University of Regina

ENSE 400/477

Project Night Terror Test Plan & Execution

Author:
Evan Geissler

Advisor:
Dr. Christine Chan

University of Regina

Last Modified
February 24, 2019

Revision History

Revision Version	Date
Version 1	February 24, 2019
Version 2	March 30, 2019

Table of Contents

List of Figures

List of Tables

Table 3.1-1. White Box Testing List

Appendices

1 Introduction

1.1 Purpose

The purpose of this section is to outline and summarize the testing processes used for Project Night Terror. The focus of the testing will be to ensure gameplay and functionality for the player works well, is desirable, usable, and playable. The testing will also help to fine tune variables involved with the player and Non-Playable Characters (NPCs).

It is important to note that there is a lot of minor and quick testing done during development as added functionally and minor changes are always tried before moving on. By doing this, I am able to detect and fix problems or bugs sooner. This also allows fine-tuning some variables better earlier on. Problems or bugs that are not resolved right away will often be left till later, but will be documented in GitHub under issues. This section will not show every case test and the appendices and GitHub issues/documents should be refereed to for more information.

1.2 Things to Test

Gameplay

Gameplay includes player movement (such as jumping, running, etc.) and anything not listed below.

AI

Testing for AI is to test every NPC. This will involve testing that animations flow well; testing that AI will respond and attack properly; that the AI will roam properly and in predefined bounds; and testing specific AI variables such as health, damage, and that attack speed are working properly. The testing methods used to test the AI will include black/white box testing and also usability testing.

Items

Item testing includes the user's flashlight, rifle, pistol, knife, and cross. The purpose of testing items is to ensure that the game does not become too easy or hard with bad variable values or mechanics; to ensure that animations and general look of items makes sense and looks good; and to ensure that the items follow good general usability ideas. This will be done with black/white box testing and also usability testing.

Collectables

Testing the collectables includes ensuring that their text displays properly, that audio files play properly, and that picking them up adds to the player's inventory properly. This will mostly be done with white box testing, but usability testing will also be used.

Saving & Loading

Testing saving and loading is important so the user can save and leave the game; so the user can come back to the game and pick up where they left off; and to ensure

that the player's progress is saved while moving across levels. There are many different variables that need to be saved or loaded and white/black box and usability testing will be used to test it.

Levels
Level testing includes going from one level to the next, ensuring that there are little to no bugs in map design (such as clipping, level holes, lighting issues, etc.), and to ensure that level traversal by the player is smooth (for example, stairs should by walked on without getting stuck). This will be done usability testing and black box testing.

User Interfaces
User interfaces (UI) include the heads up display (HUD), the main menu, pause menus, item menus, and the sub menus within these. It is important to see if the interfaces are visually appealing, if the functionally works, and if usability is present.

1.3 Test Level

A test level, that is not accessible to the player, was created very early in development to allow quick and easy testing. From early movement to working with AI, the testing map allows a large multitude of functions and features to be tried, displayed, refereed to, and more. The test level also saves time by not having to move parts that would need to be moved on a playable level or by having to build the level's lighting. Finally, there is no reason to remove items from the level as the different parts can be added to specific areas and left until needed or indefinitely. Figures 1.3-1 to Figure 1.3-3 show samples of the test level. It is important to note that features in the testing level may or may not be used in the game and is often where unused features can be seen. For example, in Figure 1.3-3 the area used to test demon AI was originally designed to view and see wooden ramps, hedgehogs, and other parts of the Atlantic Wall from WWII; a level that was originally planned early in development, but scraped in the first few months of development.

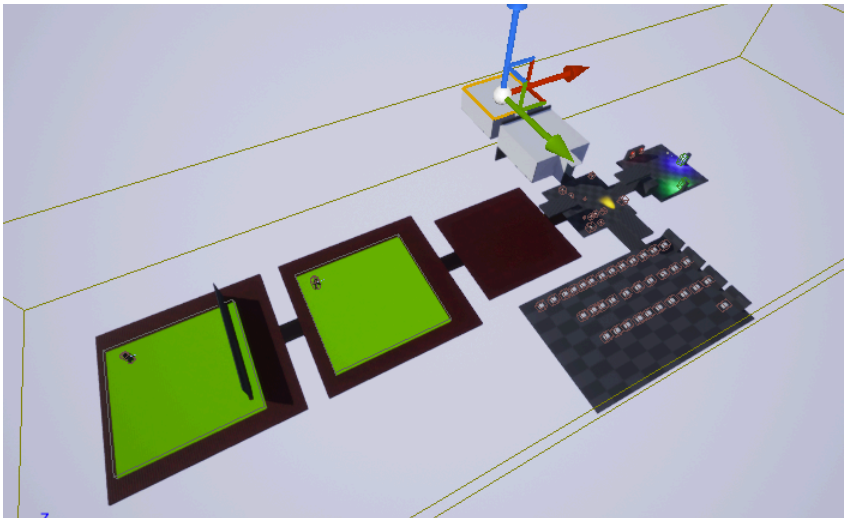

Figure 1.3-1. Entire Testing Level

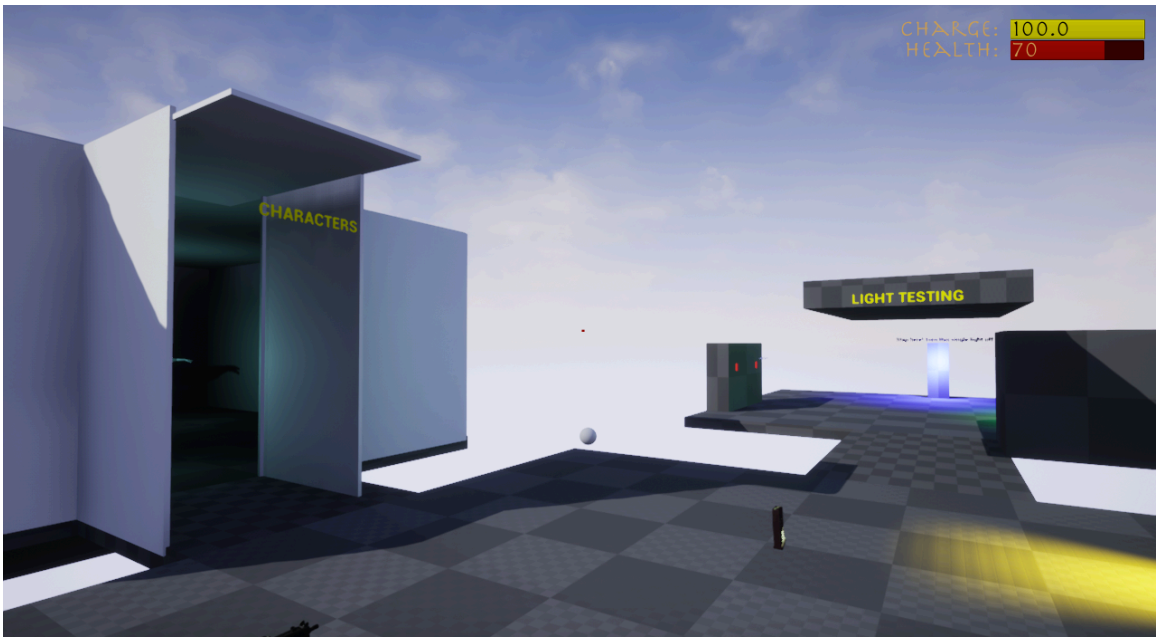

Figure 1.3-2. Test Level: Character & Light Areas

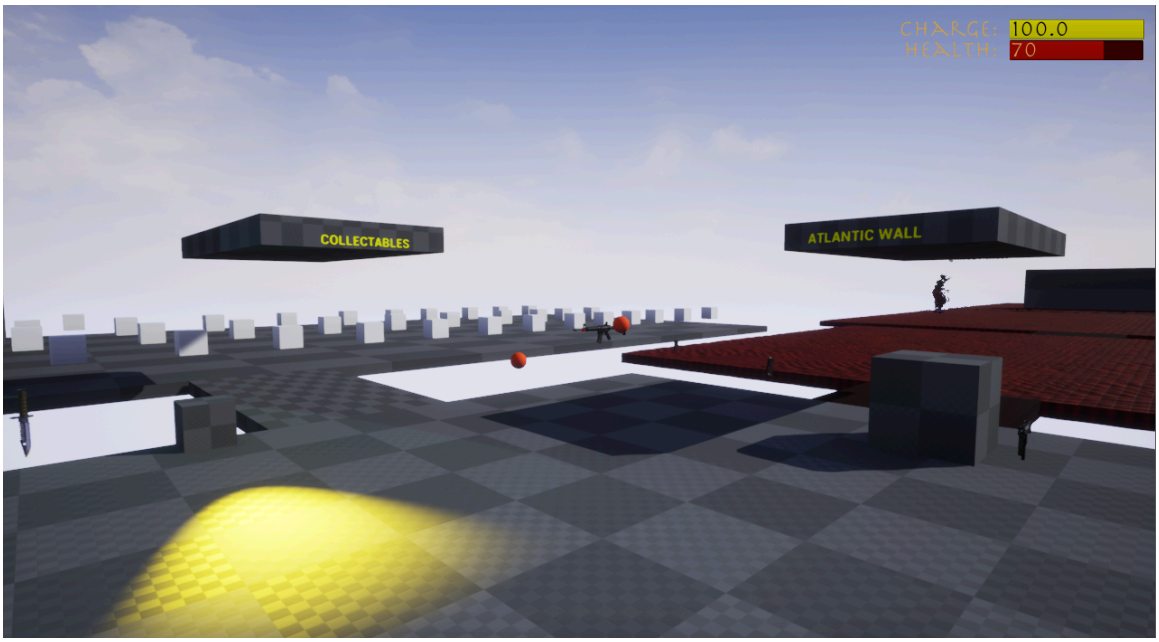

Figure 1.3-2. Test Level: Collectables & Modified Atlantic Wall Areas

2 Usability Testing

2.1 Purpose & Breakdown

The purpose of usability testing is to ensure that a range of users find the game usable, playable, and desirable. It is important that the user can learn the game easily and that the game is not hard due to controls, bad functionality, bugs, etc. Usability testing can help alleviate concerns or problems that may not arise during development and can create a better final product.

A usable game is one that the player can learn the controls without frustration and can easily understand them. A playable game is one that has few or no bugs, is not too difficult for the majority of players, and is enjoyable to play. Finally, a desirable game is one with nice user interfaces, good player interactions (such as shooting, reading text, using menus, etc.), and looks good overall.

Potential testers will be requested and any testers will be asked to fill out a consent form. Next, the tester will fill out a demographic questionnaire that will give insight into the tester's information, but more importantly their brief history with video games. The tester will then play different parts of the game and give feedback during the play test if they wish. Once playing is finished, the tester will fill out a post-questionnaire about their experience and provide feedback based on their experience.

2.2 Demographics

It is important for Project Night Terror to be playable and appealing to a large demographic. However, the game does not have a specific demographic so the testing does not have any particular demographic to test for either.

2.3 Testing Results

2.4 User Feedback

3 White Box Testing

3.1 Purpose & Breakdown

The purpose of white box testing where the tester knows about the code behind the tests is to fine tune the game, fix bugs and problems as soon as they arise, and to ensure that common events that the user will encounter are tried. There was no specific tool used in this type of testing as it was all done inside the engine. The tests were created, performed by playing the game, and the functions being tested were improved based upon the results. Table 3.1-1 below shows the different areas were white box testing was used. Brief overviews of the areas that need to be tested are below as well.

Guns

All guns need to be tested to for firing and reloading. The weapon should only fire when the player has ammo, when the player has ammo in the magazine, and when they are not reloading. The weapon should only be able to reload when the player has ammo to reload and the magazine is either empty or not full. Shooting should also be accurate enough to give the player a realistic shooting experience.

Melee Weapons

Melee weapons need to be tested to check if attacking works by dealing damage and can only deal damage when the user is within a specific range from a demon.

Flashlight
As a core item in the game, the flashlight must be able to charge and lose charge at a constant rate. The flashlight must be off when it has no charge and on if the player choses to turn it on and there is a charge. Other features such as damage or effects to the demons also need to be tested.

AI
The AI must be able to attack the player and also deal damage to the player. The combat between the player and AI can be tested while testing multiple cases for the weapons, character, and AI itself.

Pickups
Ammo and health pickups should properly add to the player's health or ammo only if those values are not currently maxed. If ammo or health is not maxed then the pickup increase should cause the player's ammo or health to become their current amount plus the pickup amount or it should become the maximum amount if the difference is less than the pickup amount.

Saving
All player and level related variables and information must be saved when the player selects the save button. The player information also needs to be saved between levels or information will be lost when a new level is loaded. Saves must also be checked that they properly delete when the user wants to delete a save file. If it is not properly deleted than there will be unexpected results with loading.

Loading
All saved variables need to be checked upon loading to ensure that the proper player and level information is being updated when the player wants to play from a save. If a load is not properly implemented then the information lost could result in the player not having an item at that point, such as the flashlight in the last level, or a level may destroy an actor that it believes the player already has found.

Perks
Checked alongside item testing and health checks, when perks are selected the related variables and values are properly updated visually and in the back end.

Objectives
Tests need to check if objectives are being updated when specific tasks are completed such as completing a level, upon a level loading, or when a user picks up an objective related item. Without these tests, if an objective does not properly update then the user may become frustrated or confused. Objectives are critical as they are the main drive for the player on what to do or where to go. Levels are also designed to move to the next once all objectives are completed so a missed or bad test could create an instance where the player is unable to advance in the game.

Table 3.1-1. White Box Testing List

White Box Test	Feature	Function
1	Rifle	Fire
2		Reload
3	Pistol	Fire
4		Reload
5	Rifle Ammo	Pickup
6	Pistol Ammo	Pickup
7	Flashlight	Charge
8		Turn on/off
9	Knife	Attacking
10	Cross	Attacking
11	Perks	Flashlight Perks Work
12		Pistol Perks Work
13		Rifle Perks Work
14		Health Perks Work

White Box Test	Feature	Function
15	Objectives	Updates upon entering level
16		Updates upon specific level events
17	Heavy Demon	Taking Damage/Dies
18		Randomly Roams
19	Light Demon	Taking Damage/Dies
20		Randomly Roams
21	Saving	Works when "Save" button pressed
22		Delete save
23		Works when "Save" button pressed
24	Loading	Works when "Load" button pressed
25		After level is loaded

3.2 Tests Performed

Although a lot of testing is performed as the game has been developed, there are some specific tests performed on features such as the items. The rifle, pistol, cross, knife, and flashlight have been tested for boundary values and specific cases. For example, the rifle's and pistol's reload function needed testing to ensure that the could not reload if they have a full magazine or if they do not have enough ammo to fill the clip. On top of this, the reload function has the potential to affect or read the specific weapon's total ammo, current magazine amount, and max magazine size. By checking specific cases, the reloading function was improved and situations that the player would commonly encounter, such as reloading at any point not just when the magazine is empty have been check to ensure there is no error. The rifle reloading testing table is shown in Table 3.2-1 and other item testing tables are shown in Appendix A.

Table 3.2-1. Rifle Reload Function Test

Conditions	Inputs		1	2	3	4	5
	Ammo	A1	–	T	–	–	–
		A2	–	–	–	T	–
		A3	–	–	–	–	T
	Total Ammo	T1	T	–	–	–	–
		T2	–	T	–	T	–
		T3	–	–	T	–	–
Actions	Outputs	Nothing	N	N	N	N	Y
	Ammo	Ammo = Total Ammo	N	Y	N	N	N
		Ammo = 0	Y	N	N	N	N
		Ammo = Magsize	N	N	Y	Y	Y
	Total Ammo	Total Ammo = 0	Y	Y	N	N	N
		Total Ammo = (Total Ammo - Magsize) + Ammo	N	N	Y	Y	N
Results			Passed	Passed	Passed	Passed	Passed

A1	Ammo = 0
A2	0 < Ammo < Magsize
A3	Ammo = Magsize
T1	Total Ammo = 0
T2	0 < Total Ammo < Magsize
T3	Magsize < Total Ammo <= Capacity

4 Black Box Testing

4.1 Purpose & Breakdown

The purpose of black box testing is to find problems that may not include code or does not require knowing the system behind the feature. For example, searching for a hole in the map requires no understanding on why or how that hole is there. This is important as there are gameplay elements or parts of the game that could affect the player's experience if not found, tested, and/or resolved. Although many black box tests are covered by usability testing, it is important as the developer to ensure that all test cases in Table 4.1-1 are properly covered.

Pickups and Visuals

Tests that require a check for pickup need to ensure that the player can properly pick it up, that the item is destroyed, and that the item is added to the player's inventory. For example, a note should prompt the user to pick it up then, upon the proper user action the note should disappear from the level and its information become visible to the player in the in game menu. Pickup checks go hand in hand with visual or audio that the user will hear or see since it is only available until after the user picks the item up or after a specific task is completed.

Lighting

Lighting tests are important as a level that is too dark can make a player frustrated that they cannot see; other problems such as strange or unrealistic reflections and shadows can also cause the player to have a lower quality experience.

Player Movement and Controls

Bad player movement, awkward controls, etc. can frustrate players, leave them confused, or lower player experience. Black box testing helps alleviate these concerns by checking player speed, animations, control placement, and more. Although listed in black box testing, this will be mostly covered in usability testing, as the testers will be covering these cases.

Menus

Menus need to work as intended and checking every single button in multiple ways ensures that the functionally properly works. Although most cases would be covered by the usability tests, the testers may not check every button or menu, which will leave specific parts untested. To avoid the lack of coverage, developer testing will include trying every user interface element.

Characters

The AI characters will need to be watched visually to check and see if they roam to places that were unintended such as out of bounds of the map, into level holes, etc. Since this does not relate to the actual functionality or code and is just visual, the only adjustments that testing will do to the characters will be level placement changes to the landscape, navigation bounds, BSPs, or static meshes.

Table 4.1-1. Black Box Testing List

Black Box Testing	Feature	Function
1	Notes	Pickup
2		Shown in inventory (hidden/visible)
3	Audio Logs	Pickup
4		Shown in inventory (hidden/visible)
5		Audio Plays
6	Artifacts	Pickup
7		Shown in inventory (hidden/visible)
8	Upgrades	Pickup
9		Shown in inventory (hidden/visible)
10	Pistol	Pickup
11		Shown in inventory (hidden/visible)
12	Rifle	Pickup
13		Shown in inventory (hidden/visible)
14	Flashlight	Pickup
15		Shown in inventory (hidden/visible)
16	Heavy Demon	Random Walk
17		Follow Player
18	Light Demon	Random Walk
19		Follow Player
19	Ghost	Random Walk
20	Menus	Buttons work as indented
21		Game paused when entering
22	Movement	Walking Speed
23		Animations look good
24		General movement look/feel
25	Levels	Lighting
26		Restricted bounds
27		Unintentional movement restrictions by level element

5 Automated Testing

5.1 Purpose & Breakdown

Automated testing could lower the time it would take to test certain functionality of the game. However, there currently is no plan for automated testing as there is nothing deemed beneficial to test automatically as almost all functionally and gameplay will be done by the user. It would be more beneficial to have myself or others see the testing first hand to give more insightful feedback. If automated testing is added, it will be added here.

Appendix A
Usability Tests
A.1

Appendix B

White Box Tests

B.1 Rifle Tests

Table B.1-1. Rifle Reload Function Test

Conditions	Inputs		1	2	3	4	5
	Ammo	A1	–	T	–	–	–
		A2	–	–	–	T	–
		A3	–	–	–	–	T
	Total Ammo	T1	T	–	–	–	–
		T2	–	T	–	T	–
		T3	–	–	T	–	–
Actions	Outputs	Nothing	N	N	N	N	Y
	Ammo	Ammo = Total Ammo	N	Y	N	N	N
		Ammo = 0	Y	N	N	N	N
		Ammo = Magsize	N	N	Y	Y	Y
	Total Ammo	Total Ammo = 0	Y	Y	N	N	N
		Total Ammo = (Total Ammo - Magsize) + Ammo	N	N	Y	Y	N
Results			Passed	Passed	Passed	Passed	Passed

A1	Ammo = 0
A2	0 < Ammo < Magsize
A3	Ammo = Magsize
T1	Total Ammo = 0
T2	0 < Total Ammo < Magsize
T3	Magsize < Total Ammo <= Capacity

Table B.1-2. Rifle Fire Function Test

Conditions	Inputs		1	2
	Ammo	A1	T	–
		A2	–	T
Actions	Firing	Gun Fired	N	Y
		Gun NOT Fired	Y	N
Results			Passed	Passed

A1	Ammo = 0
A2	Ammo > 0

Table B.1-3. Rifle Pickup Ammo Function Test

Conditions	Inputs		1	2	3	4
	Total Ammo	T1	T	–	–	–
		T2	–	T	–	–
		T3	–	–	T	–
		T4	–	–	–	T
Actions	Total Ammo	Total Ammo = Total Ammo + Ammo Pickup	Y	Y	Y	Y
		Total Ammo = Capacity	N	N	N	Y
	Ammo Pickup	Picked Up	Y	Y	Y	N
		NOT Picked Up	N	N	N	Y
Results			Passed	Passed	Passed	Passed

T1	Total Ammo = 0
T2	0 < Total Ammo < (Capacity - Ammo Pickup)
T3	(Capacity - Ammo Pickup) <= Total Ammo < Capacity
T4	Total Ammo = Capacity

B.2 Pistol Tests

Table B.2-1. Pistol Reload Function Test

Conditions	Inputs		1	2	3	4	5
	Ammo	A1	–	T	–	–	–
		A2	–	–	–	T	–
		A3	–	–	–	–	T
	Total Ammo	T1	T	–	–	–	–
		T2	–	T	–	T	–
		T3	–	–	T	–	–
Actions	Outputs	Nothing	N	N	N	N	Y
	Ammo	Ammo = Total Ammo	N	Y	N	N	N
		Ammo = 0	Y	N	N	N	N
		Ammo = Magsize	N	N	Y	Y	Y
	Total Ammo	Total Ammo = 0	Y	Y	N	N	N
		Total Ammo = (Total Ammo - Magsize) + Ammo	N	N	Y	Y	N
Results							

A1	Ammo = 0
A2	0 < Ammo < Magsize
A3	Ammo = Magsize
T1	Total Ammo = 0
T2	0 < Total Ammo < Magsize
T3	Magsize < Total Ammo <= Capacity

Table B.2-2. Pistol Fire Function Test

Conditions	Inputs		1	2
	Ammo	A1	T	–
		A2	–	T
Actions	Firing	Gun Fired	N	Y
		Gun NOT Fired	Y	N
Results			Passed	Passed

A1	Ammo = 0
A2	Ammo > 0

Table B.2-3. Rifle Pickup Ammo Function Test

Conditions	Inputs		1	2	3	4
	Total Ammo	T1	T	–	–	–
		T2	–	T	–	–
		T3	–	–	T	–
		T4	–	–	–	T
Actions	Total Ammo	Total Ammo = Total Ammo + Ammo Pickup	Y	Y	Y	Y
		Total Ammo = Capacity	N	N	N	Y
	Ammo Pickup	Picked Up	Y	Y	Y	N
		NOT Picked Up	N	N	N	Y
Results			Passed	Passed	Passed	Passed

T1	Total Ammo = 0
T2	0 < Total Ammo < (Capacity - Ammo Pickup)
T3	(Capacity - Ammo Pickup) <= Total Ammo < Capacity
T4	Total Ammo = Capacity

B.3 Cross Tests

Table B.3-1. Cross Attack Function Test

Conditions	Inputs		1	2
	Range	R1	T	–
		R2	–	T
Actions	Attacking	Attack	Y	N
		Don't Attack	N	Y
Results			Passed	Passed

R1	In Range
R2	! In Range

B.4 Knife Tests

Table B.4-1. Knife Attack Function Test

Conditions	Inputs		1	2
	Range	R1	T	–
		R2	–	T
Actions	Attacking	Attack	Y	N
		Don't Attack	N	Y
Results			Passed	Passed

R1	In Range
R2	! In Range

B.5 Flashlight Tests

Table B.5-1. Flashlight Function Test

Conditions	Inputs		1	2	3	4
	Charge	C1	T	F	–	–
		C2	–	–	–	–
		C3	–	–	F	T
	In Use	F1	T	T	–	–
		F2	–	–	T	T
Actions	Light	Light is On	N	Y	N	N
	Charge	Charging	N	N	Y	N
		Decharging	N	Y	N	N
Results			Passed	Passed	Passed	Passed

C1	Charge <= 0
C2	0 < Charge < Max
C3	Charge >= Max
F1	Turned On
F2	Turned Off

Appendix C

Black Box Tests

C.1 Collectables Tests

Table C.1-1. Note Testing

Note	Can Pick Up	NOT in Inventory Before Picked Up	In Inventory After Picked Up
1	Y	Y	Y
2	Y	Y	Y
3	Y	Y	Y
4	Y	Y	Y
5	Y	Y	Y
6	Y	Y	Y
7	Y	Y	Y
8	Y	Y	Y
9	Y	Y	Y
10	Y	Y	Y
11	Y	Y	Y
12	Y	Y	Y
13	Y	Y	Y

Table C.1-2. Audio Log Testing

Audiolog	Can Pick Up	NOT in Inventory Before Picked Up	In Inventory After Picked Up	Clip Plays	Plays Clip Properly
1	Y	Y	Y	Y	Y
2	Y	Y	Y	Y	Y
3	Y	Y	Y	Y	Y
4	Y	Y	Y	Y	Y
5	Y	Y	Y	Y	Y
6	Y	Y	Y	Y	Y
7	Y	Y	Y	Y	Y
8	Y	Y	Y	Y	Y
9	Y	Y	Y	Y	Y
10	Y	Y	Y	Y	Y

Table C.1-3. Upgrades & Perks Testing

Can Pick Up	Perk Menu Shows # of Upgrades	Flashlight Perks Works	Pistol Perks Works	Rifle Perks Works	Health Perks Works
Y	Y	Y	Y	Y	Y

Table C.1-4. Artifact Testing

Artifact	Can Pick Up	NOT in Inventory Before Picked Up	In Inventory After Picked Up
1	Y	Y	Y
2	Y	Y	Y
3	Y	Y	Y
4	Y	Y	Y
5	Y	Y	Y
6	Y	Y	Y
7	Y	Y	Y
8	Y	Y	Y
9	Y	Y	Y
10	Y	Y	Y