**Design**

My 15-112 Term Project is a horror game based off the game Silent Hill. At first the user is taking to the main menu screen which could lead to four other screens. In these other screens, except the actual game screen, there is a back button that takes the user back to the main menu. The first screen is the credits screen where I give credit to my professors and the CA’s that have helped me this semester. The second screen is the “how to play” screen where I describe how the game is played. The page tells the user to use the buttons “W,” “A,” “S,” and “D” to move around, “Space” to shoot, and “Tab” to open the doors. It is also gives the user a brief introduction to the game. The third screen is the “high scores” screen where the top 5 scores of all time are displayed. In my code, when the game is over, I save the current score, date, and time to a text file. In the draw function for the high scores screen, I take the top 5 scores from the text file that I wrote and I sort them from highest to lowest. They are then displayed on the screen.

The last screen is the screen for actually playing the game. There is a pause button that leads to a pause menu, where the user can either resume, restart, or go to the main menu. When the game is over, there is a game over menu, where the user can either restart or go to the main menu. For the actual game, there are 6 main classes that my game employs. All the classes, except the “Game” and the “PygameGame” classes, inherit from Pygame’s “pygame.sprite.Sprite” class; this is uses for collision detection and is generally more efficient when I want to update all objects on the screen.

The most basic class is the “Bullet” class, where I load 4 images of the bullet, each image showing the bullet moving either up, down, right, or left.

The “Room” class is everything that belongs to a room when the player goes into it. The class’ most important attributes include the following sprite groups: health, sanity, ammo, gold, wall, and monster. These groups are created in this “Room” class, so each room has its own unique objects in their own unique positions. The class will randomly add a number of “Health” objects, “Sanity” objects, “Gold” objects, and “Ammo” objects to their respective groups so that the player can collect them whenever they are in a certain room. The Health, Sanity, Gold, and Ammo objects all inherit from the parent “Collectibles” class. (“Wall” and “Monster” objects are added to their sprite groups later on in the Game class.)

The “Wall” class is for the walls that the players can build. The First Wall, Second Wall, and Third Wall classes all inherit from the class depending on their cost and the image they blit to the screen.

The “Monster” class is for the monsters. Its attributes include an x, y, dx, dy, which are passed in and modified in the Game class. These control where the monsters are randomly spawned and which direction they are going at what velocity. The class also includes an overlap variable which becomes True once the Game class detects the monsters overlapping. If the overlap variable becomes True, the monster will change direction or stop moving.

The “PygameGame” class is the Pygame framework created by Lukas Peraza. This parent class initializes init() which is in the child “Game” class. One modification I made to the framework is calling the preInit() function before the while loop. This function in the Game class includes bigger images and variables that won’t be constantly modified throughout the code, so they only need to be initialized once when the game is first launched. These variables won’t be unnecessarily recreated and redefined every time the game is over or the game restarts. Even though initialization is still slow in the beginning when the game launches for the first time (due to the large size of the PNG’s it needs to load), the game won't be slow whenever init() needs to be called again.

Finally, the “Game” class inherits from PygameGame framework class. This is the main class that the game needs to run from. There are many things this class does. It draws everything, depending on what the current mode/screen is. It keeps track of which button is pressed. It keeps track of when a wave ends and when a wave begins. It calculates the shortest distance by looping over a list of possible directions a monster could go towards the player. When the game is over, it saves the final score to a text file. It keeps track of what button is pressed and whether the player is moving to another room. It makes sure the monsters don’t overlap with each other. It spawns monsters and moves them. Most importantly, it keeps track of collisions between the sprite groups.

One feature that I gained from the feedback session is that my friend said the game was a bit too hard. Therefore, I made the intervals between each wave longer so that the user has more time go collect items, and I gave the user more ammo to start with.