SE 3XA3: Test Report
Zombie Survival Kit

Team 6, Group 6ix
Student 1: Brian Jonatan, jonatans
Student 2: Mohamad Hussain, hussam17
Student 3: Shivaansh Prasann, prasanns

December 4, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
December 3 2018	1.0	Document created
Date 2	1.1	Notes

This document contains the results of our initial testing plan, the information dervied from it as well as the changes that have been implemented in our project based on the results of the tests that have been conducted.

1 Functional Requirements Evaluation

1.1 Functional Requirement 1:

1.1.1 Description:

The player must be able to move forward, left, backwards and right using the WASD keys.

1.1.2 Type:

Manual

1.1.3 Observation:

By pressing the WASD keys on the keyboard, the player can move around on the game terrain.

1.1.4 Result:

Passed

1.2 Functional Requirement 2:

1.2.1 Description:

The player must be able to look in all directions by moving their mouse.

1.2.2 Type:

Manual

1.2.3 Observation:

By moving the mouse in a certain direction, it is observed that the camera turns in the same direction.

1.2.4 Result:

Passed

1.3 Functional Requirement 3:

1.3.1 Description:

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

1.3.2 Type:

Automated

1.3.3 Observation:

When the game scene is initialized, zombie enemies can be observed at various positions on the game terrain. These zombies are observed to be moving around a certain position on the map.

1.3.4 Result:

Passed

1.4 Functional Requirement 4:

1.4.1 Description:

Zombie enemies must start attacking hte player when they come within a certain radius.

1.4.2 Type:

Manual

1.4.3 Observation:

When the user controls the player and moves close to a zombie, then the zombie starts attacking the player, and the zombie character model is animated to show an attack movement.

1.4.4 Result:

Passed

1.5 Functional Requirement 5:

1.5.1 Description:

Zombie enemies must follow the player when the player is at a certain distance form the zombie.

1.5.2 Type:

Manual

1.5.3 Observation:

When the user controls the player and moves close to a zombie, then the zombie starts attacking the player, and the zombie character model is animated to show an attack movement. When the player then starts to move away, the zombie starts to follow the player.

1.5.4 Result:

Passed

1.6 Functional Requirement 6:

1.6.1 Description:

If the player runs past a certain radius from the zombie's original spawn location, the zombie must return to their circle..

1.6.2 Type:

Manual

1.6.3 Observation:

When the user controls the player and moves close to a zombie, then the zombie starts attacking the player, and the zombie character model is animated to show an attack movement. When the player then starts to move away, the zombie starts to follow the player. The user then presses and holds the shift button while moving the player away from the zombie. When the distance between the player and zombie is sufficiently large, the zombie stops chasing the player and starts returning to their own position.

1.6.4 Result:

Passed

1.7 Functional Requirement 7:

1.7.1 Description:

Different types of zombies must have different statistics (health, damage and attack speed). Different types of zombies will be distinguishable by their different appearances.

1.7.2 Type:

Manual

1.7.3 Observation:

When the user assumes control of the player and attacks various zombies in the game scene, different zombies require a different number of hits to kill.

1.7.4 Result:

Passed

1.8 Functional Requirement 8:

1.8.1 Description:

Zombies must have randomly generated chance of dropping items the player can equip or consume when killed.

1.8.2 Type:

Manual

1.8.3 Observation:

When different zombies are killed, they drop a different type of pickup after they die.

1.8.4 Result:

Passed

1.9 Functional Requirement 9:

1.9.1 Description:

The player must be able to pick up items they are looking at with the 'interact' key.

1.9.2 Type:

Manual

1.9.3 Observation:

When the user looks at a pickup and presses the 'E' button on the keyboard, the pickup item disappears from the screen.

1.9.4 Result:

Passed

1.10 Functional Requirement 10:

1.10.1 Description:

The player must be able to acces their inventory by pressing the 'inventory' key, and will have a maximum space of 20 items.

1.10.2 Type:

Manual

1.10.3 Observation:

When the user presses the 'I' button on the keyboard, the inventory UI is shown with 20 item slots.

1.10.4 Result:

Passed

1.11 Functional Requirement 11:

1.11.1 Description:

The player must be able to equip, consume and drop items in their inventory UI using the LMB.

1.11.2 Type:

Manual

1.11.3 Observation:

When the user presses the 'I' button on the keyboard, the inventory UI is shown with 20 item slots. When the player then picks up some pickups from the game scene, the inventory UI updates and shows the pickups in a slot. By clicking on an apple, the player's health increases according to the health bar and by clicking on a boot icon, it moves from the inventory UI at the right side of the screen to the equipment UI at the top left of the screen.

1.11.4 Result:

Passed

1.12 Functional Requirement 12:

1.12.1 Description:

The player must be able to unequip items in their Equipment UI using the LMB or the 'unequip all' key.

1.12.2 Type:

Dynamic, Manual

1.12.3 Observation:

When the user has items equipped, pressing the U button moves all items from the Equipment UI to the Inventory UI. When the user individually clicks on the items in the Equipment UI using the LMB, the specific item is moved from the Equipment UI to the Inventory UI.
The Test script containing the Assert statements, when executed, does not report any assertion errors.

1.12.4 Result:

Passed

1.13 Functional Requirement 13:

1.13.1 Description:

The player must be able to fire/use an equipped gun or axe by pressing the left mouse button (LMB).

1.13.2 Type:

Manual

1.13.3 Observation:

When the player has a gun equipped, pressing the LMB causes the gunshot sound to be played and the bullet UI at the bottom left corner is updated to show the ammo count in the gun. ~~When the player tries to equip an axe by opening the inventory UI, the cursor does not show up and the switch cannot~~

~~be made.~~ When the axe is equipped, clicking the LMB causes the axe swing sounds to be played.

1.13.4 Result:

~~Failed~~ Passed

1.14 Functional Requirement 14:

1.14.1 Description:

If the player has a firearm equipped, the user can reload the gun using the 'reload' key. The maximum amount of bullets will be 7.

1.14.2 Type:

Manual

1.14.3 Observation:

When the player has a gun equipped, pressing the LMB causes the gunshot sound to be played and the bullet UI at the bottom left corner is updated to show the ammo count in the gun. By pressing the 'R' button on the keyboard, the ammo UI count is set back to 7, and a reload sound effect is played.

1.14.4 Result:

Passed

1.15 Functional Requirement 15:

1.15.1 Description:

The environment must slowly go through a day and night cycle.

1.15.2 Type:

Automated, Manual

1.15.3 Observation:

A unit test script is used to test the working of the daylight controller object. This script is supposed to return assertion errors if the rotation of the daylight object in the game does not change as time passes. This test is considered to have passed as no assertion errors were returned.
While playing the game, a change in the orientation of the daylight object is visible and it's effect on the game terrain is noticeable.

1.15.4 Result:

Passed

1.16 Functional Requirement 16:

1.16.1 Description:

The player must lose health when hit by a zombie.

1.16.2 Type:

Manual

1.16.3 Observation:

When the user assumes control of the player and moves the player close to a zombie, the zombie attacks the player. When this happens, a decrease in the health level is demonstrated by the health bar UI at the bottom left corner of the screen.

1.16.4 Result:

Passed

1.17 Functional Requirement 17:

1.17.1 Description:

Zombies must lose health when hit by the player by using the LMB once the player is within attacking distance from the zombie.

1.17.2 Type:

Manual

1.17.3 Observation:

When the user assumes control of the player and moves the player close to a zombie, then uses the LMB to attack the zombie, the zombie dies after multiple clicks of the LMB, and drops a pickup object on the ground.

1.17.4 Result:

Passed

1.18 Functional Requirement 18:

1.18.1 Description:

Players or zombies must die when they reach 0 health.

1.18.2 Type:

Manual

1.18.3 Observation:

When the user assumes control of the player and moves the player close to a zombie, then uses the LMB to attack the zombie, the zombie dies after multiple clicks of the LMB, and drops a pickup object on the ground. The user then navigates the player to another zombie and allows the zombie to repeatedly damage the player. Eventually, the health level in the health bar UI (at the bottom left corner of the screen) becomes zero and a hit to the player after that causes the "Game Over" message to be displayed on the screen.

1.18.4 Result:

Passed

1.19 Functional Requirement 19:

1.19.1 Description:

An Inventory UI will give the user a visual representation of the player's inventory and will be accessible by pressing the 'inventory' key.

1.19.2 Type:

Dynamic, Manual

1.19.3 Observation:

When the user presses the "I" key on the kyeboard, an Inventory UI and an Equipment UI is displayed with 20 slots in the Inventory UI and 6 slots in the Equipment UI. If the player has picked up an item from the game, it is scene in the inventory UI.

1.19.4 Result:

Passed

1.20 Functional Requirement 20:

1.20.1 Description:

An Equipment UI will give the user a visual representation of the player's equipped items and it will appear with the Inventory UI by pressing the 'inventory' key.

1.20.2 Type:

Dynamic, Manual

1.20.3 Observation:

When the user presses the "I" key on the kyeboard, an Inventory UI and an Equipment UI is displayed with 20 slots in the Inventory UI and 6 slots in the Equipment UI. If the player has picked up and equipped an item from the game, it is scene in the Equipment UI.

1.20.4 Result:

Passed

1.21 Functional Requirement 21:

1.21.1 Description:

A Player UI will give teh user a visual representation of the player's health and remaining bullets if a gun is equipped to the player. The Player UI will also consist of a reticle.

1.21.2 Type:

Manual

1.21.3 Observation:

At all times in the game scene, a red reticle can be seen at the center of the screen. The reticle remains at the center of the screen at all times, irrespective of where the camera is moved. At the bottom left corner of the screen, a health bar UI can be seen. When a gun is equipped, the ammo count UI displays the number '7' which is the maximum capacity of a gun. When a shot is fired using the LMB, the number changes to 6.

1.21.4 Result:

Passed

2 Nonfunctional Requirements Evaluation

To evaluate our non-functional requirements, we created a Google Form to go along with an executable version of our project. These are shared with 10 volunteer testers who use our software while following a set of instructions and share their feedback through the Google Form.

2.1 Usability

2.1.1 Min-Learn-F-1

X out of 10 testers reported that the controls for our project were easy to learn.

2.2 Performance

2.2.1 Start-Time-F-1

2.2.2 FPS-F-1

2.2.3 Close-Time-F-1

2.2.4 Enough-Ram-F-1

2.3 Look and Feel

2.3.1 Graphics-F-1

2.3.2 Ease-of-Use-F-1

2.3.3 Ease-of-Use-F-1

2.4 Safety

2.4.1 Enough-Space-F-1

2.5 Precision

2.5.1 Player-Attack-Animation-to-Effect-F-1

2.5.2 Time-Day-Night-Transition-F-1

2.5.3 Zombie-Attack-Animation-to-Effect-F-1

2.5.4 Distance-to-Pickup-Item-F-1

2.6 Reliability and Availability

2.6.1 Reliable-Input-Commands-F-1

2.7 Play-When-On-F-1

2.8 Cultural Requirements

2.8.1 Culture-Discretion-F-1

3 Comparison to Existing Implementation

The existing project does not contain any documentation regarding the testing procedured in the development of the software. No unit testing has been

implemented, and the final product is not error free. For example, in the existing implementation, the axe weapon is dysfunctional and it causes the inventory UI to crash.

4 Unit Testing

Unit Testing was conducted to verify the correctness of various modules of our project. Certain modules, however, could not be tested due to restrictions imposed by the Unity 3D engine. For these modules, manual testing was used to verify that they function as intended.

Unit Testing was conducted on the following modules:

- Character Combat
- Character Stats
- Consumable Item
- Daylight Controller
- Equipment Item
- Equipment Manager
- Gun
- Inventory
- Item Store
- Player Stats
- Stat

The correctness of a module is verified by unit testing, when no assertion errors are returned by the unit test file. In case of our project, unit testing the aforementioned modules returned no assertion errors, and the modules mentioned above are therefore assumed to be correct.

[Please see the Test Plan for a detailed description of the individual Test Scripts.]

5 Changes Due to Testing

While no errors were encountered during testing, it was revealed that certain modules were preventing other modules from working correctly. Also, some undesirable behaviours were observed. These are listed below:

- The implementation of the gun firing mechanism prevented the cursor from showing up on the screen when the inventory UI was brought up, and prevented interaction with the inventory UI.
  Change made: A new function was added to the First Person Controller script to enable the cursor when the inventory UI was brought up, even if the gun was equipped.
- The initial implementation of the gun firing mechanism caused a shot to be fired even when the gun was being unequipped / swapped in the inventory UI or while equipping an item or consuming a consumable.
  Change made: The Gun module now checks whether the inventory UI is active and if that is true, a shot is not fired.
-

6 Automated Testing

For our project, the only implementation of automated testing was the Unit Testing process. The results of our unit tests are mentioned in a previous section.

7 Trace to Requirements

8 Trace to Modules

9 Code Coverage Metrics

References