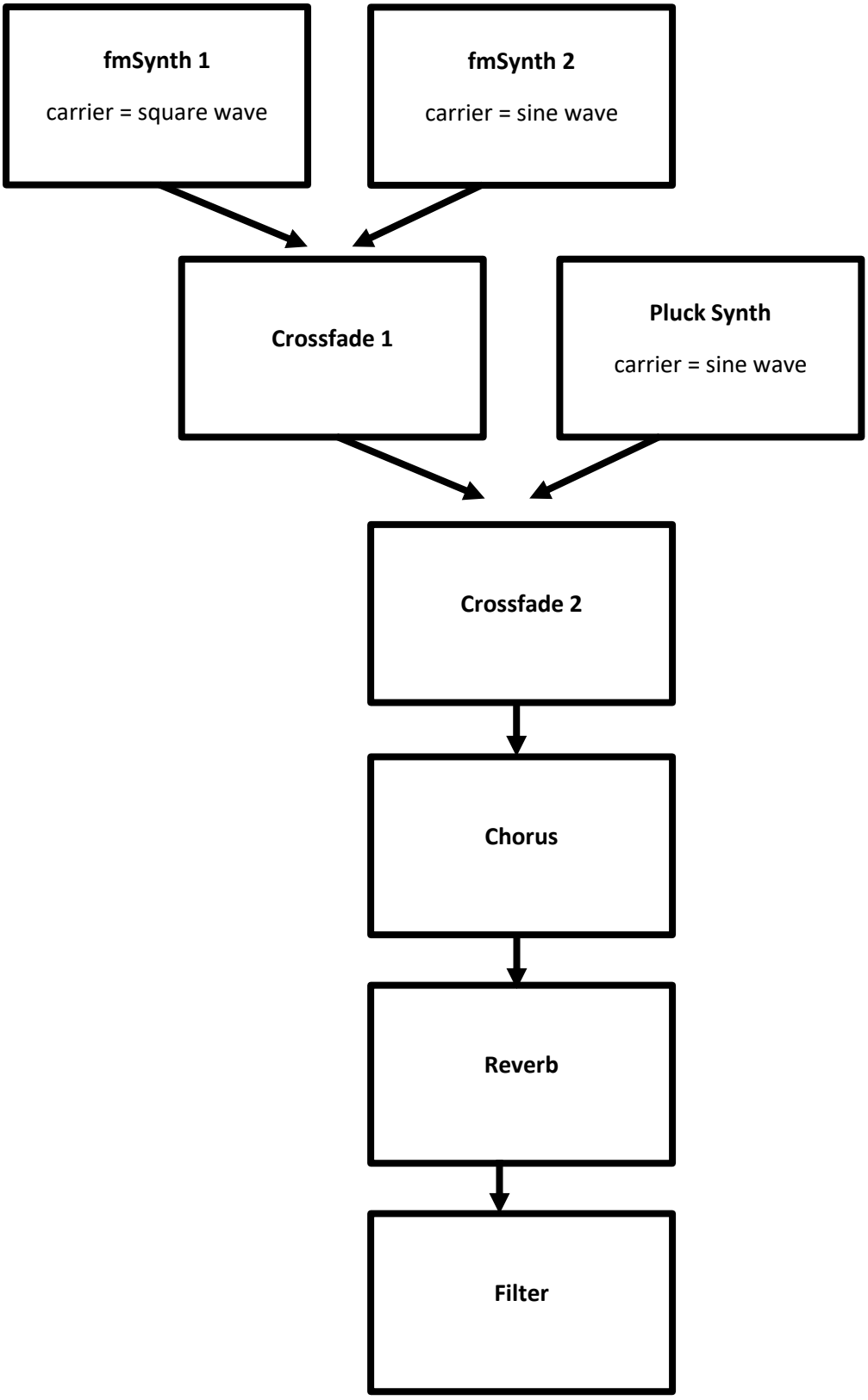
Visualizer

The width of the circle is connected to the volume of fmSynth 1, the height is connected to the volume of fmSynth 2. The colour of the circle is defined by the waveform of the audio. Each vertex is also distorted by that waveform, assigning each vertex to a new index of the waveform array.

Each particle has its own corresponding frequency and goes up according to the volume of that frequency. The colour of the particles is defined by the value of index 240 of the FFT array.

Audio



Design ideas and principle

For this assignment I had to answer the question: ‘How do you make a musical interaction playful?’.

The answer I figured for this was mainly to use many different feeling type of inputs and making sure they change the sound noticeably. This is important because I didn’t want to give labels to the settings, I wanted the user to explore the synthesizer and not think about it.

To keep it fun for a longer time I found it important to give a lot of controls, so you can keep creating new sounds without them sounding the same.

Once you change a parameter, this is immediately audible.

This music app is for children 6 to 15 and general young people on the internet who want to play with something fun.

It’s already fun to play with once you make sound, the visualizer gives extra depth to this and keeps you engaged longer. It’s also kind of fun to break the machine and make weird noises with it and to see it freak out.

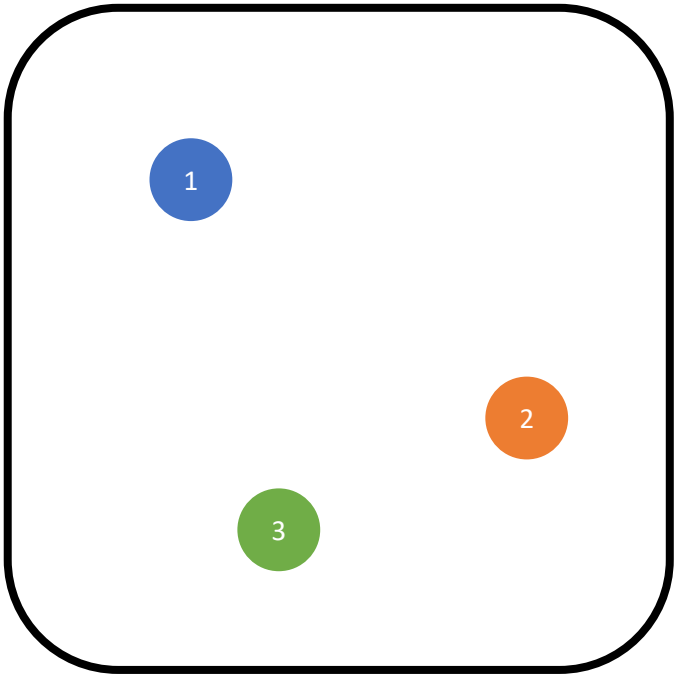
The synthesizer model is fully made in blender.

For the visualizer my first concept and idea was 3d geometry that would be distorted based on the incoming audio. I had the idea to use FFT to get multiple frequencies and map this. I wanted it to look pretty and minimalistic but also captivating.

Besides that I also wanted to look how far I could get with particles and what I could achieve with those. Get those to move around when audio is incoming.

My MVP was distorting the torus geometry, I imagined this having the most visual effect.

The technological difficulties I faced were that it’s hard in Javascript to read audio data and get useable parameters from this as a novice like me. I used tone.js for this project and their api wasn’t always as thought out as I had hoped.



1

Z -> Harmonicity

X -> Fade between Saw and Sine carrier

2

Z -> Detune FM Synth 1

X -> Dampening Pluck Synth

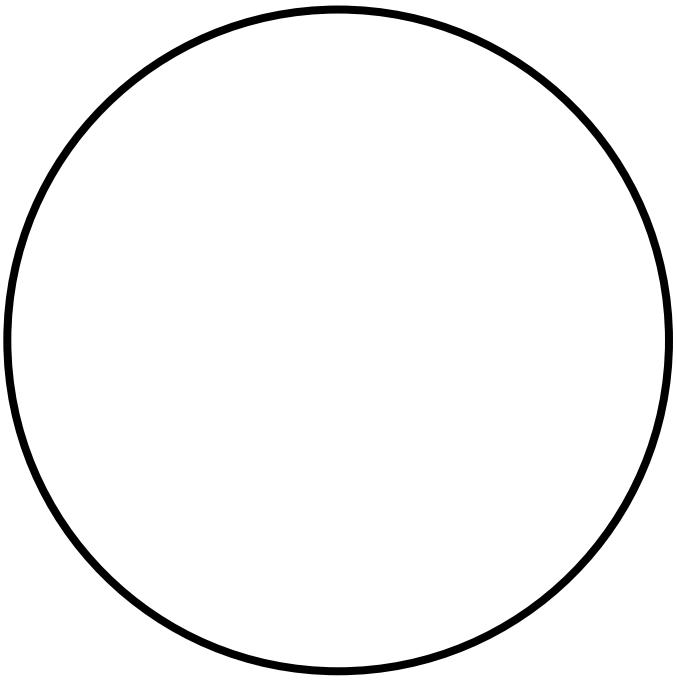
3

Z -> Chorus frequency and depth

X -> Release FM Synth 1 + 2, chorus feedback

To move the balls we cast a ray from the mouse position in the camera to the synthesizer, the location where this hits is where the ball will be moved to. I added a minimum and a maximum for this location so it stays within the square.

The balls get a random colour each time you load the page.

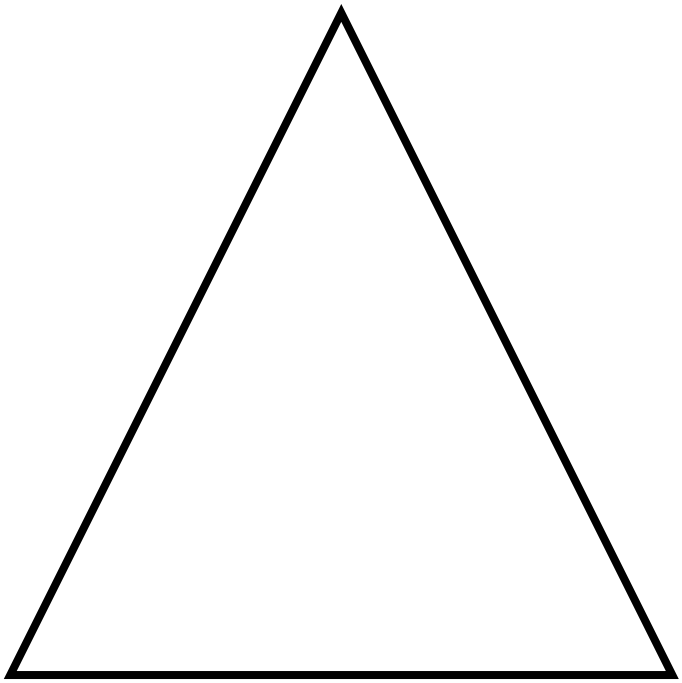


Mouse Y distance:

- reverb dampening
- chorus spread

Mouse X distance:

- reverb roomsize
- chorus delaytime



Mouse Y distance:

- fmSynth 1 + 2 modulation index
- fmSynth 1 + 2 modulation release
- fmSynth 1 + 2 modulation attack
- fmSynth 1 + 2 modulation decay
- fmSynth 1 + 2 modulation sustain
- filter frequency

Reflection on the PoC

In the end I'm very happy with how far I've come. Most of the features I had in mind are implemented and it works and looks pretty well.

There are of course a few improvements to be made.

First of all, the topology of the 3D mesh ended up pretty bad and should be remade with all edits in mind. It's also contains more vertices than it probably needs. Luckily you don't notice it hitting the performance.

The lighting and materials are also a bit amateurish and I should probably learn more about three.js materials to really get them to look polished.

The sound design could also be more thought out, right now its very easy to create harsh metallic sounds, softer sounds a little less common.

The visualiser could use some more flare, but at the same time I also kind of like the implementation as it is. It might be a bit jarring though for a new user.

One thing that could definitely improve the useability is if it had keyboard support so you can play it with you QWERTY keyboard. Midi support would also be great after that!

It is also possible to make this useable on mobile as well. Due to time constraints I wasn't able to do that part sadly.

Working on this was a bit of a challenge because I was actually working on a way different concept at first. This proved to be too hard for me because I wasn't yet comfortable with web assembly, and I invested to much time into that. I had to create a new fun idea that would look good on a resumé in the last few weeks.