Minim: Final Year Project

Bede Kelly

April 2019

Contents

1 Introduction & Motivation

Minim is a browser-based playground for making music with patterns. It's simple enough to be easy to pick up and use for people of all skill levels, but deep enough that enthusiast musicians and producers have the control they expect from professional software. Describe aims of project more concretely here.

2 Prior Art

The commercial and non-commercial space for desktop music software is saturated, but owing to the limitations of the web platform very few examples exist of full-featured audio workstations which work in-browser without any installation. This project draws more inspiration from full desktop applications such as Reason than it does from current web-based alternatives.

3 Project Planning and Workflow

Because of the project's scope I maintained a Kanban board using the Trello online software. I continuously revisited my estimates of how long future tasks were going to take, and at times I used a MSCW diagram to prioritise features. I also tracked bugs and estimated their difficulty to fix, leaving myself a 20-day buffer before the demonstration for bug fixing, code cleanup and last-minute features. Because of this and my use of the Git version control system I was able to add several features which initially I didn't project I'd have time for.

4 Overall Design Considerations

The project takes the design philosophy from Reason to put all options directly in front of the user as tweak-able dials and sliders, rather than using menus and dialogue boxes. This leads to some prioritisation and selection of the options available to the user.

5 Code Structure & Framework

Using a system of unique IDs for each instrument and effect, the React code for displaying components and the audio code for playing their sounds is effectively decoupled — updates to the page layout etc have no effect on audio processing, and the audio code is treated as authoritative when states differ.

6 Sound in the Browser with Web Audio

My project wraps web audio nodes in a higher-level abstraction which allows for multiple nodes per component. Describe how the App, Racks and Components interact with each other, with some detail about harder cases like re-ordering/deleting effects. Also describe special-case things which lie outside the regular flow of the graph e.g. the metronome and the audio recorder.

7 Hardware Control with Web MIDI

MIDI learning and note control are both big parts of the project and allow plugging in and using almost any hardware controller. Give an overview of how MIDI learning works within the AppAudio context, and how its scaling works automatically by being routed through the Knob or Slider component.

8 Audio Components in Detail

For each component, include screenshots in meaningfully different states, a description of any interesting Web Audio features used, any novel algorithms/approaches, and a short audio sample where appropriate.

8.1 Instruments

8.1.1 Tape Looper

Describe "Current Position" interpolation algorithm; include a graph of relative position over absolute time.

8.1.2 Polyphonic Synthesizer

Describe/diagram creation of an oscillator node for each note, including individual filter/amp envelopes and nodes, and also including global effects like the LFO.

8.1.3 Drum Pads

Describe Hold/Hit modes for drum pads and explain why each can be useful.

8.1.4 Ambient Sounds

Note that looping is seamless with a crossfade, and maybe diagram async asset loading with 'Promise.all()'.

8.1.5 Granular Synthesizer

Describe algorithm for granular synthesis including quick explanation of gaussian distribution.

8.2 Effects

8.2.1 Pan

8.2.2 Low and High Pass Filters

Include description of how to get a low-shelf effect using mix.

8.2.3 Volume

8.2.4 Compressor

Mention that separate side-chain inputs aren't available.

8.2.5 Convolution Reverb

Describe at a high level how a convolution reverb works and why I've included the sounds I have.

8.2.6 Echo/Delay

Include a diagram of the feedback loop with low-pass filter and gain nodes.

8.2.7 Distortion

Include a graph of the nonlinearity, and a diagram of the nodes.

8.2.8 Bit Crusher

Explain what an AudioWorklet is and why it's necessary to create this effect.

9 Precise Sequencing and Recording

For the Sequencer and Recorder, a delicate balance has to be maintained to allow scheduling blocks of future notes without using up all the browser's available memory. This should include a description of the lookahead scheduling algorithm I implemented, and how events like bpm changes affect the future-scheduled notes. Should also describe and give sample audio/screenshots of how the sequencer can lend itself to polyrhythms; and how the recorder allows for easy overdubbing to a metronome which can then be turned off.

10 Enabling Web Platform Features

This project also contains several programming decisions not directly related to audio or MIDI control. These allow for an experience that's more similar to a desktop app, like faster loading times and being able to work offline.

- Caddy on EC2 to serve with HTTPS (and movement away from AWS S3)
- ServiceWorker for offline capability
- WebPack minification and import system

11 Evaluation

Argue that the project succeeds in its goal of making it easy and fun to make music in a web browser without any installation. I'll take into account feedback/testimonials from local performing and composing musicians, and as a result critically evaluate which features were prioritised correctly/incorrectly. I'll also include snippets of audio composed by me and friends. I'll reflect on the project's availability and cross-browser support, but argue that the web platform is only getting better over time. Also look critically at the lack of a full track-based sequencer with parameter automation; reflect on the original goal to be a playground rather than a final production app.

12 Further Work

Although the project stands well on its own, there are decades of different effects and sound-generators it'd be possible to add. As well as these, it'd be great to increase the number of device types it's possible to make music with. Some future possibilities include:

- Keyboard control for people without MIDI hardware
- Parameter Automation (esp. for synth)
- Algorithmic reverb
- Monophonic Synth with Portamento
- Sidechaining
- Mobile support

13 Conclusion

Conclude by restating the aims of the project, and using the evaluation to argue that the project met those aims. Briefly say that there's scope for future work but that Minim in its current state is already useful for musicians and producers.