A Dynamic Audio Experience creation platform in Web Audio

Chinmay Pendharkar
Sonoport Asia Pte. Ltd.
Blk 71 Ayer Rajah Crescent
Ayer Rajah Industrial Estate, Singapore
chinmay.pendharkar@sonoport.com

Peter Bäck
Sonoport Asia Pte. Ltd.
Blk 71 Ayer Rajah Crescent
Ayer Rajah Industrial Estate, Singapore
peter.back@sonoport.com

Lonce Wyse
Communication and New Media Department
National University of Singapore
11 Computing Drive, Singapore
lonce.wyse@nus.edu.sg

ABSTRACT

The Web Audio platform provides a great opportunity to create rich, interactive audio experiences for the Web. The platform by itself however, is complex for an average Web developers without much audio synthesis or audio processing background to be able to use effectively.

We explore building layers of functionality over the Web Audio platform that abstract out the complexities and peculiarities of Web Audio, and furthermore, building in default behaviours which help web developers to quickly and effectively create interactions that are intuitive. In this set of demos, we showcase two of these frameworks, Sound Models and Interaction Driven Templates that can help developers use Web Audio effectively, and how such frameworks can be combined into a complete Web based interactive experience authoring tool.

Categories and Subject Descriptors

H.5.5 [**Information Systems**]: Information Interfaces and Presentation (HCI)—*sound and music computing*

General Terms

Design

Keywords

Interactive Audio, W3C Web Audio API, Audio Synthesis

1. INTRODUCTION

The modern Web platform has drastically changed the capacity of the Web to deliver rich, interactive, experiences to millions of users. With technologies like, Web Sockets, Web RTC, Web GL and Web Audio, modern browsers can deliver audio-visual experiences which can be interacted with using traditional mouse and keyboards as well as various new sensors like DeviceOrientation [3], Geolocation [2], Voice and Camera.

With such wide variety of options for content creation and interaction available, it can be challenging to create applications and experiences which feel intuitive and natural.

While moving Sonoport's Dynamic Sound Engine to the Web platform as explored by us [1], we wanted to understand how best to help developers create rich, interactive audio experiences on the Web itself. The Web Audio API although intuitive to someone in the field of computer music, can be very complex for a typical web developers to use in their application.

The approaches and ideas we came up with were then used to create an authoring platform which enables web developers to quickly, intuitively and robustly add interactive audio elements to their web applications.

In set of demos we will go through some of the novel ideas and approaches we designed into this platform to help developers easily connect user interactions to synthesized audio.

2. WEB PLATFORM AND COMPLEXITY

The openness and acessibility of the Web platform has made it an attractive platform for many developers to it. As the Web platform evolved and modernized it picked up many new technologies which were not traditionally Web based. These technologies, like Web Audio and WebGL brought along their own semantics and knowledge base into the Web platform. To be able to effectively use these new technologies, a typical web developers has to learn the semantics, and techniques of these technologies with respect to the Web platform.

For example, let's consider a scenario of adding a 'swoosh' sound when someone interacts with a specific HTML `div` tag. Detecting when a user moves the pointer over a specific HTML Element is straightforward using `DOMEvents`. However adding the interactivity with audio is significantly more complex. It would involve creating an `AudioContext`, downloading and decoding the audio file, creating `AudioBuffersSourceNodes`, starting and stopping on mouseover events. And to make it sound natural, one would have add a `GainNode` and add some basic ADSR Envelope functionality to ease In/Out the volume of the sound.

This for a simple mouse-over sound effect. However if the audio itself needs to be processed differently based on different interaction, for example changing the pitch of a sound based on where a user clicks, it would require the developers to have some understanding of audio digital signal processing.

To look at how this can be made easier for typical web developers we analyzed the various dynamic aspects of the system needed to create such a rich dynamic audio experience. Based on the two critical areas of dynamism we found, we created frameworks that help to abstract out the complexity of the dynamism. We will demoing the inner workings of these frameworks and how they can be used to create a complete dynamic audio experience authoring platform.

3. SOUND MODELS

We are used to interacting with physical things in the world around us. These things make sounds when we interact with them. This natural change of sound based on our interaction is something we're used to and expect when interacting with virtual objects on the Web.

Some of the common ways of how sounds change, or more specifically how we expect sounds to change on interaction can be abstracted out into parameterized algorithms which generate those sounds in real time. These algorithms called Sound Models can be purely algorithmic or can take on input files for sonic texture.

An example of such an Sound Model is *Trigger*. *Trigger* employs a polyphonic queue to generate a sound of something being triggered by a single user interaction. It could be a sound of a gun in a video game, a drum hit or just a sound effect for clicking on a button. However, such a sound is very commonly expected to play again if it is triggered repeatedly. More importantly if the sound caused by the previous interaction hasn't finished playing yet, the next user interaction can still start a new sound. This behaviour is encoded into a *Trigger* Sound Model along with parameters to change certain properties like the pitch of the sound.

Sound Models encode many of the common ways we expect sounds to behave and take care of all the complexities of scheduling, polyphony, enveloping from the user.

The Sound Model can also be designed to be extendible by composing them from Web Audio AudioNodes. This allows them to be chained with other AudioNodes for manually adding specific effects or processing beyond that created by the Sound Model itself.

We have created a set of Sound Models which can be used to then create rich, interactive and dynamic sound experiences. We will demo the working of these Sound Models using an Web interface, shown in Figure 1., built to develop and test these Sound Model called *JSM Player*.

4. INTERACTION DRIVEN TEMPLATES

While Sound Models abstract out the dynamism in the audio processing itself, to abstract the dynamism in user interactions, we created Interaction Models. Once again we considered the various types of interactions we had to create for the various demos and interactive media projects we had done in the past and looked at common patterns which could be encoded into some form of templates, we called these Interaction Driven Templates.

The usual Mouse and Keyboard combination for user interfaces allows us a surprisingly varied and rich set of interactions. From simple point and click, to hovering over an

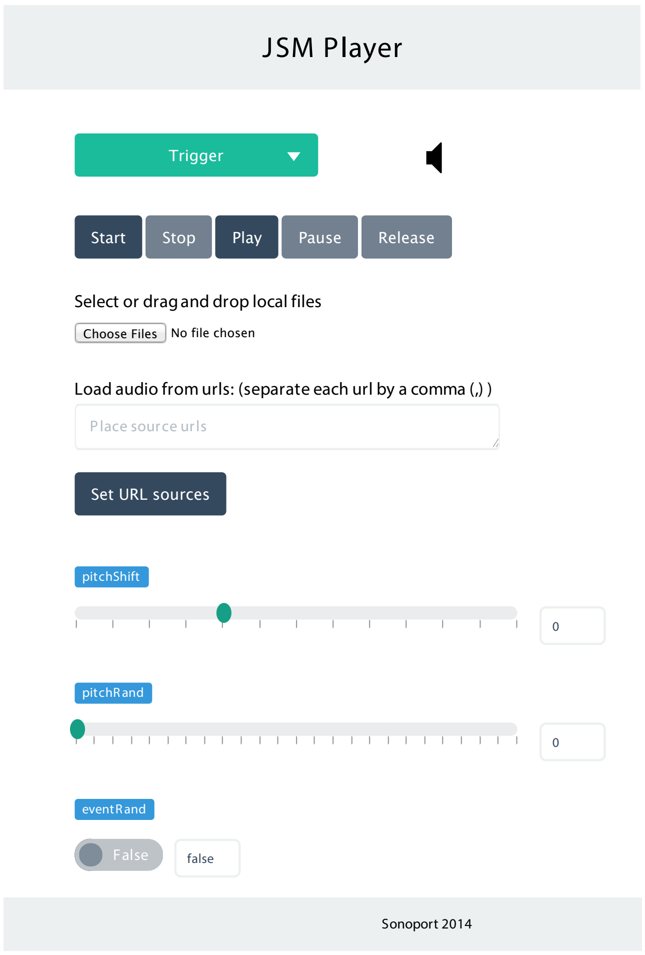


Figure 1: A Web based interface to Sound Models

object on screen, to dragging on screen objects to moving the pointer location with respect to other objects. Some of these interactions may not be explicitly supported natively by the Web platform and have to be gleaned from a combination of multiple user interface events. Abstracting this complexity out makes adding interactions to audio a lot simpler for web developers.

4.1 Interaction Levels

First step was to categorize the various types of interaction in terms of complexity of the interaction. Since finally these were to going to be templates that web developers would use in their own projects, having sorted by complexity would make it intuitive. We created levels of interactions as defined in Table 1.

For each of these *Interaction Levels*, we created multiple templates which connected the associated user interaction events with Sound Models which were complementary to corresponding Interaction Model. For example, a *Tap* or *Click* template matches the Trigger Sound Model described in section 3. In many scenarios we found multiple Sound Models being compatible with a specific Interaction Model, and hence the choice of Sound Models was exposed in the user interface.

The templates also setup default audio sources or textures

Table 1: Levels of Interaction Models

Level	Description	Example
1	very basic interactions with an onscreen object	tap, click, hover over
2	events being triggered for each time the mouse was moved	mouse-move, mouse-position,
3	onscreen object and their relation with the mouse or pointer	drag an object, move about an object
4	physical device orientation and location, external inputs like voice	device-rotate, device-move

and parameters for the Sound Models so loading a new template allows a web developers to experience the type of interactive audio that can be created using that specific template. The sounds and the parameter can, of course, be changed later to match the exact aesthetic that the web developers wants to achieve.

These templates are created using Web technologies like Canvas, and have a standardized metadata API for packaging and loading. This makes them portable and extensible, and allows easy creation of templates by third parties.

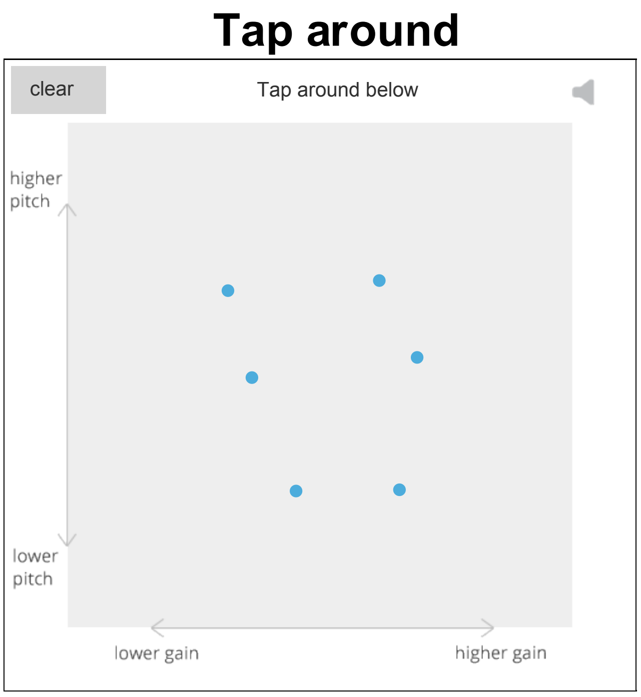


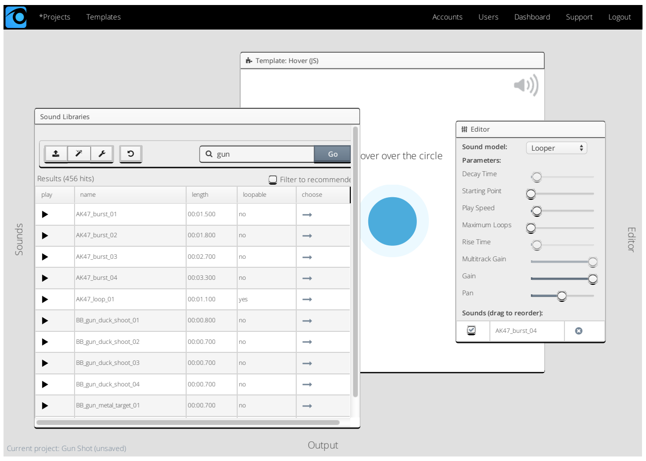
Figure 2: A Web based interface to Interaction Driven Templates

We will demo a set of *Interaction Level 1* and *Interaction Level 2* based templates that we have created for mouse based interactions as well as touch based interactions for mobile devices. Figure 2. shows the Web Interface used for testing and tweaking these templates that we will be using to demonstrate this framework.

5. AUTHORING PLATFORM

We integrated Sound Models, Interaction Driven Templates and other novel ideas like Control Mappings into the web application called Sonoport Studio. The Sonoport Studio helps web developers quickly create an interactive audio experience using dynamic Sound Models and Interaction Driven Templates using a graphical user interface to change parameters, values, mappings and sounds.

While the Sound Models and Interaction Driven Templates abstract out the dynamism in creating interactive audio, the Sonoport Studio itself is designed to help improve a web developer's workflow for creating interactive audio experience. Figure 3. shows current version of the Web interface of the Sonoport Studio.



Results (456 hits)

Filter to recommende

play	name	length	loopable	choose
▶	AK47_burst_01	00:01.500	no	→
▶	AK47_burst_02	00:01.800	no	→
▶	AK47_burst_03	00:02.700	no	→
▶	AK47_burst_04	00:03.300	no	→
▶	AK47_loop_01	00:01.100	yes	→
▶	BB_gun_duck_shoot_01	00:00.800	no	→
▶	BB_gun_duck_shoot_02	00:00.700	no	→
▶	BB_gun_duck_shoot_03	00:00.700	no	→
▶	BB_gun_duck_shoot_04	00:00.700	no	→
▶	BB_gun_metal_target_01	00:00.700	no	→

Sound model: Looper

Parameters: Decay Time, Starting Point, Play Speed, Maximum Loops, Rise Time, Multitrack Gain, Gain, Pan

Sounds (drag to reorder): AK47_burst_04

Current project: Gun Shot (unsaved)

Figure 3: Sonoport Studio authoring platform

The Sonoport Studio ties in with audio repositories to allow search, preview and importing of audio files as texture for Sound Models. These textures can then be tested with various templates and Sound Models. Sonoport Studio also has Web interface to tweak Sound Model parameters, mappings between Interactions and Models parameters to customize the experience for the intended use case.

Once developers have created an experience that they are comfortable with the project can be exported either as raw code, or as files or projects for HTML5 visual design tools like Google Web Designer or Adobe Edge.

While the Sonoport Studio is still heavily in development, we will demo a working version of the Sonoport Studio capable of using the underlying frameworks of Sound Models and Interaction Driven Templates to create a dynamic sound experience.

6. CONCLUSIONS

Web Audio has the promise of allowing rich, interactive audio experiences on the Web. However, it's complexity and semantics make it difficult for web developers without specific domain knowledge to use it effectively. Using abstractions over dynamism in sound in the form of Sound Models and over dynamism in user interactions in the form of Interaction Driven Templates, we demonstrate a platform for authoring such rich, interactive audio experiences on the

Web. Such a platform can make it easy to create an application where interactions generates audio that sounds natural and intuitive.

7. REFERENCES

[1] C. Pendharkar, P. Bäck, and L. Wyse. Adventures in scheduling, buffers and parameters : Porting a dynamic audio engine to webaudio. In *Proceedings of the Web Audio Conference*, pages XXX–XXX, Paris, France, 2015.

[2] A. Popescu. Geolocation api specification. http://dev.w3.org/geo/api/spec-source.html, July 2014.

[3] R. Tibbett, T. Volodine, S. Block, and A. Popescu. Deviceorientation event specification. http://w3c.github.io/deviceorientation/spec-source-orientation.html, March 2014.