Software Test Plan Document
Escape Keck
Video Game

Table of Contents

8.1 Introduction

This document outlines our testing strategy to make sure Escape Keck works well and as intended before it's released. In the gaming world, where player experience is crucial, we're focusing on using playtesting as a key part of our testing approach.

The remainder of this document is structured as follows, Section 8.2 contains the Unit Test Plan, Section 8.3 contains the Integration Test Plan, Section 8.4 contains the Acceptance Test Plan, Section 8.5 contains the Test Configuration Control, Section 8.6 contains the Items Not Tested and Section 8.7 contains the Test Verification Matrix.

8.2 Unit Test Plan

This section outlines the testing plan for key components of the game. We'll be focusing on testing the following parts: the player movement, the functionality of the User Interface (UI), the mechanics for object interaction and the inspect mode, collision for object pick up, and rendering for assets and materials to ensure that everything is working as intended.

8.2.1 Unit Tests Planned

Since we are developing a game, we will utilize multiple playtesting sessions throughout the development process. The playtest will be performed with individuals, within and outside of the development team, who will be playing our game while in development. The purpose of these tests is to 1) test out game mechanics and features to see if the behavior matches what was seen during development in the Unreal Engine editor, 2) receive feedback and constructive criticism from those with an outside perspective, and 3) see if any player can "break" our game by finding any unexpected bugs. In short, the playtests are designed to stress-test our game and give us useful feedback that we can utilize to our advantage as we continue the development of our game. Below is a more detailed list of what we will be looking to test in a playtest:

1. **Rendering for Assets and Materials:** Testing the graphics to make sure they look as intended in the game and work well on different devices.
2. **Player Movement:** Testing how the player character moves in the game to make sure it feels smooth and the controls are working as expected.
3. **Collision for Object Pickup:** Checking that objects react correctly when players try to pick them up in the collision area and do not react when outside of this area.
4. **Object Interaction Mechanics:** Testing how players interact with objects in the game, like pick-up, and the inspection mode mechanics such as rotating, zoom, and crafting.
5. **User Interface Functionality:** Checking that all the buttons and menus work properly, are easy to use, and are intuitive to the player.

8.2.2 Unit Test Procedures

The following test procedures require the player to 1) open the .exe file to run the game, and 2) choose the “new game” option in the main menu to start the game.

1. **Rendering for Assets and Materials:**
   - This test is simple. If the game looks as intended, then everything is fine.
   - There is no error message, but if a material looks low quality or any model looks incorrect (i.e. a table looking like a stick), that means there is an issue with the way the engine is rendering the assets.
2. **Player Movement (Top-down Controller):**
   - The player will click a legal location on the current level map where the player character can move.
   - Once they click, we should observe the character start the walk animation and make their way over to the chosen location.
   - No error message is displayed to the user, however, if the action does not work there will be visual feedback (i.e. they will not be able to move to the desired legal location)
3. **Collision for Object Pickup:**
   - The player will be able to pick up an object when they are within range (a one-meter radius in-game) and press the E key to pick up the object and it will automatically store it in their inventory
   - When the player opens their inventory, they should be able to see the object they just picked up
   - No error message will be displayed if the functionality fails. The player will be able to visually tell the interaction failed by seeing 1) the object is still on the ground after they press E or 2) the object is not in their inventory after they press E.
4. **Object Interaction Mechanics:**
   - The player will first use the Tab key to open the inventory view.
   - They will be able to use the left mouse button to drag and drop objects that are craftable onto each other to make new objects.
   - They will then select an object that they want to inspect, and from the Heads Up Display (HUD) they will click on the inspect button.
   - Once this button is clicked, the inventory UI will disappear, and the inspect mode UI will come up.
   - Once in the inspect mode, they will be able to use the middle mouse button to zoom in and out of the object and hold the right mouse button to rotate the object.
   - There is no error message to be displayed to the user, if any functionality fails to work they will be able to tell visually (i.e. object not rotating when attempting to rotate, or not zooming in, etc.)

5. **User Interface Functionality:**
    - The player will use the Tab, Esc, or T keys to open the Inventory, Pause Menu, or the Tasks Checklist respectively.
    - If the UI is working correctly, we should observe these menus open and close.
    - Additionally, each of the UI functionalities can be visually observed to determine if they are working or not such as buttons, seeing the picked-up item icons in the inventory, or seeing the completed/current tasks.
    - No error message is displayed to the user, however, if any of these functionalities fail, there will be visual feedback (i.e. nothing happening after clicking on a button, or something not disappearing when it should)
6. **Player Movement (First-person Controller):**
    - The player will use the W, A, S, and D keys to move around the current level.
    - When they are holding down one of these keys we should observe the character start the walk animation and stop moving once the key is released.
    - No error message is displayed to the user, however, if the action does not work there will be visual feedback (i.e. they will not be able to move at all or just walk in place)

8.3 Integration Test Plan

Our integration test plan is to test the functionality in the numerical order they were listed in the previous section. The reasoning for this is that each mechanic helps to build on the next one.

8.3.1 Integration Tests Planned

1. **Rendering Test:** The standalone testing of the rendering of assets and materials. Input would be the Blender assets and output would be the render given by the engine. **NOTE**: this test should be done whenever new assets are ready to be deployed into the development environment. This test could and SHOULD be carried out whenever assets are ready.
2. **Top-Down Controller Test:** The standalone testing for the foundation of the first two levels of the game. If that part of the game is not working, then the player will not be able to accomplish anything and they will not be happy. That being said, we must ensure the third-person controller is functioning as intended. Input would be left-mouse clicks and the output would be the character moving to the location of the mouse click so long as it is a valid location in the environment. If the location is not valid, the player will be blocked from entering the area.
3. **Rendering and Top-Down Test:** The first integration test between modules. This test will make sure that the player can move around the newly rendered environment. Inputs

would be a combination of tests 1 and 2. Outputs will also be a combination of tests 1 and 2.

4. **Object Collision Test:** Integrated test with the rendering and player controller. The player will move to an object and pick it up when they are within a one-meter radius. Inputs will be the same as test 2 with the addition of the E key, which the player will press to pick up the object when within range.
5. **Object Interaction and User Interface Test:** Integrated test with the rendering, player controller, and object collision mechanic. Once the player picks up an item, they should be able to see it in their inventory. Inputs will be the Tab key to open the inventory view, the HUD buttons that appear after opening the inventory view and clicking on an object icon with the mouse, the middle mouse button to zoom in and out on an object in the inspect mode, the right mouse button to rotate objects in the inspect mode, and the left mouse button to drag and drop craftable objects onto each other within the inventory view. All outputs will be visual and no error messages will be thrown if anything goes awry.
6. **Fully Integrated Top-Down Environment Test:** Integrates all the above pieces to complete the third-person portion of the game. All inputs are a combination of all previous tests and the same goes for the outputs.
7. **First-Person Controller Test:** Standalone testing (or integrated if assets are ready) for the foundation of the remaining level of the game. Inputs would be W, A, S, and D keys and the output would be the character moving up when W is pressed, left when A is pressed, down when S is pressed, and right when D is pressed.

8.3.2 Integration Test Procedures

The following section gives step-by-step directions for each of the above tests. All tests will be done within the Unreal Engine editor during development.

1. **Rendering Test:** The Blender assets must first be downloaded or pulled from GitHub. Then, load the assets into Unreal Engine and see how it renders everything. If something is not rendered correctly, then that information will be relayed to the asset designer.
2. **Top-Down Controller Test:** While assets are being developed, this testing will occur in a separate Unreal Engine environment. To test the functionality, use the left mouse click button to move the character to a valid location. You will visually see the player move to the desired location if the code is working as intended. If not, well then it’s time to debug.
3. **Rendering and Top-Down Test:** Once some assets are complete, then load them into a new environment and import the top-down player controller. Test out the new environment paired with the player controller by simply moving the player around using the controls. If anything is off, take note and try to make changes as necessary.
4. **Object Collision Test:** Using the environment from the previous test, test out the new features with the inputs described in the previous section. Make note of any bugs and try to fix them.

5. **Object Interaction and User Interface Test:** Using the environment from test 4, test out the object interaction with the inputs mentioned in the previous section. Be sure that the User Interface functions as expected. See prior sections to see the expected User Interface functionalities. Once again make note of bugs or other issues, and try to fix them.
6. **Fully Integrated Top-Down Environment Test:** Make a .exe file, and open (or download) it. Once opened, simply play the game. Be sure that all functionality matches what was seen in the Unreal Engine editor.
7. **First-Person Controller Test:** Steps for this are very similar to test 1. Just make sure the player controller is functioning correctly when the appropriate inputs are pressed. As always, make note of any bugs and try to fix them.

8.4 Acceptance Test Plan

Our acceptance test plan, for this semester, is to present the finished Minimum Viable Product (MVP) demo of the game to our audience. In the future, the acceptance test plan will cover the full game with all three levels. This will be similar to how the playtests are done, but rather than focusing on one singular functionality or mechanic of the game, they will be playing the game in its entirety.

8.4.1 Acceptance Tests Planned

We will split the acceptance tests into three sections, one for each level's playthrough. A playthrough refers to the act of playing a game from start to finish, in our case, it refers to playing an entire level from start to finish. This section outlines what aspects will be tested in the full playthrough in addition to the game's main mechanics. Completing each level will require the player to utilize the mechanics and functionalities, so the following are the general aspects of each level that will be tested.

Level 1:

- initial cutscene
- tutorial clarity (for game mechanics)
- task difficulty
- story progression
- atmosphere

Level 2:

- level transition (from level 1)
- elevator cutscene
- puzzle difficulty
- player engagement

Level 3:

- level transition (from level 2)
- player controller change

- atmosphere change
- story progression
- endings

8.4.2 Final Acceptance Test Procedures

The following test procedures require the player to 1) open the .exe file to run the game, and 2) choose the “new game” option in the main menu to start the game.

Level 1 Test:

- They will observe the starting cutscene, and hear the narrator explain why the player character is there (for their job as a teaching assistant (TA)).
- Once the game begins, they will be given a series of TA tasks to complete. These tasks will serve as a tutorial for the game mechanics, which will also be used in solving the puzzles.
- During this time, the player will follow the narrator’s instructions to complete these tasks.
- Once they complete the tasks, the player character will get locked inside Level 1 and the story progression will begin. This is also the point where the atmosphere will get darker.

Level 2 Test:

- The player character will walk through the Annex doors to transition into Level 2 from Level 1.
- The narrator will tell them to check the alternate exit through the Annex. This action will trigger the elevator cutscene where the entire Annex will begin to lower.
- After this cutscene, the player will get locked in the Annex and will need to solve a variety of puzzles to unlock the door. These puzzles will follow a chain pattern where one leads to the next.
- Once they solve all the puzzles, they will acquire the key and unlock the door to Level 3.

Level 3 Test:

- Once the player enters Level 3, the Original Keck Lab, the player controller will switch from top-down to first-person.
- The atmosphere will also change significantly into a full fledged horror game atmosphere.
- The story will progress further as the player finds out about the horrors that took place in this level and uncovers the secrets that the narrator has been hiding.
- Once they complete all the puzzles in this level, they will be able to escape and end the game.

8.5 Test Configuration Control

For playtesting our game, since GitHub is our chosen version control system, we are focusing on utilizing GitHub to facilitate the distribution and coordination of playtesting sessions. GitHub will serve as a central hub for managing the game's source code, assets, and builds, ensuring that testers have access to the latest version of the game for evaluation.

Unlike traditional software development scenarios where separate areas for tests are maintained within the version control system, playtesting of a game involves a more direct approach. Testers can simply download the .exe file from the GitHub repository onto their laptops and run it locally for playtesting purposes. This streamlined process eliminates the need for maintaining separate areas for tests within GitHub, as the focus is primarily on assessing the user experience and gameplay mechanics rather than conducting unit or integration tests. To receive feedback after the playtesting, we will provide testers with a Google Form where they can rate the functionality, mechanics, story, and the overall feel of the game.

8.6 Items Not Tested

We will not be testing the music and the audio for character/narrator conversations because it is not a priority at the moment. We need to have the functionality and mechanics working perfectly before we can consider these aspects.

8.7 Test Verification Matrix

5.2.1.1	The character shall move to the location on the current level that the player clicks with their left mouse button.	**Player Movement (Top-down Controller)**
5.2.1.2	The camera shall follow the player character from the top looking down at them as they move.	**Player Movement (Top-down Controller)**
5.2.1.3	The player shall move forward by pressing the W key on the keyboard.	**Player Movement (First-person Controller)**
5.2.1.4	The player shall move left by pressing the A key on the keyboard.	**Player Movement (First-person Controller)**

5.2.1.5	The player shall move right by pressing the D key on the keyboard.	**Player Movement (First-person Controller)**
5.2.1.6	The player shall move backward by pressing the S key on the keyboard.	**Player Movement (First-person Controller)**
5.2.1.7	The camera shall show the level map from the eye level of the character.	**Player Movement (First-person Controller)**
5.2.2.1	There shall be a backpack icon on the top left corner of the screen that the character can click with the left mouse button to open the inventory view.	**User Interface Functionality**
5.2.2.2	The player shall also be able to use the Tab key on the keyboard to toggle the inventory view.	**User Interface Functionality**
5.2.2.3	There shall be a drop-down menu on the top left corner of the screen with a gear icon for settings.	**User Interface Functionality**
5.2.2.4	There shall be a drop-down menu on the top left corner of the screen with a magnifying glass icon for current-level tasks.	**User Interface Functionality**
5.2.2.5	There shall be a currency indicator on the top right corner of the screen to let the player know how much currency they have accumulated from completing their tasks.	**User Interface Functionality**
5.2.3.1	The player shall be able to pick things up and put them into their inventory by pressing the E key on their keyboard when standing on top of an object.	**Collision for Object Pickup**
5.2.3.2	The player shall be able to inspect objects by choosing the inspect mode from the aforementioned menu.	**Object Interaction Mechanics**
5.2.3.3	The player shall be able to rotate items in inspect mode by holding down their right mouse button to drag around the object.	**Object Interaction Mechanics**

5.2.3.4	The player shall be able to view the description for each item when they hover over the item icon with their cursor in the inventory view.	**User Interface Functionality**
5.2.3.5	The player shall be able to view if an item is craftable when they hover over the item icon with their cursor in the inventory view right under the description.	**User Interface Functionality**
5.2.3.6	The player shall be able to combine items to make new ones when an item specifies “craftable” by holding down the left mouse button to drag and drop/ let go of one item onto another in the inventory view.	**Object Interaction Mechanics**
5.2.3.7	The player shall be able to zoom in and out of objects in inspect mode by scrolling using the middle mouse button.	**Object Interaction Mechanics**
5.2.3.8	The player shall be able to drop items from their inventory by clicking on the drop option from the aforementioned menu.	**Object Interaction Mechanics**
5.2.4.1	The transition between Level 1 (Main Keck Lab) and Level 2 (Annex) shall occur by passing through the Annex doors.	**Level 2 Test**
5.2.4.2	The transition between Level 2 (Annex) and Level 3 (2nd Keck Lab Below) shall occur once you unlock the Annex doors and pass through.	**Level 3 Test**
5.3.1	Character movement shall start instantaneously from the moment the player clicks with the left mouse button on the location they want the player to go to.	**Player Movement (Top-down Controller)**
5.3.2	Picking up items shall happen instantaneously.	**Collision for Object Pickup**
5.3.3	Using the inventory shall happen instantaneously.	**User Interface Functionality**

5.3.4	Opening the inspect mode shall happen instantaneously.	**User Interface Functionality**
5.3.5	Using the mechanics in inspect mode shall happen instantaneously.	**Object Interaction Mechanics**
5.3.6	Dropping an object shall happen instantaneously.	**Object Interaction Mechanics**
5.3.7	The initial game loading time shall be no longer 30 seconds.	**Level 1 Test**
5.3.8	The level transitions shall happen instantaneously.	**Level 2 Test & Level 3 Test**
5.3.9	The Annex elevator cutscene shall take approximately 10 seconds.	**Level 2 Test**