Ernes Railey

12/14/2018

GAME 235 Final Project

Project Postmortem

**the cure**

“the cure” is a game during which the player develops and tests the durability of some unknown anti-virus referred to as “the cure” under the direction of the mysterious second party. After completing the initial development, the player must keep the testing cross-hairs (which spawn every time the minim FFT analyzer detects a certain frequency range) from completely destroying their work. Passing the durability test results in a relatively optimistic ending. Failing, on the other hand, dooms humanity.

**Development**

When I began the project, I originally planned to recreate a particle swarming effect that I’d come across while browsing projects on Open Processing and use it as an audio visualizer. At first, I approached the project using the standard rules of particle flocking (e.g. cohesion, seeking, alignment, etc.) and created a preliminary version that came close to what I was looking for, but wasn’t quite there. However, after reviewing examples of the swarming effect, I took a different approach that plotted each particle a set distance away from other particles using the circle equation. Once I completed that particle effect, I tuned various fields (mass (for the radius sizes), viscosity, max number of particles on the screen, etc.) until I felt comfortable with the effect that I’d created).

I wasn’t sure how I would implement the audio analyzer until I’d nearly completed the swarm effect. While tuning the particles, I had a sudden idea and began looking into Processing libraries that would handle user input. From there, I developed a narrative where the player was a scientific doctor of some sort that would develop and test a cure (the particle swarm).

The crosshairs weren’t mobile in my original idea. Mobility was something they inherited from the parent class (that I forgot was there), but the result was interesting, so I left it in. I also planned for the radius of each crosshair to increase in size in relation to their opacity. I created this effect in a separate sketch but decided not to use it in the end.

**Learning Points**

Ultimately, I learned that:

1) There’s usually an easier way to do whatever you’ve set out to do. I was really excited to try writing a custom flocking algorithm, but it turned out to be a lot more difficult to create the effect that I wanted using that approach. In the end, I had to throw away all the code that I’d initially written for particle interaction. I tried implementing some of it for what I thought would be a cool effect in the intro, but between the controlP5 library and minim, Processing’s performance on my computer took a nosedive, so it didn’t make it to the final product.

2) Trying to make a spawner using a frequency analyzer is a lot more complicated than I thought, and requires more testing than I was willing to put in. To test the spawner, I created a different sketch that drew particles in random places on the sketch whenever the frequency analyzer detected a certain range of frequencies from the music. I was trying to make each play through a little different by having the crosshairs spawn in accordance to the music rather than using random() or a timer. However, when I implemented it with the crosshair’s destroy function, the game became impossible, so I had to limit the frequency range even further. The crosshairs still spawned too frequently for my liking, but I was content enough to work on the fading aspect.

I think I benefited from the flexibility of my original idea and my willingness to pivot early instead of doubling down on a specific aspect of my idea. Furthermore, separating game state/new feature into separate sketches and testing their functionality before adding them to the final sketch was the best approach for this project. I do realize that I could have used GitHub’s branches, but most of the time, creating a new sketch to try out an idea was faster.

I really enjoyed creating this (although debugging was a nightmare), and I plan to keep working on it eventually. I’d like to make the spawning closer to what I’d originally intended and make more interesting tests for the player.

3/23/2023 Update

This project has been on my mind for the past few years, and after *finally* tracking down the original ControlP5 library, I managed to fix a few persistent bugs. I also changed the FFT analyzer detection, opting for detecting “loudness” dB rather than a specific range of frequencies (the implementation of which was a little arbitrary and too sensitive for my liking).