Maverick Espinosa
04/27/24
Pure Data Final Project

Commentary:

Initially I wanted to create a vocoder that could load in an audio sample and translate the audio signals to the nearest note on a C major scale to see if it could generate an interesting melody, however I also wanted to utilize Gem and create a synthesizer with various effects. I sought out to do both but then ended up only getting a fraction of my audio player vocoder finished and was able to implement my synthesizer with an arpeggiator and visualizer. I first made an overall hierarchical class diagram of the different abstractions used in my synthesizer/ visualizer patch so that I could abstract any components like the synthesizer, visualizer, effects, Gem objects, that I would use a lot. Abstracting also really helped clean and group similar parts together so that adding new components would be a lot more efficient.

Guide:

This patch uses Gem and pure data.

This patch works by connecting your midi keyboard to the control box at the top. My midi keyboard has 8 different knobs, thus I mapped each one to a specific effect that manipulates the audio signal produced by playing a note on my keyboard. Here are the knobs and what they map to:

1) Volume
2) Arp tempo rate (beats per min)
3) Sleuth/glide amount
4) Resonance frequency cutoff (like a band pass filter)
5) Reverb amount
6) Chorus amount
7) Delay Amount
8) Panning (centered at middle, turn left to move left etc)

The synthesizer has an arpeggiator feature that can be turned on or off. When off the synthesizer plays notes with all the effects mentioned available for the user to use while playing a note.

When the Arp is turned on, it maps the first 4 notes played to be repeated. The last note of the 4 should be held down if you want the 4 arpeggiated notes to continuously play. Once you lift your finger off the key the notes will naturally decay and end

For example, if the first 4 notes I play are B,C, F, G, and I want to hear the notes repeat I would hold down on G to hear the sequence repeat.

For the Visualizer, I mapped the 8 parameters mentioned to different elements of the visuals. The main visuals associated in the Visualizer can be classified into GemShapes and GemParticles. The former include shapes like Cubes, Spheres, Cones, etc and I made it so that the first C,D,E,F,G notes when played can spawn in or delete an object. The effects that 1-8 map to are as follows:

1) Volume -> Object Size
2) BPM -> spawn rate of the objects (uses the hits of the bpm to trigger a respawn action)
3) # of hits (this is the bpm translated into a metronome tick)
4) Sleuth -> some objects have a segments amount that make the granularity of the shape higher or lower)
5) ResFrequency -> increases the speed of objects
6) Reverb -> this slows down the speed of the objects
7) Chorus -> Currently doesn't have a mapping
8) Delay -> Currently doesn't have a mapping
9) Pan -> shifts objects to the left or right

The latter includes particle effects like spawning line and cube particles.

The effects that 1-8 map to are as follows:

1) Volume -> Object Size
2) BPM -> spawn rate of particles
3) # of hits (this is the bpm translated into a metronome tick)
4) Sleuth -> increases the life of a particle
5) Res freq -> Currently doesnt have a mapping
6) Reverb amt -> this slows down the speed of the objects
7) Chorus -> changes the size of the particles
8) Delay -> changes the rotate speed of objects
9) Pan -> shifts objects to the left or right

Future Directions:

I want to add more effects like a granular synthesizer effect or even bitcrushing and extend the amount of knobs I can use to control effects on the audio signal. I want to try to possibly create a plugin out of this to have when I load up fl studio or any daw. I want to also improve the visualizer and the various kinds of visual scenes available and better learn how to use Gem. I might also look more into my vocoder patch and try to improve the timbre sometime over the summer.