Friday the 13th Coding Project Final Report

For use in CS 440
University of Illinois at Chicago
Prepared by Group 27:
Alberto Ramirez, Aqsa Arif, and Lucy Harck
Fall 2022

Table of Contents

List of Figures

I Project Description

1 Project Overview

Friday the 13th is a horror survival game in which players will take the role of a police officer attempting to solve a missing person's case. A monster will threaten their life as they search for clues. The player will interact with NPCs and collect items in order to complete puzzles in a meaningful way and to test their decision making skills. Players can pick up items, open their inventory, drop items, combine items, shoot objects, and use items on the environment around them. The player will solve puzzles and make choices that will help them navigate new places or escape from dangerous situations. The items collected throughout the game will be assigned a weight and the player will only be able to carry a limited weight. The outcome of the game is to survive until the end, the possible endings range from escaping the killer, becoming the killer or dying by the killer.

2 Project Domain

The game is intended for people who enjoy horror or survival games on PC that can be played on the average build to provide access to more users. Intended age group would be 17 years or older.

3 Relationship to Other Documents

Group 15's Friday The 13th Development Project Report was used as a foundation for objectives and theme of the project.

4 Naming Conventions and Definitions

API: This stands for Application Programming Interface, it's how different computers are able to call each other's functions without predefining them in their own program.

NPC: This term stands for Non-playable character. They usually assist in the progression of the narrative in a story.

Maze/Main Map: The puzzle level of the game consists of a maze the player has to navigate through. They'll need to find items on this stage and avoid the monster.

Tutorial: This is the introductory level of the game where the player learns the basic mechanisms of gameplay.

Main Menu: The opening menu of the game that allows a player to choose between new game/load game/ exit game.

Pumpkin Monster: The primary antagonist of the game. A humanoid pumpkin headed creature roaming the forest maze looking for more victims.

Main Character/Hero: Our main character is a police officer named Jason who's investigating the forest. He's looking for a missing child named Mary.

Mary el Sacrificio: The game's unfortunate victim. The game centers around finding her and ends upon finding her dead body in the forest.

4a Definitions of Key Terms

NPC: Non-Playable Character. Characters within a game that exist in the environment but cannot be controlled by the player.

Unity: Game engine that the developers used to create the game in.

Plastic SCM: A program which syncs the developers changes to the game across their different devices.

UI: Short for user interface. The text, boxes, and buttons that are shown on screen that the user uses to interact with the program.

Lower/High/Mid Tier Systems: Synonyms for computers on the lower, middle, or higher end of the average performance spectrums.

Framerate: How many individual still frames from the video game a user's computer displays per second.

New game: The menu option a player can select to start a new version of the game.

Load game: The menu option a player can select to load one of the three saved slots in the game.

Save game: The menu option a player can select to save their current game into one of the three game slots.

Exit game: The menu option a player can select to end and shut down the game application.

Police station: This is the tutorial stage of the game. The player is introduced to the basic mechanisms of the gameplay in this stage level.

Merge/Combine: This is a player item option to combine two items in their inventory to create a completely new item. This may make the previous version of the item

inaccessible.

Equip: This is an inventory option a player has that enables them to hold the item in their knapsack. For example, a player has the option to take a flashlight out of their knapsack and use it on the level by having the main character hold it. Other items can be held as well, with a limit of 1 item being equipped per hand, but may not have any visual effects like the flashlight which can be turned on or off.

Dialogue: This is the story narration of the game. Can consist of the player thinking to themselves or interacting with an NPC.

Kitchen: This is a room in the tutorial level where the player learns about combining items. In the kitchen of the police station the player is prompted to pick up a cup and milk from the fridge, select the cup and put it in the coffee machine to obtain a cup of coffee. From there the player is prompted to combine the cup of coffee with milk and successfully combine the items to obtain a latte.

Office: This is a room in the police station where the main character initiates a dialogue with their boss, the chief of police, about a case file. They receive the case file here and are prompted to "look/inspect" the item.

Look/inspect: This is an inventory option where players can take a closer look at the items in their inventory. Specifically used to read case files and look at items to figure out puzzle passcodes or hints.

Forest: The main puzzle stage of the game. The level after the tutorial where the player would encounter a pumpkin monster and look for the dead body of the victim.

Forest pathway: This is the maze pathway of the forest where the main gameplay occurs and mainly where the player navigates to find helpful items and tries to avoid monsters.

Item: Usable items in the game that have a weight and that the player can possibly equip, unequip, consume, drop, or pick up.

Monster: The antagonist of the game. A pumpkin monster that lurks around the maze looking for the player.

Victim: This is Mary el Sacrificio who is initially reported missing. The game ends upon finding her dead body.

Player: The main protagonist of the game, a police officer named Jason. He navigates the forest in search of the missing little girl Mary.

Case file: This is the inventory item that the player can inspect to initiate the task of looking for the missing girl Mary.

Knapsack: The backpack of the game that holds all the items and has a weight

carrying limit.

Police car: This is the stage item that lets the player go back to the tutorial stage. Upon interacting with this item they return to the police station.

Dead body: This is the body of Mary el Sacrificio, the victim of the missing person's case our player is working on. Upon finding her the game is over and we have an ending narration of what happened after they found her body.

4b UML and Other Notation Used in This Document

UML features we used include hard edged rectangles and soft edge rectangles to represent game scenes or levels. The circle represents an item and the stick figure of a person represents the player. The rectangle with a label on top represents the class layout with the main class on top as a header and inner items as subclasses of the main class. Lastly is the pointing arrow, this represents a specific sequence of events usually with a text label describing what event in particular or showing structure inheritance between different classes.

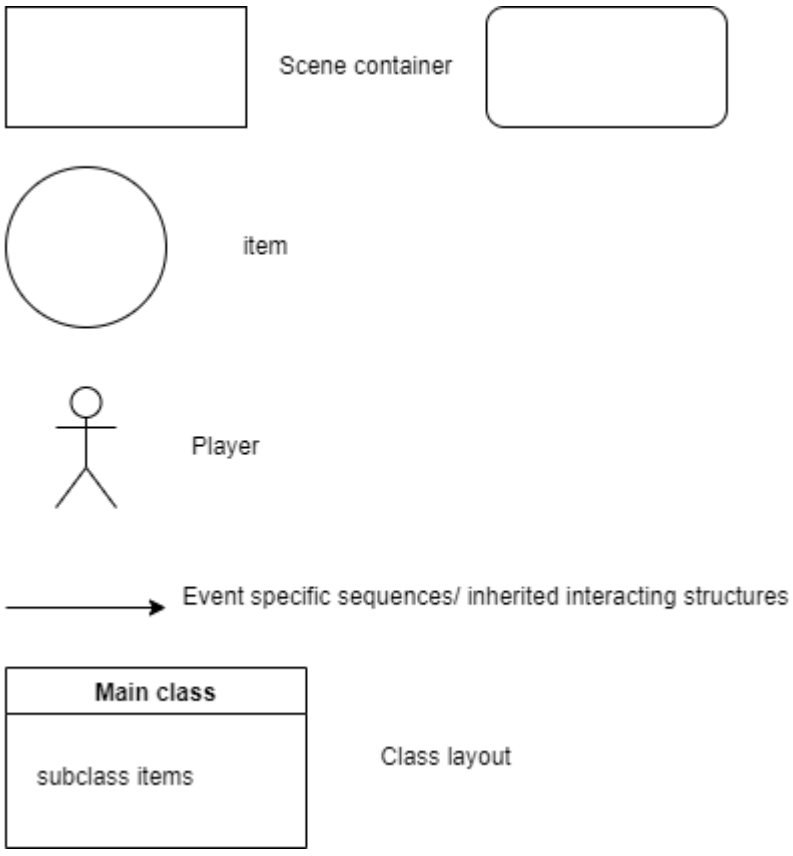

Figure 1 - UML symbol notations.

4c Data Dictionary for Any Included Models

Items in the game are structures to track map position and edit game interaction of being enabled to combine with knapsack items or map stage items. Certain items, like

the flashlight, have special features to enable being a light source upon being equipped. All items are assigned a weight.

It should also be mentioned that a user's save data is saved / loaded into a struct while the game is being played, and then converted from and to a XML file in between play sessions.

Dataset of all items consists of a structure with the list of all items of the game.

Knapsack items consist of a structure that is present in player inventory. These items are referenced from the dataset of all the items in the game and cross checked if they can be added to inventory by checking the weight of the item.

Dropped items location is a structure property of the item itself, the x,y,z coordinates of the item upon dropping it are updated as the player drops the item from their knapsack to a spot on the map.

Monster pathway is a group of cubes in game that act as positions for the monster to follow along its path. In the monster's movement script, the script keeps track of its next position to move the monster to. Once it reaches the last cube in the group, the monster starts back at the beginning of the path.

Event prompts are dialogues that pop up on the character's screen as player actions initiate event triggers. These are initiated by function calls of lambda assignment upon sequence of events.

Character interaction sequence this is a basic boolean upon first interaction with an NPC. The NPC gives an opening dialogue, upon second and third interaction they have a follow up dialogue in the form of gameplay hints to navigate level.

Game menu and scenes are separate scene files. They are only similar in the API and data structures they use for gameplay.

Item combined dataset is the feature in the item structure itself that enables it to be combined with another item. The item combination is specified by individual items itself. For example, a cup of coffee + milk = latte. Upon a non-valid combination nothing happens, for example, cup + flashlight = cup + flashlight + prompt that nothing happened.

Useful equipment is an item feature that is specific to an item structure. Certain items, like flashlights, have a special function upon being equipped. A function call to act as a light source or item specific function upon being equipped.

Items that can interact with stage items are also a part of the item structure. For example, using a cup to interact with the coffee machine to get a cup of coffee back. These functionalities are set by event triggers when items are selected to interact with stage items. Another example is using a key to interact with a door to trigger the door being unlocked.

Dataset for item weights are a part of the item structure. It's a basic integer/float value assigned to the item.

II Project Deliverables

The game is developed through Unity's engine in a 3D environment. Collaboration for version control of the assets and scripts are done through Plastic SCM to check-in/check-out changes.

5 First Release

The first release involved developing basic gameplay functionality, and having the developers get situated with Unity and the Plastic SCM infrastructure. Once situated, the following abilities were implemented:

- Ability for the user to pick up items,
- Ability for the monster NPC to follow predetermined paths,
- Ability for the monster NPC to diminish the players healthplayer's health while in distance
- Implementation of saving and loading save files.
- Implementation of a UI for health and knapsack item count.
- Creation of the map including a forest layout with trail path and trees.
- Creation of monster and item pick-up assets.
- Ability to load into the game from the main menu
- Creation of the vignette effect when the player gets close to the monster NPC
- Implementation of first person character controls for keyboard/mouse, and gamepad using Unity's standard first person controller

Additionally, below a UML diagram can be seen, representing the players inventory, and their interactions that they may take while playing the game.

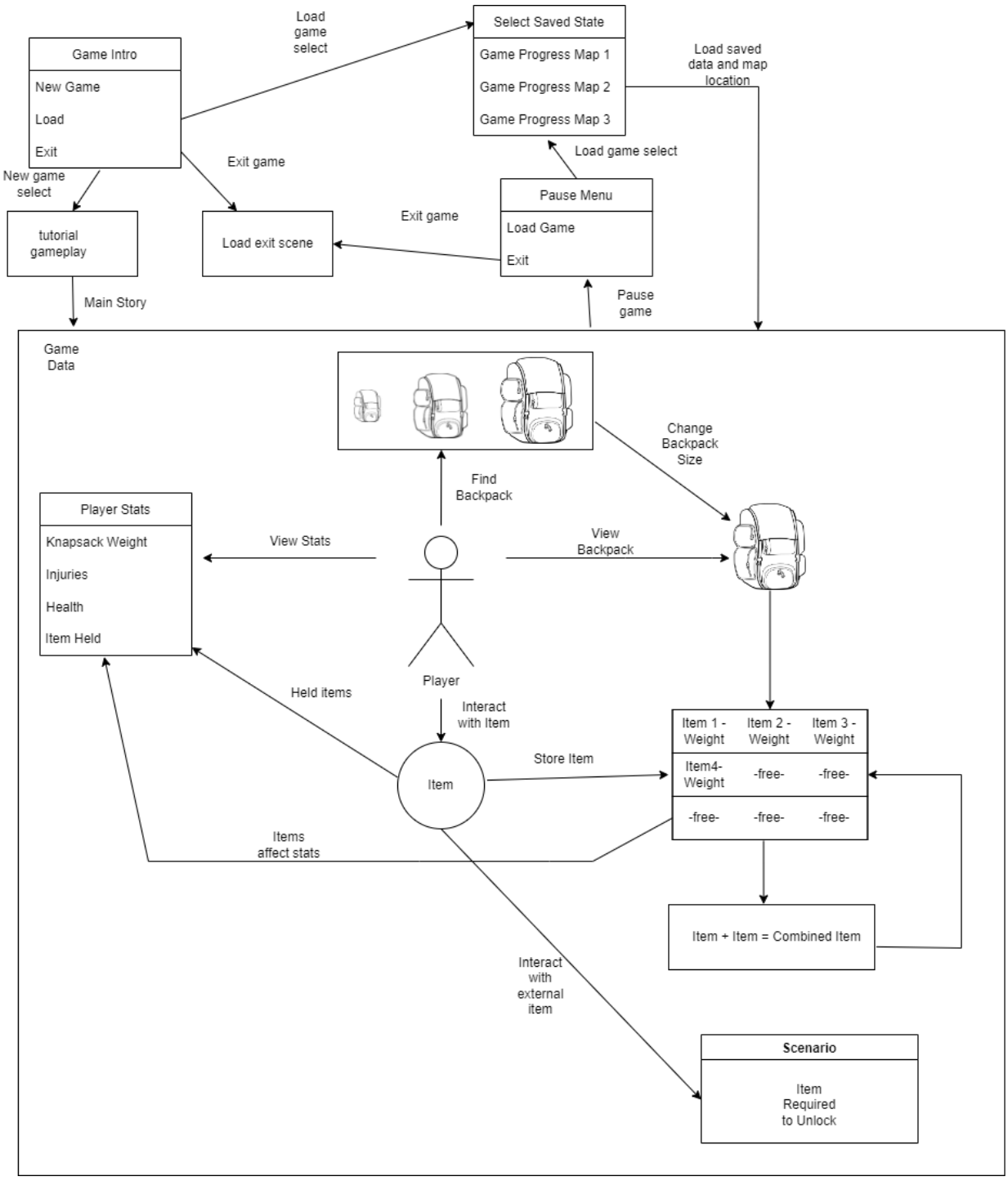

Figure 2 - Scenario 1 diagram.

6 Second Release

The second release involved adding functionality to the previous release, and adding new features to the game. The following game mechanics were implemented:

- Creation of tutorial area map.
- Creation of first level map.
- Implementation of inventory UI design
- Created procedural inventory item generation for inventory
- Implementation of equip, drop, open, and close, buttons for each inventory item.
- Creation of new character model for the monster NPC
- Change of level ordering to make the progression go from the main menu to the tutorial phase, to the first level.
- Selected assets to be added to the map in the next release
- More concretely designed the flow of the gameplay with UML diagram (seen below) to determine how the game should commence.

Seen below is a diagram stating the flow of the gameplay, as mentioned in the bullet points above.

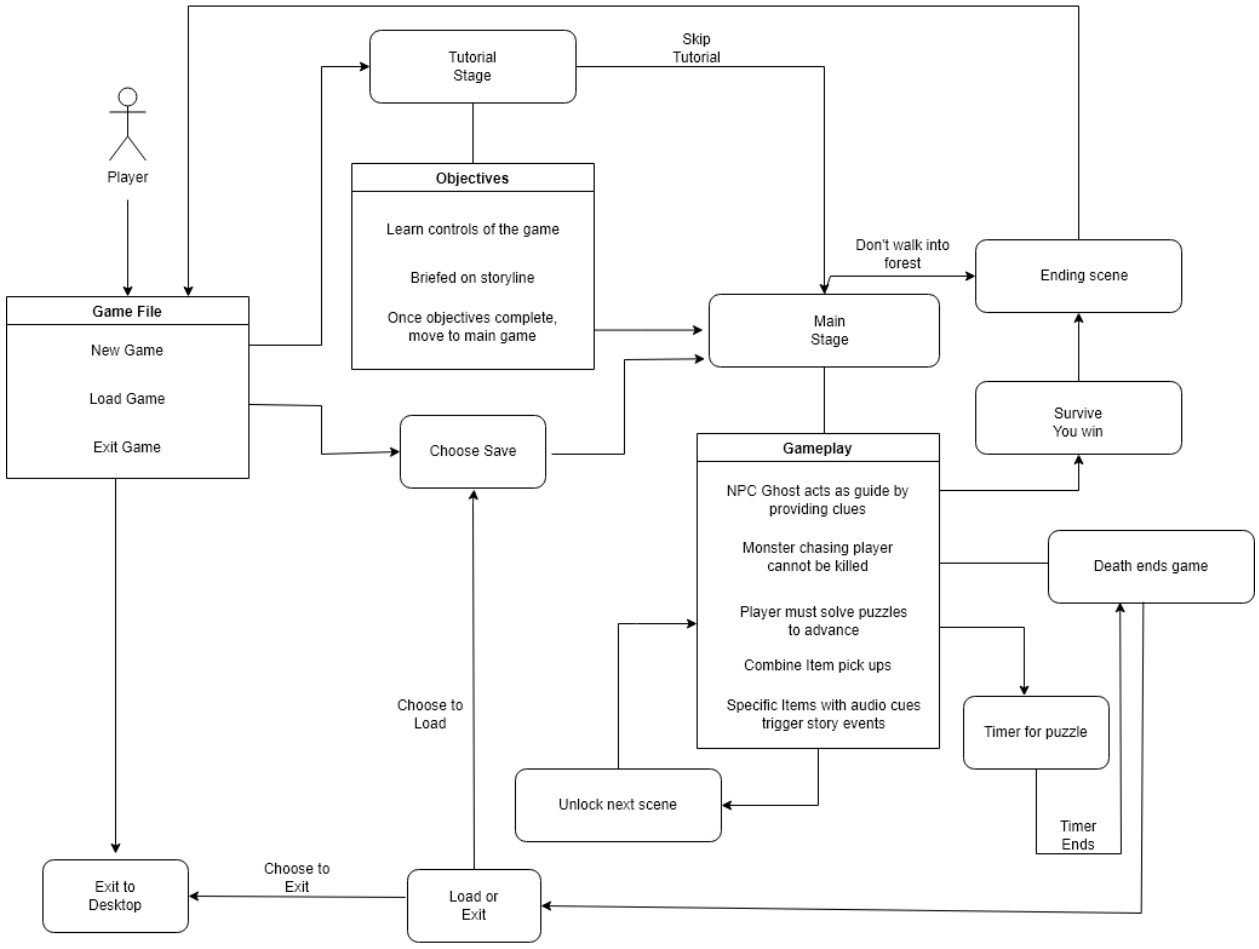

Figure 3 - Scenario 2 diagram.

7 Comparison with Original Project Design Document

This rendition of Friday the 13th went a slightly different direction than its predecessor. The main difference being that the previous design was completely text based, and this rendition

takes place in a 3d game. In addition, the game was changed from a multiplayer game to a singleplayer game to give the player a sense of feeling alone. Also, it should be noted that the Jason character in the previous game was changed to a pumpkin themed monster to avoid possible copyright issues with the 'Jason' character. Another change that was added was the disallowance of the player's ability to kill the killer, only allowing the player to repel the monster as a non-killable monster allows for more of a sense of dread while the player is playing the game.

Additions that were added include giving the game a story. The game will be expected to have a visual novel aspect where the player can interact with NPCS along it's quest to complete the game, and a story will be added where the player will be given the role of a police officer who's job is to find a missing girl. A ghost NPC will lead the character along their path, giving them visual and auditory hints as they explore the levels.

Regardless of these changes, the core principles of collecting objects and navigating around the map, with a limited amount of inventory space, based on weights remains the same.

III Testing

8 Items to be Tested

ID 1 - Game Items

ID 2 - Monster Path

ID 3 - Player Health

ID 4 - Map Restrictions

ID 5 - Item Weight

ID 6 - UI Items

ID 7 - Pick up items

ID 8 - Code Boundary Testing

9 Test Specifications

ID 1 - Game Items

Description: Items contain the right attributes.

Items covered by this test: Game items, Knapsack UI

Requirements addressed by this test: Functional and Usable requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the items produce repeated accurate results.

Environmental needs: Free Banana Unity Asset (Any asset can be used to test.)

Intercase Dependencies: Player pick up, Knapsack inventory.

Test Procedures: Debug log the item attributes.

Input Specification: Pick up 2 identical items.

Output Specifications: Console log prints info for attributes of item. Items are displayed in the inventory UI slot.

Pass/Fail Criteria: Pass : Item attributes match info displayed in console log. Player picking up the item adds the correct weight to the knapsack and the sprite and name in the inventory UI represents the item.

Fail: Item attributes do not match info in console log.

Player picking up the item adds the incorrect weight to the knapsack and the sprite and the name does not represent the item in the inventory UI.

ID 2 - Monster Path

Description: The monster has a set path to follow around the map.

Items covered by this test: Monster, Player health, Player model.

Requirements addressed by this test: Functional requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the monster produce repeated accurate results.

Environmental needs: Monster path layout predefined.

Intercase Dependencies: Map Restrictions.

Test Procedures: Monster stays on path.

Input Specification: Player model intercepts monster path. Monster's path has items or objects in the way.

Output Specifications: Monster stays on path by moving.

Pass/Fail Criteria: Pass: Monter stays on path even with player model in the way. Monster does not behave irregularly when moving across items and does not go

through objects.

Fail: Monster's path halts when the player gets in the way. Monster behaves irregularly when coming across items and walks through objects.

ID 3 - Player Health

Description: Player health decreases when monster approaches.

Items covered by this test: Player, Player Health UI, Monster, Monster path.

Requirements addressed by this test: Functional and Usable requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the health related features produce repeated accurate results.

Environmental needs: Player is on a valid map and not interacting with health reducing objects.

Intercase Dependencies: Monster path.

Test Procedures: The player model needs to be within a set distance from the monster.

Input Specification: Player moves player model close to the monster on its path.

Output Specifications: The Player's health bar decreases and screen gets red.

Pass/Fail Criteria: Pass: Player's health UI bar decreases when in close range of monster or dies.

Fail : Player's health UI bar does not decrease when in range of monster or decreases when outside of range of monster.

ID 4 - Map Restrictions

Description: Player model is not allowed to go outside of map.

Items covered by this test: Player model. Map objects. Map.

Requirements addressed by this test: Functional and Usable requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the map produce repeated accurate results.

Environmental needs: Assets / Objects used to design maps.

Intercase Dependencies: Valid scene transition functional.

Test Procedures: Move player model to areas that should not be accessed or walk/fall through.

Input Specification: Player moves player model to areas around that map. Moving to edges of the map.

Output Specifications: Player walks around map close to map restrictions.

Pass/Fail Criteria: Pass: Player does not walk or fall through the map.

Fail: Player walks/falls through the map.

ID 5 - Item Weight

Description: Each item provides the correct weight assigned.

Items covered by this test: knapsack UI, game items.

Requirements addressed by this test: Functional and Usable requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the items produce repeated accurate results.

Environmental needs: Free Banana Unity Asset (Any asset can be used to test.)

Intercase Dependencies: Player pick up, Knapsack inventory.

Test Procedures: Debug log the item attributes.

Input Specification: Pick up 2 identical items, check if weight is the same.

Output Specifications: Console log prints info for attributes of item. Items are displayed in the inventory UI slot.

Pass/Fail Criteria: Pass : Item attributes match info displayed in console log.
Player picking up the item adds the correct weight to the knapsack and the sprite and name in the inventory UI represents the item.

Fail: Item attributes do not match info in console log.

Player picking up the item adds the incorrect weight to the knapsack and the sprite and the name does not represent the item in the inventory UI.

ID 6 - UI Items

Description: Items players interact with have correct properties.

Items covered by this test: knapsack UI, game items.

Requirements addressed by this test: Debug log the item attributes.

Environmental needs: Free Banana Unity Asset (Any asset can be used to test.)

Intercase Dependencies: Player pick up, Knapsack inventory.

Test Procedures: Debug log the item attributes.

Input Specification: Pick up 2 identical items.

Output Specifications: Console log prints info for attributes of item. Items are displayed in the inventory UI slot.

Pass/Fail Criteria: Pass : Item attributes match info displayed in console log. Player picking up the item adds the correct weight to the knapsack and the sprite and name in the inventory UI represents the item.

Fail: Item attributes do not match info in console log.

Player picking up the item adds the incorrect weight to the knapsack and the sprite and the name does not represent the item in the inventory UI.

ID 7 - Pick up Items

Description: Pick up items from map, see it disappear from map and see item reflected in knapsack and drop item from knapsack to see it's location on map.

Items covered by this test: knapsack UI, game items.

Requirements addressed by this test: Debug log the item attributes.

Environmental needs: Free Banana Unity Asset (Any asset can be used to test.)

Intercase Dependencies: Player pick up, Knapsack inventory.

Test Procedures: Debug log the item attributes.

Input Specification: Pick up 2 identical items.

Output Specifications: Console log prints info for attributes of item. Items are displayed in the inventory UI slot.

Pass/Fail Criteria: Pass : Item attributes match info displayed in console log. Player picking up the item adds the correct weight to the knapsack and the sprite

and name in the inventory UI represents the item. See if item disappears from map upon pickup. Similarly, watch item appear upon drop from inventory.

Fail: Item attributes do not match info in console log.

Player picking up the item adds the incorrect weight to the knapsack and the sprite and the name does not represent the item in the inventory UI. Item disapppears upon pickup or drop or location invalid.

ID 8 - Code Boundary Testing

Description: Boundary testing for knapsack slots and item weight.

Items covered by this test: knapsack UI, game items.

Requirements addressed by this test: Functional and Usable requirements are met through in game testing of repeated gameplay results. Reliability and Performance requirements deal with testing if different interactions with the feature produce repeated accurate results.

Environmental needs: Free Banana Unity Asset (Any asset can be used to test,) game script accessible.

Intercase Dependencies: Player pick up, Knapsack inventory.

Test Procedures: Debug log the item attributes.

Input Specification: Pick up a number of items that would exceed weight and number of slots available in the knapsack.

Output Specifications: Console log prints info for attributes of item. Items are displayed in the inventory UI slot.

Pass/Fail Criteria: Pass : Item attributes match info displayed in console log. Player picking up the item adds the correct weight to the knapsack and the sprite and name in the inventory UI represents the item. No more items are added to knapsack upon weight limit being met. No more item pickup upon slot capacity being full.

Fail: Item attributes do not match info in console log.

Player picking up the item adds the incorrect weight to the knapsack and the sprite and the name does not represent the item in the inventory UI. Item slots replace each other in the knapsack. Item is added to the knapsack despite weight limit being met.

10 Test Results

ID 1 - Game Items

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy and Alberto.

Expected Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully.

Actual Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully.

Test Status: Success.

ID 2 - Monster Path

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy.

Expected Results: Monster follows predefined path at assigned speed.

Actual Results: Monster follows predefined path at assigned speed.

Test Status: Success.

ID 3 - Player Health

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy, Alberto, and Aqsa.

Expected Results: Player model approaches monster and loses health.

Actual Results: Player model approaches monster and loses health.

Test Status: Success.

ID 4 - Map Restriction

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy, Alberto, and Aqsa.

Expected Results: Player model cannot exceed bounds of map.

Actual Results: Box collider plane stops player model from exceeding limits.

Test Status: Success.

ID 5 - Item Weight

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy and Alberto.

Expected Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned.

Actual Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned.

Test Status: Success.

ID 6 - UI Items

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy and Alberto.

Expected Results: Asset loaded into scene and item features like weight, appearance, and location are correct.

Actual Results: Asset loaded into scene and item features like weight, appearance, and location are correct.

Test Status: Success.

ID 7 - Pick up Items

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy and Alberto.

Expected Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned.

Actual Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned.

Test Status: Success.

ID 8 - Code Boundary Testing

Date(s) of Execution: November 16, 2022.

Staff conducting tests: Lucy, Alberto, and Aqsa.

Expected Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned. Excess items are picked up and no more items exceed number of available slots.

Actual Results: Item present placed on map, picked up by player, disappears from map, appears in knapsack successfully, and correct weight assigned. Excess items are picked up and no more items exceed number of available slots.

Test Status: Success.

11 Regression Testing

Repeated tests involved testing the item pick up code whenever the inventory code was tested since we had to make sure that the items in the inventory matched what the player had and that involved having the player gain access to items.

Also, the monster's path code was tightly tested with the player's health code as whenever the player had to get damaged, we also had to make sure that the player was moving.

Lastly, the player movement code was always tested whenever we played the game as we had to move to certain positions in the game to test other sections of the game.

IV Inspection

12 Items to be Inspected

Lucy's code that should be inspected is her inventory UI code.

Aqsa's code that should be inspected should be her scene transition code.

Alberto's code that should be inspected is his knapsack implementation.

13 Inspection Procedures

The following procedures were followed when going through with the inspection:

- Play game as normal to see if the game has any bugs with the new feature(s)
- Test the boundaries of inputs while playing the game
- Create test cases with different inputs for each function's inputs / varying values

14 Inspection Results

Lucy's code inspection: Aqsa and Alberto followed the instructions listed above, trying out different varying values, and inputs. These inspections were performed on

an estimated date of the 15th of November. No bugs were found during inspection.

Alberto's code inspection: Aqsa and Lucy tested alberts code, picking up a varying amount of items. No bugs were found during inspection.

Aqsa's code inspection: Alberto and Lucy inspected Aqsas code, moving between different scenes from different points in the map at different time frames in the game. No bugs were found during inspection.

V Recommendations and Conclusions

As of the current date of inspection, all of the code that has been tested has passed their corresponding tests. In regards to future tests, previous tests will need to be run periodically to ensure that no tests were harmed while creating new code in the application. If any new functionality is added to the previous sections, new tests must be added to account for these changes.

VI Project Issues

15 Open Issues

Due to the intensive nature of 3D graphics, there is no guarantee that the build will run on lower tier systems, and users may experience lag while playing the game on some systems. This issue was found while testing the build on different laptops, where some laptops would refuse to load the game due, or would have a low framerate due to the graphics being too intensive.

Large-scale testing has not been done on different tier systems and different operating systems so full compatibility cannot be guaranteed yet.

While using version control, the developers experienced issues with merging their results due to set up errors. The developers were not able to find the solution to this without brute forcing their changes into Plastic SCM. The error will need to be resolved to be able to continue development quickly in the future.

16 Waiting Room

Due to the size of the project, there are a vast amount of features that still need to be implemented. Features are grouped by pending further release number, and are listed by priority. The features are listed below:

Pending Release #1:

- Backend code to allow for merging items needs to be implemented. Additions to the item structures should be made to specify what types of

items become different items after crafting.
- The inventory interface needs to allow for item crafting to commence using the crafting code mentioned above.
- Mockup user dialog trees for all dialog interactions in game

Pending Release #2:

- Program dialog system
- Program NPC interaction buttons
- Apply existing monster NPC movement code to any moving NPCS (ghost, tutorial policemen, etc.)
- Program dropping and picking up items from the player's inventory
- Allow for saving and loading the player's game state when the user loads and exits the game.
- Implement audio hints system.
- Create an 'item inspection' system.

Pending Release #3:

- The features in the previous releases should be demonstrated in the user tutorial room.
- Use the 'item inspection' system to allow the player to inspect any relevant items for the plot.
- Use the dialog and pathfinding systems entailed in the previous releases to implement the story gameplay.
- Implement audio hints from the system described in the previous release.
- Program new maps applying the mentioned features above.
- Build new maps for each level (including end of game scenes)
- Use a dialog system to program the 'end of game' scene when the user completes the game.
- Implement 'go home' feature where the user decides to not enter the forest at the beginning of the game, ending the game early.
- Tie up any possible loose ends that were not implemented in the previous releases.

17 Ideas for Solutions

It is recommended that future developers stick to using the same tool sets as the previous developers, those tool sets being the Unity game engine, using Visual Studio Code or Visual Studio for text editing (as that is what Unity mainly supports and both are feature rich.), and Plastic SCM, assuming these toolsets are still up-to-date and relevant when the project is being further implemented. Using these tools should give the developers a smooth workflow, reducing error usage and reducing time spent on this project compared to other workflows which may take longer and not help the

developer as much.

If the future developers decide to stray away from Unity, it is recommended that the programmers use an already existing engine that is well documented with robust features and a plethora of free assets to avoid having to 'reinvent the wheel' for features that are already readily available.

Programatically, the code should use an Entity Component System design, where each class is represented as a part of an object as recommended by Unity. Singletons are encouraged when there is, and will only be one instance of an object in the game. These programming styles will help keep the game consistent, and modular throughout its lifetime.

18 Project Retrospective

Reflecting on the project, there were a lot of moments where we had issues, and a lot of moments where we played to our strengths.

For our strengths, we all communicated fairly well and got along nicely. We always met up punctually, and made sure we put in an equal amount of effort. There were a lot of issues that we encountered within development, but we made sure to stay positive, and tackled the issues head on as a team, communicating and problem solving together. In the future, those who take on the projects should communicate extensively with their time likewise to allow for effective problem solving and teamwork.

Given our small amount of experience with Unity, we quickly learned how to use the tool. Also, we ended up taking a creative approach to the project, making it our own regardless of the small amount of information that we were given from the previous project.

As for what could be improved, the project has some aspects of the Traveling Salesman problem, but due to its unfinished state, it does not fully meet the requirements for Traveling Salesman. If more planning was put into place with specific times, we could have gotten more done, but because we lacked specific dates, it was easy to push the weekly assignments towards the end of the week, making it more difficult to get the tasks done. We could have eliminated this issue by putting deadlines in Jira instead of keeping the tasks open-ended and vague. The game description does meet the requirements for the traveling salesman problem, but at the current state, it will need more work.

VII Glossary

NPC: A character within a game that the user cannot control. Short for non-playable characters. Non-Playable-Characters are characters controlled by the computer instead of another user.

Unity: A game development application that the developers used to create the game in.

Plastic SCM: A program which syncs the people's progress on a video game that they created across their different devices.

UI: Short for user interface. The text, boxes, and buttons that are shown on screen that the user uses to interact with the program.

Lower/High/Mid Tier Systems: Another word for slow, average, or fast computers.

Framerate: How many individual frames from the video game show on someone's computer displays per second.

New game: The menu option a player can select to start a new version of the game.

Load game: The menu option a player can select to load one of the three saved slots in the game.

Save game: The menu option a player can select to save their current game into one of the three game slots.

Exit game: The menu option a player can select to end and shut down the game application.

Police station: This is the tutorial stage of the game. The player is introduced to the basic mechanisms of the gameplay in this stage level.

Merge/Combine: This is a player item option to combine two items in their inventory to create a completely new item. This may make the previous version of the item inaccessible.

Equip: This is an inventory option a player has that enables them to hold the item in their knapsack. For example, a player has the option to take a flashlight out of their knapsack and use it on the level by having the main character hold it. Other items can be held as well, with a limit of 1 item being equipped per hand, but may not have any visual effects like the flashlight which can be turned on or off.

Dialogue: This is the story narration of the game. Can consist of the player thinking to themselves or interacting with an NPC.

Kitchen: This is a room in the tutorial level where the player learns about combining items. In the kitchen of the police station the player is prompted to pick up a cup and milk from the fridge, select the cup and put it in the coffee machine to obtain a cup of coffee. From there the player is prompted to combine the cup of coffee with milk and successfully combine the items to obtain a latte.

Office: This is a room in the police station where the main character initiates a

dialogue with their boss, the chief of police, about a case file. They receive the case file here and are prompted to “look/inspect” the item.

Look/inspect: This is an inventory option where players can take a closer look at the items in their inventory. Specifically used to read case files and look at items to figure out puzzle passcodes or hints.

Forest: The main puzzle stage of the game. The level after the tutorial where the player would encounter a pumpkin monster and look for the dead body of the victim.

Forest pathway: This is the maze pathway of the forest where the main gameplay occurs and mainly where the player navigates to find helpful items and tries to avoid monsters.

Item: Usable items in the game that have a weight and that the player can possibly equip, unequip, consume, drop, or pick up.

Monster: The antagonist of the game. A pumpkin monster that lurks around the maze looking for the player.

Victim: This is Mary el Sacrificio who is initially reported missing. The game ends upon finding her dead body.

Player: The main protagonist of a game. In this game in particular, the player is a police officer named Jason. He navigates the forest in search of the missing little girl Mary.

Case file: This is the inventory item that the player can inspect to initiate the task of looking for the missing girl Mary.

Knapsack: The backpack of the game that holds all the items and has a weight carrying limit.

Police car: This is the stage item that lets the player go back to the tutorial stage. Upon interacting with this item they return to the police station.

Dead body: This is the body of Mary el Sacrificio, the victim of the missing person’s case our player is working on. Upon finding her, the game is over and we have an ending narration of what happened after they found her body

VIII References / Bibliography

1. **R. Barrera, C. Dominguez, B. Nguyen, R. Tonkin et al. “Friday The 13th Project Report.” *University of Illinois at Chicago*, Dec. 2019.**
2. **Robertson and Robertson, Mastering the Requirements Process.**
3. **A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth edWiley, 2013.**
4. **J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document**

for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

5. **M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.**
6. **Brackeys. "How to Make Terrain in Unity!" *YouTube*, YouTube, 23 June 2019, https://www.youtube.com/watch?v=MWQv2Bagwgk.**
7. **"Creating the Backend | Unity Inventory System Tutorial - Part 1." *YouTube*, YouTube, 26 Dec. 2021, https://www.youtube.com/watch?v=svoXugGLFwU&ab_channel=DanPos-GameDevTutorials%21.**
8. **"First Person Movement in 10 Minutes - Unity Tutorial." *YouTube*, YouTube, 7 Feb. 2022, https://www.youtube.com/watch?v=f473C43s8nE&t=180s.**
9. **"Flexible Inventory System in Unity with Events and Scriptable Objects." *YouTube*, YouTube, 23 Mar. 2022, https://www.youtube.com/watch?v=geq7lQSBDAE&t=662s&ab_channel=BMo.**
10. **"Flexible Pickup / Collectible System in Unity with Events." *YouTube*, YouTube, 22 Mar. 2022, https://www.youtube.com/watch?v=f75Wcwu33OY&t=643s&ab_channel=BMo.**
11. **GameDevGuide. "Creating an Inventory System in Unity." *YouTube*, YouTube, 30 Nov. 2021, https://www.youtube.com/watch?v=SGz3sbZkfkg&t=394s&ab_channel=GameDevGuide.**
12. **"How to Fix Pink Materials in Unity." *YouTube*, YouTube, 10 May 2021, https://www.youtube.com/watch?v=nB0r0c-SIVg&t=257s.**
13. **"How to Make Beautiful Terrain in Unity 2020 | Beginner Tutorial." *YouTube*, YouTube, 15 July 2020, https://www.youtube.com/watch?v=ddy12WHqt-M.**
14. **jayanamgames. "Unity Moving Platform Tutorial." *YouTube*, YouTube, 18 Apr. 2018, https://www.youtube.com/watch?v=rO19dA2jksk&t=5s.**
15. **KetraGames. "How to Collect Items (Unity Tutorial)." *YouTube*, YouTube, 25 Jan. 2022, https://www.youtube.com/watch?v=EfUCEwKmcjc&t=1s&ab_channel=KetraGames**
16. **"Unity - How to Use a Button to Switch between Scenes." *YouTube*, YouTube, 20 Jan. 2019, https://www.youtube.com/watch?v=PpIkrff7bKU&t=29s.**
17. **"Unity Inventory System - Easy Tutorial (2022)." *YouTube*, YouTube, 6 Jan. 2022, https://www.youtube.com/watch?v=AoD_F1fSFFg&t=362s&ab_channel=SoloGameDev.**
18. **"Unity Scriptable Objects - Easy Tutorial." *YouTube*, YouTube, 19 Dec. 2021, https://www.youtube.com/watch?v=JyU3qPn9_O0&t=109s&ab_channel=SoloGameDev.**
19. **"Unity Spot Light | Spot Light in Unity - Unity Lighting Tutorial 05." *YouTube*, YouTube, 25 Nov. 2018, https://www.youtube.com/watch?v=EVRpUQLEkJk&t=244s.**
20. **"Unity Tutorial - How to Do Multiple Scenes." *YouTube*, YouTube, 17 May 2020, https://www.youtube.com/watch?v=R_ipogbSv5s.**

IX Index