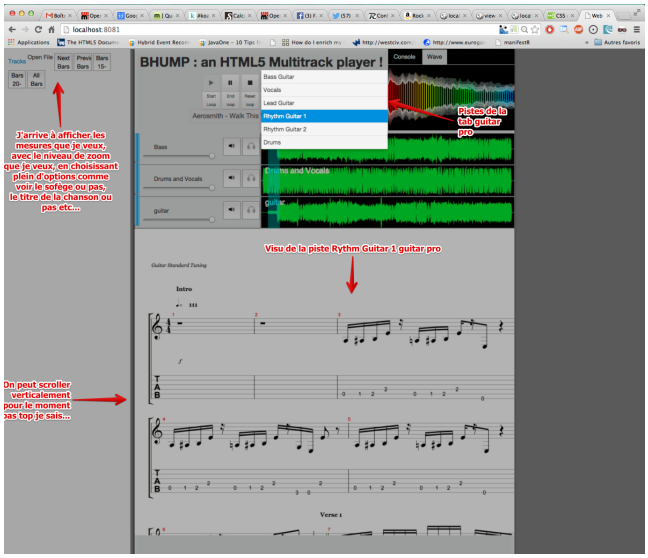


Figure 15 Preliminary experimentations: displaying musical score sections in sync to the audio, using the alphatab.net guitar pro file rendering library.

7. Acknowledgements

We would like to thank Pr. Roger Lartheau from the Biot music school for his advices and patience.

8. REFERENCES

[1] Web audio API. http://www.w3.org/TR/webaudio/

[2] The 'Mixing Secrets' Free Multitrack Download Library: http://www.cambridge-mt.com/ms-mtk.htm

[3] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam and J. P. Bello, "MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research", in 15th International Society for Music Information Retrieval Conference, Taipei, Taiwan, Oct. 2014.

[4] Soundtrap: https://www.soundtrap.com, presentation video https://www.youtube.com/watch?v=PMH1vM-dSc0

[5] Gabriel Cardoso: Cloud Audio Workstation, http://www.claw-studio.com/

[6] Adam Levine and Piettro Verrecchia, OpenDAW: https://github.com/pverrecchia/OpenDAW

[7] Proposal for a W3C Media Multitrack API: http://www.w3.org/WAI/PF/HTML/wiki/Media_Multitrack_Media_API

[8] Helios Audio Mixer JavaScript library: https://github.com/heliosdesign/helios-audio-mixer

[9] Mix.js – Chrome Experiment - https://github.com/kevincennis/Mix.js

[10] VexFlow time stretching: http://vexflow.com/vexwarp/

[11] VexFlow, Music engraving in JavaScript and HTML5, http://www.vexflow.com

[12] Keyboard-Singer: sing along and pitch your voice in real time: https://github.com/bioball/Keyboard-Singer

[13] NoteFlight, “Finale in a web browser”, http://www.noteflight.com/

[14] Chris Wilson - A Tale of Two Clocks - Scheduling Web Audio with Precision, 2013, http://www.html5rocks.com/en/tutorials/audio/scheduling/

[15] Pavel Arapov, Michel Buffa, and Amel Ben Othmane. 2014. WikiNEXT: a wiki for exploiting the web of data. In Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14)

[16] Alphatab.net a library that renders Guitar Pro files in HTML5 canvas or SVG, http://www.alphatab.net