ProcJam Writeup

Cole Granof Joseph Petitti

November 10, 2019

1. The goal of our project was originally to make a procedurally generated top-down twinstick shooter game in the style of *The Binding of Isaac* or *Nuclear Throne* using parameterized design. We wanted to have levels be generated by using Conway's Game of Life to create natural-looking cave-like structures, and generate the enemies by randomly assigning them a number of points in various attributes. We wanted to make a fun game with simple gameplay that is unique each time you play it.

   We planned to implement this project using plain JavaScript and HTML, without the use of any external libraries or game engines.

2. The project we ended up creating has a lot of the functionality of the game we intended to create, but is missing a few features due to time constraints. We implemented a level generator, enemy generator, and basic physics engine using plain JavaScript and HTML. The JavaScript code is organized into classes according to object-oriented programming concepts. We utilize ES6 modules to smartly include them in the HTML document.

   Our project can generate cave-like levels by running a modified version of Conway's Game of Life for several generations with specific rules. It then fills these levels with enemies that are also procedurally generated. Each enemy has a randomly chosen difficulty level which determines the number of points it gets to put into each of movement speed, shot speed, accuracy, and rate of fire. Enemies also have procedurally generated shapes, colors, and faces.

   Due to the time constraints of ProcJam, we weren't able to finish the interactive aspects of the project in time. For now, it's just a rudimentary engine and level generator, but it would not be too difficult to add more gameplay components in the future.

3. We chose to use JavaScript because that is the language we have the most experience with in creating graphical applications. JavaScript's canvas API makes it easy to draw simple shapes on the screen without having to configure external libraries or deal with platform-specific issues. Making the project as a static HTML document means it's easy to serve and show off over the web.

The downside of our approach was that we had to implement a lot of basic features that a game engine would normally have from scratch, such as a Vector class and a system for updating game object positions at a fixed rate. If we had to start from scratch we might decide to use a simple JavaScript game engine to save us the time of reimplementing these features.

4. In developing this project we learned a lot about developing procedurally generated content like enemies and levels. Cellular automata can be a powerful technique for creating natural-looking content. Further, a few basic variations in color, eye shapes, and mouth shapes can lead to a lot of variety in enemy types.

5. Cole Granof monitored the #procjam hashtag on Twitter in order to draw inspiration from other peoples' projects. When the project neared a state of completion, he created a tweet with screenshots, and threaded that tweet with a link to interact with the generator on his personal website. Cole doesn't have the most impressive Twitter following so the tweet did not gain a lot of traction. However, one person who liked the tweet was user @KilledByAPixel. Completely coincidentally, Cole recognized this handle from dwitter.net, a website where you can post 240-character JavaScript scripts that render to a canvas. This user has created a significant amount of generative art in the past.