**Salvation**

Keltin Grimes

**Project Description**

Salvation will be a 3-dimensional dungeon-like first-person shooter where you must battle through multiple monster-filled levels to escape to salvation. Hellish creatures will continually spawn as you blast your way to the exit with various hit-scan weapons, picking up useful items and powerups on your way. Upon completion of the campaign, a series of increasingly difficult levels, you will be able to create your own maps, with all the features of the pre-made levels. User map-creation will be possible thanks to the ray-casting technique, which will render a 2D map into a 3D perspective.

**Competitive Analysis**

Ray-casting was a technique used by game developers in the 1990’s to render 3-dimensional graphics on limited hardware. The most famous example is Wolfenstein 3D, a pioneer in the use of ray-casting which had textured walls, animated sprites, collectible items, and back and forth shooting.[[1]](#footnote-1) Improvements in processing and storage have now left ray-casting a largely obsolete technology, but it can still be used to create a retro feel. While the quality of Wolfenstein will be difficult to match, as is was the product of an entire development studio, and my gameplay will be fairly similar, the level-editor will distinguish my project from this classic, as the game had only premade levels.

There have been a couple of 15-112 term projects that have used the ray-casting technique. Abscond: The Vaults of Vija is a fast-paced 3D maze running game, but the gameplay is not very interactive as there are no enemies or items. In terms of features, Maze is the most similar to my proposed project. It has a ray-casted 3D world with path-finding enemies, collectible items, and a level editor. Additionally, it has random map generation and the ability to throw objects, which I will likely not be implementing. However, the graphics are simplistic and the game is winnable in just a few seconds, whereas all elements of my game will be textured and will have a meaningful progression. The gameplay of PYDOOM is quite similar to what I envision for Salvation, but has no level editor and uses a different rendering technique with an entirely different feel.

**Structural Plan**

I have created a modal-based structure around a standard pygame loop, which allows me develop various modes independently. The main file contains the Salvation class which initializes the various modes and contains the game loop which distributes information to the current mode. Another file contains the code for the different modes. The Mode superclass contains a general event handler, with various functions to be overridden by specific game modes. Within this file are the MainMenuMode, CampaignMode, and LevelEditorMode sub-classes, which each have unique properties and user interactions. Additional files will be for the Button class which allows for interactive use of the graphical interface, the Enemy class for path-finding enemies, and the Item class for the various collectibles and powerups. The ‘Assets’ folder holds all of the textures for the game.

**Algorithmic Plan**

The most difficult part of the project will be the interaction between the player and the enemy monsters. When ray-casting for item or enemy sprites, a buffer is created to store the distance to the nearest wall, and a sprite is only drawn if it is closer than the wall. If that is the case and the sprite is in the middle of the screen it will be flagged for being able to be shot. The enemies will have melee attacks, attempted at timed intervals, and will only injure the player if within a certain distance. Checking the distance for enemy attacks will only occur if the enemy and player are in adjacent grid cells for greater speed. The monsters will move by randomly picking a forward cell to move to, with directions being weighted by their closeness to the player, incentivizing lifelike and non-repetitive movement.

**Timeline Plan**

The modal shell of the game has already been developed, so remaining work will focus on specific game modes. As of now the main menu is complete. The timeline is as follows:

Wednesday 11/20: Submit TP1 Deliverable and complete enemy superclass

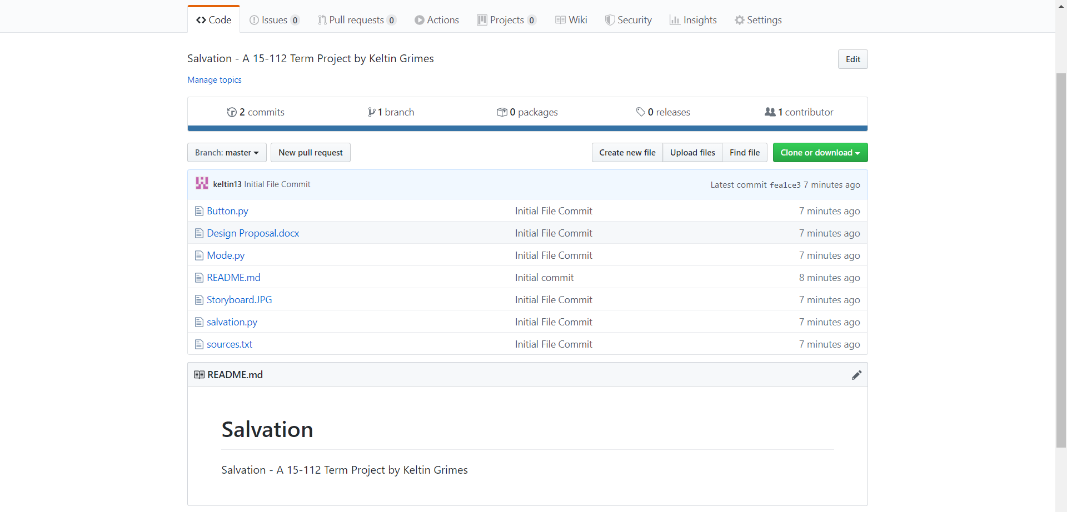
Saturday 11/23: Create monster-spawner and enemy movement

Sunday 11/24: Implement items and backpack, design a number of levels for the campaign

Tuesday 11/26: Finish level-editor and submit TP2 Deliverable

**Version Control Plan**

I am using GitHub to manage and save various versions of my code, as shown below:



**Module List**

I will be using pygame.

**TP2 Updates**

I have added an InterlevelMode for navigating between levels and added an Animations file which holds the Bullet class for animating the firing of the gun.

**TP3 Updates**

I have added random map generation, constraints to make sure user-made levels don’t crash, openable doors, improved wall collision detection, a more robust GUI, two new types of enemies with different looks, health, speed, and attack strength, and generally fixed bugs and made the experience more fluid. From the peer feedback session, I learned that the double damage powerup should have a more obvious effect, so I added a gold ring to the screen when it is picked up.

1. <https://store.steampowered.com/app/2270/Wolfenstein_3D/> [↑](#footnote-ref-1)