fmSynth Design Process

Summary of Design Decisions

At the start, since we wanted the plugin to be able to be used regardless of the user's operating system, we decided to only edit and push the source code to Github. We would then compile the source code on our own machines and be able to run it in a host application or digital audio workstation (DAW) such as Ableton Live or FL Studio. Since the build for each operating system is different, this was the most simple and efficient way to cooperatively work together.

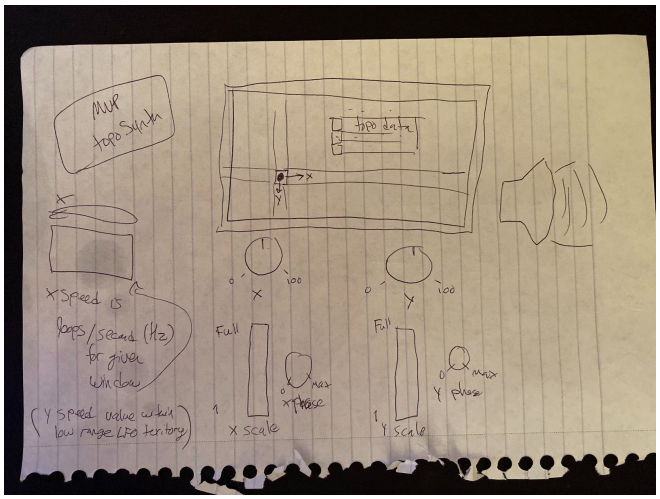
In terms of design features, as a frequency modulation synthesizer, things can get complicated very fast, so our focus was always to keep a simple design. This meant reducing the amount of knobs to only what was necessary and making it clear to the user what was being used as the carrier signal and what was being used as the modulator signal. We also wanted to implement the most necessary effects in our plugin, since so many great effect plugins already exist in the market and they could easily be used in tandem with our plugin. There would be no point in implementing a subpar effect feature, when users can easily just place a professionally made effect plugin in the effects chain, so we just focused on what our synth was meant to do - frequency modulating audio synthesis. We only added the most basic effect, which is a filter to take out any unwanted frequencies. We also added an ADSR envelope which would allow the user to "shape" the sound, and this had to be native to our plugin since it is the source of the audio. Another feature we wanted to add were presets, so that users could save a certain sound they created and can easily access it the next time they want to use it. Additionally, every parameter in the synth is able to be modulated by the host program. The most common use of this would be by using Low Frequency Oscillators (LFOs) that can control the value of a specific parameter based on a function's output over time.

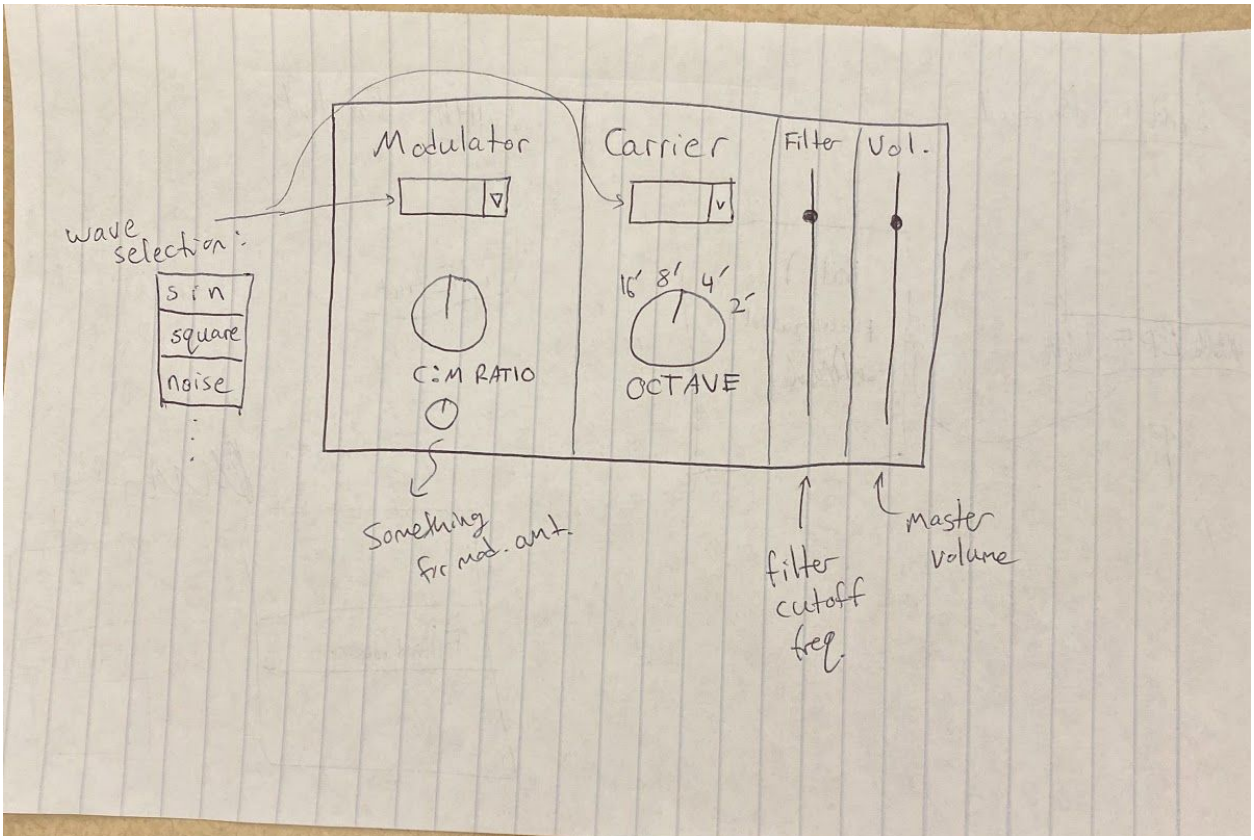
Our UI/UX focus was to have as little going on as possible, so that the user could easily navigate around the plugin. We separated the plug in into three main sections - one the carrier, one the modulator, and another for the filter/envelope/volume gain. This way the user can easily tell how the audio flow works and can clearly understand what each knob does to the audio. Our simple layout and the neutral colors of gray/blue make the plugin easy on the user’s eye and the important things are accentuated in white text so the user knows what to focus on.

Stages of Design

Sprint 01

At the end of our first sprint, we had our user stories set up and we had the overall design of our end product in mind. We had an established work flow and a working “Hello World” plugin that demonstrated that our product was working and runnable across different platforms. This is where we spent most of our time

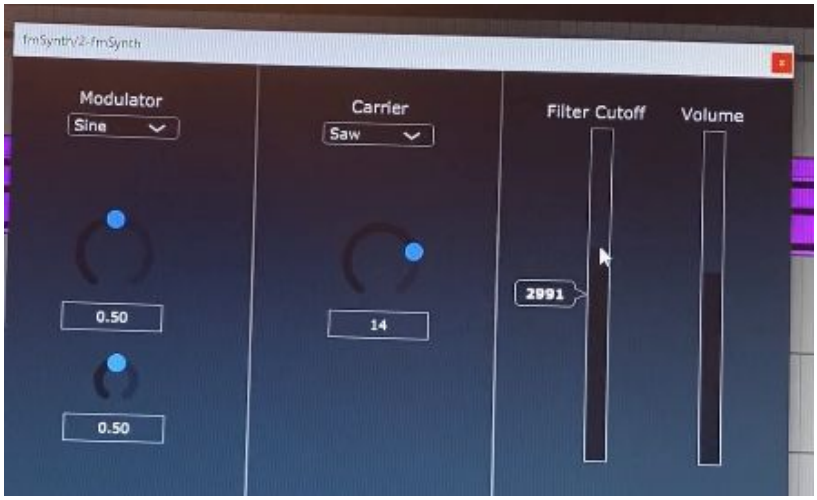




getting accustomed to working with JUCE and its development process. We made sure that each of our team's members experimented with the framework's online tutorials so that we would all have a baseline understanding and could later branch out into more specific tasks.

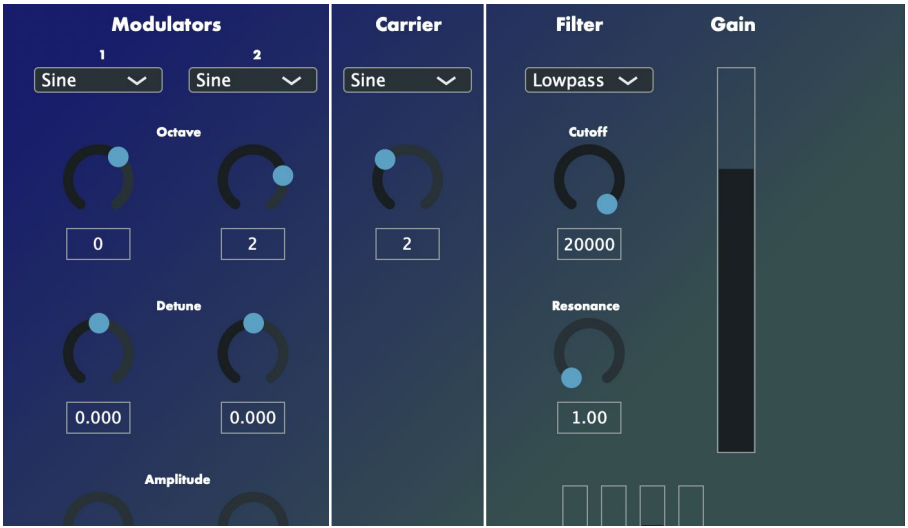
Sprint 02

At the end of our second sprint, we had our minimum viable product, which was a generic synthesizer that had three different waveforms available for the user to pick from. We also had a filter knob (only capable of low pass filtering at the time) and a master volume knob so that the user at least had some control over the sound.



Sprint 03

By the end of our third sprint/release date, we had a fully working FM synthesizer with five different waveforms, multiple filters and a working ADSR envelope. We also finished the testing process and fixed any major bugs in our product.



Testing Results

Since this is a audio plugin, doing things such as unit testing was not the easiest, as there wasn't really a "right" output for the program. Most of the testing of functionality was done by ear, and basic unit tests were implemented to test very basic functionality of the code. One small difficulty encountered was that the UnitTest suite in JUCE was bugged and would segfault on assert statements. It was easily solved by simply using assert.h from the C standard library.

Github Actions was used to automate the testing process. It first checks the cache for JUCE/FRUT and if it does not exist, downloads and compiles it. Caching the compiled version of JUCE and FRUT saves about 2 minutes each time the workflow runs. It then checks the cache for compiled code for the synth. Since not all of the code is changed each commit, caching the compiled code and rebuilding it after each workflow also saves on time.

Implementation Difficulties

When implementing the different waveforms, we initially started by creating different classes for each waveform, but this was difficult to keep track of. Since the differences between each waveform was just small differences in the math, we eventually decided to consolidate all the waveforms into one class to keep our code concise.

When implementing the ADSR envelope, it was difficult to get the ADSR envelope to work with polyphony (multiple voices). Some notes would work, when pressing other notes, but notes would not work by themselves. The bug was extremely inconsistent and hard to keep track of. We eventually realized that we weren’t incrementing correctly when passing the audio buffer.

External Resources

The JUCE framework provided us with the starter code for our synth and we also followed their online starter tutorials to start off the project. We also watched online tutorials on youtube from the official JUCE channel and The Audio Programmer Channel. During the implementation of our synth, we extensively used the JUCE library to borrow different classes, such as the ADSR class for the envelope and the building of our filters.

The Audio Programmer Youtube Channel:
https://www.youtube.com/channel/UCpKb02FsH4WH4X_2xhIoJ1A

JUCE Youtube Channel:
https://www.youtube.com/channel/UCaF6fKdDrSmPDmiZcl9KLnQ

JUCE Website:
https://juce.com/

JUCE Forum:
https://forum.juce.com/