Functional Test Plan & Test Case Descriptions

for

HumanVsZombies, Release 1.0

Prepared by:
Kaan Akduman
Jamie Booker
Rounak Chawla
Yadira Gonzalez
Minh Pham
Sarah Yurick

EECS 393 Software Engineering
Case Western Reserve University

April 1, 2020

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Kaan, Jamie, Rounak, Yadira, Minh, Sarah	April 1, 2020	Initial draft	1.0 approved

Functional Test Plan & Test Case Descriptions

1. Introduction

1.1 Purpose
This document will summarize the objectives of testing the HumanVsZombies game. It will lay out the methods and procedures used to test the software and will specify the test cases that are to be used in order to verify that the requirements from the Software Requirements Specification document [1] are fully met.

1.2 Scope
HumanVsZombies will be tested on a Windows PC using the Unity Testing Framework. This document thoroughly lists each of the use cases, their respective test cases, and the use cases that will not be tested. Refer to the Test Specifications in section 4 and the Functional Tests in section 5 to see detailed descriptions of each test that will be performed.

1.3 Abbreviations, Acronyms, and Definitions

Term	**Definition**
Boundary	An area on the map that the player cannot walk through such as the edge of the map and buildings
Collision	Refers to when any objects (the player, zombies, weapons, ammo fired from weapons, movable objects, and buildings) interact with each other during gameplay. Collisions have varying effects depending on the objects that are colliding, as outlined in section 5.4
GitHub	Website which hosts software development version control
Movable object	An object in the setting which can be moved when the player collides with it
Player Sprite	The player's character, i.e. the Spartan
Score/Player Score	A numerical value that is a function of the time the player has lasted without losing all health and number of zombies killed
Software Requirements Specification	A document that clearly outlines the software to be developed

"The System"	The HumanVsZombies software
Unity	The game engine used to develop HumanVsZombies
Use Case	Any scenario in which the user interacts directly with the software
Wave	Next level of difficulty during gameplay. A new wave is triggered as a function of player score. At the beginning of a wave, new zombies and new (worse) weapons are spawned on the map

1.4 References

1. Akduman, et al. Software Requirements Specification. *Software Requirements Specification for HumanVsZombies*.
2. Akduman, et al. Software Design Document. *Software Design Document for HumanVsZombies.*

1.5 Overview of Contents of Document

Test Plan Description

This section gives a very brief overview of the HumanVsZombies software for a high-level idea of what will be tested. The testing schedule is also listed here.

Test Design Specifications

This section outlines the methods of testing to be used on this software, environmental requirements, which features are not to be tested, criteria for the suspension and resumption of testing, and risks that may interfere with successful testing.

Test Specifications

This section provides a descriptive overview of how the software responds as the user provides input while progressing through HumanVsZombies.

Functional Tests

This section steps through the process by which the software responds to each possible action by the user. For each test case tag there will be an attributed description, setup, and an expected result in response to the setup.

2. Test Plan Description

The HumanVsZombies game application includes both old features of the seen-before shooter game and creative ones designed for players to enjoy and have a fun time. Each functionality is made up of a number of use cases and can be tested in different ways. This document will list each of the features, their respective use cases, our plan to test the use cases, and other related details.

2.1 Product Summary

HumanVsZombies is a zombie shooter game application that aims to provide users with an enjoyable and relaxing experience. It incorporates various levels of challenges while still preserving the fun, creative element that a game should have. Having Case Western Reserve University's campus as its setting, HumanVsZombies is an elevated version of the original shooter game, with new features such as wave functionality, which weakens weapon damage after each round, and a free-for-all scoring system with an increasing level of difficulty to keep the game captivating.

2.2 Responsibilities

The roles and responsibilities of each team member during the testing process are listed in the table below. These roles are intended to be adaptable, and each member may assume different, flexible roles as needed during testing.

Name	Role	Responsibility
Sarah Yurick	Test Lead	Oversee and coordinate test process.
Kaan Akduman	Tester	Execute test procedures and augment automated tests.
Jamie Booker	Tester	Execute test procedures and augment automated tests.
Rownak Chowla	Tester	Execute test procedures and augment automated tests.
Yadira Gonzalez	Tester	Execute test procedures and augment automated tests.
Minh Pham	Recorder	Record test results.

3. Test Design Specifications

3.1 Testing Approach

The authors of this document used the Software Requirements Specification [1] and Software Design Document [2] to enumerate the test cases and their procedures, which are intended to test the functional requirements of the application. All test cases listed are designed to verify the implementations of features listed in previous documents for HumanVsZombies.

3.2 Feature or Combination of Features Not To Be Tested

The following list describes features that will be excluded during the testing phase:

- Application load testing - It will be assumed that if the testers' devices are able to properly render the application as a whole (including user interfaces, and gameplay features such as player control, zombie movements, game status information, and sound and music) then likewise devices (Windows PCs) should be able to do the same.

- Response time - It will be assumed that the Unity platform will handle user inputs (for both navigating user interfaces as well as during gameplay) in a reasonable amount of time.
- Animation rendering - It will be assumed that the animations (i.e. player and zombie sprites running in their direction of movement) specified in the player's and zombies' Blend Trees (a built-in tool provided by Unity) will be properly handled by the Unity platform.

3.3 Environmental Needs

All tests are to be administered on a Windows PC. In order to test the HumanVsZombies application, the user must have Unity Version 2019.2.19f1 installed on their device. HumanVsZombies will not be tested with any other version of Unity. Tests will be run with Unity's built-in Test Runner, which utilizes Unity's Testing Framework, an integration of NUnit.

3.4 Suspension/Resumption Criteria

Any test suspended before completion must be restarted and resumed from the first step of the test, with initial preconditions reestablished. If any test cannot be completed, this must be recorded.

3.5 Risks and Contingencies

- Schedule restraints - The authors of this document recognize that in attempting to achieve completeness and comprehensiveness in listing test cases, the number of tests outlined in this document may be overwhelming given the short timeline the testers have to perform them. In light of this, testers may choose to consolidate overlapping cases which are expected to behave similarly (e.g. players and zombies should both be unable to move through buildings) as they see fit.
- Testing user input - Because the Unity API code is so embedded in the game code, it may be difficult to test user input (e.g. that pressing the W key moves the player up). Thus in lieu of making tests that require input, the testers may have to decouple the code from the user input and instead rely on the results of the input.

4. Test Specifications

4.1 Navigating the Main Menu

1. When the user opens the game application, a main menu that contains visuals of the game title and a background image (left to the creativity of the game developers) will appear.
2. The main menu will give the user three options to click: Play, Settings, and Highscores.
3. If the user clicks the Play button:
    a. The system will display a short story sequence that outlines the zombie apocalypse.

   b. After the story sequence, the system will place the player sprite on the map with an initial score of zero.
   c. The system will randomly place zombies, weapons, and movable objects on the map. The user can also now control the player as gameplay begins.
4. If the user clicks the Settings button, the screen will show the Settings Menu.
5. If the user clicks the Highscores button, the screen will show the Leaderboard.
6. The user can only click one button at a time. If the user does not click a button, the main menu will remain on the screen.

4.2 Navigating the Settings Menu

1. The user can click the Settings button either on the Main Menu or on the Pause Menu.
2. When the user clicks the Settings button, the Settings Menu will be displayed on the screen.
3. The Settings Menu will give the user three options to click: Toggle Sound, Toggle Music, and Back.
4. If the user clicks the Toggle Sound button, the sound effects played during gameplay will be turned "on" if they were "off" or "off" if they were "on."
   a. The system will update the visuals of the Toggle Sound button to reflect these changes.
   b. When the user starts or resumes the game, the gameplay sound effects will reflect these changes.
5. If the user clicks the Toggle Music button, the music played during gameplay will be turned "on" if it was "off" or "off" if it was "on."
   a. The system will update the visuals of the Toggle Music button to reflect these changes.
   b. When the user clicks the button, the system immediately stops or plays the music.
6. If the user clicks the Back button, the system will display the Main Menu or the Pause Menu, depending on how the user got to the Settings Menu.
7. The user can only click one button at a time. If the user does not click a button, the Settings Menu will remain on the screen.

4.3 Navigating the Leaderboard

1. The user can click the Highscores button from the Main Menu.
2. When the user clicks the Highscores button, the Leaderboard will be displayed on the screen.
3. The leaderboard will display three blocks: the Top Scores table, the Main Menu button, and the Reset button.
4. The Top Scores table will show the top three scores obtained in previous games.
5. If the user clicks the Top Scores table, the Leaderboard will remain on the screen.
6. If the user clicks the Main Menu button, the system will go back and display the Main Menu.
7. If the user clicks the Reset button, all three top scores will be reset to zero.
   a. The system will reset the variables that hold the top three scores to zero.

   b. The system will immediately reflect these changes on the Top Scores table.
8. The user can only click one button at a time. If the user does not click the Main Menu button or the Reset button, the Leaderboard will remain on the screen.

4.4 User Interactions with the System during Gameplay

4.4.1 Moving the player

1. The user can choose where they want the player to move on the map by pressing an arrow key or WASD key on the keyboard.
2. Each key corresponds to a direction that the player can move on the map:
   a. Up arrow/W key moves the player up.
   b. Left arrow/A key moves the player to the left.
   c. Down arrow/S key moves the player down.
   d. Right arrow/D key moves the player to the right.
   e. Pressing the up arrow/W key or down arrow/D key and left arrow/A key or right arrow/D key simultaneously moves the player diagonally.
   f. Pressing the up arrow/W key and the down arrow/S key does not move the player up or down.
   g. Pressing the left arrow/A key and the right arrow/D key does not move the player left or right.
3. The system will update the player's location (visually as well as the player's position variable in the code) every time the user presses the key, and the user must release the key for the player to stop moving.
4. Moving the player renders an animation depicting the character running in the corresponding direction.
5. If the player is located at the edge of the map, the system will recognize that the player is at a boundary.
   a. The system will not allow the user to move the player beyond the boundary.
   b. The system will not update the player's location.
   c. The user must press a key that is not in the direction of the boundary for the player to move.
6. If the user presses keys that are not the arrow keys or WASD keys, the player will not move.

4.4.2 Moving objects in the world

1. The user can make the player move objects on the map by pressing an arrow key or WASD key on the keyboard.
2. Each key corresponds to a direction that the player can move the object in, given that the player is facing the object:
   a. Up arrow/W key allows the player to move the object up.
   b. Left arrow/A key allows the player to move the object to the left.
   c. Down arrow/S key allows the player to move the object down.
   d. Right arrow/D key allows the player to the object to the left.

   e. Pressing the up arrow/W key or down arrow/D key and left arrow/A key or right arrow/D key simultaneously moves the player and the object diagonally.
   f. Pressing the up arrow/W key and the down arrow/S key does not move the player or the object up or down.
   g. Pressing the left arrow/A key and the right arrow/D key does not move the player or the object left or right.
3. The system will update the location of the object (visually as well as the object and player position variables in the code) every time the user presses the key, and the user must release the key for the object to stop moving.
4. If the object is located at the edge of the map, the system will recognize that it is at a boundary.
   a. The system will not allow the player to move the object beyond the map.
   b. The system will not update the location of the object.
   c. The player must face the object in a direction that is not toward the boundary and press the corresponding key for the object to move.
5. If the user presses keys that are not the arrow keys or WASD keys, the movable object will not move.
6. If the player does not collide with the object, it will not move.

4.4.3 Colliding with buildings in the world

1. The user cannot control the player to move buildings on the map because buildings are not designated as movable objects.
2. If the user presses an arrow key or WASD key while facing a building, the building will not move nor will the player move beyond the building.
3. The system will recognize the building as a boundary and the player will collide with the building.
4. The building's location will not change and the player's location will not be updated.

4.4.4 Colliding with a zombie and updating player health

1. If a user moves the player to collide with a zombie, or if the user does not move the player and a zombie walks into the player, the player's health is decremented and the static health bar at the bottom of the screen updates to show a health bar that has ⅓ of the max capacity less than what it showed previously.
2. A player loses all health when the player collides with a zombie 3 times.
3. At the start of the game, the system gives the player full health, which is displayed on a static bar at the bottom of the screen.
4. The player does not lose health when it collides with a building or movable object.
5. The player cannot regain health once it has lost it.

4.4.5 Picking up weapons

1. The user can make the player pick up weapons that are scattered on the map.
2. There must be a weapon on the map, and the player cannot already be holding a weapon.
3. The user must move the player to the location of the desired weapon.
4. Once the player collides with the weapon, the system will update the player sprite so that the player is now holding the weapon.
5. The system will update the screen to show a static bar that displays the remaining ammo that the weapon has; the weapon will start out with full ammo.
6. The system will then remove the weapon from the map and register that the player is now holding a weapon that can be aimed or fired.
7. The user cannot make the player pick up a new weapon if the player is already holding one. If the user attempts to do so by colliding with another weapon on the map:
    a. The system will recognize that the player is already holding a weapon and will not change the player sprite, weapon ammo bar, etc.
    b. The desired weapon will remain on the map.
8. The user cannot make the player drop the weapon until it is completely used and its remaining ammo is zero.

4.4.6 Aiming weapon

1. The user can make the player aim their weapon in the direction of the mouse located on the screen.
2. The direction the player is aiming the weapon is indicated by a crosshair that encircles the player.
3. The direction of the crosshair that encircles the player corresponds to the direction of the mouse on the screen.
4. If the player is holding a weapon, the user cannot make the player aim at itself. This means that if the user moves the mouse to hover over the player, the crosshair encircling the player will maintain its most recent direction.
    a. Related to section 4.4.7, if the user clicks the mouse, the weapon will fire in the most recent direction.
5. If the player is not holding a weapon, the system will not display the crosshair around the player and will not keep track of the direction of the mouse.

4.4.7 Firing weapon

1. As long as the player is holding a weapon (meaning that its ammo is not zero), the user can make the player fire their weapon in the direction of the crosshair that encircles the player.
2. When the user clicks the mouse, the weapon will fire in the direction of the crosshair.
3. When the user clicks the mouse, the system will display an animation of the player firing the weapon.

4. The system will then decrease the ammo count on the static ammo bar located at the bottom of the screen.
5. If a zombie is within range and is in the direction of the firing, the zombie may be killed or it can sustain damage (see section 4.4.8).
6. If the player is not holding a weapon and the user clicks the mouse, nothing on the screen will change.
7. If the user does not click the mouse, the weapon will not fire.

4.4.8 Firing weapon at zombie and updating zombie's health

1. Each zombie's health is constant and is full at the start of a game (the initial health of zombies is left to the decision of the game developers).
2. When the player clicks to fire ammo and a zombie is within range and in the direction of the firing, that zombie loses health when it collides with the weapon's ammo.
    a. The damage done to the zombie's health depends on the weapon that is used.
    b. If that zombie's health is drained to zero, it is killed, meaning that it is removed from the map and the zombie kill counter is incremented by 1.
3. The zombie can collide with buildings and movable objects without losing health.
    a. Zombie collisions with buildings and movable objects behave the same as the player's collisions with buildings and movable objects.
4. The zombie cannot regain health once it has lost it.

4.4.9 Increasing player score

1. The user can increase the player score by killing zombies (see 4.4.8.2).
2. The user can increase the player score by continuing to survive in the game (i.e. by achieving a longer gameplay time).
3. The player score is a function of the number of zombies the player has killed so far and how long the player has been in the game. The exact function used to calculate the score is left to the decision of the game developers.
    a. The system increases the player score and displays this on the upper left corner of the screen when the player successfully kills a zombie.
    b. The system will increase the score and display this on the upper left corner of the screen the longer the player is in the game.
4. Zombie kills and game duration are the only aspects that will increment the player score.
5. Player score cannot decrease while playing the game.

4.4.10 Game status

1. Whether or not the user can win is left to the choice of the game developers. There are 2 options:

        a. The user wins the game once the player has gone through all the waves and used every available weapon.
        b. The last available weapon has infinite ammo, and zombies continue to spawn as a function of zombies killed and gameplay time until the player dies.
    2. The user loses the game when its player has lost all of its health (by colliding with zombies 3 times), and the system displays the Game Over Menu.
    3. When the player has reached a certain score (exact qualifying scores TBD) during gameplay, the system will initiate a new wave which will spawn more zombies and new weapons on the map.
        a. The weapons that spawn with each new wave are worse than previous weapons, meaning that they will do less damage when fired at a zombie.

4.5 Navigating the Pause Menu

1. The user can pause the game by clicking the Pause Game button located in the upper right hand corner of the screen during gameplay.
2. When the user clicks the Pause Game button, the Pause Menu will be displayed on the screen.
3. When the user is on the Pause Menu, the system will pause its internal timer and freeze the player's and zombies' movements in place.
4. If the user presses an arrow key/WASD key or clicks the mouse while in the pause menu, nothing on the screen will change.
5. The Pause Screen will show the user three options to click: Resume, Settings, and Quit To Main Menu.
6. If the user clicks the Resume button, the game will resume to its last state and gameplay will resume as if the game had never been paused:
    a. The system will unpause its internal timer and the player and zombies will be unfrozen.
    b. The user is now free to use the arrow keys/WASD keys to move the player and click the mouse to fire their weapon.
7. If the user clicks the Settings button, the system will display the Settings Menu.
8. If the user clicks the Quit To Main Menu button, the game will end and the system will display the Main Menu.
    a. Before terminating the game, the system will compare the current score to the Top Scores.
    b. If the score is a new high score, the system will change the top three scores accordingly. The Leaderboard table will reflect these changes.
    c. When the game is terminated, the system will reset the current score and internal timer to zero.
9. The user can only click one button at a time. If the user does not click a button, the Pause Menu will remain on the screen.

4.6 Navigating the Game Over Menu

1. The system will display the Game Over Menu when the player loses all their health (or if the player wins, if the game developers decide to implement that possibility) and will display a Game Over message alongside the menu. Also see 4.4.10.1-2.
2. The player loses all their health when they come into contact with a zombie 3 times.
3. If the score that the player obtained before the game ending is a new highscore, the system will change the top three scores accordingly. The Top Scores table on the Leaderboard Menu will reflect these changes.
4. The system will reset the current score and internal timer to zero.
5. The Game Over Menu will give the user two options: Replay and Exit.
6. If the user clicks the Replay button, the system will display the Main Menu.
7. If the user clicks the Exit button, the game application will close.
8. The user can only click one button at a time. If the user does not click a button, the Game Over Menu will remain on the screen.

5. Functional Tests

5.1 User Interface Tests

Test	**Description**	**Setup**	**Expected Result**
UIT-1	Navigate from Main Menu to gameplay	Go to main menu and click “Play”	Gameplay begins
UIT-2	Navigate from Main Menu to Settings	Go to main menu and click “Settings”	Settings Menu opens
UIT-3	Navigate from Main Menu to Leaderboard	Go to main menu and click “Highscores”	Leaderboard opens
UIT-4	Toggle sound	While in the Settings Menu, click “Toggle Sound”	Sound effects played during gameplay turned off if on, or on if off
UIT-5	Toggle music	While in the Settings Menu, click “Toggle Music”	Music is immediately turned off if on, or on if off
UIT-6	Navigate from Settings Menu to previous page	While in the Settings Menu, click “Back”	Previous page is opened (either the Main Menu or Pause Menu)
UIT-7	Navigate from Leaderboard to Main Menu	While in the Leaderboard, click the “Main Menu” button	Main Menu opens

UIT-8	Reset Leaderboard scores to zero	While in the Leaderboard, click “Reset”	Top Scores table values are changed to zero, and the variables that hold these scores are also updated to zero
UIT-9	Pausing the game	During gameplay, click the pause button	Pause Menu is displayed. The game pauses, meaning that the system will pause its internal timer and freeze the player’s and zombies’ movements
UIT-10	Navigate from paused to resuming gameplay	While in the Pause Menu, click the Resume button	Game resumes: the system will unpause its internal timer and the player and zombies will be unfrozen
UIT-11	Navigate from paused to the Settings Menu	While in the Pause Menu, click the Settings button	Settings Menu opens
UIT-12	Quit gameplay by navigating from paused to Main Menu	While in the Pause Menu, click “Quit To Main Menu”	The game ends, checks the current score against the Top Scores and adjusts the Leaderboard if necessary, resets the current score and internal timer to zero, and returns the user to the Main Menu.
UIT-13	Navigate from the Game Over Menu to the Main Menu	On the Game Over Menu, click “Replay”	Main Menu opens
UIT-14	Exit the game from the Game Over Menu	On the Game Over Menu, click “Exit”	The application closes

5.2 PlayerController Class Tests

Test	**Description**	**Setup**	**Expected Result**
PCT-1	Moving the player	During gameplay, press	The player’s position is

		the arrow keys or WASD keys	updated accordingly. The character should move up if W or up is pressed, left if A or left is pressed, down if S or down is pressed, and right if D or right is pressed. See section 4.4.1.2 for more details
PCT-2	Aiming a weapon	During gameplay, the user moves the mouse to aim	The crosshair which encircles the player should follow the direction of the mouse
PCT-3	Firing a weapon	During gameplay, the player is holding a weapon and the user clicks to shoot	Ammo is launched in the direction of the firing. Ammo left decrements, and the static bar at the bottom of the screen indicating weapon ammo decrements

5.3 Zombie Class Tests

Test	**Description**	**Setup**	**Expected Result**
ZCT-1	Zombies spawn at the beginning of the game	The player starts a new game	There are zombies on the map (count TBD)
ZCT-2	Zombies spawn at the beginning of a new wave	During gameplay, the player has reached a score which triggers a new wave (see GST-3)	New zombies are spawned on the map (count TBD)
ZCT-3	Each zombie on the map moves	Zombies have spawned on the map	The zombies should move around the map as their positions are continuously changing
ZCT-4	Zombie dies	During gameplay, the player attacks a zombie and its health decrements to zero	The zombie disappears from the map and the zombie kill count is incremented (see overlap with GST-1)

5.4 Collision Tests

Test	**Description**	**Setup**	**Expected Result**
CT-1	Player collides with a zombie	The player runs into a zombie during gameplay	The player's health decrements, which is indicated by the static health bar at the bottom of the screen decrementing
CT-2	Player collides with a movable object	Player runs into a movable object during gameplay	The movable object's position is updated and moves in the direction that it was pushed
CT-3	Player collides with a building	Player runs into a building during gameplay	The player's position should not change; the player cannot move through buildings
CT-4	Zombie collides with a movable object	Zombie walks into a movable object during gameplay	The movable object's position is updated and moves in the direction that it was pushed
CT-5	Zombie collides with a building	Zombie walks into a building during gameplay	The zombie's position should not change; the zombie cannot walk through buildings
CT-6	Movable object collides with a building	The player attempts to push a movable object through a building during gameplay	The movable object's position should not change; the movable object cannot be moved through buildings
CT-7	Object collides with boundary	During gameplay, the player, a zombie, or a movable object tries to move off of the map	The object's position should not change; nothing can leave the map
CT-8	Weaponless player collides with a weapon on the map	Player runs over the weapon during gameplay	The player's sprite is updated to depict it holding that weapon. A static bar at the bottom of the screen appears to depict weapon ammo.

			The weapon disappears from the map
CT-9	Player collides with a weapon on the map while already holding a weapon	Player is holding a weapon and runs over a weapon on the map during gameplay	Nothing happens: the player sprite, weapon ammo bar, and weapon on the map remain
CT-10	Ammo fired collides with a zombie	The player is holding a weapon and clicks the mouse in the direction of a zombie	The zombie's health decrements according to the weapon being used; see ZCT-4 concerning zombie deaths

5.5 Game Status Tests

Test	**Description**	**Setup**	**Expected Result**
GST-1	Player score increases with zombie kill	During gameplay, the player kills a zombie (see overlap with ZCT-4)	The zombie kill increments and is used to calculate the player score (exact method TBD)
GST-2	Player score increases with gameplay time	An internal timer keeps track of how long the user has been in gameplay, and the timer reaches a certain mark (which is TBD; for example could be every 30 seconds)	The player score increases with the internal timer (exact method for calculating TBD)
GST-3	A new wave is triggered	During gameplay, the player reaches some score which triggers a new wave (exact scores TBD)	New zombies (count TBD) and new worse weapons spawn on the map (see overlap with ZCT-2)
GST-4	Game over	The player's health drains to zero (see CT-1 for decrementing player score). Whether or not the player can win is TBD, see 4.4.10.1	The current score is checked against Highscores, which are updated if necessary. The current score and internal timer are reset to zero. The Game

			Over Menu opens

6. Defect Resolution

6.1 Issue Tracking

We are planning to use our GitHub respiratory and its built-in issue tracking to track and label all issues, as well as making the code accessible to all teammates.

6.2 Issue Resolutions

In most cases, the recorder is responsible for revising the code consistently, tracking and recording the status of all errors identified. Once identified, the testers will be in charge of resolving the problem and marking it as completed. That being said, all developers are expected to contribute in identifying and reporting any defects found in any parts of the code.

7. Exit Criteria

- All tests defined above have been run and have passed review.
- Any tests found to be unable to be tested (excluding those mentioned in section 3.2) through Unity's Testing Framework have been tested manually.

8. Inspection Report

Team Member	**Findings**	**Resolution**
Kaan Akduman	4.4.4.1 did not mention updating the health bar	Added a clause to talk about that
Jamie Booker	1.3 Boundary definition did not include buildings	Included buildings
Rounak Chawla	None	
Yadira Gonzalez	4.4.2.3 did not include updating the object position variable	Added updating the object position variable
Minh Pham	None	
Sarah Yurick	Section 3 Test Design Specifications had not been done; found various unclear or incorrect descriptions throughout the document	Completed section 3; restructured and rewrote sections of the document accordingly