SE 3XA3: Test Plan
Zombie Survival Kit

Team #, Team Name
Mohammad Hussain hussam17
Brian Jonatan jonatans
Shivaansh Prasann prasanns

October 26, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2018-10-26	1.0	First Revision of Test Plan
Date 2	1.1	Notes

This document ...

1 General Information

1.1 Purpose

1.2 Scope

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation1	Definition1
Abbreviation2	Definition2

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

1.4 Overview of Document

2 Plan

2.1 Software Description

2.2 Test Team

2.3 Automated Testing Approach

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Func. Req. 1: The player must be able to move forward, left, down, and right using the WASD keys

Player Movement

1. Player-Movement-M-1

   Type: Manual

   Initial State: A scene is initially paused. The scene will have the player initialized at the center of the playable map, and a tree will be initialized at a position beside the player to act as a stationary marker.

   Input: Start scene button, WASD keys

   Output: Changes on screen.

How test will be performed:
The user will start the scene and the user will press down on the WASD keys. If the user notices that the position of the player changes in the scene relative to the tree, the test is concluded and is considered to have passed. If the player position stays the same relative to the tree's position while the user is pressing down on the WASD keys, the test is considered to have failed.

3.1.2 Func. Req. 2: The player must be able to lookin all directions by moving their mouse

Camera Movement

1. Camera-Movement-M-1

   Type: Manual

   Initial State: A scene is initially paused. The scene will have the player initialized at the center of the playable map, and a tree will be initialized at a position beside the player to act as a stationary marker.

   Input: Start scene button, mouse movement

   Output: Changes on screen.

   How test will be performed:
   The user will start the scene and the user will move the mouse in any direction. If the user notices that the perspective of the player changes in the scene using the tree as a marker, the test is concluded and is considered to have passed. If the tree stays at the same position on screen while the mouse is moving, the test is considered to have failed.

3.1.3 Func. Req. 3: Zombie enemies must walk back and forth between random points in their 'spawn' circle, which imitates them walking around

'Spawn' Circle

1. Spawn-Circle-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will have a predetermined position on the playable map.

   Input: void

   Output: void or assertion error.

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given for the zombie to move in any direction on the x, y plane of the the playable map, and will continue to move in that same direction until the zombie's position hits any point on the 'spawn' circle's perimeter; to which the zombie will change its directional movement and the process starts over again. Assert statements will be used to check if the zombie's position never passes any point along the 'spawn' circle's perimeter. After each change in direction occurs, a counter will be recorded. Once the counter reaches a specified amount, the test is concluded and the test is considered to have passed and nothing is returned (void). Any assertion errors will be considered as a failed test.

3.1.4 Func. Req. 4: Zombie enemies must start attacking the player when they come within a certain radius

Attack Radius

1. Attack-Radius-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized outside the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the player to move towards the zombie's attack radius. Once the player is within the zombie enemy's attack radius, the zombie will be given commands to attack the player, and the zombie's attack animation will be enabled. Assertion statements will be used to check if the zombie's attack animation is enabled. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.5 Func. Req. 5: Zombie enemies must follow the player if they start running away while the zombie is in it's attack state

Zombie Follows Player

1. Zombie-Follows-Player-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized at a distance outside of

the zombie's 'spawn' circle.

Input: void

Output: void or assertion error

How test will be performed:
A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given for the zombie to move in any direction on the x, y plane of the the playable map and will have its position constrained by it's spawn circle. Commands will be given to the player to move towards the zombie's 'spawn' circle center. If the position of the player and the zombie is at a certain distance from each other, commands will be given to the zombie to move its position towards the player and for the player to move away from the zombie's position. The change in position of the player and the zombie will be the same. Assertion statements will be used to check if the distance between the zombie's position and the player's position remains the same, and if the position of the zombie and player are not the same; while the zombie's position is not greater than a specified distance from the center of the 'spawn' circle. Once the zombie's position reaches this distance, the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.6 Func. Req. 6: If the player runs past a certain radius from the zombie's original spawn loaction, the zombie must return to their circle

Zombie Returns to 'Spawn' Circle Center

1. Zombie-Return-Spawn-Circle-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will be a predetermined position on the playable

map. The player object is initialized at a distance outside of the zombie's 'spawn' circle.

Input: void

Output: void or assertion error

How test will be performed:
A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given for the zombie to move in any direction on the x, y plane of the the playable map and will have its position constrained by it's spawn circle. Commands will be given to the player to move towards the zombie's 'spawn' circle center. If the position of the player and the zombie is at a certain distance from each other, commands will be given to the zombie to move its position towards the player and for the player to move away from the zombie's position. The change in position of the player and the zombie will be the same. Assertion statements will be used to check if the distance between the zombie's position and the player's position remains the same, and if the position of the zombie and player are not the same; while the zombie's position is not greater than a specified distance from the center of the 'spawn' circle. Once the zombie's position reaches this distance, the zombie will return to the center of its 'spawn' circle, concluding the test. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.7 Func. Req. 7: Different types of zombies must have different statistics (health, damage,etc.)

Zombie Stats

1. Zombie-Stats-A-1

   Type: Automated

   Initial State: All different zombie enemy type objects are initialized.

Input: void

Output: void or assertion error

How test will be performed:
A test script will be created that initializes the test with the specified initial state. Within the test script, instructions will be given to compare all the different stats fields of each zombie enemy type. An assertion statement will be used to check if all values within a stat field are different for each zombie enemy type. Once all assertion tests are passed, the test is conlcuded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.8 Func. Req. 8: Zombies must have randomly generated chance of dropping items the player can use or consume when killed.

Appearing Drops

1. Appearing-Drops-A-1

   Type: Automated

   Initial State: Two zombie enemy objects are initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized at a distance of the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the

player to attack one of the zombies. An assertion statement will be used to check if that zombie's current health value is lower than than its initial health value. Once the zombie's health value reaches zero, an item object will appear at a chance of 100% at the position of the zombie before the zombie object is set to null (representing the first zombie's death). An assertion statement will be used to check if there exists an item object at the position where the zombie had died. The second zombie will undergo the same process as the first zombie, but an item object appearing at this zombie's position will have a chance of 0%. An assertion statement will be used to check if there does not exists an item object at the position of the zombie before the zombie object is set to null (representing the second zombie's death), and the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.9 Func. Req. 9: Dropped items must be automatically deleted if not picked up within a certain time

Disappearing Drops

1. Disappearing-Drops-A-1

   Type: Automated

   Initial State: A zombie enemy objects is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized at a distance of the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the

player to attack one of the zombies. An assertion statement will be used to check if that zombie's current health value is lower than than its initial health value. Once the zombie's health value reaches zero, an item object will appear at a chance of 100% at the position of the zombie before the zombie object is set to null (representing the zombie's death). An assertion statement will be used to check if there exists an item object at the position where the zombie had died. Instructions to wait for the amount of time it takes for the item to disappear are given. An assertion statement will be used to check if there does not exists an item object at the same position, and the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.10 Func. Req. 10: The player must be able to pick up tiems they are looking at with the 'interact' key

Pickup Drops

1. Pickup-Drops-A-1

   Type: Automated

   Initial State: A player object is initialized at any predetermined position, and an item object is also initialized near the player's position.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the player to move towards the item and, once close enough, to change the camera's foward position to the item. Then, the 'interact' key will be inputted. An assertion statement will be used to check if there exists

the item in the player's inventory, and the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.11 Func. Req. 11: The player must be able to access their inventory by pressing the 'invenotry' key

Inventory Access

1. Inventory-Access-A-1

   Type: Automated

   Initial State: A player object is initialized at any predetermined position

   Input: void

   Output: void or assertion error

   How test will be performed:
   The user will start the scene and the user will press down on the WASD keys. If the user notices thatthe position of the player changes in the scene relative to the tree, the test is concluded and is considered to have passed. If the player position stays the same relative to the tree's position while the user is pressing down on th WASD keys, the test is considered to have failed.

3.1.12 Func. Req. 12: The player must by able to equip, consume, delete, and move around items in their inventory using the mouse.

Use Items Move Items

1. Use-Move-Items-M-1

Type: Manual

Initial State: A scene is initially paused. A player object is initialized at the center of the playable map. Equipment and consumable item objects are initialized close to the player's position.

Input: Start scene button, WASD keys, mouse movement, mouse left-click, 'inventory' key, 'interact' key, 'de-equip' button

Output: Changes on screen.

How test will be performed:
The user will start the scene and the user will press down on the WASD keys to move the player towards the items and stop once the player is close enough to the item to interact with it. The user will then use the mouse to change the camera view until the item is in the center of screen and will press the 'interact' key on the item to pick it up. The user will do this for both items. The user will then press the 'inventory' key to open the inventory UI, and use the mouse to hover over either item icon. The user will press down on the left mouse button to use the item. If the item used is the consumable, the user will check the player's health value to see if it has increased by a specified amount. The user will also check if the item no longer exists in the player's inventory. The user will then press down on the left mouse button to use the equipment item. The user will check if the equipment has been equipped in one of the player's equipment slot and if the equipment item has disappeared from the player's inventory. The user will then press the 'de-equip' button and will check if the equipment is no longer in the player's equpment slot, but back in the player's inventory. The user will move the equipment item to another slot by pressing and holding down the left mouse button on the item icon, and drag the mouse to an empty slot in the player's inventory. The user will check if the item has moved to its intended spot. The user will then press the left mouse button on the remove button in the top right corner of the slot the item is in. The user will check if the item has been removed from the player's inventory and has appeared at some close distance in front

of the player. If any of the user's check fails, the test is considered to have failed. Otherwise, the test was passed.

3.1.13 Func. Req. 13: The player must by able to fire/use their equipped weapon by pressing the left mouse button.

Attack Items

1. Attack-Items-M-1

   Type: Manual

   Initial State: A scene is initially paused. A player object is initialized at the center of the playable map. A weapon type equipment is initialized to the player's corresponding equipment slot. A zombie enemy object is initialized close to the enemy

   Input: Start scene button, WASD keys, mouse movement, left mouse button

   Output: Changes on screen.

   How test will be performed:
   The user will start the scene and the user will press down on the WASD keys to move the player towards the zombie until the player is within attacking range from the zombie. The user will move the mouse until the zombie is in the middle of the user's screen and will then press the left mouse button. The user will observe if the zombie's health is decreasing after each attack. If any of the user's check fails, the test is considered to have failed. Otherwise, the test was passed.

3.1.14 Func. Req. 14: If the player has a firearm equipped, they must be able to aimdown sights using the right mouse button.

Aiming

1. Aiming-M-1

   Type: Changes on screen

   Initial State: A scene is initially paused. A player object is initialized at the center of the playable map. A weapon type equipment is initialized to the player's corresponding equipment slot.

   Input: Start scene button, mouse movement, right left-click

   Output: void

   How test will be performed:
   The user will start the scene and the user will press down and hold on the right mouse button. The user will check if the firearm object moves to a position that shows the player is "aiming down their sights". The player will then move the mouse while still holding down the right mouse button and will check if where the firearm is pointing at changes. If any of the user's check fails, the test is considered to have failed. Otherwise, the test was passed.

3.1.15 Func. Req. 15: The environment must slowly go through a day and night cycle.

Day and Night

1. Day-Night-A-1

   Type: Automated

   Initial State: Day time at the current system time.

   Input: void

   Output: void or assertion error

How test will be performed:
A test script will be created that initializes the test with the specified initial state. Within the test script, instructions will be given for the function that changes day time into night time to run, and to check how much time has passed from when the test script started. Once the time needed for day time to become night time has passed, an assertion statement will check if it is night time. Once again, instructions will be given to check how much time has passed after the previously mentioned assertion statement has been checked. Once the time needed for night time to become day time, an assertion statement will check if it is day time. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.16 Func. Req. 16: The player must lose health when hit by a zombie

Zombie Attack

1. Zombie-Attack-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized outside the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the player to move towards the zombie's attack radius. Once the player is within the zombie enemy's attack radius, the zombie will be given commands to attack the player and the zombie's attack animation will

be enabled. An assertion statements will be used to check after each zombie attack if the player's health is lower than its previous health. Once the player's health reaches zero, the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.17 Func. Req. 17: Zombies must lose health when hit by the player

Player Attack

1. Player-Attack-A-1

   Type: Automated

   Initial State: A zombie enemy object is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized outside the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the player to move and change its camera forward positon towards the zombie. Once the player is within attacking distance, the player will be given commands to attack the zombie and the player's attack animation will be enabled. An assertion statement will be used to check, after each player attack, if the zombie's health is lower than its previous health. Once the zombie's health reaches zero, the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate a failed test.

3.1.18 Func. Req. 18: Players and zombies must die when they reach 0 health

Death

1. Death-A-1

   Type: Automated

   Initial State: Two zombie enemy objects is initialized at the center of its 'spawn' circle that will be have a predetermined position on the playable map. The player object is initialized outside the zombie's attack radius.

   Input: void

   Output: void or assertion error

   How test will be performed:
   A test script will be created that initializes the test with the specified initial state. Within the test script, commands will be given to the player to move and change its camera forward positon towards the first zombie. Once the player is within attacking distance, the player will be given commands to attack the first zombie and the player's attack animation will be enabled. Assertion statements will be used to check, after each player attack, if the first zombie's health is lower than its previous health. Once the first zombie's health reaches zero, the first zombie will have died. An assertion statement will be used to check that the first zombie not longer exists (null). Commands will then be given to the second zombie to move towards the player. Once the player is within the zombie's attack radius, the zombie's attack animation will begin. An assertion statements will be used to check after each zombie attack if the player's health is lower than its previous health. Once the player's health reaches zero, the player is dead. An assertion statement will be used to check if the player is dead and the test is concluded. No assertion errors indicate a passed test, whereas assertion errors indicate

a failed test.

3.1.19 Func. Req. 19: Upon player death, the game must reset

Reset

1. Reset-M-1

   Type: Manual

   Initial State: A player object is initialized in the center of the playable map and a zombie enemy object is initialized at a distance from the player that is within the zombie's attack radius.

   Input: Scene start button

   Output: Changes on screen

   How test will be performed:
   The user will start the scene. The user will check if the zombie is attacking the player and if the player's health is decreasing after each zombie attack. Once the player's health decreases to zero, the user will check if the scene resets to its inital state where the process begins all over again. If any of the user's check fails, the test is considered to have failed. Otherwise, the test was passed.

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.2 Unit testing of output files

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.