Score : 92/100
the Test ID not named properly(digit missing)

Zombie Dash Final Report

Prepared by
Nick Toledo, Omar Flores,
Yash Shah, and Stephen Lambert
in CS 440
at the
University of Illinois Chicago

November 2020

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

Zombie Dash is a map-based zombie survival game with strategy elements. The strategy elements are added as puzzles in the form of mazes as the maps. Players start the game with limited ammo and must proceed through the maze encountering or avoiding zombies and picking up resources along the way. Players must escape from the maze within the time limit before the map is overrun with zombies and the game ends.

2 Project Domain

The domain of the project is the video game industry, specifically the audience of gamers that are interested in the Zombie genre. This project will deliver a new Zombie game experience not currently available on the market. Inspired by the love of games from the original group to deliver a product worthy of other gamers.

3 Relationship to Other Documents

This document makes references and draws inspiration from the original Zombie Dash development report written by Group 14 in CS 440, Spring 2019 [1].

4 Naming Conventions and Definitions

4a Definitions of Key Terms

AI: Artificial intelligence, referring to the movement and behavior of NPCs.

Non-Playable Character: Refers to characters not controlled by the player such as the zombies.

Tiled: A tile-based map editor used to create tile-based maps.

Ads: Advertisements/commercials.

Energy: A type of in-game currency used to play the game.

4b UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [2]. Any exceptions are noted where used.

4c Data Dictionary for Any Included Models

Starting player equipment includes a pistol. machine-gun, and shotgun.

Player starts with 50 machine-gun ammo, and 10 shotgun ammo.

Player Energy ranges from values 0-100.

Player Health ranges from value 0-100.

Zombie Health starts value 30 and can drop down to 0.

Player does 10 damage to zombies by using gunfire bullets.

Zombies do 2.5 damage to the player's health by melee or through gunfire bullets.

Game time starts at 300 seconds and drops down to 0.

II Project Deliverables

5 First Release

Our product was set to release October 12, 2020 at 8am. For our first release we focused mainly on the user interface of the system. This included a login menu for the application, as well as the main menu, and menus for other options. Other menu interfaces were created for the multiplayer, and single player. We implemented an energy system into the game which limited playtime to users, however players were able to refill energy multiple ways such as watching ads or buying more energy.

We also created an in-game shop and implemented an in-game currency which could be used at the shop, during this release you could only purchase additional energy to continue playing. An options menu was also designed, however none of the actual options changed any gameplay. Throughout all the menu interfaces the player's current energy level is displayed as well as the player's username on the top. Every time the player would start a game, the player's energy would drop by 10. When a player's energy dropped to 0, the player could no longer start any more games and had to refill his energy.

Our main menu draws inspiration from the original group's main menu concept from the Zombie Dash report shown below.

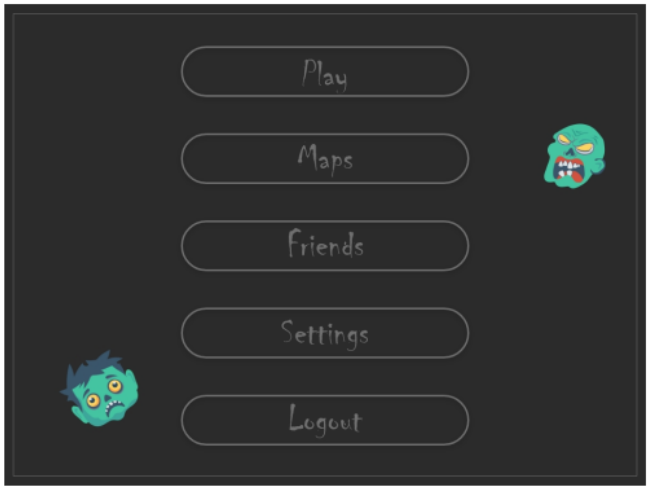

Figure 1 - Menu GUI Concept

6 Second Release

The second release primarily focused on the gameplay elements of the game. The second release allowed for a player to start a game. Upon starting he would be spawned with limited ammo and surrounded by zombies. The gameplay interface displayed a health bar at the top left, and information about the current weapon and ammo on the top right. The player could move throughout the map using arrow keys, shoot with a space bar, and change weapons with the left shift button.

Zombie AIs were also introduced in this demo. Zombies would spawn with random movement speeds, as well as have a chance to spawn with or without a weapon. Zombies would begin to follow the player when getting too close and stop chasing when distance grew too large. However, zombies would continue to chase indefinitely if they were shot.

The energy system was expanded further by implementing ads into the game to refill energy. These ads were added as 11 separate videos all part of a single ad campaign. When first watching an ad, you would get the first ad in the series. When watching another ad, the next ad in the series would be played instead of the same as previously watched. This feature makes the ad watching experience much more enjoyable by having a linear storyline between all the ads. With the bonus of being rewarded 10 energy for watching the ad.

Additional features included in-game shop functionality allowing users to purchase new character sprites to play as. As well as the addition of consumables/items dropped randomly by zombies for the player's use. These include med kits, as well as ammo for both the shotgun and machine-gun. This release also included the addition of 4 maps, along with an in-game timer, and a finish line for each respective maze.

7 Comparison with Original Project Design Document

Our prototype compares well to the full project described, we hit most of the general ideas presented by the previous group. The previous group wanted to include puzzle elements into a zombie game, which we did in the form of mazes as the maps to play on. The previous group wanted a way to log in and out which we were able to accomplish, however no authentication was created at this point of the project to focus on more important aspects of the system. The previous group also wanted there to be a noise system which notified zombies of close players which we were able to implement by having the zombies begin chasing the player after getting too close.

However, there were some elements that we were not able to get working. One of these being the inclusion of real-time based maps. The original group wanted maps to be generated using resources such as google maps or uploading maps. Unfortunately, the previous group did not make it clear as to how this functionality would be implemented, and as such we had a very hard time getting this to work properly. Instead we went with creating maps in a tile editor program called tiled, as a way to get the initial gameplay working. In the future we would like to find a way to generate tile-based maps from real-time based maps. We were also not able to complete the

multiplayer aspect of the game, instead we focused primarily on getting the single player working correctly.

Some personal inclusions we added to the project were the addition of an in-game store which drew inspiration from the original groups idea to potentially add cosmetics in the future. Additional items we added to the in-game shop such as energy refills and different characters to play as. We also added the in-game energy system to promote the use of the in-game store and watching ads.

Below is an activity diagram from the original group showcasing some of the use cases for the player.

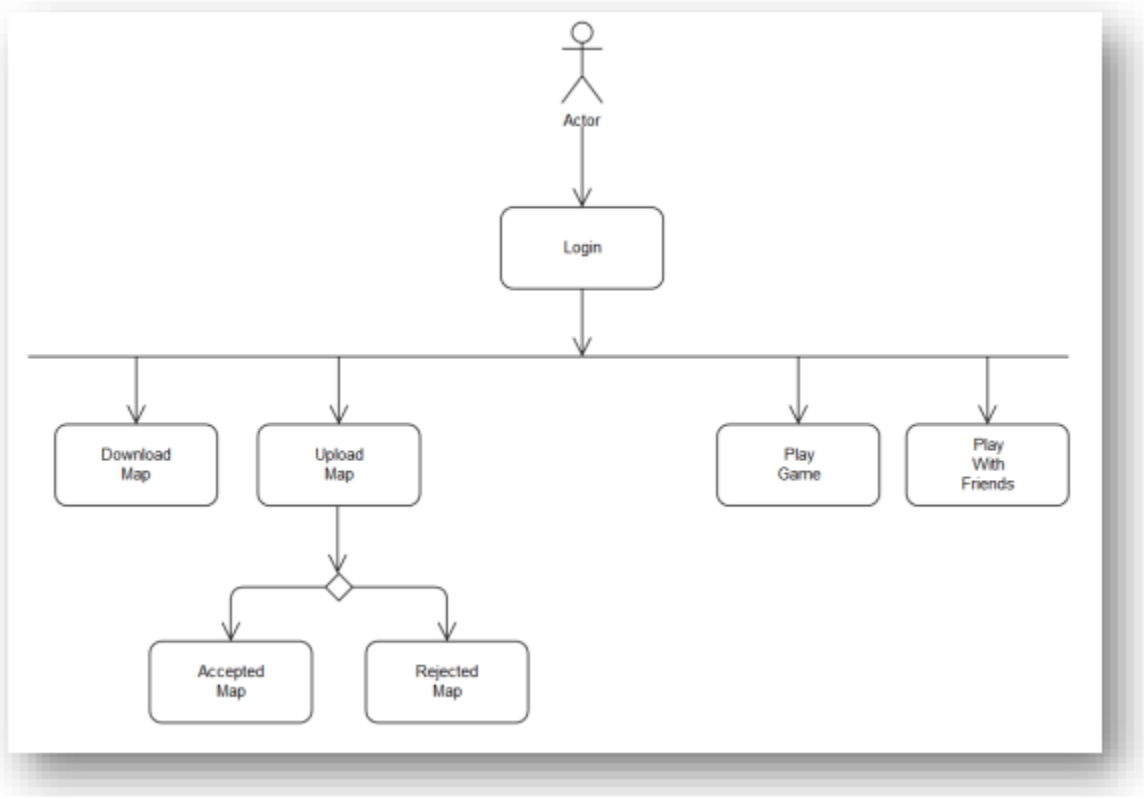

Figure 2 - Activity Diagram

III Testing

8 Items to be Tested

The functionality that we tested in this game is the login, main menu, in game shop, energy system, gameplay, advertisements, selecting maps, and starting a game. It was important to test these functions, so that we could ensure that we met our functionality goals. We tested the logging and main menu functionality by ensuring that players could type a username, password, and login using a button. The main menu was tested by ensuring that buttons appeared when the game started and by testing different

buttons, such as, in game shops, ads, and maps to ensure that players could select items from the shop, watch ads for energy, and select maps for the game.

The energy system and ads were tested by watching ads in the game and testing players' weapons, to make certain that players actually gained energy by watching ads and killing zombies in the game. Game play and selecting maps were tested by selecting a map from the main menu and playing the game, to make sure that zombies attacked the player, to guarantee that players could collect ammo and kill zombies, and to make sure that players could successfully find the end of the maze.

9 Test Specifications

ID# - Map Selection

Description: Tests the selection and addition of maps.

Items covered by this test: Map selection and additions.

Requirements addressed by this test: This test addresses requirement #40 from the Zombie Dash Project Report.

Environmental needs: Map files must be tsx files created using the Tiled map editor application.

Intercase Dependencies: NA

Test Procedures: Start game, log in, select play game, select one of four maps to play on.

Input Specification: This test takes in a tsx file as input with objects for the player, walls, and zombies.

Output Specifications: Expected output is the loading and display of the correct map chosen by the user.

Pass/Fail Criteria: This test passes when add a tsx file to the map folder, and can start the map from the map selection screen.

ID# - Movement

Description: Test player movement based on user input.

Items covered by this test: This test covers player rotation as well as forward and back movement on user inputs.

Requirements addressed by this test: N/A.

Environmental needs: Player must use keyboard as input device.

Intercase Dependencies: N/A

Test Procedures: User starts up application, chooses "Play Game" option, selects a map, and proceeds to play the game controlling character using movement controls.

Input Specification: Input movement includes keyboard arrow keys (Left, Right, Up, and Down).

Output Specifications: Output expected for this test included the rotation of the character sprite based on Left and Right arrow key inputs. As well as forward and backward movement using the up and down arrow keys.

Pass/Fail Criteria: This test passes when the user is able to control character movement correctly using arrow keys.

ID# - Energy Depletion

Description: Test relating to using energy to play a game.

Items covered by this test: This test covers the energy depletion mechanic of the game when starting a game.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: N/A

Test Procedures: Test is conducted by beginning application and starting a game. After finishing a game either by completing the maze, dying, or running out of time, player energy will have decreased by 10.

Input Specification: Input for this test is the player's current energy level.

Output Specifications: Output expected is the player's energy level reduced by 10.

Pass/Fail Criteria: This test passes when starting and returning from a game results in the player's energy level dropping by 10. The user will not be able to start a game if energy level is less than 10, preventing energy level from reaching negative values.

ID# - Watching Ads

Description: Allows the user to watch ads for in-game rewards.

Items covered by this test: This test covers the ad feature of the game as well as the reward feature for watching an ad.

Requirements addressed by this test: N/A

Environmental needs: This test requires a VLC media player to be installed, as well as a 64-bit Python interpreter to launch the ad feature.

Intercase Dependencies: N/A

Test Procedures: After starting application and logging in, press the "Watch Ad" button in the main menu.

Input Specification: This test takes in video files supported by VLC media player.

Output Specifications: This tests output involves displaying an unskippable ad on the main screen for the user to watch. Returns to the main menu after the ad is finished.

Pass/Fail Criteria: This test passes when the user is able to watch ads from the main menu. After watching the ad the player's energy will be increased by 10. Ads will not increase energy above 100.

ID# - Subsequent Ads

Description: Showing different ads each viewing.

Items covered by this test: This test covers the feature for showing the next ad in the series instead of the same as previously watched.

Requirements addressed by this test: N/A.

Environmental needs: This test requires a VLC media player to be installed, as well as a 64-bit Python interpreter to launch the ad feature.

Intercase Dependencies: "Watching Ads" test must be completed successfully before running this test.

Test Procedures: After starting the application and logging in. The user will watch an ad from the main menu. After finishing the ad the user will watch another ad.

Input Specification: This test takes in video files supported by VLC media player.

Output Specifications: Output from this test will be an ad displayed on the main screen. After watching a subsequent ad, the same ad will not play, instead the next ad in the series will be shown.

Pass/Fail Criteria: This test passes when watching multiple ads results in no repeating ads, and showing the next ad in the series.

ID# - Player Damage

Description: This test relates to players getting damage from colliding with zombies or getting shot by the zombies.

Items covered by this test: This test covers gameplay functionality; it tests whether or not a players health goes down when damaged.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: Zombie shooting, AI, and weapons for the player should be tested beforehand.

Test Procedures: After logging in, selecting a map character, and starting the game, the player should move around the map and start getting attacked by zombies.

Input Specification: The input values for this test are the user's current health.

Output Specifications: The output for this test is a decreasing health bar, when attacked by zombies. The health bar should go down, each time a zombie attack.

Pass/Fail Criteria: This test pass when attacked by a zombie shooting or colliding with the users.

ID# - Player Death

Description: This test relates to a player losing the game when they are getting damaged by a zombie.

Items covered by this test: This tests gameplay functionality, it tests if players can lose the game, and if the application correctly ends the game.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: Player damage should be tested beforehand.

Test Procedures: Login to the application, select a map, and start moving throughout the maze while letting the zombie cause damage. The health bar should go down and when it's emptied the game should return to the main menu.

Input Specification: The input values is the player's current health.

Output Specifications: The output for this test is a decreasing health bar, when attacked by zombies. The health bar should go down, each time a zombie attacks.

Pass/Fail Criteria: Pass.

ID# - Reaching Finish Line

Description: The test relates to the player's reaching the end of the maze.

Items covered by this test: The test covers gameplay functionality, it tests the functionality of the game ending when reaching the finish line.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: Tests that are related to player and zombie damage, ai, and shooting, should be tested beforehand.

Test Procedures: Login to the application, select a map, survive the zombies and reach the end of the maze. The application should reset to the main menu.

Input Specification: The input values is the player's current position in the game.

Output Specifications: The output for this test is the application resetting to the main menu.

Pass/Fail Criteria: Pass.

ID# - Switching Weapons

Description: This test relates to the player being able to select different weapons.

Items covered by this test: This is another gameplay functionality test, it tests players selecting weapons.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: Ammo pickups and zombie/player shooting needs to be tested beforehand.

Test Procedures: Login to the application, select a map, pickup weapons and switch to different weapons.

Input Specification: The input value is the player's current weapon.

Output Specifications: The output for this test is a new weapon.

Pass/Fail Criteria: Pass.

ID# - Displaying Weapon Info

Description: This test relates to the application displaying the correct weapon.

Items covered by this test: This tests gameplay and visuals in the game.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: Switching weapons should be tested beforehand.

Test Procedures: Login, select a map, collect and switch weapons, the application should display the correct weapon.

Input Specification: The input values are the player's current weapon.

Output Specifications: The output should be the correct weapon that was selected.

Pass/Fail Criteria: Pass.

ID# - In-game Timer

Description: This test relates to the time limit in the game, players should reach the end of the maze before the timer runs out, if not the game will end and the application will return to the main menu.

Items covered by this test: This test the timer functionality in the game, to ensure whether or not the game ends, if the time runs out.

Requirements addressed by this test: N/A

Environmental needs: N/A

Intercase Dependencies: N/A

Test Procedures: After logging in, selecting a map, character, and starting the game, stand still till the timer runs out, when it does the application should reset to the main menu.

Input Specification: The input for this test is the current time in the game.

Output Specifications: The output for this test is a decreasing timer in the game, and the application resetting to the main menu.

Pass/Fail Criteria: Pass.

ID# - Ammo Pick-ups

Description: This is to test the pick up abilities of individuals to get the ammos.

Items covered by this test: picking up ammos.

Requirements addressed by this test: Ability to pick up ammos

Environmental needs: Any computer or laptop can be used to run this test.

Intercase Dependencies: NA

Test Procedures: Run the code->open the game menu->Select a map->Play the game->Pick up ammos in the map.

Input Specification: map.tml is used to run this test.

Output Specifications: The player will be able to pick up ammos from the map while moving.

Pass/Fail Criteria: If the player picks up ammos while moving around, this test passes.

ID# - Health Packs

Description: The player can pick up health packs while moving around in the map only if the player lost some health.

Items covered by this test: Ability to pick up health packs.

Requirements addressed by this test: This test addresses the ability of a player to pick up health packs if the player is injured.

Environmental needs: Any computer or laptop can be used to run this test.

Intercase Dependencies: NA

Test Procedures: Run the code->open the game menu->Select a map->Play the game->Pick up ammos in the map.

Input Specification: medkit.png will be used.

Output Specifications: The player can pick up medkits which can be done using medkit.png

Pass/Fail Criteria: The test passes if the player is able to pick up health packs while moving around in the map.

ID# - Pistol

Description: The player can pick up a pistol which will be available in the game shop.

Items covered by this test: ability to select and use pistol in game

Requirements addressed by this test: This test addresses the ability of a player to pick up pistol from the shop

Environmental needs: Any computer or laptop can be used to run this test.

Intercase Dependencies: NA

Test Procedures: Run the code->open the game menu->Select shop menu->select pistol

Input Specification: manOld_gun.png will be used

Output Specifications: The player can select pistol from shop which can be done using manOld_gun.png

Pass/Fail Criteria: The player will be able to pick up pistol from shop

ID# - Shotgun

Description: The player can pick up a shotgun which will be available in the game shop.

Items covered by this test: ability to select and use shotgun in game

Requirements addressed by this test: This test addresses the ability of a player to pick up shotgun from the shop

Environmental needs: Any computer or laptop can be used to run this test.

Intercase Dependencies: NA

Test Procedures: Run the code->open the game menu->Select shop menu->select shotgun menu which gives you 3 shotgun options

Input Specification: hitman1_shotgun.png, manBlue_shotgun.png, robot1_shotgun.png will be used

Output Specifications: The player can select shotgun from shop

Pass/Fail Criteria: The player will be able to pick up any type of shotgun from shop

ID# - Machine-Gun

Description: The player can pick up a machine gun which will be available in the game shop.

Items covered by this test: ability to select and use machine gun in game

Requirements addressed by this test: This test addresses the ability of a player to pick up machine gun from the shop

Environmental needs: Any computer or laptop can be used to run this test.

Intercase Dependencies: NA

Test Procedures: Run the code->open the game menu->Select shop menu->select machine gun menu which gives you 3 shotgun options

Input Specification: hitman1_machinegun.png, manBlue_machinegun.png, robot1_machinegun.png will be used

Output Specifications: The player can select machine gun from shop

Pass/Fail Criteria: The player will be able to pick up any type of machine gun from shop

ID# - Zombie Damage

Description: Game will test zombies receiving damage from the player.

Items covered by this test: Zombie's health will decrease upon player shooting them.

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: Ensuring spawn of zombies on the map.

Test Procedures: Run the code -> Click login -> Click Play -> Click any map -> Use arrow keys to approach a zombie

Input Specification: Space bar will be pressed until the player lands three shots on the zombie.

Output Specifications: The zombie will die and disappear from the map.

Pass/Fail Criteria: Upon landing three shots on the zombie, the zombie will die and disappear from the map.

ID# - Zombie Shooting

Description: Game will test the zombie shooting the player.

Items covered by this test: Player's health will decrease upon zombies shooting the player.

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: Ensuring spawn of zombies on the map.

Test Procedures: Run the code -> Click login -> Click Play -> Click any map -> Use arrow keys to approach a zombie with a gun

Input Specification: NA

Output Specifications: Zombie will shoot the player

Pass/Fail Criteria: Upon the player getting close to a zombie with a gun, the zombie will begin to shoot the player thus damaging and decreasing the player's health.

ID# - Zombie AI

Description: Game will test zombie AI

Items covered by this test: Zombie detection of player movement

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: Ensuring spawn of zombies on the map.

Test Procedures: Run the code -> Click login -> Click Play -> Click any map -> Use arrow keys to move around the map -> Gets close to zombie -> Grows distant from the zombie

Input Specification: NA

Output Specifications: Zombies within the player's radius will follow the player. Otherwise, it will not follow.

Pass/Fail Criteria: Upon the player getting near a zombie, the zombie will start to follow the player. Otherwise, if the player is distant, the zombie will not follow.

ID# - Wall Collisions

Description: Game will test the ability to not go through walls.

Items covered by this test: Player and zombie ability to not go through walls.

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: NA

Test Procedures: Run the code -> Click login -> Click Play -> Click any map -> Use arrow keys towards a wall

Input Specification: NA

Output Specifications: Player and zombie inability to go through the walls.

Pass/Fail Criteria: Upon the player approaching a wall, the player will not be able to go through the wall. Likewise for the zombie.

ID# - Purchasing Energy from In-game Shop

Description: Game will test the ability to purchase energy from the in-game store.

Items covered by this test: In-game currency utilized to purchase items.

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: Ensuring in-game currency transactions work properly.

Test Procedures: Run the code -> Click login -> Click Play -> Click any map -> Press ESC -> Click Shop -> Click the energy icon

Input Specification: NA

Output Specifications: Player's energy will refill to 100.

Pass/Fail Criteria: Upon entering the in-game store, the player will have the ability to purchase more energy with the in-game currency.

ID# - Purchasing Character from In-Game Shop

Description: Game will test the ability to purchase characters from the in-game store.

Items covered by this test: In-game currency utilized to purchase items.

Requirements addressed by this test: NA

Environmental needs: Any Python IDE utilizing the PyGame library. As well as any capable PC or laptop.

Intercase Dependencies: Ensuring in-game currency transactions work properly.

Test Procedures: Run the code -> Click login -> Click Shop -> Choose any character to purchase -> Go back to previous screen -> Click Play -> Choose a map

Input Specification: NA

Output Specifications: Characters purchased and selected will be used when playing a game.

Pass/Fail Criteria: Upon purchase and selection of a character in the in-game store, while entering a game the selected player will be used to play the game.

10 Test Results

ID# - Map Selection

Date(s) of Execution:11/10/20

Staff conducting tests: Nick Toledo

Expected Results: Expected all in-game maps to be selectable and playable from the map selection screen.

Actual Results: All added maps were able to be loaded and played from the map selection screen.

Test Status: Pass .

ID# - Movement

Date(s) of Execution: 11/10/20

Staff conducting tests: Nick Toledo

Expected Results: Expected user to be able to move character sprite using arrow key inputs.

Actual Results: User was able to control character correctly user arrow key inputs.

Test Status: Pass .

ID# - Energy Depletion

Date(s) of Execution: 11/11/20 2:10pm, 2:35pm

Staff conducting tests: Nick Toledo.

Expected Results: Expected energy to reduce correctly after starting and returning from a game.

Actual Results: First test energy depleted correctly after starting games, but would allow games to continue playing at zero energy level and go negative. After the second test, energy depletion was working as anticipated.

Test Status: Pass

ID# - Watching Ads

Date(s) of Execution: 11/12/20.

Staff conducting tests: Nick Toledo.

Expected Results: Expected unskippable ad to be displayed on the main menu .

Actual Results: Ads were displayed as expected.

Test Status: Pass.

ID# - Subsequent Ads

Date(s) of Execution: 11/12/20.

Staff conducting tests: Nick Toledo.

Expected Results: Expected no repeat ads to be shown when watching subsequent ads.

Actual Results: As expected, no repeat ads were shown. Instead the next ad in the series was shown to the user..

Test Status: Pass.

ID# - Player Damage

Date(s) of Execution: 11/13/2020

Staff conducting tests: Stephen.

Expected Results: Expected to see health bar decrease when attacked by zombies.

Actual Results: The players health bar decreases each time a zombie attacks.

Test Status: Pass.

ID# - Player Death

Date(s) of Execution: 11/14/2020

Staff conducting tests: Stephen

Expected Results: Expected to see a decreasing health bar, and the application resetting to the main menu.

Actual Results: The players health decreases and the game resets to the main menu.

Test Status: Pass.

ID# - Reach Finish Line

Date(s) of Execution: 11/14/2020

Staff conducting tests: Stephen

Expected Results: Expected the application to switch to the main menu gui.

Actual Results: The game ended and the application switched to the main menu gui.

Test Status: Pass.

ID# - Switching Weapons

Date(s) of Execution: 11/14/2020

Staff conducting tests: Stephen.

Expected Results: Expected the game to display a new weapon when switched.

Actual Results: The game displayed a new weapon when switched.

Test Status: Pass.

ID# - Displaying Weapon Info

Date(s) of Execution: 11/14/2020

Staff conducting tests: Stephen.

Expected Results: Expected to see the weapons and amount of ammo.

Actual Results: The game displays weapons and shows the amount of ammo.

Test Status: Pass.

ID# - In-game Timer

Date(s) of Execution: 11/13/2020

Staff conducting tests: Stephen.

Expected Results: The expected results are the timer ending, and the game resetting to the main menu.

Actual Results: The timer decreases, and the game resets to the main menu.

Test Status: Pass.

ID# - Ammo Pick-Ups

Date(s) of Execution: 11/25/20

Staff conducting tests: yas shah

Expected Results: player should be able to pick up ammos while moving around in the map.

Actual Results: Player was able to pick up ammos while moving around

Test Status: Pass

ID# - Health Packs

Date(s) of Execution: 11/25/20

Staff conducting tests: yash shah

Expected Results: Player should be able to pick up health packs when injured

Actual Results: The player was able to pick up health packs when injured

Test Status: Pass

ID# - Pistol

Date(s) of Execution: 11/25/20

Staff conducting tests: yash shah

Expected Results: Player should be able to select a pistol from the store.

Actual Results: Player was able to pick up a pistol from the store.

Test Status: Pass

ID# - Shotgun

Date(s) of Execution: 11/25/20

Staff conducting tests: yash shah

Expected Results: Player should be able to select a shotgun from the store.

Actual Results: Player was able to pick up a shotgun from the store.

Test Status: Pass

ID# - Machine-gun

Date(s) of Execution: 11/25/20

Staff conducting tests: yash shah

Expected Results: Players should be able to select a machine gun from the store.

Actual Results: Player was able to pick up a pistol from the store.

Test Status: Pass

ID# - Zombie Damage

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: Zombies receive damage from the player.

Actual Results: Zombies health depleted upon getting shot at from the player.

Test Status: Pass

ID# - Zombie Shooting

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: Zombies with guns deal damage by shooting at the player if they're in range of the player.

Actual Results: Zombies shoot the player if the player is in range.

Test Status: Pass

ID# - Zombie AI

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: Zombies will follow the player if the player is in range, and will stop following if the player goes distant. If the player shoots at a zombie, the zombie will keep following regardless of the distance.

Actual Results: Zombies follow the player if the player is in range. Zombies stop following the player as soon as the player is out of range. A zombie shot by the player will continue to follow the player regardless of the distance.

Test Status: Pass

ID# - Wall Collisions

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: The player and zombies will not be able to go through walls.

Actual Results: The player and zombies successfully aren't able to go through walls.

Test Status: Pass

ID# - Purchasing Energy from in-game shop

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: The player will be able to purchase more energy using the in-game currency.

Actual Results: When the player isn't at max energy, the player is able to successfully purchase energy through the in-game store using in-game currency.

Test Status: Pass

ID# - Purchasing Character from in-game Shop

Date(s) of Execution: 11/04/2020

Staff conducting tests: Omar Flores

Expected Results: The player will be able to purchase different characters to use in game with the in-game currency.

Actual Results: The player is able to successfully purchase different characters and toggle between selected characters to use in different games.

Test Status: Pass

11 Regression Testing

Regression tests for changes/additions to the advertisement system.

- Watching Ads Test
- Subsequent Ads Test

Regression tests for future in-game shop updates.

- Purchasing Energy from an in-game shop.
- Purchasing Characters from in-game shop.

Regression tests for future Zombie behavior Changes.

- Zombie AI Test.
- Zombie Damage Test.
- Zombie Shooting Test.

Regression test for Player attribute changes.

- Player Damage Test
- Player Death Test
- Movement Test
- Pistol/Shotgun/Machine Gun Tests

IV Inspection

12 Items to be Inspected

Subsection 1

- In-game Shop Code
- Game Shop Purchases Code
- Player and Zombie Spawn Code

Subsection 2

- Main Menu Code
- Zombie Behavior Code
- Consumable Items Code

Subsection 3

- Log-in Menu
- Character Movement Code
- Gun Fire Code
- Advertisement Code

Subsection 4

- Multiplayer Menu Code
- Energy System Code
- Options Menu Code
- In-Game GUI Code

13 Inspection Procedures

For the inspections we created a checklist that is originally based on a generic checklist for code reviews written by Karl E, Wiegers. Information about this checklist is provided in the bibliography [3].

The checklist items that we chose for our inspections are..

Structure

- Is the code well-structured?
- Is there any unreachable code or unused methods?
- Is there any leftover/test routine code?
- Is their repeated code which could be moved to a single method?
- Are symbolics used instead of magic numbers?

Documentation

- Is code clear and well commented?
- Are comments consistent?

Variables

- Do variables have meaningful names?
- Are there unused variables

14 Inspection Results

Subsections Inspected by Nick Toledo	**Structure**	**Documentation**	**Variables**
Subsection 1 11/16/20 2:30pm	Code is well structured and easy to read and understand. Some repeated code for rendering images and text could have been condensed into a function.	Shop code contains little comments, the rest of the subsection seems to be commented accordingly.	The lack of comments is made up for by well named variables and methods. Making it easy to read and understand the code.

Subsection 2 11/16/20 6:00pm	Code maintains a good structure, all code is used and valuable to the system. Some repeated display code could have been condensed into a function.	Blocks of code are well commented and provide valuable information to understanding the code well.	All variable names are named meaningfully and compliment the comments within the code.
Subsection 4 11/17/20 4:00pm	Very well organized, shows strong oop principles. Inheritance used well to minimize rewriting methods and attributes.	Code contains an adequate amount of comments, and is consistent throughout the subsection.	Very well named variables and methods

Table 1 - Nick's Inspections

Subsections Inspected by Omar Flores	**Structure**	**Documentation**	**Variables**
Subsection 1 11/14/20 9:00pm	Code is well structured. Readers will be able to understand the logic and flow of the program.	In-game shop has minimal comments, but readers will still be able to understand it.	Variables are given proper meaningful names. Thus, allowing readers to understand the use of each variable.
Subsection 2 11/14/20 11pm	Code is well structured. All written code is meaningful for the product to work as expected. Functions could've been used to reduce code repetition.	Main menu, zombie behavior, and consumable code have great comments to allow readers to understand what certain sections are doing.	Variables are given proper meaningful names. Thus, allowing readers to understand the use of each variable.
Subsection 3 11/15/20 8pm	Code has a great structure. Great use of Python libraries such as PyGame is shown as well as providing ad support using VLC.	All items in this subsection have sufficient comments that will allow readers greater understanding of the code.	All variables used are given meaningful names corresponding to the use of the variable.

Table 2 - Omar's Inspections

Subsections Inspected by Stephen Lambert	Structure	Documentation	Variables
Subsection 1 11/13/2020 11am	Code is organized efficiently by using functions. Files are organized and named specifically for the functionality that the code runs. Collaborates will be able to understand the code.	The main menu and watch ads code has great comments that collaborators will be able to understand.	Variables as well as function names are specifically named for their functionality.
Subsection 3 11/3/2020 12pm	Code as well as functions continue to be organized and named specifically for its functionality. The addition of more functions would have been beneficial.	Code continues to have great comments that collaborators will be able to understand. The main code could have included more beneficial comments.	Variables and functions continue to be named precisely for the functionality that the code runs.
Subsection 4 11/13/2020 6pm	Code, functions, and their names remain organized. The use of pygame libraries and tile to make the maps were beneficial to the game.	Comments exist throughout the code, collaborators will have greater understanding of the functionality.	All of the variables and function names are relevant.

Table 3 - Stephen's Inspections

Subsections Inspected by Yash Shah	Structure	Documentation	Variables
Subsection 2 11/26/20 2pm	Code is very well sorted out and detailed. Everything is properly organized throughout.	All the comments are very meaningful which helps understand the code.	All names of the variables are meaningful.

Subsection 3 11/26/20 6pm	Code is very detailed and well organized from start to the end.	Everything in this section has more than enough comments.	Variables used are meaningful names which represent the variables.
Subsection 4 11/26/20 8:30pm	Well organized and the code structure is properly maintained.	Code has a nice amount of comments to explain the code.	Nicely named variables and methods throughout the code.

Table 4 - Yash's Inspections

V Recommendations and Conclusions

Items covered during testing all seemed to have passed testing with no problem. Inspections also went well for each subsection of the codebase, we did have some subsections lacking in comments and having repeated code. However, these are issues that can be addressed easily and do not warrant any further testing or inspection.

VI Project Issues

15 Open Issues

A current open issue with our system involves the use of Python for accessing VLC media player's API from the application to display in-game ads. Currently users must have VLC media player installed to watch the ads as expected, however it's not realistic to expect users to install additional software to watch ads. Other options we explored involved other libraries such as Pyglet, and Pygame's built in video viewer. Unfortunately Pyglet would close out the application when viewing an ad, and Pygame's video feature has been deprecated due to instability. An alternative method not involving installation of additional software should be found to reduce further user complication with the application.

16 Waiting Room

A requirement we would like to see addressed in future releases are those pertaining to use real-time map data to generate maps to play in. This is a great idea and would give the game a lot more originality and playability. Unfortunately this requirement is not as simple as making some calls to google map API's. At the moment the system tile-based maps are created in a map editor called Tiled which must be added manually in the project.

17 Ideas for Solutions

In order to implement a real-time map feature envisioned by the original group, we would have to obtain information about a current location, and somehow create a fully

playable map complete with starting and ending locations, as well as zombie spawns. One approach to hopefully tackling this feature in the future would be to further understand how the application reads in the map tile files, and finding a way to dynamically create a tiled map from map inputs in the application. If creating a map from the application becomes possible, then the next step to developing this feature would be planning a way to process map data to a tile based map. One suggestion being using the roads on a map to create the paths in the map, and perhaps using an actual image of the map as the floor of the map as opposed to grass in other maps.

18 Project Retrospective

Although we managed to get a fairly good looking functional product completed, it did not also come without its shortcomings. If we could change one thing about our project it would have been the technology used for creating the game. For our application we chose to use Python along with a library called Pygame. Although the word game is in the name, it is a very basic game library, and mostly serves as a way to generate and render text/images on a screen. In the future, when building game applications we would like to explore much more robust tools such as Unity. Unity is a very powerful tool and allows for much more visually appealing gameplay which makes a great first impression for both users and stakeholders..

VII Glossary

AI: Artificial intelligence, referring to the movement and behavior of NPCs.

Non-Playable Character: Refers to characters not controlled by the player such as the zombies.

Tiled: A tile-based map editor used to create tile-based maps.

Ads: Advertisements/commercials.

Energy: A type of in-game currency used to play the game.

Consumables: In-game items dropped by zombies during gameplay. These included ammo and med kits.

TSX File: A filetype for tile based maps created using the Tiled application.

VLC Media Player: A third party media plate application. It's APIs are used to display in-game ads.

Tile-based Map: A map created using layers of tiles/squares on a grid structure.

Pygame: A Python library for game development.

VIII References / Bibliography

[1] D. Hatten, M. Ryan, M. Torres and N. Gelezinis, "Zombie Dash Project Report," Chicago, 2019.

[2] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[3] K. E. Wiegers, "Generic Checklist for Code Reviews," 2001. [Online]. Available: https://www.liberty.edu/media/1414/%5B6401%5Dcode_review_checklist.pdf. [Accessed 10 November 2020].

IX Index