College of Engineering

Ms. Daphne

Mr. Dexter

Mr. Drake

Ms. Lauren

Mr. Travis

CS4390 I Senior Capstone Project

Robot and I: Learning with Gaming

Supervised by:

Dr. Mohammad Faridul H. Siddiqui. Ph.D.

Fall 2022

Department of Computer Science

Abstract

The goal for this software is to introduce fundamental aspects of programming. Starting out with simple concepts, such as assignment statements and if statements. As the levels progress, the game goes deeper into programming concepts and specific languages. The game runs on the Unity game engine, an industry standard that helps build games of this sort. During the levels, the user will play a two-dimensional game as a programmer who controls an in-game robot that has to accomplish certain tasks. The user controls the robot by finishing in-game code fragments. The first phase of the project focuses on creating a rough prototype of the game. Next, the project design and implementation of levels the user will see, and create one or two beginning levels of the game using pseudo code. The second phase of our project will consist of creating seven or more levels that are either gradually harder than the first or use the programming languages Python and C# to implement what was taught in pseudo code.

Plagiarism Declaration

With the exception of any statement to the contrary, all the material presented in this report is the result of my own efforts. In addition, no parts of this report are copied from other sources. I understand that any evidence of plagiarism and/or the use of unacknowledged third party materials will be dealt with as a serious matter.

Signed 

Acknowledgements

We would like to express our gratitude and apparition to everyone that helped guide us in writing this report. To all of our professors that helped us throughout the last few years and to all of the people that made tutorials for us to watch when we were lost or confused. A special thanks must go to Dr. Mohammad Faridul H. Siddiqui whose continuous guidance, resources, and suggestions, helped us build this project and report. We would also like to acknowledge our fellow teammates. None of us could have done this without each others continuous support.

Table 1: Revision History

Name	Date	Reason for Change	Version
All	9/13	Initial revision	1.0
All	9/14	Small edits	1.1
Drake, Lauren, Travis	09/15	Abstract, Section 3, Section 4	1.1.1
Drake	09/16 12am	Section 4, cont. (charts)	1.1.2
Drake, Travis, Dexter, Lauren	09/16 10am	Meetup: sect 1-3 review	1.2
Drake	09/17 12am	Section 4, more charts	1.2.1
Drake, Travis, Daphne, Lauren	09/17 10am	Meetup: sect 4-5 review	1.3
Dexter and Daphne	09/18 11pm	English, Grammar, edits to LaTeX	1.3.1
Daphne	09/19 9am	English and Grammar	1.3.2
Daphne and Dexter	09/19 3pm	English and Grammar	1.3.3
Daphne and Dexter	09/20 1pm	English and Grammar	1.3.4
All	09/20 4pm	Team Review	1.3.5
Daphne	9/20 10pm	Move everything to LaTeX file	1.4
Daphne	10/9 3pm	Update LaTeX file	2.0
Daphne	10/17 3pm	Updated Sprint backlog	2.1
Daphne	10/31 3pm	Updated Sprint backlog	2.2
All	11/11 12pm	Updates to entire Document	2.3
Daphne	11/14 3pm	Updated Sprint backlog	2.4
Daphne	11/21 11am	Added in all test cases	2.5
Dexter	02/14 5pm	Made phase 2 changes	3.0
Daphne	02/28 1pm	changes to scrum report	3.1
Daphne	03/14 1pm	changes to scrum report	3.2
Daphne	03/28 1pm	changes to scrum report	3.3
Daphne	04/25 1pm	changes to scrum report	3.4
Daphne	04/26 5pm	Adding in test cases	3.5
All	05/8 1pm	combining the phases	3.6
Daphne	05/9 3pm	adding in user stories and last minute things	4.0

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to give a detailed specification report of the Unity game; Robot and I. This paper will describe the purpose, the features of the game, what the game will be used for, the interfaces and the constraints of the software being built. The end goal for the game will be to help others learn programming through gaming.

1.2 Document Conventions

This document was based on the template for IEEE SRS and the capstone project report template. The file specifications such as font size, headers and footer, ect. are pre-defined from the capstone project report template.

1.3 Intended Audience and Reading Suggestions

The intended audience for this SRS is individuals who are new to computer programming that want to learn the basic concepts and other computer science students that want to play a game with programming. This document will explain the features and concepts used to make the game in Unity, specifically how the player interacts with the game. For those unfamiliar with SRS reports, the standard way of reading is a top-down approach. Developers and those interested in the specifics of the game should focus on sections three and four. Section two gives a standard overview for those who do not wish to read the entire document, and section five

covers other required information that is not directly pertinent to the game.

1.4 Project Scope

The goal of this software is to teach users with little to no programming skills the concepts of computer science and put those concepts to use in a fun and exciting way. While the game is set up for newer programmers, those with experience will also have a fun and challenging time as the levels increase in difficulty. Our goal is that the end result from the user's experience is, they will end the game with an increased knowledge of programming concepts or reinforce what they already know. A more detailed explanation on how this software will accomplish this goal is outlined in the abstract.

1.5 References

This document references the Abstract.

Chapter 2

Overall Description

2.1 Product Perspective

Robot and I is a new self-contained software project that is intended for anyone wanting to learn programming in a fun environment. Robot and I was decided upon by a group of five students who wanted to challenge themselves in developing a game that would help teach others the basics and advanced concepts of programming.

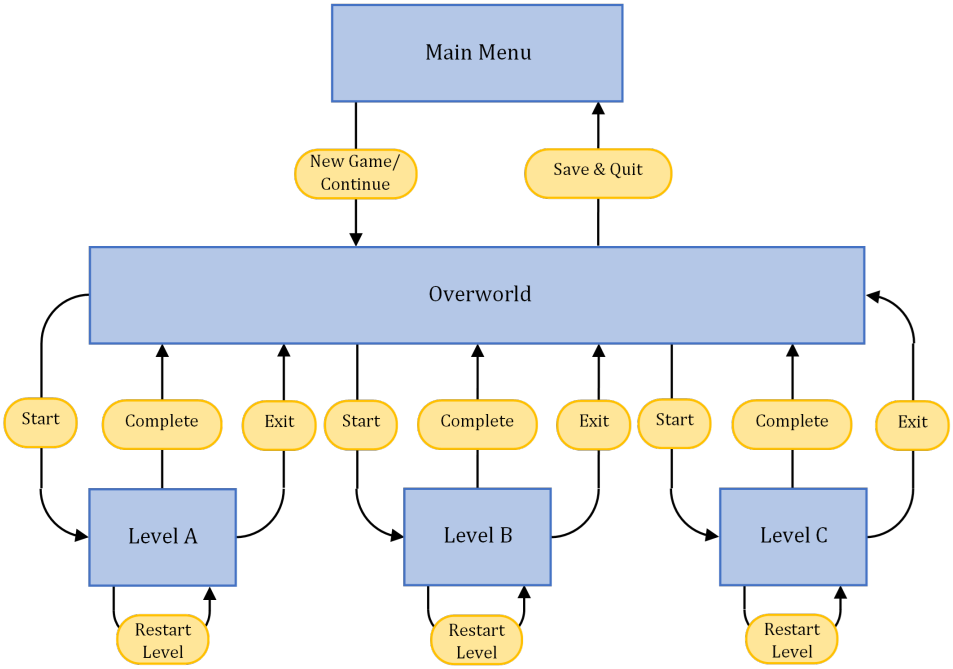

Figure 2.1: Early Graph of Project Scope.

2.2 Product Features

Some of the product features will include player movement in a top-down player view (island). The player will communicate with NPC characters who will guide the player to the different start areas for levels. Upon entering a level, the player will discover the view changes to a 2D plat-former perspective. The levels will be composed of blocks, trees, houses, etc. In the beginning levels, the player will answer fill-in-the-blank code fragments, and the player will need to fill in the correct response in the boxes. Once the responses are correct, the player will gain a gadget which will allow bit to complete the level in application. The difficulty of the levels will gradually increase, and in the later levels the player will be writing full code segments that must run and perform the correct result in order to finish the level. When the user completes a certain number of levels, boats will appear that will take the user to another island. There are two other islands with new and more challenging levels. There are also mini-maps and notes the user can look at/use to help the user along their journey.

2.3 User Classes and Characteristics

This game was made for beginner programmers looking to learn and practice rudimentary topics of programming. These programmers will begin their learning journey by completing levels in pseudo code and gradually moving upwards to actual programming languages, such as Python and C#. Also, experienced programmers will have an enjoyable time completing the levels, and it may help reinforce some of their knowledge in computer programming.

2.4 Operating Environment

This software will operate on the user's computer as an executable file. Upon running the software, a standard .json file will be created on the user's computer in the Temp directory. This software will use the .json file to load and save game data as the user progresses in the game. All other components of the game are handled within the installation directory, and this software will not access files outside of the directory and the .json file.

2.5 Design and Implementation Constraints

The design and implementation of this software is limited to the tools available in the Unity package manager and tools acquired from the Nuget package manager. These tools include IronPython3 and Microsoft's C# Codedom Compiler. The hardware and software limitations are those that are restricted to the running of Unity software and the version of its creation. At the time of this writing, there are no other constraints on the design and implementation of the software. There are currently no security requirements for our game design and implementation.

2.6 User Documentation

For this project, there will be a user manual and tutorials available on the game itself, located inside the options menu called user manual. These documents will provide the user with a detailed description of how to navigate around and how to use the game. The tutorials will also contribute to the story line of the game in order to feel more natural.

2.7 Assumptions and Dependencies

This software assumes Unity will be able to work under the weight of this software and not crash unexpectedly. This software is also dependent on the user not inducing a security failure by writing malicious code or by performing some unintended action while completing the tasks given in the level games.

Chapter 3

System Features

3.1 Main Menu

3.1.1 Description and Priority

The main menu is a high priority (2/9 risk). This will be one of the first screens that the user sees. The main menu will have four buttons that allow the user to either start a new game, continue a previously saved game, go to the options that will allow the user to change the audio settings, or quit the game.

3.1.2 Stimulus/Response Sequences

If the user clicks the "New Game" button, it will reset the data and connect them to the tutorial level. If the user clicks on the "Continue" button, it will connect them to their current level. If the user clicks on the "Options" button, it will bring them to the options menu. If they click the "Quit" button, the game will close.

3.1.3 Functional Requirements

The user needs to be able to run Unity games and have a display screen and mouse to change the options. If the main menu crashes, Unity should give an error message.

REQ-1: Design the Main Menu

REQ-2: Add 4 buttons for New Game, Continue, Options, and Quit

REQ-3: Attach actions to each button

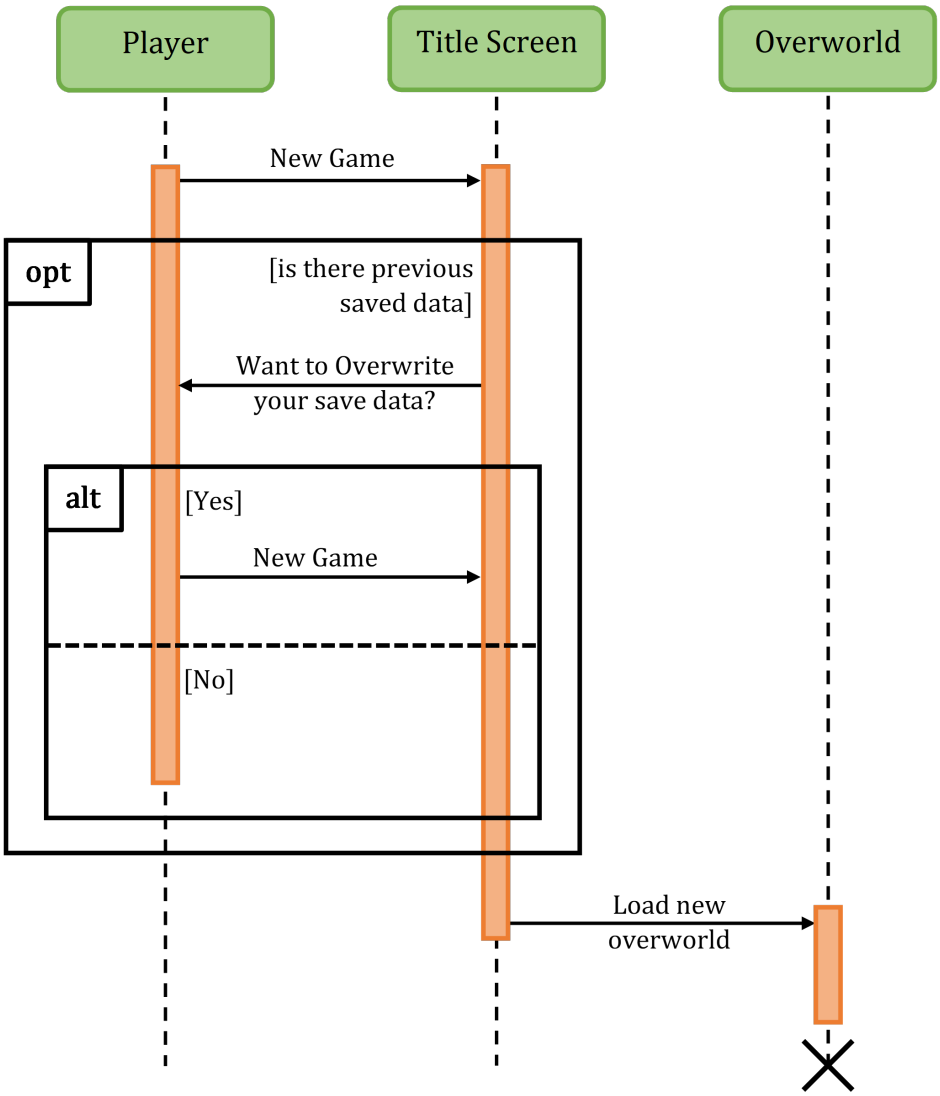

Figure 3.1: New Game Sequence Diagram

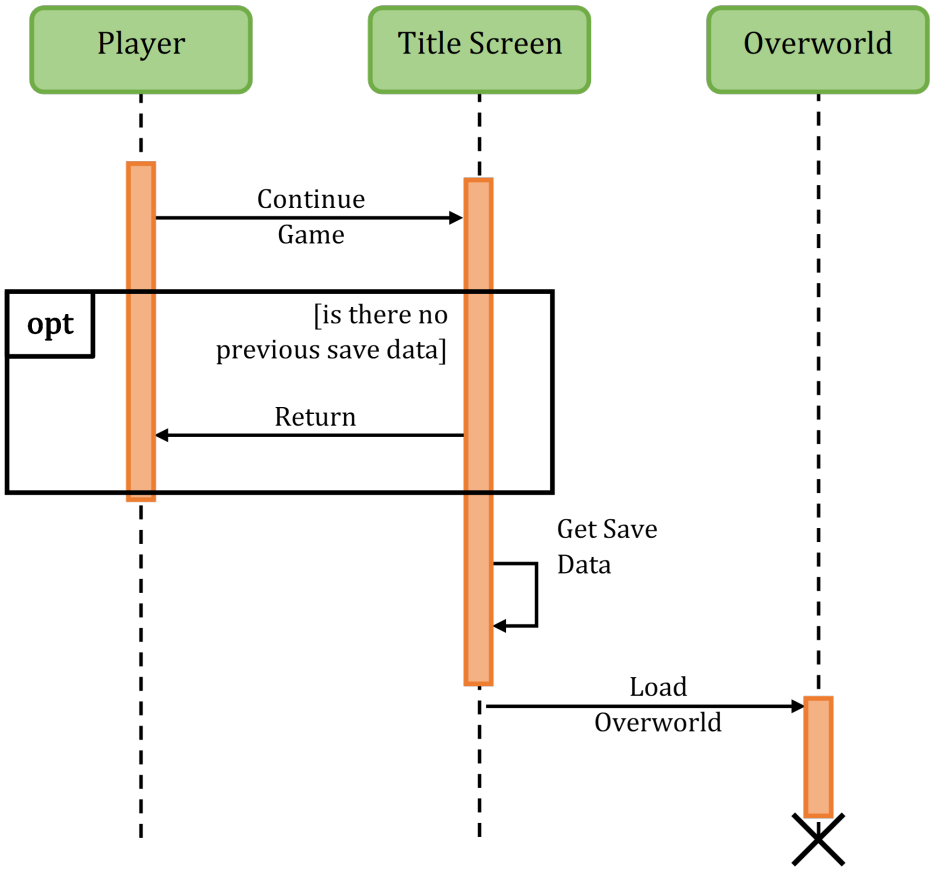

Figure 3.2: Continue Game Sequence Diagram

3.2 Options Menu

3.2.1 Description and Priority

The options menu is a medium priority (1/9 risk). This screen will be linked to the main menu screen. This menu will allow the user to change the audio setting such as overall volume, sound effects and background music for the map and levels. There will be a back button that goes back to the main menu.

3.2.2 Stimulus/Response Sequences

If the user clicks on the “Back” button, it will bring them to the main menu. If the user moves the “Master Volume” slider under the “Master volume” text to the right/left, it will increase/decrease the volume respectively. If the slider goes as far left as it can, it will turn off the background music, and there will be a check box to mute the volume. The ”Master Volume” sliders controls the volume for the other sliders. The ”Music Volume” and ”SFX Volume” sliders

act in the same way.

3.2.3 Functional Requirements

The user needs an audio device to be able to hear the music and a mouse to utilize volume sliders. If the option screen crashes, Unity should give a crash message.

REQ-1: Create a button and link it to a separate screen

REQ-2: Design the Options Menu

REQ-3: Add a volume slider attached to the overall volume of the game called "Master Volume"

REQ-4: Add a volume slider attached to the music volume of the game called "Music Volume"

REQ-5: Add a volume slider attached to the overall sound effects of the game called "SFX Volume"

REQ-6: check box for "Master Volume"

REQ-7: check box for "Music Volume"

REQ-8: check box for "SFX Volume"

REQ-9: Add back button that is linked to the main menu

REQ-10: Add User Manual button that is linked to PDF the will open in users default browser.

REQ-11: Create a menu section graphic

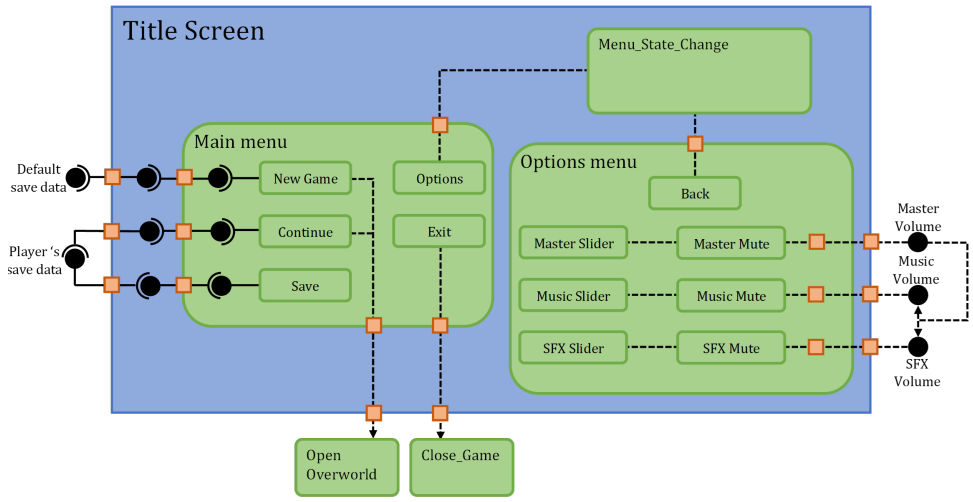

Figure 3.3: Title Screen Component Diagram

3.3 Over World Map

3.3.1 Description and Priority

The map is a high priority (5/9 risk). The map is what the user will see when they travel the world looking for NPCs to help. The map will have three islands, Pseudo City (Pseudo Code), Python Jungle (Python), and The Sharp Seas (C#). There will be a menu button at the top of the screen that will lead to the menu section. There is also a progress bar at the top that shows how many levels have been completed.

3.3.2 Stimulus/Response Sequences

The user will be able to move around the map as they please. There will be a back button that when pressed will allow the user to go back to the main menu.

3.3.3 Functional Requirements

The user will need to have access to a computer screen, mouse and keyboard to move around. If the map screen crashes, Unity should give a crash message.

REQ-1: Design the maps

REQ-2: Implement design

REQ-3: Add characters

REQ-4: Add the different maps and the names

REQ-5: Create each level in the associated map

REQ-6: Create menu button that is linked to menu section

REQ-7: Create Progress Bar

3.4 Map

3.4.1 Description and Priority

The map is a medium priority (3/9 risk). The map is available to the user through the use of the ”m” key on the user’s keyboard. The map will open up and display the current state of

that islands completion status. There will be a separate map for each island, and the map is exited through another press of the "m" key.

3.4.2 Stimulus/Response Sequences

The user will be able to look around the map as they please. All current available, unavailable and completed levels will be visible to the player while they look around.

3.4.3 Functional Requirements

The user will need to have access to a computer screen, mouse and keyboard to look around. If the map screen crashes, Unity should give a crash message.

REQ-1: Design the maps

REQ-2: Implement design

REQ-3: Add characters

REQ-4: Add the different maps and the names

REQ-5: Create each level in the associated map

REQ-6: Create icons and colors for each character

3.5 Levels

3.5.1 Description and Priority

The levels are a high priority (6/9 risk). The levels are the main focus of the game and will be where the user will complete tasks and learn programming concepts. A story will play out that explains the task at hand with dialogue that will use a start lesson button, continue button and back button to explain the problem. Then the user will be presented with the problem they have to solve. Once the problem has been solved, the task will then be completed on the screen and the user will be brought back to the island. During the levels, there will be an options menu button to handle audio settings. Another button we will add is a Notebook button for users to write notes in. Also, there will be a reset button if the user want to restart the level.

3.5.2 Stimulus/Response Sequences

The user will be able to move around the level as they please. There will be a back button that when pressed will allow the user to go back to the island screen. During the levels, the user will be presented with a problem that will be solved either by multiple choice question or a free keyboard. If the user answers incorrectly, what he has chosen or typed out can now be changed. If the user answers correctly, the task at hand will be completed on the screen. There will be an options button that connects to the options menu for audio settings, and a notes button that will connect to the notebook.

3.5.3 Functional Requirements

The user will need to have access to a computer screen, mouse and keyboard to move around and interact with the prompt. If the level crashes, Unity should give a crash message.

REQ-1: Decide on what programming topic the level will be over and what task the user will have to do

REQ-2: Design the level

REQ-3: Implement design

REQ-4: Add character

REQ-5: Add dialogue

REQ-6: Add start lesson button

REQ-7: Add continue button

REQ-8: Add back button for dialogue

REQ-9: Provide the problem for the user to solve

REQ-10: Once the task is complete, send the user back to the map

REQ-11: Add a back button that is linked to the map screen

REQ-12: Add a notes button that is linked to the notebook for the level

REQ-13: Add reset button

REQ-14: Add an options button that is linked to the options menu in level

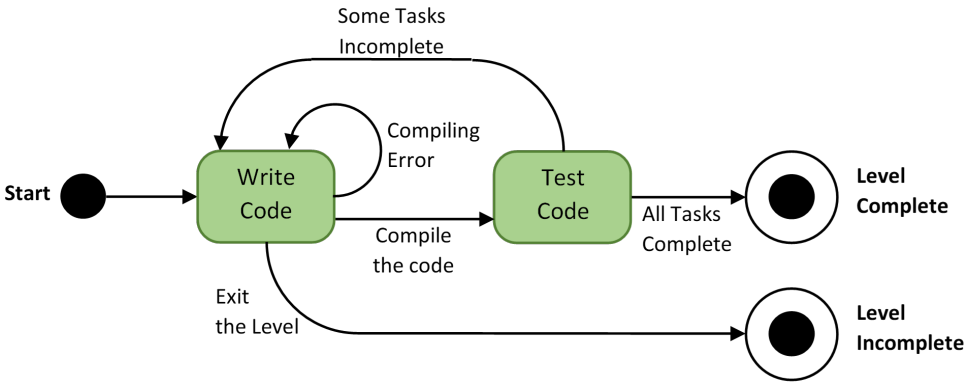

Figure 3.4: Diagram for Level State Machine

3.6 Options Menu in Level

3.6.1 Description and Priority

The options menu in level is a low priority (2/9 risk). This menu will have a ”save” button, ”options” button, ”back” button, and ”exit to world” button.

3.6.2 Stimulus/Response Sequences

If the user clicks on the “Back” button, it will bring them to the level. If the user presses the ”Save” button it will save the game. If the User presses the ”Options” button it will take them to the options menu mentioned in section 3.2. If the user presses the ”Exit to world” button it will take them to the overworld island.

3.6.3 Functional Requirements

The user needs an audio device to be able to hear the music. The user will also need to have access to a mouse to press buttons and move sliders. If the option screen crashes, Unity should give a crash message.

REQ-1: Create a button in the level and link it to a separate screen

REQ-2: Design the Options Menu

REQ-3: Add ”save” button

REQ-4: Add ”options” button

REQ-5: Add ”back” button

REQ-6: Add ”exit to world” button

REQ-7: Create back button that is linked to the level

REQ-8: Create a restart level button that will reset the level with a clean state

3.7 Options Menu in Map

3.7.1 Description and Priority

The options menu in map is a low priority (2/9 risk). In the menu there is a ”save” button, ”options” button, ”notebook” button, ”back” button, and ”exit to title” button.

3.7.2 Stimulus/Response Sequences

If the user clicks on the “Back” button, it will bring them to the island. If the user presses the ”save” button it will save the game. If the user presses the ”options” button it will take them to the options menu discussed in section 3.2. If the user presses the ”notebook” button it will bring up the notebook. If the user presses the ”exit to title” button it will take them to the main menu.

3.7.3 Functional Requirements

The user needs an audio device to be able to hear the music. If the option screen crashes, Unity should give a crash message.

REQ-1: Create a button on the map screen and link it to a separate screen

REQ-2: Design the Options Menu

REQ-3: Add ”save” button

REQ-4: Add ”options” button

REQ-5: Add ”notebook” button

REQ-6: Add ”exit to title” button

REQ-7: Create a back button to get out of the option menu.

REQ-8: Create a menu section graphic

3.8 Dialogue

3.8.1 Description and Priority

The dialogue is a high priority (1/9 risk). This will be on the screen of the map and the levels. The in level dialogue will describe what problem the user will be solving and the task at hand. The NPCs will have dialog in the overworld that prompts the player what direction to go in.

3.8.2 Stimulus/Response Sequences

When the user starts a level, there will be dialogue to go along with the level on the screen. In the overworld, when the user interacts with an NPC, their dialogues will appear. There will be a button to continue the dialog.

3.8.3 Functional Requirements

The user will need to have a display screen and mouse.

REQ-1: Dialog boxes in level

REQ-2: Dialog boxes in overworld

REQ-3: Progress/skip buttons, click screen to continue button to skip lesson

3.9 Notebook

3.9.1 Description and Priority

The Note Book button is a low priority (2/9 risk). This will be a button in the levels of the game that the users can write notes of what the learned. Along with the options menu discussed in 3.7.

3.9.2 Stimulus/Response Sequences

If the user clicks on the “Notes” button, a text box will pop up where they can write into. The data will be saved throughout levels so the user can look back at their notes at any time.

3.9.3 Functional Requirements

The user will need access to a mouse to press the button. If the Notes button on the screen crashes, Unity should give a crash message.

REQ-1: Design and implement a button on the level for a Notes

REQ-2: Link it to a pop up text box

REQ-3: Creating Notes for each level to go with the task

3.10 Music

3.10.1 Description and Priority

The music is a medium priority (1/9 risk). Music will play in the background of levels and the map. There will be different music for the levels and for each map that the user goes to. The music will be original and not distracting to the player.

3.10.2 Stimulus/Response Sequences

The only actions required by the user is to be in the game. The user also has the ability to change the volume for the music.

3.10.3 Functional Requirements

The user needs an audio device to be able to hear the music.

REQ-1: Compose music

REQ-2: Record the music

REQ-3: Attach the music to the map and levels

3.11 Sound Effect

3.11.1 Description and Priority

The sound effects are low priority (1/9 risk). Sound effects are the sounds that play when the users get a level correct or incorrect.

3.11.2 Stimulus/Response Sequences

When the user gets the level correct the "correct" sound effect will play. When the user gets the level incorrect the "incorrect" sound effect will play.

3.11.3 Functional Requirements

The user needs an audio device to be able to hear the music.

REQ-1: Write SFX

REQ-2: Record the SFX

REQ-3: Attach the SFX to the objects

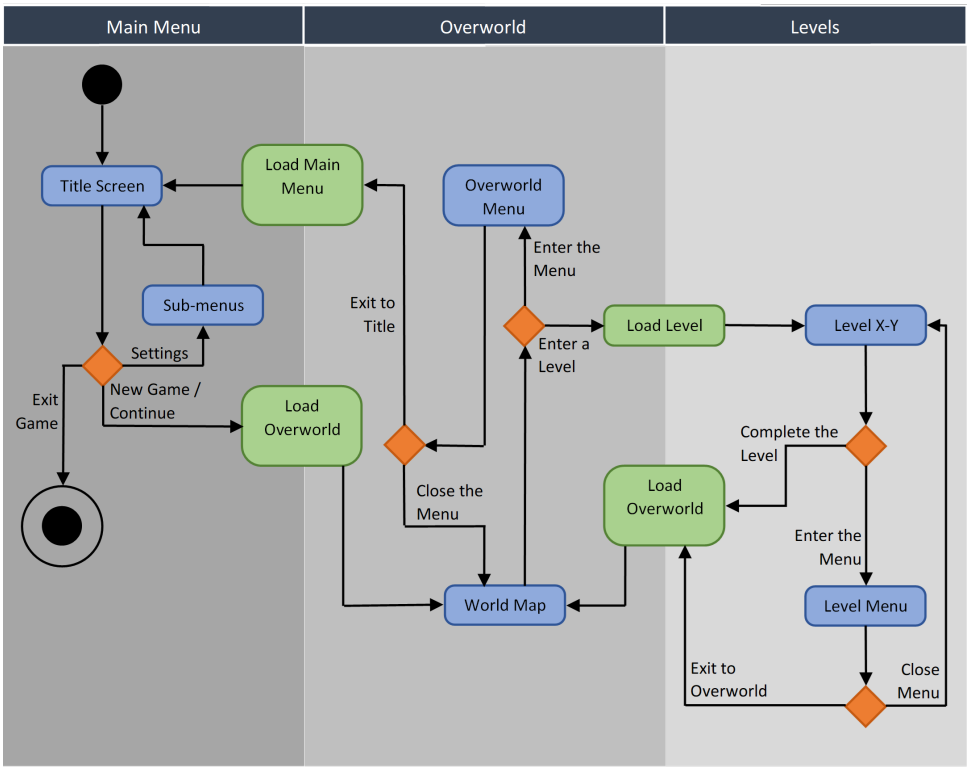

Figure 3.5: Diagram of Project Activity

Chapter 4

External Interface Requirements

4.1 User Interfaces

A video game that lacks in it's user interfaces, lacks in either its video or gaming aspects; as such UI will be crucial in the development of our application. Identifying how our users will relay information to and from the application will propel our project to success.

4.1.1 Output Interfaces

GUI

Our application will be constructed of five types of scenes: menus, tutorial level, overworld, maps, and levels. The main menu will contain buttons that the user can interact with to activate certain functions, some will open sub menus. A mock-up for our main menu is shown in [Figure 4.1].

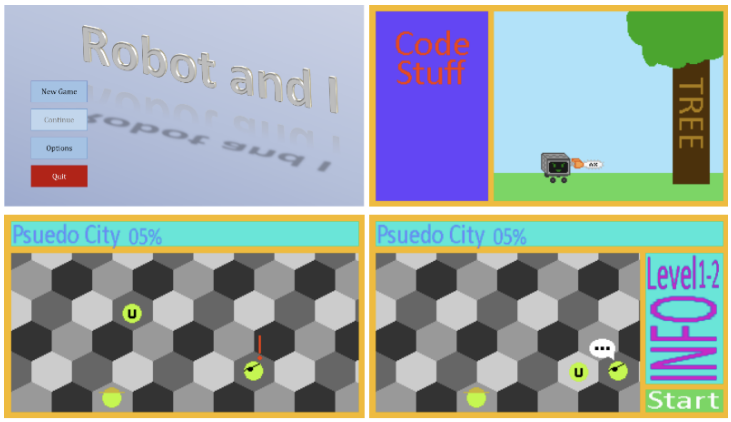

Figure 4.1: Output interfaces example.

Audio

Using audio as feedback but not relying on it.

4.1.2 Input Interfaces

Mouse Controls

The user will use the mouse to point and click, and to click and drag items.

Keyboard Controls

The keyboard will be used for enter data along with controlling the characters movements in the overworld.

Monitor Controls

The game will be in landscape view.

4.2 Hardware Interfaces

The software will require access and communication to the: Monitor, Keyboard and Mouse. The logical connection between these allows the user to input data into the game through keyboard and/or mouse input as well as seeing the results of this input through the monitor. It's recommended to have an audio device but it is not required.

4.3 Software Interfaces

This software will communicate with Unity and its libraries to perform and execute all functions of the game. This software will also connect to and read/write to a .json file that serves as the software's data storage. Finally, this software will access the user's default browser when they click the manual button.

4.4 Communications Interfaces

This software will communicate to Microsoft's C# compiler and Mono's official executable for running the C# compiler. These executables should be found inside the directory of this

software, else an error will display. Other communication will be to the user manual provided in-game to the user.

Chapter 5

Other Nonfunctional Requirements

5.1 Performance Requirements

To be able to run Robot and I, the requirements would be a newer operating system no older than Windows 10, but not yet tested for Windows 11. The user will have access to the software through an executable file on the user's machine. The software will be completely internet independent and will have no need for internet connection. Unless the user clicks the User manual button that opens to default browser.

5.2 Safety Requirements

We will be implementing auto-saving throughout the game. In the case of a catastrophic event, the user can load up an emergency backup that was created so that the user's progress will not be destroyed.

5.3 Security Requirements

The executable form of this game will not store any form of personal information, and does not require internet access. This software will only access the .json file found in the Temp folder of the user's computer and any files found in the installation directory of the software.

5.4 Software Quality Attributes

The game will be available via an executable through acquired means. Due to this, once the executable is created or the file is uploaded, changes to the game can not be made until another version is posted. The game will contain as many levels that can be produced in the given time, and the game will be able to execute only on the proper answer. The game will be simple enough that individuals who are not tech savvy can use and operate the game.

Chapter 6

User Stories Documentation

6.1 Epic 1 - Animations

6.1.1 Overworld Player

This sprint, we focused on implementing animation for our overworld character, Bit. Bit is fairly simple to control in this mode, with only four distinct actions, two of which require animation: idle ("standing") and walking. We started with the more complex walking animation, which involved making Bit hop around. Originally, we had planned for Bit to move primarily via propeller or wheels, but we scrapped that idea earlier on. Through our research on animation, we learned about the importance of avoiding stilted or robotic movements. To achieve a more natural look, we utilized a technique called "squash and stretch," which involves drawing the character as if they were a liquid, almost like a bouncing ball. For each of Bit's walking animations (one for each direction), we used ten frames of animation, with only five unique frames. The sequence is as follows: 1) neutral, 2) squash in anticipation, 3) stretch on release, 4) neutral frame but ascended, and 5) peak of jump. The order of these frames is 1-2-3-4-5-5-4-3-2-1, and each sequence repeats as long as Bit moves in that direction. For our idle animations, we keep Bit facing the same direction as before and give them a bouncy look by cycling frames 1 and 2 in that direction.

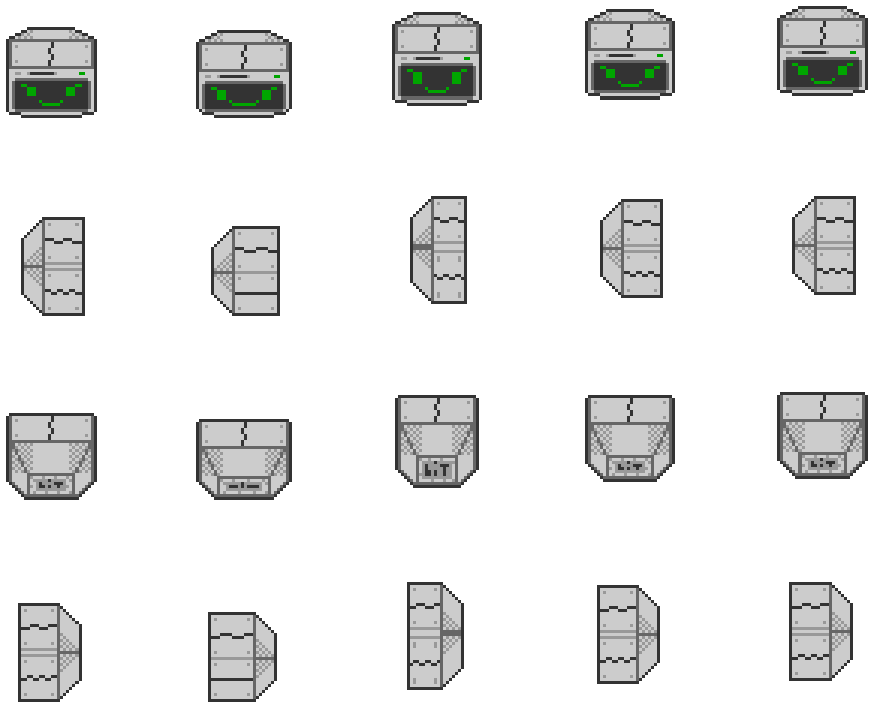

Figure 6.1: Bit Sprites for Overworld Animation

6.1.2 Overworld Misc.

As a user, I want the dialogue box in the overworld to be animated so that it is hidden until I run into an NPC. The animation should consist of two parts: open and close. When the NPC is hit, a boolean variable "isOpen" should be set to true and the open animation should bring the dialogue box onto the screen. When I hit the back button, the "isOpen" variable should be set to false and the close animation should bring the dialogue box below the screen.

6.1.3 Level Robot

This sprint, we tackled the most complex animation of the project so far. We began by planning out the required actions and illustrating each frame for those actions. Additionally, we needed to incorporate triggers for each action based on the player's inputs. Our planned actions include not moving (facing left, centered, and right), walking (left and right), jumping (left, centered, and right), ascending in air (left, centered, and right), descending in air (left, centered, and

right), landing (left, centered, and right), pushing into a wall (left and right), hitting the head on the ceiling (left, centered, and right), returning gaze to center (left and right), glitching (idle), smiling (idle), blinking (idle), nodding off (idle), sleeping (idle), screen shut off, and a green screen of death. We explored Unity's features for interactive animation, but many of these added unnecessary complexity for 2D animations. While integrating these features was easy enough for the top-down with eight animations, player controls required a more complex method. Eventually, we settled on a method that involved setting each animation through Bit's movement code.

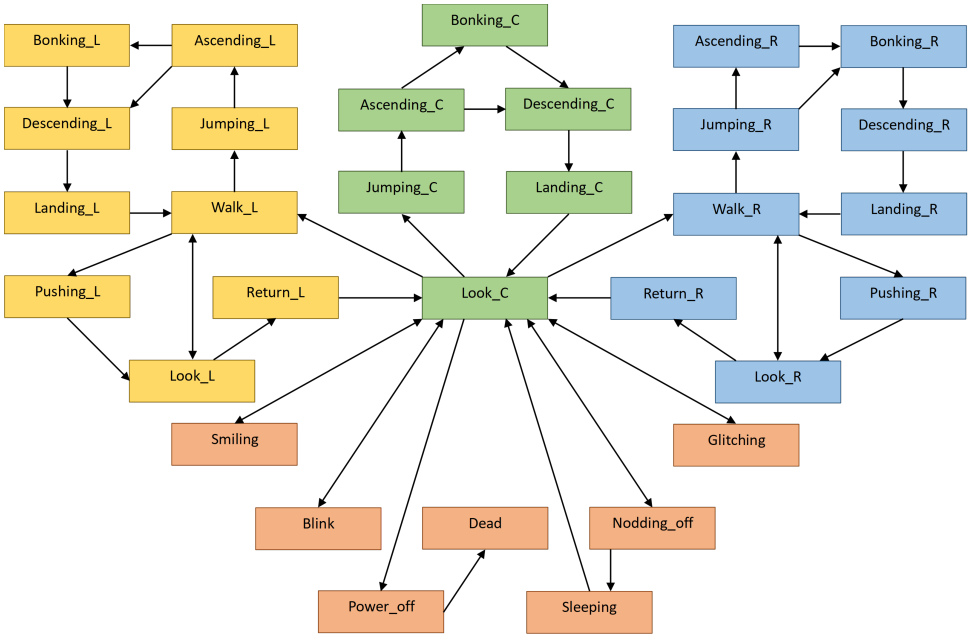

Figure 6.2: Flow Chart for Bit Animation

6.1.4 Level Tools

This sprint, our focus was on creating animations for tools that Bit would use in levels. We used Asprite, a program that allowed us to test animations in the program. Initially, creating the animations for the tools proved challenging and time-consuming, especially when designing the tool's hand. However, once we had the actual tool part complete, the animating process became much simpler. We then created a bendy metal tube for the tool's "arm" which was relatively straightforward.

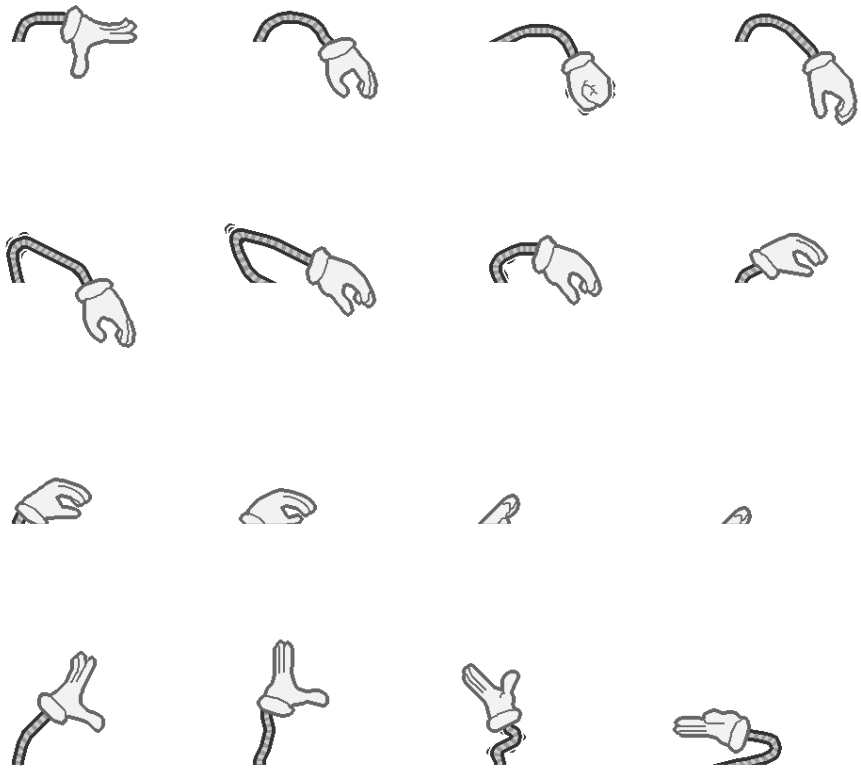

Figure 6.3: Tool Sprites for Animation

6.1.5 Level Objects

This sprint, we focused on creating animations for objects that required them. In general, we found that most of the animations required only one to three simple movements. Many of the animations consisted of static start and stop states with a single play animation or a looping animation.

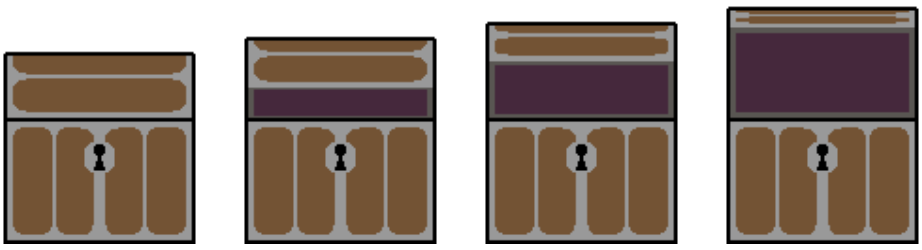

Figure 6.4: Chest Sprites for Animation

6.2 Epic 2 - Audio

6.2.1 Menu Music

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game and created three scripts. The first script, Audio_Manager, accesses the sounds and creates a stop and play music function that will be used in the second script. The second script, Audio_Options_Manager, is used to change the volume with the actions of the user sliding a slider or checking/unchecking the mute box. The second script also sets the player's initial volume to fifty percent. The third script, Sounds, was created to create a public class that allows us to access all aspects of the sounds. The Audio_Manager script will be used in every other audio aspect and will need to be minorly updated throughout. Through Audio_Manager, we were able to store the values the user has set the volume to and keep those values through every scene. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.2 Overworld Music

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include this sound. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.3 Level Music (type 1)

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added

the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include this sound. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.4 Level Music (type 2)

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include this sound. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.5 Level Music (type 1)

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include this sound. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.6 Boss theme

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include this sound. After that, we tested the audio aspect of the game and confirmed it worked as needed.

6.2.7 SFX

For this user story, I started off by playing around with a piano on the app GarageBand on my phone. After I played a tune that I liked, I recorded it and imported it onto a file on my phone that was type m4a. I did this with both sound effects created, one for correct and one for incorrect. I wanted all the sounds in our game to be mp3, so I used the website https://cloudconvert.com/m4a-to-mp3 to convert the m4a file to a mp3 file. From there, I added the mp3 file into the game. I then updated the previously written script, Audio_Manager, to include these sounds.

6.3 Epic 3 - Game Text

6.3.1 NPC Dialogue

This user story involves creating dialogue tasks for the level. There are currently 46 NPC character dialogues that have been written and decided upon. The dialogues will provide the user with an incentive to complete the problem and help the NPC. To progress in the level, the user will have to complete the task assigned by the NPC.

6.3.2 Lesson Dialogue (Part 1)

This story will go over the dialogue of The Duke in the Level. This story will take multiple sprints to finish this story as it is very long and needs to be done well to properly explain programming concepts to the user. This will be part one that goes over the first three programming concepts. We will have multiple people speaking during each level but this story will only go into what The Duke says. A NPC will have a task that needs to be completed by the Robot, Bit, that will be explained by the NPC. One character, whose name is The Duke, will describe the problem the user has to answer.

6.3.3 Lesson Dialogue (Part 2)

This story will go over the dialogue of The Duke in the Level. This story will take multiple sprints to finish this story as it is very long and needs to be done well to properly explain programming concepts to the user. This will be part two that goes over the next three programming concepts. We will have multiple people speaking during each level but this story will only go into what The Duke says. A NPC will have a task that needs to be completed by the Robot, Bit, that will

be explained by the NPC. One character, whose name is The Duke, will describe the problem the user has to answer.

6.3.4 Lesson Dialogue (Part 3)

After having someone read over the dialogue, we were given the feedback that it sounded too much like a textbook. So, for this story I will go over previously written dialogue and add more character to it. We also have more of an idea of what we want the tasks to be for every level. I will be adding more dialogue from The Duke and NPC's referring to those tasks while adding more lesson explanations for some lessons that have not been done.

6.3.5 Lesson Dialogue (Part 4)

For this user story, I will be going over what has already been written to add dialogue transitions from the lessons to the questions. This will add more character to the game and make it sound less like a textbook. I will also be adding dialogue for two more programming concepts and editing the lessons that have been reviewed by other teammates that require a little more explanation.

6.3.6 Lesson Dialogue (Part 5)

For this user story, I will be writing dialogue for one more programming concept. I will also be writing a tutorial for how to use the map of the overworld that will be given by a NPC, Luna. The user will meet Luna the first time they enter the map. Along with writing these dialogues, I will be implementing the dialogue for the levels that are complete into the game and testing them to ensure they are working properly.

6.3.7 Lesson Dialogue (Part 6)

For this user story, I will be writing dialogue for one more programming concept. I will also be writing a tutorial for how to use the ships in the overworld that will be given by a NPC. Along with writing these dialogues, I will be implementing the dialogue for the levels that are complete into the game and testing them to ensure they are working properly.

6.3.8 Lesson Dialogue (Part 7)

For this user story, I will be going over what has already been written to check for consistency in dialogue between the levels. This will ensure that every level has the same look. I will also be adding dialogue for seven more programming concepts. This will complete all the dialogue needed for the game. I will then read over the document and recheck all my work.

6.3.9 Manual / Readme

For this user story, I wrote the user manual for our game. I went through the game starting with the beginning page, the main menu, and articulated every option that the user could possibly make and told them what the outcome would be throughout the game. This allows the user to fully know their way around the game if they chose to read the manual. After finishing, I had the manual proofread by another team member to double check my work. We plan on having a button in the game that allows the user to access the user manual at all times.

6.4 Epic 4 - Game Functions

6.4.1 Sliders in Option Menus

For our sliders we needed to create two textures, one for the bar, and another for the fill. In unity a slider object was default, we just needed to apply our textures, resize and place it in our options prefab. For the mute checkbox we need two other textures, for both the “check” and the box. Just like the slider unity included a basic checkbox we could use. When we include music and sound effects we will have our sliders output floats from zero to one-hundred, and the mute checkboxes will output boolean true/false (true muting, false for sound). For our music and sound effects we will determine an upper threshold for the loudest the sound can output, and our music/SFX sliders will determine the percentage of how loud that sound will be. The master slider will control the max volume the other two slides will reference. The mute checkboxes will override the percentage to zero, despite what the slider is set to.

6.4.2 Graphics Options

This sprint was focused on creating graphics for the Options menus throughout our game. Early on we needed to decide what we needed to have between each of the three different options menus (those being those in each of the different scene types). From our group discussion we decided

we needed access to our user manual, the volume controls, and a close options menu button. The textures were simple to create for the sliders and integrating was easy as a slider was a built in Unity object.The sliders are actually made up of 3 different graphical parts, the bar, the bar's fill, and the sliding nub, though they are currently non-functional as we have no sounds to control yet. The user manual button does nothing for the moment as we have nothing to connect it to yet. The exit button works as it hides the options menu and reveals the main menu.

6.4.3 Save Functionality and Load Functionality

A saving and loading feature for our game is necessary for the player enjoyment of our game. It would be frustrating to complete levels 1-4 and then come back and have to replay the levels again. The game can be saved manually, and is saved every time the player enters a level/moves to another island. The game is also saved automatically every 10 minutes (if the player stays static for 10 minutes). The data that is saved is the player's last scene, position, notebook and the state of completion for the levels.

6.4.4 Maps for Islands

For the maps of the islands I first made a new scene for each area and copied the tile maps from each area to their respective map scene. Next I copied the script camera zoom and edited the code to allow for the camera to zoom further out. I also copied all of the NPCS and ships over to the maps. I made a new object with a black circle and text to replace the NPCs. Then I made a copy of islandNPCs and named it mapNPCs. I changed the code up so that when a level is unlocked the text changes from red with an x symbol to yellow with a circle symbol, and when a level is complete the text changes to blue with a check symbol. The player can not move bit in the maps, instead they use camera panning.

6.4.5 Notebook

For this sprint my primary goal was to develop an in-game notebook. The implementation of the notebook was fairly straight forward. Pull any text from the save file and send it to the notebook. Then upon leaving, save the current notebook text to the save file. The only difficulties were getting graphics for the notebook and where to place it in the levels and overworld. Along with the notebook, I added in some features to the game. Drake developed a "Game Saved" pop up

button. I implemented it into the game so the save button pops up the pop up. I also attached our Manual to the manual button so that when the player clicks the button, the manual pops up. The final feature I added to the game was the progress bar registering completed levels. That way as the player completes the game, they can track their progress using the progress bar.

6.5 Epic 5 - Graphics

6.5.1 Executable Icon

When the team developed the C compiler to be used in our game for dynamic C compilation of code, we also introduced a slight defect to the game. The defect was such that playing the game locally through the Unity editor yielded no issues, but building the game and running the executable caused issues when the dynamic compilation was used. During the winter break, the issue was diagnosed and fixed. Diagnosis: When the code is run to use the dynamic compilation of C, Unity calls the mcs.exe (Microsoft C Sharp Executable), which in turn calls a Mono.exe file. When the game is built however, Unity does not include the mcs.exe or mono.exe files for the code to find. Upon further investigation, it was found that even when the files were included in the build, the actual mono code to find these executables were bugged. Fix: The fix included three steps. The first step was to fix the mono code so that the executables were found correctly. This included understanding and then rewriting the mono code to better find the executables when the game was built. The second step was to find the correct executables to send with the game. It was found that the executables that Unity uses during the editor gameplay are only available to the editor, and will not run properly out of their designated location. A simple download of the official Mono release fixed this. The final step was getting all the dependencies that the executables needed to run, and then ensuring that the executables are placed in the correct location when we send the game to another person.

6.5.2 Menu UI

For this sprint, we worked on creating graphics for our title screen. After making our menu UI, we discovered that we needed to make graphics for buttons, background, title, and a slider. The buttons were the simplest to create as we recycled some previously made assets to make them. The slider is actually made up of 3 different graphical parts, the bar, the bar's fill,

and the sliding nub. However, creating the background proved to be more challenging than anticipated due to our ambitious design. Nevertheless, we were able to incorporate the title into the background design.

6.5.3 Overworld UI

For the overworld(UI) sprint, we created the dialogue box for the overworld dialogue. The dialogue box consists of two pieces: a light tan for the conversations and a medium brown for the name tag and buttons.

6.5.4 Level UI

For this sprint, we focused on creating graphics for our title screen. After designing our menu UI, we realized that we needed to create graphics for buttons, a background, and the title. The buttons were the easiest to create as we reused some previously made assets and made them look like small displays, similar to our player character. However, creating the background was more challenging than we anticipated, as we were used to creating smaller textures and not a full-screen graphic. To make it less empty, we added the title to the background, although we may go back and refine it further.

6.5.5 Overworld Tileset

The overworld islands are a top-down view and there are about eighty different tiles that will be used to create the three islands. Each island has different color schemes with Pseudo city being lighter colors, python jungle being darker colors, and the sharp seas reusing some from basic city and docks. Inspiration was taken from multiple different tilesets found on google.

6.5.6 Area 1 Level Tileset

During this sprint, we worked on the level Tilesets for areas 1-3. For this document, we will discuss what went into the tilesets for area 1.

For our tilesets, we need to make 47 tiles at 128 X 128 px. Those tiles are then arranged into a texture: 6 rows, 8 columns(one tile is duplicated).

Under this sprint, we made 3 textures for primary use in area 1(though we could still use them elsewhere). Bricks, Log House, and Siding House. The Siding House texture was designed to

be versatile, as its siding color can be easily changed to create other variants.

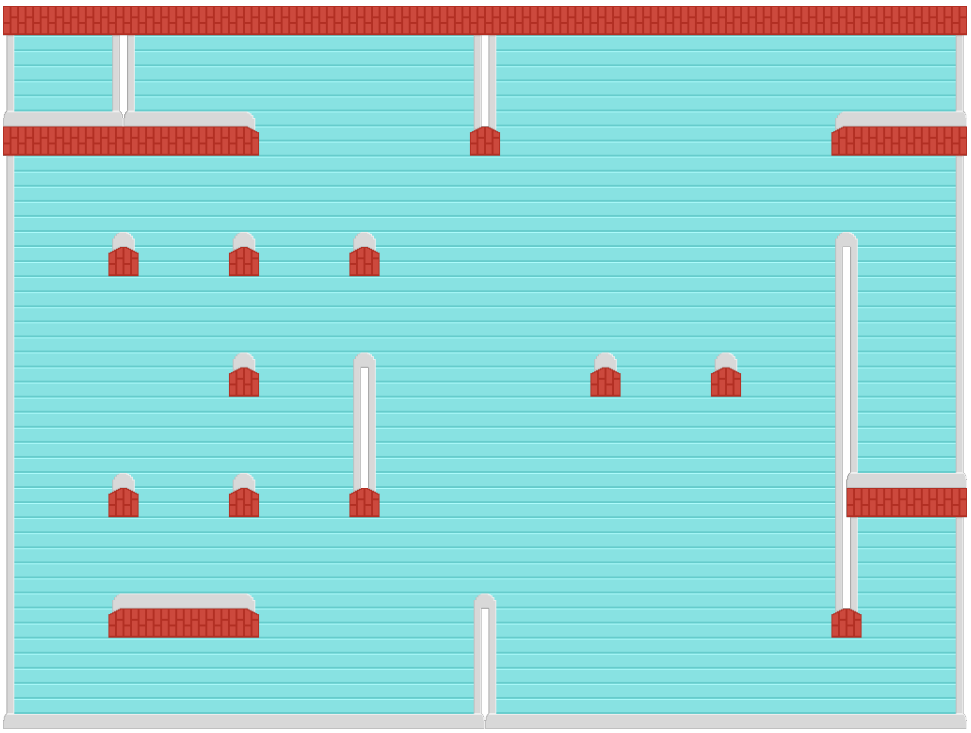

Figure 6.5: Example of sprites in Unity

6.5.7 Area 1 Level Backgrounds

For this sprint, we created the backgrounds of Pseudo City. These backgrounds were designed with accessibility in mind. The low complexity of the background will allow our player's character and other necessary elements to stand out. Each part of the background shown is supposed to repeat every 640 px.

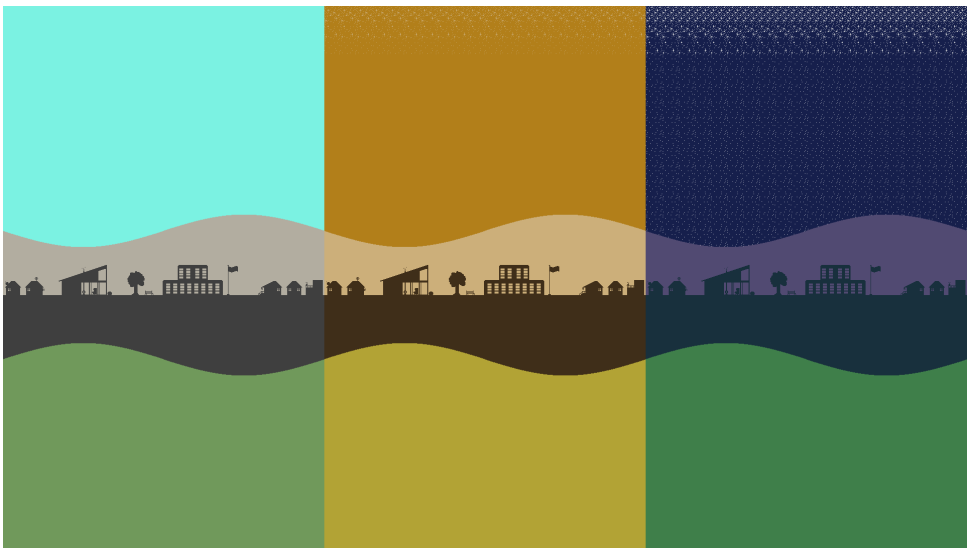

Figure 6.6: Background for Levels for Pseudo City

6.5.8 Area 2 Level Tileset

During this sprint, we worked on the level Tilesets for areas 1-3. For this document, we will discuss what went into the tilesets for area 2, and it was later finished up during sprint 3.

Under this sprint, we made 5 textures for primary use in area 2(though we could still use them elsewhere). Tree Trunk, Tree Leaves, and Ground(Plain, w/ Grass, w/ Stone path). For the Tree Trunk, it was originally thought we would need 2 textures for vertical/horizontal bark. Eventually, it was decided that a diagonal bark would be used to bridge the difference and reduce the types of bark needed. Also, for our ground, we knew we would need different ground types, so we made some minor alterations to the same texture to create the cobble and grass variants.

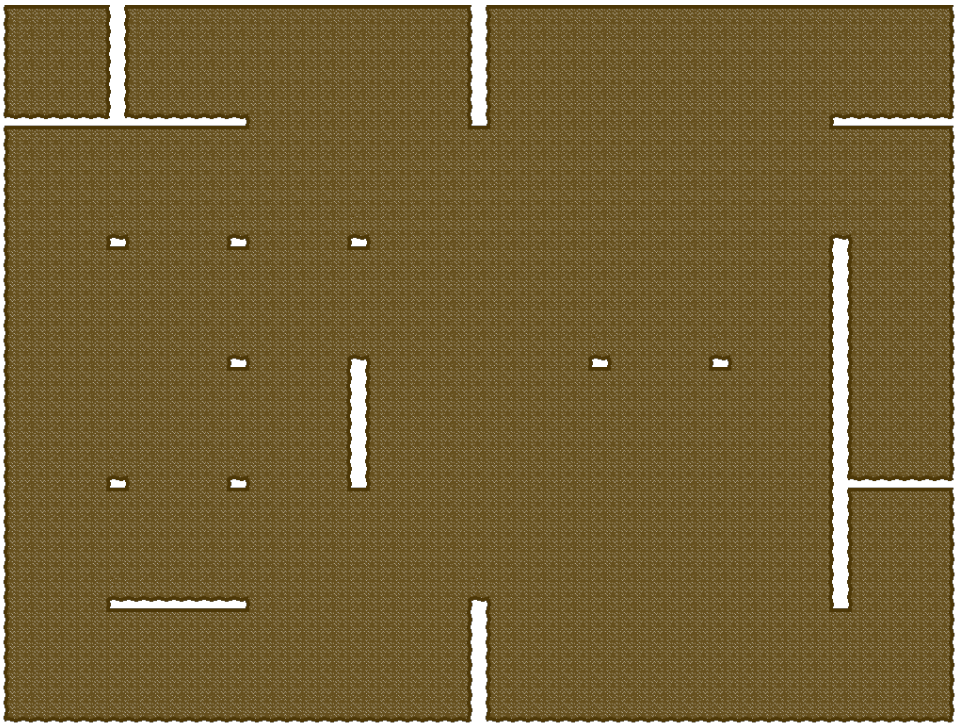

Figure 6.7: Example of sprites in Unity

6.5.9 Area 2 Level Backgrounds

6.5.10 Area 3 Level Tileset

During this sprint, we worked on the level Tilesets for areas 1-3. For this document, we will discuss what went into the tilesets for area 3.

Under this sprint, we made 3 textures for primary use in area 3 (though we could still use them elsewhere). Sand, Wood Planks, and Pier. The Pier tileset was one the most unique texture

that was made during this sprint as most of it is transparent not only that, but most of the Pier is recycled from Wood Planks and Log House.

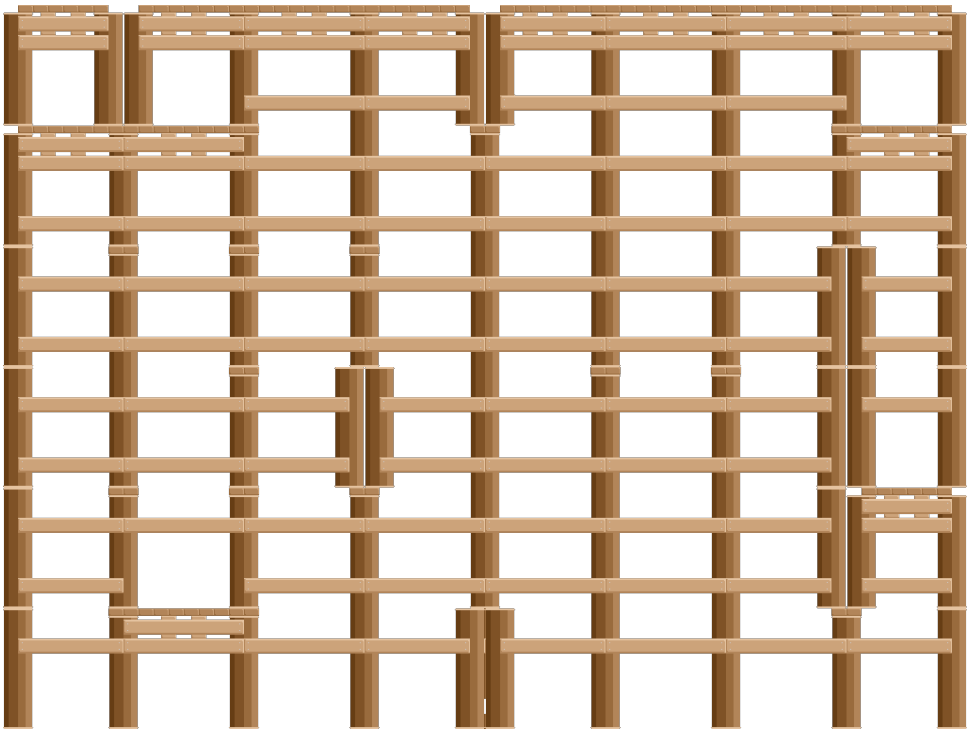

Figure 6.8: Example of sprites in Unity

6.5.11 Area 3 Level Backgrounds

For this sprint, we created the backgrounds of Sharp Seas. The screen size we are designing for is 1920 x 1080 pixels. Each part of the background shown is supposed to repeat every 640 px. We created multiple version So not every level needs to look the same.

6.5.12 Overload Player

For this sprint we worked on making the graphics for Bit in the overworld. Utilizing the design for Bit from previous Sprints we worked to scale it down to match that of other assets we have on the islands. Referencing previous drawing where we drew Bit from all sides, we made 4 different perspective shots of Bit, similar to the perspective we had with other overworld assets.

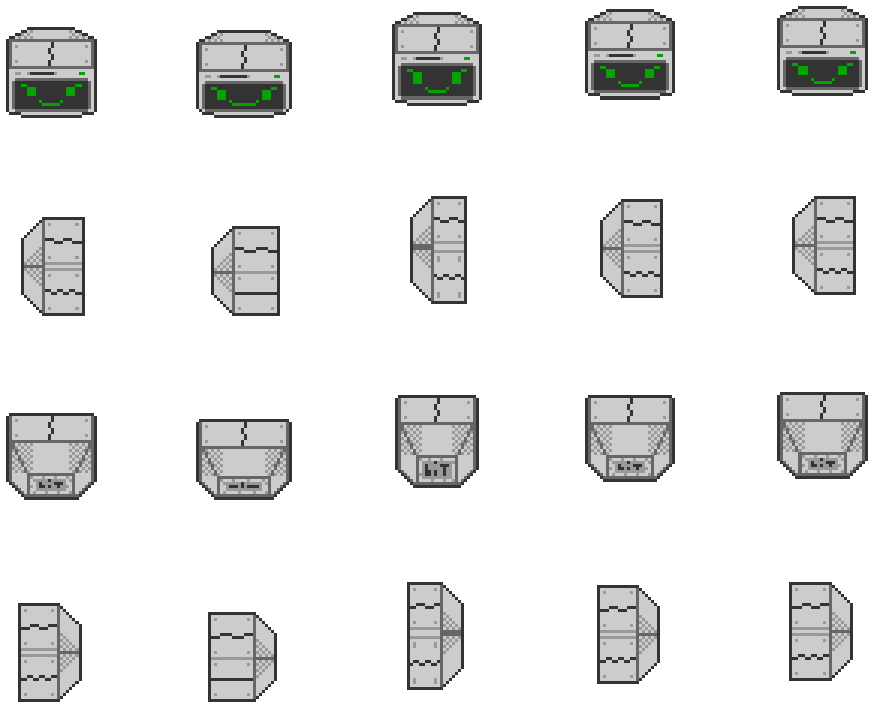

Figure 6.9: Example of sprites in Unity

6.5.13 Overworld NPCs

The overworld NPCs will be the entrance for the levels so we will need quite a few NPCs, because of this we have decided to make the NPCs parts interchangeable. This way we can make more NPCs with fewer items. For inspiration, we looked at other top down games to see the different types of character designs. Below is an example of the interchangeable parts and them put together. Also an example of how the NPCs will look in unity.

Figure 6.10: Example of Sprites of the NPCs

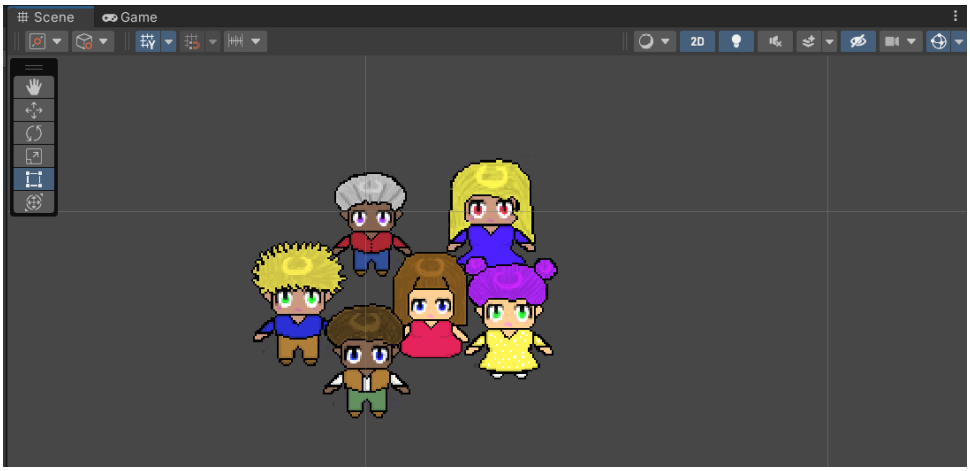

Figure 6.11: Example of sprites in Unity

6.5.14 Overworld Misc. (Part 1)

For overworld misc. (part 1) is the main decorations like houses, trees, bushes. It took awhile to get the trees to look right, inspiration came from many different games until finally settling on a style. For the bushes they are the same design as the trees, just smaller and no tree trunk. The houses have the same style as each other, just different colors.

Figure 6.12: Example Decorations for Island 1

6.5.15 Overworld Misc. (Part 2)

For overworld misc. part 2, the graphics needed was a ship to take the player to the different islands. The ship is based on various different top-down ships in other games. The dock is a part of the ship in unity but in paint.net it is in a separate layer so it can be adjusted later on. Also, added different types of flowers for the overworld as decorations.

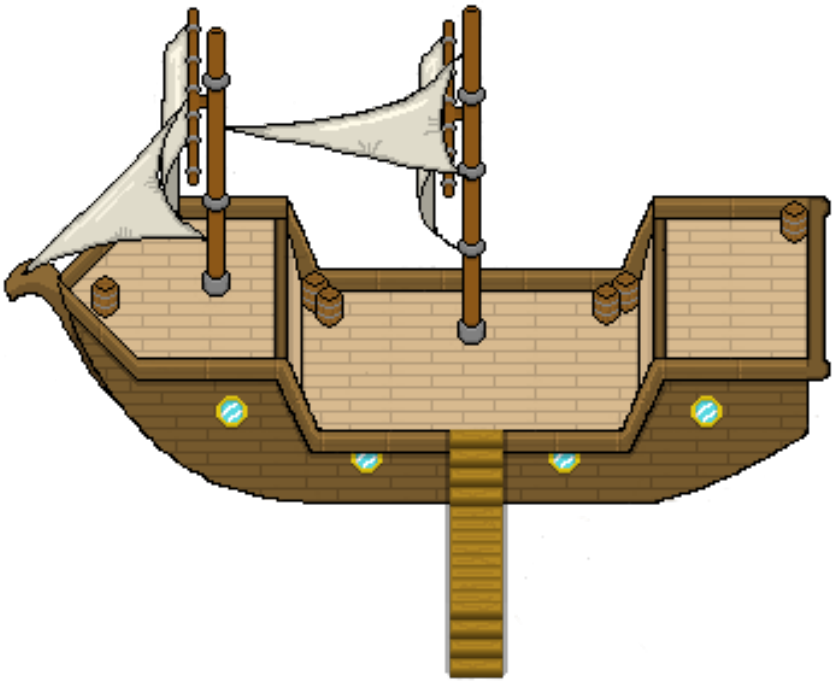

Figure 6.13: Example of ships to transport player

6.5.16 Overworld Misc. (Part 3)

6.5.17 Island 1

For island one, we loaded in the tilesets for area 1 into the tile palette. Next, we added a grid to the scene, set the grid to .64 by .64 then added in the 2d object tilemap. For the design of the island, we took the grass tile and did a complete random shape then added the sand tiles around it. For the boundary lines of the map, we used a 2d edge collider. After that was in, we added water to the map so that's all the player sees when they hit a boundary line.

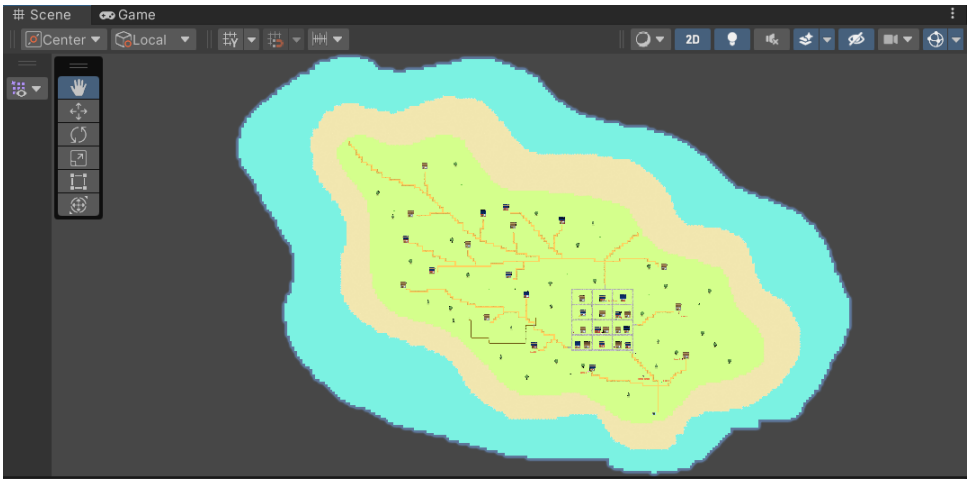

Figure 6.14: Island 1 in Unity

6.5.18 Island 2

For island 2 we loaded in the tilesets for area 2 into the tile palette, which are the same as the area 1 tilesets just re-colored. Next we added a grid to the scene, set the grid to .64 by .64 then added in the 2d object tilemap. For the design of the island we took the grass tile and did two complete random shapes then added the sand ocean tiles around it. For the boundary lines of the map we used a 2d edge collider. After the boundary lines were put in we added more ocean outside of the boundary lines so that when the player zooms out all they can see is ocean.

6.5.19 Island 3

For island 3 we loaded in the tilesets for area 1 and area 2 into the tile palette, since island 3 reuses tiles from both previous islands. Next we added a grid to the scene, set the grid to .64 by .64 then added in the 2d object tilemap. For the design of the island we took the grass tile and did multiple random shapes to create mini islands then added the sand ocean tiles around it. For the boundary lines of the map we used a 2d edge collider. After the boundary lines were put in we added more ocean outside of the boundary lines so that when the player zooms out all they can see is ocean.

6.5.20 Level Robot

For this sprint, we needed to develop a sprite for the player character Bit. Through research and discussion, we discovered some constraints we would need to put on this task for completion. We knew before our first sprint, we wanted to employ a pixel art style, for its low development

time and its general popularity. In order to capture this style, we applied some self-imposed constraints, for example, we limited the character sprite to 128x128 pixels, and used as few colors as we could. Our beta sprite (A1) fit these criteria, but failed by being just a little small, and using isometric perspective. The perspective was the big issue as we decided an orthographic view would work better as we did not need to worry about vanishing points when designing levels. Sprites (A2) and (A3) were used to test the size we would want for Bit. Sprite (B2) through (B3) we attempted to emulate an old crt television with the faux-wood paneling. In tandem, we also experimented with the screen size for Bit because of not just the color, but size of dials and sliders we went in a new direction for inspiration. All later sprites took inspiration from computer monitors from the 80's/90's. Instead of the bulky controls on the television, computer monitors had sleeker devices built in the face of the monitor, like floppy drives, indicator lights, and buttons. Sprite (C2) we got a fix on Bit's screen size, and the only difference between it and the final design are minor details and color. Finally, we went with the color scheme introduced in (D2), with just recoloring of some decor for the final sprite (D3). We created the faces alongside (C3). The last task we wanted to accomplish was designing Bit's other five sides (so we had something to reference for later Bit sprites).

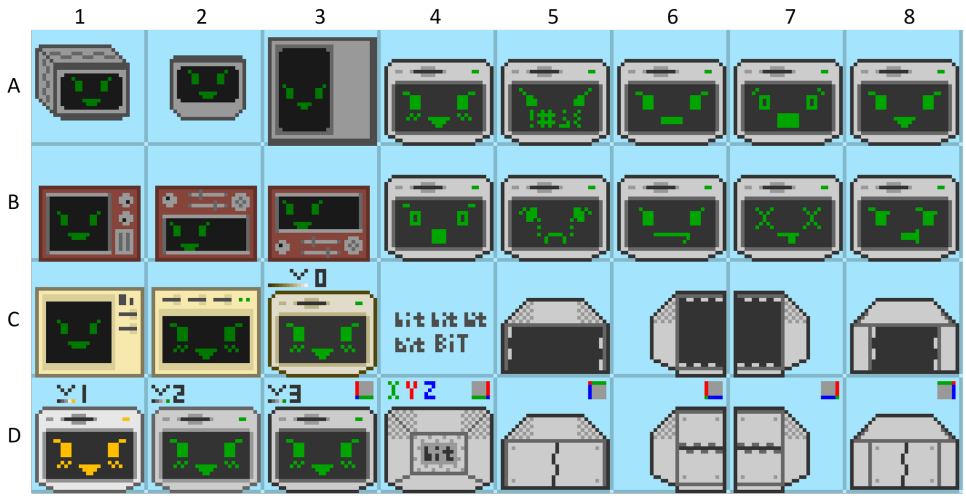

Figure 6.15: Timeline of the Sprite of Bit

6.5.21 Level Tools

For this sprint, we created many different graphical assets for our levels. To keep track of what graphics were needed, we made a list where team members could add what graphics they needed for the levels they were creating. The tools were some of the most detailed graphics we made, each being 256x256 pixels. Although the tools didn't take up a majority of the space, this was

done to help with placing them in the game.

6.5.22 Level Objects

For this sprint, we created various graphical assets for our levels. To ensure efficient task management, we developed a list where team members could add the specific graphics they required for their assigned levels. As we worked on the assets, we took into consideration the size requirements of each object and adjusted the sprite size accordingly. The smallest objects were 16x16 pixels, while the largest was 256x512 pixels.

6.6 Epic 6 - Level Design

6.6.1 Lesson Discovery

Below is a table that shows what concepts we plan on using as the questions for our user. First, the user will learn the concept in pseudo code, then the questions about that topic will open in Python then C# when Python is completed. The user can choose which island they would like to practice the concept in. For concepts that are specific to the language, such as ternary in C#, the user will have access to that part of the island after learning what is needed to complete that task. For this example, the user will need to learn variable types, variable assignments, relational operators, and if-else statements.

Pseudo	Python	C#
Variable type	Variable Type	Variable Type
Variable Assignment	Variable Assignment	Variable Assignment
Arithmetic Operators	Arithmetic Operators	Arithmetic Operators
Logical Operators	Logical Operators	Logical Operators
Relational Operators	Relational Operators	Relational Operators
If Statements	If Statements	If Statements
If-Else Statements	If-Else Statements	If-Else Statements
Nested If-Else	Nested If-Else	Nested If-Else
Switch Statements	Switch Statements	Switch Statements
Input/Output	Input/Output	Input/Output
While	While	While
Do-While	Do-While	Do-While
For	For	For
Array 1D-2D	Array 1D-2D	Array 1D-2D
Function Declaration	Function Declaration	Function Declaration
Pointers		Pointers

Table 6.1: Lesson Discovery

6.6.2 Gameplay Feasibility

It is feasibly possible to implement a form of parser/compiler for Python and C#. In the following sprints I will attempt to use "ICodeCompiler" and "IronPython3" as my form of runtime compilation and parsing. I expect great difficulty in the implementation since the hundreds of sources I have found are outdated (2003 to 2015), and no person has exactly done the work we attempt to do in our game. If all else fails, the game can run on fill-in-the-blank boxes and code can be written to manually parse the syntax of the language (grammar and lexeme creation through Sprache and ANTLR).

6.6.3 C# Parser and Object Movement

My Task for this sprint was to add a working C# compiler inside Unity. Since building a lexeme and language parser would have been extremely difficult/time consuming, I decided to attempt my hand at runtime scripting using the CodeDomCompilerClass. Through hours of reading over fifty or so webpages trying to learn how to get this project started, I came across a really helpful example at StackExchange.com. Using this example and the information I gained from the many other webpages, I was able to attach a .cs file to a Unity object and run it during runtime of the game. After the addition of the file, I was able to use the CodeDomCompileClass to compile and execute the script during runtime of the game.

6.6.4 Python Parser and Object Movement

My task during this sprint was to implement a Python compiler inside our Unity project. Instead of implementing my own form of lexeme and grammar parser, I researched and found a neat package developed by an independent group named IronPython. For the longest time IronPython only supported Python2, but just this year a beta version for Python3 was released. Due to this, all documentation for using IronPython3 was non-existant in the seventy or so webpages I researched through. After a good week of websearching, I was able to gather snippets of code that allowed me to reproduce a working compiler for Python3 inside of our Unity project.

6.6.5 World 1, Level 1

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets

that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. The baskets cannot move, but can register if the correct box is in it. The boxes appear one at a time. When the current box is put into the current basket, it will disappear and the next box will be visible. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.6 World 1, Level 2

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The box can be picked up if the player character is close to the box and presses “e”. The baskets cannot move, but can register if the correct box is in it. The baskets are changed to look like the outside walls of a house, and the walls change colors when they detect the box. Once the player changes the color of all walls, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.7 World 1, Level 3

This level teaches the concept of if statements. Through a dialog box explaining the concept, 2 fill in the blank portions, and gameplay portion. The fill in the blank portions have code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. The baskets cannot move, but can register if the correct box is in it. Once the player character places the correct amount of boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.8 World 1, Level 4

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. The baskets cannot move, but can register if the correct box is in it. Once the player character places the correct amount of boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.9 World 1, Level 5

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. The baskets cannot move, but can register if the correct box is in it. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.10 World 1, Level 6

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”,box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”, the boxes look like chickens or roosters. The baskets cannot move, but can register if the correct box is in it. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.11 World 1, Level 7

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!",box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The basket cannot move, but can register if the correct box is in it. Each box has a color related to it. The baskets are changed to look like the fences, and the fences change colors when they detect a box to the color of the box. Once the player changes the color of all fences to the correct color, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.12 World 1, Level 8

This level teaches the concept of if statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", box objects will appear for the gameplay section. The gameplay section has a non-playable character, boxes and baskets. The boxes can be picked up if the player character is close to the box and presses "e". When the box is picked up, smaller boxes appear. The basket cannot move, but can register if the correct box is in it. Once the player character has the middle smaller box in the basket for 5 seconds, the non-playable character can be interacted with, having a visible indicator appearing over its head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.13 World 1, Level 9

This level teaches the concept of switch statements. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The baskets cannot move, but can register if the correct box is in it. There are three rounds where each round the two boxes have to be placed in the correct basket and the instructions update to guide the player. Once the player character places the boxes in the correct basket for each round, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.14 World 1, Level 10

This level teaches the concept of while loops. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", the box object will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, a box and baskets. The instructions appear in a text box above the non-playable character's head. The box can be picked up if the player character is close to the box and presses "e". The baskets cannot move, but can register if the box is in it. The box has a text input field attached and a text box attached to each basket. Once the player character the box in the range of a basket, the text box attached to the basket will update from "Name" to the text in the box. Once all baskets have a new name, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character

touches the non-playable character, they will beat the level.

6.6.15 World 1, Level 11

This level teaches the concept of input/output. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", the box object will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, a box and baskets. The instructions appear in a text box above the non-playable character's head. The box can be picked up if the player character is close to the box and presses "e". The baskets cannot move, but can register if the correct box is in it. When the box is in a basket the color of the basket changes to gray. Once the player character changes the color of all baskets, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.16 World 1, Level 12

This level teaches the concept of do while loops. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", the box objects will appear. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The baskets cannot move, but can register if the correct box is in it; moreover the box the basket is looking for looks like a cat. Once the player character places the cat in the basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.17 World 1, Level 13

This level teaches the concept of for loops. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. If the player presses the left mouse button with the box in hand, the object can be used. The baskets cannot move, but can register if the correct box is in it. Once the player character picks up and puts down the box 5 times in each of the 5 baskets, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.18 World 1, Level 14

This level teaches the concept of 1D arrays. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or “Invalid”, if one value creates an error. When all sections show “Correct!”, box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character’s head. The boxes can be picked up if the player character is close to the box and presses “e”. The basket cannot move, but can register if the correct box is in it. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character’s head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.19 World 1, Level 15

This level teaches the concept of 2D arrays. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon with a chicken and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The basket cannot move, but can register if the correct box is in it. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions with a chicken object to the left of the non-playable character. When the player character touches the non-playable character, they will beat the level.

6.6.20 World 1, Level 16

This level teaches the concept of function calls. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The basket cannot move, but can register if the correct box is in it. There are four baskets with one holding three box objects that look like chickens and three the player can use. There is a box object the player can pick that makes a wall appear around the baskets with the chickens. There is a box object that looks like a cat slowly coming towards the chickens, but the cat will flee if the cat hits a wall. If the cat gets to the chickens, all of the chickens disappear, a red "X" appears over the non-playable character, and level has to be reset.Once the chickens last 60 seconds, the cat

will disappear. The non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.21 World 1, Level 17

This level teaches the concept of pointers. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 4 sections. Once the player puts input into every blank in a section, it will produce a message saying, "Correct!", if it is all correct, "Incorrect!", if any value was not correct, or "Invalid", if one value creates an error. When all sections show "Correct!", box objects will appear for the gameplay section. The gameplay section has a non-playable character with an icon and instructions, boxes and baskets. The instructions appear in a text box above the non-playable character's head. The boxes can be picked up if the player character is close to the box and presses "e". The basket cannot move, but can register if the correct box is in it. The boxes are attached to each other in three groups of three with the middle box looking like a rope, but the rope does not get counted by the baskets. Once the player character places all the boxes in the correct basket, the non-playable character can be interacted with, and an icon will appear over the non-playable character's head; although, below the instructions. When the player character touches the non-playable character, they will beat the level.

6.6.22 World 2, Level 1

For this level, the coding portion is the player will translate binary into it integer and assign it to the correct var. There is no gameplay for this level.

6.6.23 World 2, Level 2

For this level, the player will look at the blank puzzle pieces and assign them to the correct spot. If they get it correct they get to see the finished puzzle as the gameplay.

6.6.24 World 2, Level 3

For this level the player is solving math equations to get supplies ready to build a house. Once the level is complete the supplies will show up and the user can exit the level.

6.6.25 World 2, Level 4

For this level, the player uses logic operators to turn bits scanner on. Then the player walks across the screen picking up eggs. Then they come to part 2 of the coding portion where they use logic operators to sort which snake eats which eggs.

6.6.26 World 2, Level 5

For this level, the player uses relational operators to program a lawn mover what it can and can't cut down. The player then mows the NPCs lawn.

6.6.27 World 2, Level 6

for this level, the player uses if statements to decided which fireflies the can and can not catch. Once the level is correct the player will be able to catch fireflies.

6.6.28 World 2, Level 7

For this level, the player uses if else statements to decided if the dirt is good dirt to plant a tree. Once the coding portion is correct they player can plant some trees.

6.6.29 World 2, Level 8

For this level, the player uses nested if statements to sort snakes by color and stripes. Once they get the level correct the player will see snakes getting sorted.

6.6.30 World 2, Level 9

For this level, the player uses switch statements to assign how much each flower get watered. Once they get the level right the player walks across the screen to water flowers.

6.6.31 World 2, Level 10

For this Level, the player uses input and output statements to get the input for an email and use the output to send it. For the gameplay, the user writes in what they want to be sent and see the output of what they entered.

6.7 World 2, Level 11

For this level the player writes code to replace a dirt road to cobblestone with while loops. The player then places in the cobble road. For the second codding part the player write the code to place fences at each new road built. Then they place the fences and can exit the level.

6.7.1 World 2, Level 12

This level teaches the concept of for loops. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 2 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or an error message, if one or values create an error. When all sections show “Correct!”, an staircase looking object will appear. The gameplay section has a non-playable character, the staircase object, and a tree with snake objects in it. The staircase can be climbed up. If the player character reaches the center of the tree where the leaves are located, the snakes will leave the tree, and the non-playable character can be interacted with. When the player character touches the non-playable character, they will beat the level.

6.7.2 World 2, Level 13

This level teaches the concept of 1D arrays. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 2 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or an error message, if one or values create an error. When all sections show “Correct!”, a basket looking object will appear. The gameplay section has a non-playable character, the basket object, and flowers. The basket can be picked up. If the player character reaches a flower holding the basket, the flowers will be hidden. Once all of the flowers are hidden, the non-playable character can be interacted with. When the player character touches the non-playable character, they will beat the level.

6.7.3 World 2, Level 14

This level teaches the concept of 2D arrays. Through a dialog box explaining the concept, a fill in the blank portion, and gameplay portion. The fill in the blank portion has code snippets that need to be filled in. The code snippets are split up into 2 sections. Once the player puts input into every blank in a section, it will produce a message saying, “Correct!”, if it is all correct, “Incorrect!”, if any value was not correct, or an error message, if one or values create an error. When all sections show “Correct!”, a snake looking object will appear. The gameplay section has a non-playable character, 2 snake objects, and nests. The snake that appeared can be picked up. If the player character reaches the other snake holding the snake that appeared, the other will be hidden and appear at another nest. Once the player character brings the snake that appeared at the start to every spot the other snakes appears in, the snake will appear in front of the non-playable character, and the non-playable character can be interacted with. When the player character touches the non-playable character, they will beat the level.

6.7.4 World, Level 15

This level teaches the concepts of functions to the player in the Python language. We wanted to incorporate the different aspects of the function (definition and call), as well and show that Python is not restricted to just one variable return like many other languages. This level simply allows the player to write their own swap function and their own function call. We designed this level to not immediately be difficult, but instead lead up to the difficulty of writing all the code for a swap function and function call.

6.7.5 World 3, Level 1

This level functions similarly to world 1, level 1, but with small adjustments to the questions to fix the context of C#. It teaches them 6 data types, int, char, String, bool, float and double; moreover, how they can be assigned to a variable. The level contains platforms, buttons ,a player object, tmp textObjects and tmp InputObjects. The platforms are used for the player object to stand on; moreover, they allow the player to move toward the complete level object, but one platform blocks the player from reaching it. The scriptController object connects the inputObjects and textObjects to the code, and the compileCode is a button object that runs the script when clicked. My script receives input for each type in the related InputObject, and puts them into try blocks. The try block makes sure the value being inputted is valid, or it will

throw it to its corresponding catch block. In the catch block, it prints “incorrect” in yellow in its corresponding textObject for output. If the value is valid, it will print “correct” in cyan in its corresponding textObject for output. If the input is null, the output will remain empty. If all inputObjects are inputted correctly, the level will be able to be completed, for the platform blocking the complete level object will be removed.

6.7.6 World 3, Level 2

It introduces them to how variables can interact with each other. The level contains platforms, buttons ,a player object, tmp textObjects and tmp InputObjects. The level shows 4 pseudo code sections, and they are expected to insert the correct input. The platforms are used for the player object to stand on; moreover, they allow the player to move toward the complete level object, but one platform blocks the player from reaching it. The scriptController object connects the inputObjects and textObjects to the code, and the compileCode is a button object that runs the script when clicked. My script receives input for each type in the related InputObject, and puts them into try blocks. The try block makes sure the value being inputted is valid, or it will throw it to its corresponding catch block. In the catch block, it prints “invalid” in red text in its corresponding textObject for output. If the value is valid and the correct value, it will print “correct” in cyan text in its corresponding textObject for output. If the value is not correct, “incorrect” in yellow text in its corresponding textObject for output. If all inputObjects are inputted correctly, the level will be able to be completed, for the platform blocking the complete level object will be removed.

6.7.7 World 3, Level 3

The goal for this story was to create implementations for the CSharp Levels 3. I wanted to design the levels to have more gameplay interaction than just complete the level and move on. For level 3, I added it so that as the problems were completed, pieces of the boat would appear. The goal is as the the levels get more complex, the interaction improves more and more.

6.7.8 World 3, Level 4

The goal for this story was to create implementations for the CSharp Levels 4. I wanted to design the levels to have more gameplay interaction than just complete the level and move on. For level 4, I added it so that Styrofoam would disappear as the player completes problems.

This gives the level a little more interaction. The goal is as the the levels get more complex, the interaction improves more and more.

6.7.9 World 3, Level 5 and Level 6

The inspiration for this Story came from the requirements given for the level. For level 5, the requirement was to have fishing rods and fish. So, the design for this level was placed around having fish and fishing rods in the level. The fish will randomly come up and out of the water. When they do that, the player can write the correct Boolean expression to show that the fish is there. There are 6 problems to solve for this level. For level 6, the inspiration was on chests popping up out of the ground. So the design was that BIt would walk across the level. When he comes upon a chest, the chest would pop up out of the ground and the code portion would be available. The player then writes the correct if-statement to open the chest and then move on in the level to the next chest. The NPC at the end unlocks once all chests are found.

6.7.10 World 3, Level 7 and Level 8

The inspiration for this Story came from the requirements given for the level. For level 7, the requirement was to have dog food sorted. So I came up with the idea that the dog food will get sorted when the player runs their code. The player writes an if-statement to sort the food, and when the code is run, dog food gets sorted on the screen. If the code resulted in the correct answer, the dog food sorts in the correct baskets. If the answer is wrong though, the dog food is also sorted incorrectly. For level 8, the idea was to have the player buy fish at a fish market. The player writes the correct if-else-if statements to simulate the purchasing of fish and adding up the total. When the code is ran, if the answer is correct, NPC will walk across the screen and purchase fish at the shop. The prices will get shown on the shop stand, and will be accurate to the test data. However if the code is incorrect, the prices shown will be ridiculous. This helps encourage the player to fix their code to be correct.

6.7.11 World 3, Level 9, Level 10, Level 11, Level 12, and Level 13

The inspiration for these stories came from thought about what I want the player to learn. Level 9: To me, they had already gotten through if and if-else statements. All a switch statement is is a glorified if-statement. I wanted the player to have some fun. So, I wrote it so that the player would only have to write a correct switch statement and watch some Crabs race across

the screen. Level 10: This level was difficult, and certainly the least fun to develop. It supports input and output to a screen and the player talks to Pirates and pre-defined text messages. Level 11: For the first of the loops, I wanted to keep it simple. Just increment some variables. The kick here is that the player sees that while the loop is executing, two crabs are playing Tug-A-War. Level 12: I wanted to have some more fun again. With this level, the player writes a do-while loop to time the runtime that Turtles race across the screen. Level 13: This level I wanted to draw back on what I did once I learned how to do a for-loop. It was a faster While loop in reality. For this level I had them write up some code to find the operation for Number and the next 5 numbers in sequence.

6.7.12 World 3, Level 14, Level 15, Level 16, and Level 17

Just as in the other sprints, this sprint was focused on developing new levels. The inspiration for these levels did not come through the dialogue design. For the 1 and 2 Dimensional Arrays, I wanted the user to define their own array and use it. I designed the level to be as such the player would enjoy writing the array and watching the animation play through. I also wanted to focus on "If not found keep searching" concept. Both levels go through that example. For Functions, I wanted the player to see the importance of writing code separately and just calling it to do a mundane task. The code and level plays as such where the user has to add up five sets of arrays. Something so repetitious and long, they will be happy that they used a function. For level 17, this was the most complicated level to implement. I wanted it to be the last level, and most fun of them. As such, I included a bunch of things that went into it. The hardest part however was getting pointers to work in C. Since C does have a garbage collector and it is managed, you can only use pointers in a strict context. Sadly, everything I was trying to do in the level was outside the context that C allowed for pointers. So, for this level, I imitate pointers through the use of a 2D and 1D array. I also used a bunch of string manipulation techniques so that I could replace the user's code with that of what would work in the level. After the level worked, I touched it up to be sentimental for how the last level in our game should go.

6.7.13 Rework of Past Levels

For this sprint, I went in and added a background to all the levels, changed out some graphics, redid levels world 2, level 1 and world 2, level 2.

6.7.14 Assign Gameplay to Levels

For this sprint, I came up with ides for gameplay for levels and assign what i thought would be the best fit for each level.

6.8 Epic 7 - UI

6.8.1 Main Menu

The main menu is a scene in unity that contains five buttons, “New Game”, “Load”, “Options” and “Quit”; moreover, the scene has a hidden options menu with text boxes, a slider and a “Back” button. Only three buttons perform an action when clicked, “New Game”, “Options” and “Quit”. First, The “New Game” button changes scenes to the overworld when clicked. Second, when the “Options” button is clicked, it hides the main menu buttons and shows the options menu button, text boxes and sliders. Second, The options menu has a textbox showing “Options” in the top section and “Volume” in a textbox above the slider. The slider is in the middle of the scene; moreover, it can slide left to right. The “”Back” button hides the option menu assets and shows the main menu assets. Third, the “Quit” button exits the game when clicked. It uses a C script to close the game when the “Quit” button is clicked, and hide/show the buttons, textboxes and slider when the “Options” and “Back” button is clicked.

6.8.2 Overworld

For the overworld(UI) sprint, we created the dialogue box for the overworld dialogue. The dialogue box consists of two pieces: a light tan for the conversations and a medium brown for the name tag and buttons.

6.8.3 Levels

The purpose of this story was to modify all the current implemented levels so that they look professional. In order to do this, we needed to refactor the whole level from the ground up. We went with a solid color for the ground, and two walls on the ends to enclose the player in the level. After this, we modified the camera to use a 2D border that it can’t go past. This created the scene and level effect we desired. Finally, we needed to put the level together. We decided to use a big TV display to show the code boxes and the dialogue for the level. This gives the visual to the player that they are in the world that Bit lives in. We also added in buttons at the

top that the player can click to: leave the level, review their notebook and save their progress before leaving.

6.8.4 Opening and Ending Credits

For this sprint we worked on the opening and ending credits to our game.The Opening Credits were fairly simple with Unity as all we needed was a team logo, and add it in as a splash screen. The ending credits were a little more involved as we needed to make a new scene for it. For this scene we made a screen sized black background, that we would program the text to rise in front of. We also added the functionality where if the player presses any button the credits will end early.

6.8.5 Overlay Menus

For this sprint, we worked on creating a menu that pops up when the player clicks on the menu button. The pop up menus will be for the overworld and levels.

6.8.6 Error Messages

This story was focused on creating error messages in the game that provided the player with better understanding of what happened then simply "Object Reference not set to an object". The error messages primarily involved work with the C compilers, and some with the Python compilers.

6.8.7 Dialog Overlay

For this story, I used a useful youtube video created by Brackeys called How to make a Dialogue System in Unity. With this video and help from other team members, I was able to create the dialogue box that was triggered by a 'Start Lesson' button and create a 'Continue' button that triggers the next part of the dialogue. I implemented this dialogue in the first three levels of Pseudo City. Using the same youtube video we created an overworld dialogue box that has a back button to exit out of the conversation. A continue button that lets the player keep talking to the npc, and a start button that when pressed takes the player to the corresponding level using the load level script.

Chapter 7

Test Cases

7.1 Navigating through the Main Menu

Navigating through the Main Menu

Purpose: Verify the user story Main Menu (all parts).	
Test Run Information: Tester Name: Lauren Taylor Date(s) of Test: 04/12/22 Location/server being used: Unity website	Prerequisites for this test: None
	Software Versions: Application: Unity website Browser(used and those COTS supports): N/A Database: N/A Operating System: Windows 10
	Required Configuration: (browser setup, security or user ID roles) No special setup needed
Notes and Results	
TEST SCRIPT STEPS/RESULTS	

STEP	TEST STEP/ INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS or FAIL
1.	User opens the game.	The Unity Logo appears with the Best Cap opening credit logo appearing next. After, the main menu pops up and the user will only have access to the mouse. The main menu background music starts playing.	The Unity Logo appears with the Best Cap opening credit logo appearing next. After, the main menu pops up and the user will only have access to the mouse. The main menu background music starts playing.		PASS
2.	User presses the new game button.	The user is sent to Level 0.	The user is sent to Level 0.		PASS
3.	User presses continue button.	The user is sent to the last place they were in the game such as a level or a spot on an island.	The user is sent to the last place they were in the game such as a level or a spot on an island.		PASS
4.	User presses the options button.	The user is shown the options menu.	The user is shown the options menu.		PASS

5.	User presses the quit button.	The game closes and progress is saved.	The game closes and progress is saved.		PASS
6.	User uses a keyboard.	Nothing should happen.	Nothing happens		PASS
7.	User clicks on a graphic.	Nothing should happen.	Nothing happens		PASS

7.2 Navigating through the Options Menu

Navigating through the Options Menu

Purpose: Verify the user story Option Menu (all parts).	
Test Run Information: Tester Name: Lauren Taylor Date(s) of Test: 04/12/22 Location/server being used: Unity website	Prerequisites for this test: None
	Software Versions: Application: Unity website Browser(used and those COTS supports): N/A Database: N/A Operating System: Windows 10
	Required Configuration: (browser setup, security or user ID roles) No special setup needed
Notes and Results	
TEST SCRIPT STEPS/RESULTS	

STEP	TEST STEP/ INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS or FAIL
1.	Option menu open up.	The options menu is correctly sized and the user should only have access to the mouse.	The options menu is correctly sized and the user should only have access to the mouse.		PASS
2.	User slides the master volume slider to the right.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.		PASS
3.	User slides the master volume slider to the left.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.		PASS
4.	User slides the music volume slider to the right.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.		PASS

5.	User slides the music volume slider to the left.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.		PASS
6.	User slides the SFX volume slider to the right.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.	When the user slides the volume slider to the right, the volume will increase and the value will show you what percentage you have set it at.		PASS
7.	User slides the SFX volume slider to the left.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.	When the user slides the volume slider to the left, the volume will turn down and the value will show you what percentage you have set it at.		PASS
8.	User clicks mute button on the master volume when the box is not checked.	Every sound in the game will be muted.	Every sound in the game will be muted.		PASS

9.	User clicks mute button on the master volume when the box is checked.	Every sound in the game will be unmuted.	Every sound in the game will be unmuted.		PASS
10.	User clicks mute button on the music volume when the box is not checked.	Music in the game will be muted.	Music in the game will be muted.		PASS
11.	User clicks mute button on the music volume when the box is checked.	Music in the game will be unmuted.	Music in the game will be unmuted.		PASS
12.	User clicks mute button on sfx volume when the box is not checked.	Sound effects in the game are muted.	Sound effects in the game are muted.		PASS

13.	User clicks mute button on sfx volume when the box is checked.	Sound effects in the game are unmuted.	Sound effects in the game are unmuted.		PASS
14.	User clicks the User Manual button.	The User Manual will pop up on the screen.	The User Manual will pop up on the screen.		PASS
15.	User presses enter.	Nothing should happen.	Nothing happens		PASS
16.	User presses anything on the keyboard.	Nothing should happen.	Nothing happens		PASS
17.	User clicks on a graphic.	Nothing should happen.	Nothing happens		PASS
18.	User presses the back button.	The screen will return to where the user was before pressing the options menu.	The screen will return to where the user was before pressing the options menu.		PASS

7.3 Navigating through Level 0

Navigating through Level 0

<table>
<tr><td colspan="6">Purpose: Verify the user story Level 0.</td></tr>
<tr><td colspan="3">Test Run Information:
Tester Name: Lauren Taylor
Date(s) of Test: 04/17/22
Location/server being used: Unity website</td><td colspan="3">Prerequisites for this test: None</td></tr>
<tr><td colspan="3"></td><td colspan="3">Software Versions:
Application: Unity website
Browser(used and those COTS supports): N/A
Database: N/A
Operating System: Windows 10</td></tr>
<tr><td colspan="3"></td><td colspan="3">Required Configuration:
(browser setup, security or user ID roles)
No special setup needed</td></tr>
<tr><td colspan="6">Notes and Results</td></tr>
<tr><td colspan="6">TEST SCRIPT STEPS/RESULTS</td></tr>
<tr><td>STEP</td><td>TEST STEP/ INPUT</td><td>EXPECTED RESULTS</td><td>ACTUAL RESULTS</td><td>Requirements Validated</td><td>PASS or FAIL</td></tr>
<tr><td>1.</td><td>Level opens.</td><td>Opens with Bit, some objects, a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.</td><td>Opens with Bit, some objects, a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.</td><td></td><td>PASS</td></tr>
</table>

2.	User uses arrows.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.		PASS
3.	User uses the space bar.	Bit jumps.	Bit jumps.		PASS
4.	User presses A/a on the keyboard.	Bit moves to the left on the screen.	Bit moves to the left on the screen.		PASS
5.	User presses D/d on the keyboard.	Bit moves right on the screen.	Bit moves right on the screen.		PASS
6.	User clicks the Continue button while there still is dialogue to be said.	The dialogue box will show the next portion of the dialogue.	The dialogue box will show the next portion of the dialogue.		PASS

7.	User clicks the Continue button before the Start Lesson button.	The dialogue will disappear and the question will appear.	The dialogue is skipped and the question appears.		PASS
8.	User clicks the Continue button after the Start Lesson button while there still is dialogue to be said.	The dialogue box will show the next portion of the dialogue.	The dialogue box will show the next portion of the dialogue.		PASS
9.	User clicks the Continue button after the Start Lesson button when there is no more dialogue to be said.	The dialogue box disappears and the question appears.	The dialogue box disappears and the question appears.		PASS

10.	The user clicks inside the question box.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.		PASS
11.	The user clicks outside of the question box.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.		PASS
12.	User gets the question correct.	The screen will indicate that the user got the question correct and the correct sound effect will play.	The screen will indicate that the user got the question correct and the correct sound effect will play.		PASS
13.	User gets the question wrong.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.		PASS

14.	User clicks on a non-button or non-playable object.	Nothing happens.	Nothing happens.		PASS
15.	The user clicks on the R (reset) button.	The level is reset fully.	The level is reset fully.		PASS
16.	The user clicks on the N (notebook) button.	The notebook pops up.	The notebook pops up.		PASS
17.	While in the notebook, the user presses the close button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.		PASS
18.	The user clicks the M (menu) button.	The menu pops up over the current level screen.	The menu pops up over the current level screen.		PASS
19.	The user clicks on the back button in the menu.	The menu that is over the level screen disappears.	The menu that is over the level screen disappears.		PASS

20.	The user clicks on the save button in the menu.	The user's progress is saved.	The user's progress is saved.		PASS
21.	The user clicks on the options button in the menu.	The options menu appears over the menu screen.	The options menu appears over the menu screen.		PASS
22.	The user clicks on the notebook button in the menu.	The notebook pops up.	The notebook pops up.		PASS
23.	The user clicks on the exit to world button in the menu.	The user is brought back to the over-world.	The user is brought back to the over-world.		PASS
24.	The user clicks on the exit to the main menu button in the menu.	The user is brought back to the main menu.	The user is brought back to the main menu.		PASS

25.	The user moves Bit into the trees passing the bridge.	The user is brought back to the NPC on Pseudo Island and Level 1 is made available for them to play.	The user is brought back to the NPC on Pseudo Island and Level 1 is made available for them to play.		PASS

7.4 Navigating through the Overworld

Navigating through the Overworld

Purpose: Verify the user story Overworld (all parts).					
Test Run Information: Tester Name: Lauren Taylor Date(s) of Test: 04/12/22 Location/server being used: Unity website			Prerequisites for this test: None		
			Software Versions: Application: Unity website Browser(used and those COTS supports): N/A Database: N/A Operating System: Windows 10		
			Required Configuration: (browser setup, security or user ID roles) No special setup needed		
Notes and Results					
TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/ INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS or FAIL

1.	Overworld opens after the user plays Level 0.	The overworld opens to Pseudo City with every Level One ready to play, and every other level not shown.	The overworld opens to Pseudo City with every Level One ready to play, and every other level not shown.		PASS
2.	Overworld opens after the user presses the continue button from the main menu.	The user is brought back to the last position they were in the overworld. The levels that have been completed are available for the user to play again. The newest level that was unlocked is available for the user to play. The levels that have not been unlocked are not shown.	The user is brought back to the last position they were in the overworld. The levels that have been completed are available for the user to play again. The newest level that was unlocked is available for the user to play. The levels that have not been unlocked are not shown.		PASS
3.	User runs into a boat.	A dialogue box pops up asking the question “Do you want to travel to (island name)?”	A dialogue box pops up asking the question “Do you want to travel to (island name)?”		PASS

4.	User clicks the start button while talking to a NPC on a boat.	The user will go to the island that is stated in the dialogue box.	The user goes to the island that is stated in the dialogue box.		PASS
5.	User presses enter on the keyboard.	Nothing should happen.	Nothing happens		PASS
6.	User clicks on a non-button object.	Nothing should happen.	Nothing happens		PASS
7.	User uses arrows.	Bit will move in the direction that the arrow the user pressed indicates.	Bit will move in the direction that the arrow the user pressed indicates.		PASS
8.	User presses A/a on the keyboard.	Bit moves to the left on the screen.	Bit moves to the left on the screen.		PASS
9.	User presses S/s on the keyboard.	Bit moves down on the screen.	Bit moves down on the screen.		PASS

10.	User presses W/w on the keyboard.	Bit moves up on the screen.	Bit moves up on the screen.		PASS
11.	User presses D/d on the keyboard.	Bit moves right on the screen.	Bit moves right on the screen.		PASS
12.	User presses M/m on the keyboard.	A map appears on the screen.	A map appears on the screen.		PASS
13.	User presses M/m on the keyboard while the map is already open.	The map will disappear and the screen that was shown before the user pressed M/m will appear.	The map will disappear and the screen that was shown before the user pressed M/m will appear.		PASS
14.	User presses anything else on the keyboard.	Nothing should happen	Nothing happens		PASS
15.	User runs into an NPC.	A dialogue box pops up with the NPC's dialogue.	A dialogue box pops up with the NPC's dialogue.		PASS

16.	User clicks the start button on a NPC dialogue for a level.	The user is brought to the level that is associated with that NPC.	The user is brought to the level that is associated with that NPC.		PASS
17.	While talking to an NPC through the dialogue box, the user presses the continue button when there is no more dialogue to be said.	The button has no effect.	The button has no effect.		PASS

18.	While talking to an NPC through the dialogue box, the user presses the continue button when there is more dialogue to be said.	New dialogue appears in the dialogue box.	New dialogue appears in the dialogue box.		PASS
19.	While talking to an NPC through the dialogue box, the user presses the back button.	The dialogue box disappears from the screen.	The dialogue box disappears from the screen.		PASS
20.	User clicks the "M" button on the top right corner of the screen.	A menu will appear on top of the overworld.	A menu will appear on top of the overworld.		PASS

21.	While in the menu in the overworld, the user presses the back button.	The user is brought back to the overworld.	The user is brought back to the overworld.		PASS
22.	While in the menu in the overworld, the user presses the save button.	The user's progress will be saved.	The user's progress will be saved.		PASS
23.	While in the menu in the overworld, the user presses the options button.	The user is shown the options menu.	The user is shown the options menu.		PASS
24.	While in the menu in the overworld, the user presses the notebook button.	The user is shown their notes and they are allowed to update them.	The user is shown their notes and they are allowed to update them.		PASS

25.	While in the notebook, the user presses the close button.	The user's notes are saved and they are brought back to the menu screen.	The user's notes are saved and they are brought back to the menu screen.		PASS
26.	While in the menu in the overworld, the user presses the exit to title button.	The user is brought to the main menu.	The user is brought to the main menu.		PASS
27.	User scrolls up with the mouse.	The map zooms in.	The map zooms in.		PASS
28.	User scrolls down with the mouse.	The map zooms out.	The map zooms out.		PASS

7.5 Navigating through the Levels in Pseudo City

Navigating through the Levels in Pseudo City

Purpose: Verify the user story Levels in Pseudo City(all parts).	
Test Run Information: Tester Name: Lauren Taylor Date(s) of Test: 04/14/22 Location/server being used: Unity website	Prerequisites for this test: None
	Software Versions: Application: Unity website

			Browser(used and those COTS supports): N/A Database: N/A Operating System: Windows 10		
			Required Configuration: (browser setup, security or user ID roles) No special setup needed		
Notes and Results					
TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/ INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS or FAIL
1.	Level opens.	Opens with the NPC, Bit, objects (certain levels), a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.	Opens with the NPC, Bit, a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.		PASS
2.	User uses arrows.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.		PASS
3.	User uses the space bar.	Bit jumps.	Bit jumps.		PASS

4.	User presses A/a on the keyboard.	Bit moves to the left on the screen.	Bit moves to the left on the screen.		PASS
5.	User presses D/d on the keyboard.	Bit moves right on the screen.	Bit moves right on the screen.		PASS
6.	User clicks the Start Lesson button.	The Duke will start explaining the concept to the user using text bubbles.	The Duke will start explaining the concept to the user using text bubbles.		PASS
7.	User clicks the Continue button before the Start Lesson button.	The dialogue will disappear and the question will appear.	The dialogue is skipped and the question appears.		PASS
8.	User clicks the Continue button after the Start Lesson button while there still is dialogue to be said.	The dialogue box will show the next portion of the dialogue.	The dialogue box will show the next portion of the dialogue.		PASS

9.	User clicks the Continue button after the Start Lesson button when there is no more dialogue to be said.	The dialogue box disappears and the question appears.	The dialogue box disappears and the question appears.		PASS
10.	The user clicks inside the question box.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.		PASS
11.	The user clicks outside of the question box.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.		PASS

12.	User gets the question correct.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen.		PASS
13.	The user is prompted to press E/e on the keyboard during gameplay and an object is next to Bit.	Bit picks up the object next to him. Bit can move around while holding the object.	Bit picks up the object next to him. Bit can move around while holding the object.		PASS
14.	The user is prompted to press E/e on the keyboard during gameplay and Bit is holding an object.	Bit puts down the object in his hands.	Bit puts down the object in his hands.		PASS

15.	User gets the question wrong.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.		PASS
16.	User clicks on a non-button or non-playable object.	Nothing happens.	Nothing happens		PASS
17.	The user clicks on the R (reset) button.	The level is reset fully.	The level is reset fully.		PASS
18.	The user clicks on the N (notebook) button.	The notebook pops up.	The notebook pops up.		PASS

19.	While in the note-book, the user presses the close button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.		PASS
20.	The user clicks the M (menu) button.	The menu pops up over the current level screen.	The menu pops up over the current level screen.		PASS
21.	The user clicks on the back button in the menu.	The menu that is over the level screen disappears.	The menu that is over the level screen disappears.		PASS
22.	The user clicks on the save button in the menu.	The user's progress is saved.	The user's progress is saved.		PASS
23.	The user clicks on the options button in the menu.	The options menu appears over the menu screen.	The options menu appears over the menu screen.		PASS
24.	The user clicks on the notebook button in the menu.	The notebook pops up.	The notebook pops up.		PASS

25.	The user clicks on the exit to world button in the menu.	The user is brought back to the over-world.	The user is brought back to the over-world.		PASS
26.	The user moves Bit into the NPC and the level has been completed.	The user is brought back to the NPC on the island and the next level is made available for them to play.	The user is brought back to the NPC on the island and the next level is made available for them to play.		PASS
27.	The user moves Bit into the NPC and the level has not been completed.	Nothing happens.	Nothing happens.		PASS

7.6 Navigating through the Levels in Python Jungle

Navigating through the Levels in Python Jungle

Purpose: Verify the user story Levels in Python Jungle (all parts).	
Test Run Information: Tester Name: Lauren Taylor Date(s) of Test: 04/13/22 Location/server being used: Unity website	Prerequisites for this test: None
	Software Versions:

			Application: Unity website Browser(used and those COTS supports): N/A Database: N/A Operating System: Windows 10		
			Required Configuration: (browser setup, security or user ID roles) No special setup needed		
Notes and Results					
TEST SCRIPT STEPS/RESULTS					
STEP	TEST STEP/ INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS or FAIL
1.	Level opens.	Opens with the NPC, Bit, objects (certain levels), a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.	Opens with the NPC, Bit, a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.		PASS
2.	User uses arrows.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.		PASS

3.	User uses the space bar.	Bit jumps.	Bit jumps.		PASS
4.	User presses A/a on the keyboard.	Bit moves to the left on the screen.	Bit moves to the left on the screen.		PASS
5.	User presses D/d on the keyboard.	Bit moves right on the screen.	Bit moves right on the screen.		PASS
6.	User clicks the Start Lesson button.	The Duke will start explaining the concept to the user using text bubbles.	The Duke will start explaining the concept to the user using text bubbles.		PASS
7.	User clicks the Continue button before the Start Lesson button.	The dialogue will disappear and the question will appear.	The dialogue is skipped and the question appears.		PASS

8.	User clicks the Continue button after the Start Lesson button while there still is dialogue to be said.	The dialogue box will show the next portion of the dialogue.	The dialogue box will show the next portion of the dialogue.		PASS
9.	User clicks the Continue button after the Start Lesson button when there is no more dialogue to be said.	The dialogue box disappears and the question appears.	The dialogue box disappears and the question appears.		PASS
10.	The user clicks inside the question box.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.		PASS

11.	The user clicks outside of the question box.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.		PASS
12.	User gets the question correct.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen. Some gameplay may require the user to move Bit across the screen.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen. Some gameplay may require the user to move Bit across the screen.		PASS

13.	User gets the question wrong.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.		PASS
14.	User clicks on a non-button or non-playable object.	Nothing happens.	Nothing happens		PASS
15.	The user clicks on the R (reset) button.	The level is reset fully.	The level is reset fully.		PASS
16.	The user clicks on the N (notebook) button.	The notebook pops up.	The notebook pops up.		PASS

17.	While in the note-book, the user presses the close button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.		PASS
18.	The user clicks the M (menu) button.	The menu pops up over the current level screen.	The menu pops up over the current level screen.		PASS
19.	The user clicks on the back button in the menu.	The menu that is over the level screen disappears.	The menu that is over the level screen disappears.		PASS
20.	The user clicks on the save button in the menu.	The user's progress is saved.	The user's progress is saved.		PASS
21.	The user clicks on the options button in the menu.	The options menu appears over the menu screen.	The options menu appears over the menu screen.		PASS
22.	The user clicks on the notebook button in the menu.	The notebook pops up.	The notebook pops up.		PASS

23.	The user clicks on the exit to world button in the menu.	The user is brought back to the over-world.	The user is brought back to the over-world.		PASS
24.	The user clicks the compiler button while their code is correct and there are more questions left for the user to answer.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the next question is shown to the user.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the next question is shown to the user.		PASS

25.	The user clicks the compiler button while their code is correct and there are no more questions left for the user to answer.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the gameplay associated with that level is played out on the screen. Some gameplay may require the user to move Bit across the screen.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the gameplay associated with that level is played out on the screen. Some gameplay may require the user to move Bit across the screen.		PASS
26.	The user clicks the compiler button and their code has a bug.	The error is printed out on the screen for the user to read. The incorrect sound effect is played as well.	The error is printed out on the screen for the user to read. The incorrect sound effect is played as well.		PASS
27.	The user clicks the compiler and creates an infinite loop.	The compiler will wait 7 seconds then terminate the process and inform the user they have created an infinite loop.	The compiler will wait 7 seconds then terminate the process and inform the user they have created an infinite loop.		PASS

28.	The user presses the compile button and there isn't any code written.	Nothing happens.	Nothing happens.		PASS
29.	The user tries to write malicious code to break the game.	The user has only access to what we allow them to and an error will be printed out to the screen informing them they are referring to something that is out of context.	The user has only access to what we allow them to and an error will be printed out to the screen informing them they are referring to something that is out of context.		PASS
30.	The user moves Bit into the NPC and the level has been completed.	The user is brought back to the NPC on the island and the next level is made available for them to play.	The user is brought back to the NPC on the island and the next level is made available for them to play.		PASS
31.	The user moves Bit into the NPC and the level has not been completed.	Nothing happens.	Nothing happens.		PASS

7.7 Navigating through the Levels in Sharp Seas

Navigating through the Levels in Sharp Seas

<table>
<tr><td colspan="6">Purpose: Verify the user story Levels in Sharp Seas(all parts).</td></tr>
<tr><td colspan="3">Test Run Information:
Tester Name: Lauren Taylor
Date(s) of Test: 04/13/22
Location/server being used: Unity website</td><td colspan="3">Prerequisites for this test: None</td></tr>
<tr><td colspan="3"></td><td colspan="3">Software Versions:
Application: Unity website
Browser(used and those COTS supports): N/A
Database: N/A
Operating System: Windows 10</td></tr>
<tr><td colspan="3"></td><td colspan="3">Required Configuration:
(browser setup, security or user ID roles)
No special setup needed</td></tr>
<tr><td colspan="6">Notes and Results</td></tr>
<tr><td colspan="6">TEST SCRIPT STEPS/RESULTS</td></tr>
<tr><td>STEP</td><td>TEST STEP/ INPUT</td><td>EXPECTED RESULTS</td><td>ACTUAL RESULTS</td><td>Requirements Validated</td><td>PASS or FAIL</td></tr>
<tr><td>1.</td><td>Level opens.</td><td>Opens with the NPC, Bit, objects (certain levels), a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.</td><td>Opens with the NPC, Bit, a blank dialogue box, a start lesson button, a continue button, a N button, a M button, and a R button.</td><td></td><td>PASS</td></tr>
</table>

2.	User uses arrows.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.	Bit will move in the direction that the arrow indicates when the left and right arrow is pressed. The up and down arrow won't do anything.		PASS
3.	User uses the space bar.	Bit jumps.	Bit jumps.		PASS
4.	User presses A/a on the keyboard.	Bit moves to the left on the screen.	Bit moves to the left on the screen.		PASS
5.	User presses D/d on the keyboard.	Bit moves right on the screen.	Bit moves right on the screen.		PASS
6.	User clicks the Start Lesson button.	The Duke will start explaining the concept to the user using text bubbles.	The Duke will start explaining the concept to the user using text bubbles.		PASS
7.	User clicks the Continue button before the Start Lesson button.	The dialogue will disappear and the question will appear.	The dialogue is skipped and the question appears.		PASS

8.	User clicks the Continue button after the Start Lesson button while there still is dialogue to be said.	The dialogue box will show the next portion of the dialogue.	The dialogue box will show the next portion of the dialogue.		PASS
9.	User clicks the Continue button after the Start Lesson button when there is no more dialogue to be said.	The dialogue box disappears and the question appears.	The dialogue box disappears and the question appears.		PASS
10.	The user clicks inside the question box.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.	The user will have access to the entire keyboard to answer the question and Bit won't be able to move.		PASS

11.	The user clicks outside of the question box.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.	The user will now be able to move around with the arrow keys and the A and D on the keyboard or jump.		PASS
12.	User gets the question correct.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen. Some gameplay may require the user to move Bit across the screen.	The screen will indicate that the user got the question correct and the correct sound effect will play. The user will either be presented with a new question or the gameplay will appear on the screen. Some gameplay may require the user to move Bit across the screen.		PASS

13.	User gets the question wrong.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.	When there are multiple parts to the question, the screen will state which questions were incorrect and which were correct. When there is only one part, the screen will indicate an incorrect answer. The incorrect sound effect will play as well.		PASS
14.	User clicks on a non-button or non-playable object.	Nothing happens.	Nothing happens		PASS
15.	The user clicks on the R (reset) button.	The level is reset fully.	The level is reset fully.		PASS
16.	The user clicks on the N (notebook) button.	The notebook pops up.	The notebook pops up.		PASS

17.	While in the note-book, the user presses the close button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.	The user's notes are saved and they are brought back to the screen they saw before pressing the notebook button.		PASS
18.	The user clicks the M (menu) button.	The menu pops up over the current level screen.	The menu pops up over the current level screen.		PASS
19.	The user clicks on the back button in the menu.	The menu that is over the level screen disappears.	The menu that is over the level screen disappears.		PASS
20.	The user clicks on the save button in the menu.	The user's progress is saved.	The user's progress is saved.		PASS
21.	The user clicks on the options button in the menu.	The options menu appears over the menu screen.	The options menu appears over the menu screen.		PASS
22.	The user clicks on the notebook button in the menu.	The notebook pops up.	The notebook pops up.		PASS

23.	The user clicks on the exit to world button in the menu.	The user is brought back to the over-world.	The user is brought back to the over-world.		PASS
24.	The user clicks the compiler button while their code is correct and there are more questions left for the user to answer.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the next question is shown to the user.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the next question is shown to the user.		PASS
25.	The user clicks the compiler button while their code is correct and there are no more questions left for the user to answer.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the gameplay associated with that level is played out on the screen.	The code the user has typed is run and the output is printed with an indicator that their answer is correct. The correct sound effect is played and the gameplay associated with that level is played out on the screen.		PASS

26.	The user clicks the compiler button and their code has a bug.	The error is printed out on the screen for the user to read. The incorrect sound effect is played as well.	The error is printed out on the screen for the user to read. The incorrect sound effect is played as well.		PASS
27.	The user clicks the compiler and creates an infinite loop.	The compiler will wait 7 seconds then terminate the process and inform the user they have created an infinite loop.	The compiler will wait 7 seconds then terminate the process and inform the user they have created an infinite loop.		PASS
28.	The user presses the compile button and there isn't any code written.	Nothing happens.	Nothing happens.		PASS
29.	The user tries to write malicious code to break the game.	The user has only access to what we allow them to and an error will be printed out to the screen informing them they are referring to something that is out of context.	The user has only access to what we allow them to and an error will be printed out to the screen informing them they are referring to something that is out of context.		PASS

30.	The user moves Bit into the NPC and the level has been completed.	The user is brought back to the NPC on the island and the next level is made available for them to play.	The user is brought back to the NPC on the island and the next level is made available for them to play.		PASS
31.	The user moves Bit into the NPC and the level has not been completed.	Nothing happens.	Nothing happens.		PASS

Appendix A

Scrum Report

A.1 Sprint Dates - Phase 1

S. No.	Date	Status
1	9/19/22 - 10/03/22	Complete
2	10/03/22 - 10/17/22	Complete
3	10/17/22 - 10/31/22	Complete
4	10/31/22 - 11/14/22	Complete
5	11/14/22 - 11/28/22	Complete

A.2 Sprint Dates - Phase 2

S. No.	Date	Status
1	01/17/23 - 01/31/23	Complete
2	01/31/23 - 02/14/23	Complete
3	02/14/23 - 02/28/23	Complete
4	02/28/23 - 03/14/23	Complete
5	03/14/23 - 04/04/23	Complete

6	04/04/23 - 04/18/23	Started
7	04/18/23 - 05/02/23	Not Started

A.3 User Stories

S. No.	Epic / User Stories	Effort	Priority	Sprint	Owner
E1	**Animations**	**8**	**high**	**N/A**	**N/A**
U1.1	Overworld Player	5		1	Drake
U1.2	Overworld Misc.	5		5	Daphne
U1.3	Level Robot	9		7	Drake
U1.4	Level Tools	8		8	Drake
U1.5	Level Objects	5		9	Drake
E2	**Audio**	**1**	**Low**	**N/A**	**N/A**
U2.1	Menu music	3		9	Lauren
U2.2	Overworld music	5		9	Lauren
U2.3	Level music (type 1)	4		9	Lauren
U2.4	Level music (type 2)	4		9	Lauren
U2.5	Level music (type 3)	4		9	Lauren
U2.6	Boss theme	4		10	Lauren
U2.7	SFX	6		10	Lauren
E3	**Game Text**	**3**	**high**	**N/A**	**N/A**
U3.1	NPC Dialogue	4		3	Lauren
U3.2	Lesson Dialogue (part 1)	8		2	Lauren
U3.3	Lesson Dialogue (part 2)	8		3	Lauren
U3.4	Lesson Dialogue (part 3)	8		6	Lauren
U3.5	Lesson Dialogue (part 4)	8		7	Lauren
U3.6	Lesson Dialogue (part 5)	8		8	Lauren
U3.7	Lesson Dialogue (part 6)	8		11	Lauren

U3.8	Lesson Dialogue (part 7)	8		12	Lauren
U3.9	Manual/ReadMe	7		5	Lauren
E4	**Game Functions**	**1**	**med**	**N/A**	**N/A**
U4.1	Master Sound Slider	3		6	Drake
U4.2	Music Slider/Mute	3		6	Drake
U4.3	SFX Slider/Mute	3		6	Drake
U4.4	Graphics Options	6		6	Drake
U4.5	Save functionality	4		6	Dexter
U4.6	Load functionality	4		6	Dexter
U4.7	Maps for Island	4		7	Daphne
U4.8	Notebook	4		7	Dexter
U4.9	Last Minute Items (Phase 1)	7		5	All
U4.10	Last Minute Items (Phase 2)	7		12	All
E5	**Graphics**	**9**	**high**	**N/A**	**N/A**
U5.1	Executable Icon	1		5	Drake, Dexter
U5.2	Menu UI	4		3	Drake
U5.3	Overworld UI	6		4	Daphne
U5.4	Level UI	6		9	Drake
U5.5	World Tileset	6		2	Daphne
U5.6	Area 1 Level Tileset	4		2	Drake
U5.7	Area 1 Level Backgrounds	4		4	Drake
U5.8	Area 2 Level Tileset	4		2,3	Drake
U5.9	Area 2 Level Backgrounds	4		5	Drake
U5.10	Area 3 Level Tileset	4		2	Drake
U5.11	Area 3 Level Backgrounds	4		9	Drake
U5.12	Overworld Player	3		5	Drake
U5.13	Overworld NPCs	4		1	Daphne
U5.14	Overworld Misc. (Part 1)	5		3	Daphne
U5.15	Overworld Misc. (Part 2)	5		4	Daphne

U5.16	Overworld Misc. (Part 3)	5		7	Daphne
U5.17	Island 1	5		3	Daphne
U5.18	Island 2	5		6	Daphne
U5.19	Island 3	5		6	Daphne
U5.20	Level Robot	5		1	Drake
U5.21	Level Tools	5		8	Drake
U5.22	Level Objects	8		7	Drake
E6	**Level Design**	**5**	**high**	**N/A**	**N/A**
U6.1	Lesson Discovery	9		1	Dexter, Lauren, Travis
U6.2	Gameplay Feasibility	8		1	Dexter, Lauren, Travis
U6.3	C# Parser and Object Movement	9		2	Dexter
U6.4	Python Parser and Object Movement	9		3	Dexter
U6.5	World 1, Lesson 1, Variables	6		2	Travis
U6.6	World 1, Lesson 2, Assignment	6		2	Travis
U6.7	World 1, Lesson 3, Operators	6		2	Travis
U6.8	World 1, Lesson 4, Logic Operators	6		3	Travis
U6.9	World 1, Lesson 5, Relation Operators	6		3	Travis
U6.10	World 1, Lesson 6, If statements	6		7	Travis
U6.11	World 1, Lesson 7, If Else statements	6		7	Travis
U6.12	World 1, Lesson 8, Nested If else statements	6		7	Travis
U6.13	World 1, Lesson 9, Switch statements	6		8	Travis
U6.14	World 1, Lesson 10, Input / Output	6		8	Travis
U6.15	World 1, Lesson 11, While Loops	6		8	Travis
U6.16	World 1, Lesson 12, Do while loops	6		8	Travis
U6.17	World 1, Lesson 13, For loops	6		9	Travis
U6.18	World 1, Lesson 14, 1D Arrays	6		9	Travis

U6.19	World 1, Lesson 15, 2D Arrays	6		9	Travis
U6.20	World 1, Lesson 16, Function Declaration	6		10	Travis
U6.21	World 1, Lesson 17, Pointers	6		10	Travis
U6.22	World 2, Lesson 1, Variables	6		4	Travis
U6.23	World 2, Lesson 2, Assignment	6		4	Travis
U6.24	World 2, Lesson 3, Operators	6		8	Daphne
U6.25	World 2, Lesson 4, Logic Operators	6		8	Daphne
U6.26	World 2, Lesson 5, Relation Operators	6		8	Daphne
U6.27	World 2, Lesson 6, If statements	6		9	Daphne
U6.28	World 2, Lesson 7, If Else statements	6		9	Daphne
U6.29	World 2, Lesson 8, Nested If else statements	6		9	Daphne
U6.30	World 2, Lesson 9, Switch statements	6		10	Daphne
U6.31	World 2, Lesson 10, Input/Output	6		10	Daphne
U6.32	World 2, Lesson 11, While Loops	6		10	Daphne
U6.33	World 2, Lesson 12, For loops	6		11	Travis
U6.34	World 2, Lesson 13, 1D Arrays	6		11	Travis
U6.35	World 2, Lesson 14, 2D Arrays	6		11	Travis
U6.36	World 2, Lesson 17, Function call/return	6		4,5	Dexter
U6.37	World 3, Lesson 1, Variables	6		4	Travis
U6.38	World 3, Lesson 2, Assignment	6		4	Travis
U6.39	World 3, Lesson 3, Operators	6		8	Dexter
U6.40	World 3, Lesson 4, Logic Operators	6		8	Dexter
U6.41	World 3, Lesson 5, Relation Operators	6		9	Dexter
U6.42	World 3, Lesson 6, If statements	6		9	Dexter
U6.43	World 3, Lesson 7, If Else statements	6		10	Dexter
U6.44	World 3, Lesson 8, Nested If else statements	6		10	Dexter
U6.45	World 3, Lesson 9, Switch statements	6		11	Dexter
U6.46	World 3, Lesson 10, Input/Output	6		11	Dexter
U6.47	World 3, Lesson 11, While Loops	6		11	Dexter
U6.48	World 3, Lesson 12, Do while loops	6		11	Dexter

U6.49	World 3, Lesson 13, For loops	6		11	Dexter
U6.50	World 3, Lesson 14, 1D Arrays	6		12	Dexter
U6.51	World 3, Lesson 15, 2D Arrays	6		12	Dexter
U6.52	World 3, Lesson 12, While Loops	6		4,5	Dexter
U6.53	World 3, Lesson 17, Pointers	6		12	Dexter
U6.54	Rework of Past Levels	6		11	Daphne
U6.55	Assign Gameplay to Levels	6		6	Travis
E7	**UI**	**7**	**high**	**N/A**	**N/A**
U7.1	Main Menu	6		3	Travis
U7.2	Overworld	6		6	Daphne
U7.3	Levels	6		4	Dexter, Drake
U7.4	Opening Credits	2		10	Drake
U7.5	Overlay Menus	5		4	Drake
U7.6	Error Messages	4		10	Dexter
U7.7	Dialog Overlay	5		4,5	Daphne, Lauren

Appendix B

Glossary

Glossary

Word	Definition	Page No.
NPC	Non-player character	4, 12, 14
IEEE	Institute of Electrical and Electronics Engineers	1
SRS	System Requirements Specification Documents	1
SFX	Sound effects	14
Slider	an object that can be moved by the mouse when it is grabbed, and changes qualities as it goes towards one of the edges.	7, 10, 11
Boolean, bool	a value that is set to true or false or set to 0 or 1	22, 34