CSCI 4940 – Capstone Project

Group #5

Members: Morgan Channel – Raymond Chui – Christopher Clark – Cedric Crawford

Professor: Dr. Ge

March Progress Report

# Problem Formulation

For the month of March, we worked on making the project more object oriented. We decided on this change because previously the different objects in the rooms would need to have it’s code rewritten for every room. This change involves making most objects and assets their own scenes so that implementing them in different rooms would be easier.

# Analysis of the Problem

With the way the project is currently designed, we would have to rewrite all the code every time we would like to reuse an entity in our game. This could make the game more unstable and it would take a significant amount of time to complete if we decided to continue in that direction. We decided that it was necessary to make the game more object oriented to ease the process of creating more rooms in the dungeon and speed up our overall progress. This change will also trim down the amount of storage space the game will use and allow the game run more consistently.

# Implementation

We have already started on making each entity in the game its own scene to make it easier when we start creating new rooms with recycled resources.

# April Goals

The goal for next month is to have all the dungeon rooms connected so that the player will be able to fully traverse the dungeon. We would like to have game audio implemented into the game in the coming month. This would be enemy attack audio and background music for the dungeon. We will also work on designing a mini boss and finishing the scene for the main boss fight.

# Member Contributions

# Christopher Clark

This month the focus of my work has been to develop a structure for the project to allow us to reuse most of our creations in the game. The model we are using for our game now involves programming and designing every entity in our game inside of its own scene. A scene in Godot is a collection of sprites, animations, and similar things, all of which are commonly called Nodes. A scene also contains the configuration for each node, including a reference to the code attached to it, stored in a separate script file. Each scene can be instanced as a child in multiple other scenes, with changes to the entity’s scene being persistent in every scene it is used. So far, entities that have been implemented to follow this model include the Player entity, the targeting pigeon which is spawned for targeting, the Katydid enemy, the walls, the floors, the torches, and doors. Since the walls and some of the floors contain collision boxes setup with them, it saves a massive amount of time to make these reusable, as the collision boxes don’t need to be setup manually in every scene that includes them.

I also set up some code in a Global.gd script file. Code in here is meant to be accessible from scripts anywhere in the game. It includes a variable storing the instance of the Player entity because the Player should only ever be loaded once and remain persistent across multiple scenes. In addition, there is code in Global for changing the current scene, that way scene changing remains consistent no matter what initiates it. It also contains code for spawning entities as children to the current scene, for entities that follow the player from scene-to-scene.

Finally, I rewrote all of the prototype code for the Player entity so that it is more efficient and organized.

# Raymond Chui

# Morgan Channell

# Cedric Crawford