Alex Henri - 40108348
CART 253 Project 2 - Progress Report

Where we left off:

The last time you saw my project, I had made a simple generative system using p5.voronoi. It consisted of sites that would orbit the center and when they went offscreen, they would respawn in the center. They could be manipulated by the user directly by jittering them, playing, pausing and toggling the stroke for both the cells and the sites. I had also implemented a very basic color shading system so that the cells would also change shade as they moved.

Where we're at:

Since my last iteration, I've done a lot of cleanup, putting the previous iteration into a class-controlled system. The cells and diagram are now controlled as objects. I also added in a music class which allows me to load music samples and allows me to get info on various parts of the audio stream.

Some noteworthy features:

- Individually spawning cells with R.
- 5 Demo music tracks to listen and watch with 1, 2, 3, 4 and 5.
- Added "behaviors" to cells, allowing them to behave differently.
- Cell behaviors are given on spawn and persist until respawn.
- All parameters of any cell can be accessed and changed at any time.
- Basic beat detection using PeakDetect from p5.sound
- Added basic effects: whiteout, blackout, addjitter
- Bass hits = jitter
- Claps/Snares = whiteout

The road here wasn't easy:

I designed the audio system independently of my last prototype iteration to make sure that I could implement it easily. I wanted to make this feel modular and manageable so I designed the music class which tracks the current song, prints info about the song and can tell me anything I need to know about the song using fast fourier transform, FFT from p5.sound. This class stores the songs and can switch between them based on keyboard input using a class method. It was difficult trying to get the audio to play properly. For example, to switch tracks, I had to stop the previous track, clear the current song, set the current song to the new song and then start looping it. The play/pause feature loved interrupting this process but I *think* I have it in a stable state.

In the same stroke, I wanted to also clean up the prototype and have it run in its own class as well. I had already started doing this with my cell class but I wanted no code in the driver program besides the "update the diagram" method. I had to rewrite the code, replacing variables meticulously with “this. ____”. Besides that being annoying, I also took the time to streamline the code a bit. I will be honest, this transition broke my prototype and I was panicking because suddenly, my computer would lag after only 20 cells when my prototype had been able to run 250 with no issues. Of course, it was a misplaced curly bracket causing all the trouble, as is most of the time.

Jokes aside, this was a difficult process but also a totally-necessary one. I rebuilt how the cells evolve by giving them individual behaviors. So far, I have 3 behaviors, the orbiting spiral from the prototype, a new one called “shooting star” which spawns a cell and shoots it off the screen and one more called “blink” which will spawn a cell somewhere on the canvas randomly and when it’s given a queue, it will teleport the cell a small distance away in a random direction. I have not implemented it into the actual program just yet but the code works and is ready to be added once I figure out how and when I can give the cells their queues.

Queues is the way I would describe the way my program functions at the moment. I have 5 frequencies I’m analyzing for “energy” (thanks, peakdetect!). The hardest part of the designing up until now has been designing the ways that energy is visualized in the diagram. For example, when the bass hits, *What does that look like?* At least for me, it’s a low, visceral tremble so I’ve made it such that the diagram will shake. The same reasoning applies to a clap/snare hit. For this, I designed effects. Effects live in the diagram class and are applied to the entirety of the diagram. So far, I have 2 effects: whiteout and blackout. Whiteout makes everything white and blackout makes everything black. I have the program setup in such a way that the cells want to return to their original state, meaning that if they were to speed up, slow down, change color, etc. they would undo these effects as they evolve.

I had to tweak the sensitivity and ranges of the frequency bands to capture the claps and bass alike. This will surely take more tweaking but for the time being, the idea is there and it functions.

Going forward:

Going forward, I will be designing more effects and behaviors for the cells so that we can begin seeing some more interesting visualizations. I will spend a great deal of time tweaking these to make sure they capture the essence of the music. I will also add mic input and a more accessible control panel to adjust parameters of the diagram on the fly.