Hans Heidmann
CSCI 4448
Fall 2022

Blood Red Skies

Video Demonstration Link - https://youtu.be/r08W2V7sXMI

1. Info
    - Project Name: “Blood Red Skies” (formerly “top-down-survival-shooter”)
    - Team Members: Hans Heidmann (former member Peter Loden was not with me during this project, he told me he does not need the points)

2. Final State of System Statement
    - Implementing an MVC system to control view switching between the main scenes (menu, game, leaderboard, etc) was a minor challenge, but starting off by building a nice clean system made all of the following development rapid and implementation of new parts was surprisingly simple. I haven’t done much game development in JavaScript but I did a lot in Macromedia/Adobe Flash back in the day and have done a bit in Unity more recently, so I’m aware of common practices and general videogame architecture. While implementing the Strategy and Flyweight patterns during this project, I quickly realized more than ever that using design patterns could have prevented so many headaches in the past that were due to messy, unorganized, hacked-together code! While programming games that include visual rendering like this one, I’ve noticed that it is easy to accidentally fall into a habit of using the visual elements to inspect code changes when in reality deeper checks need to be implemented in order to avoid code smells and awful bugs that are a nightmare to debug. The image rendering/manipulation library I used (PIXI.js) is actually an exciting little gem that the creators actually built to resemble many of the features of Flash, which I find exciting and plan to explore its potential much more in the future than I had time to during this project. I honestly could have done without the audio library I used (Howler.js), since the only features I ended up using were the playback-speed modifier to make the player’s footsteps sound unique and the loop feature to repeat the music when it ends. At the time of writing this, I haven’t yet had time [but plan] to add “powerup” drops from the enemies such as new guns, gun-specific ammo, health packs, and so on. I will also be adding a bit more to the GUI to show which gun is currently being held, so the player doesn’t need to rely on the ammo indicator to know. The monster AI could use some work to improve/balance gameplay a bit so that scoring high is more so due to player strategy than randomness of events. Then I will be sharing it with my friends and students to have them test it and provide feedback, so I can improve it and possibly monetize it at some point.

Hans Heidmann
CSCI 4448
Fall 2022

3. Final Class Diagram and Comparison Statement
    - Updated UML Class Diagram (it's also in the project repo if link breaks)
    - Project 5 UML Class Diagram (it's also in the project repo if link breaks)
    - Key Changes:
        i. I ended up making quite a few changes as my project matured. One of the major changes was the way I handled guns and shooting. I was originally planning on using only inheritance with abstract gun and bullet objects, each having concrete extensions for different gun/bullet types. When the time came around to write the code, I decided to use the Strategy Pattern, since I made 3 guns that shoot() differently. Turns out using a single bullet object was fine since I was able to give it attributes for gunType, damage, display size, speed, etc.
        ii. The other big change I will mention is that I didn't get a chance to implement the Factory Pattern for creation of my enemy monsters. I had already implemented 4 other patterns and was basically just out of time even though I want to add it, but I do plan to continue working on this game for fun and will be trying either Factory or Builder for the originally intended purpose, especially because I will be adding in multiple types of monsters to make the game more interesting.

4. Third-Party code vs. Original code Statement
    - All of the code written for this game was completely original!
    - Examples were referenced (but not copied) from the official documents for Firebase, PIXI.js, and Howler.js

5. Statement on the OOAD process for your overall Semester Project
    - Choosing the ideal design pattern for a certain aspect of a game can be tough and while there is probably not a "right" answer in many cases, I am now aware that certain patterns will generally scale better for certain game architectures.
    - As I mentioned in my Final Statement, it is easy to fall into a bad habit of assuming changes of visual elements accurately reflect major changes in code, when in fact this is not the case. It took a couple terrifying debugging sessions due to falling victim to this practice to really make me want to implement better tests and checkpoints in my code to avoid future headaches and panic!