

Software Requirements Specification (SRS)
Pixel Path: The Road to Code

Team: 1

Authors: Joseph Beausoleil, Kristina Russell, Dylan Silk, Kevin Chen, Juel Teixeira Baptista

Customer: Middle School grade kids

Instructor: Dr James Daly

1 Introduction

Our SRS document will give an overview of the project as a whole and the work that we have done since the beginning. This includes all of our diagrams that we have. The diagrams that we have are much more fleshed out in this document with much more descriptions on what each part means.

- Section 1: This is a general introduction to all of the topics for the SRS document.
- Section 2: This is a description of the project and the software that we used to make it.
- Section 3: This is a list of all of our requirements that we have to go into our project.
- Section 4: Includes our use case diagram, class diagram, and sequence diagrams. These provide descriptions for each topic on each diagram.
- Section 5: This is information about or prototype that will be submitted.
- Section 6: This is a list of references for our project.
- Section 7: This is the point of contact.

1.1 Purpose

The purpose of the SRS is to identify and describe the purpose of the software and its features. The SRS document explains how the system is going to behave and the documents involved in the planning process.

Our intended audience are kids in grades 4th-8th. Some of those kids will not know how to code or use a computer and have yet to learn a lot of concepts in general. From our point of view, we do not expect our audience to have the same attention span as a college student. Therefore, our teaching method will be done through games. Kids love games, thus prolonging their attention span.

1.2 Scope

Our software, Pixel Path: The Road to Code, will be educational game driven software. The game's design will be mini-games directed towards children in about middle school. The player will progress through a map of a multitude of levels where a user would complete many different kinds of activities. These include things like sorting different variables, playing Asteroids, or finishing lines of code. With this style of game it would make it more accessible for kids of all ages to be able to play up to whatever level they are comfortable with. Meaning anyone would be able to play this game up until the point that they naturally can't anymore.

1.3 Definitions, acronyms, and abbreviations

Game map - Overworld view of different paths containing multiple levels

Path - A series of levels that focus on one specific concept

Concept - A fundamental idea that programmers should understand. Consists of several smaller topics.

Level - One space on the game map. Could contain mini-games, lesson, or boss level

Lesson - Teaches a topic meant to prepare the user to complete the mini-games. A lesson is made up of multiple Levels

Mini-game - An interactive activity where the user will be able to perform actions such as matching terms, selecting an option from several choices, and moving their character.

The mini-games available are Asteroids, Whack a mole, Puzzle mini-game, and sorting mini-game. Upon completion, it awards the user with key-fragments

Key-fragment - Currency used to obtain customizable cosmetic items for the character. Also will unlock new levels when enough key-fragments are collected.

Boss level - A level at the end of a lesson that combines all of the topics that were talked about into one harder level.

1.4 Organization

The SRS is a container for all of the information regarding our game and how it works. Section 2 will contain general info about the game. Section 3 will have the requirements for how the game is supposed to run and include major functionality. Section 4 contains the UML diagrams with their descriptions of what it means and how things work. Section 5 contains information about the prototype that has been submitted. And finally section 6 will contain any sources that we used for our projects.

2 Overall Description

This section of the SRS is going to be a description of the project. The Product Perspective will be a description of what the project is going to look like and what the inspirations were to make it. The Product Functions is going to be a description for how the systems work and work together. The User Characteristics section will go over what we expect out of a user, and what we want them to be able to do for our game. The Constraints section will go over some constraints that might occur while playing or interacting with the game. The Assumptions and Dependencies section will go over the assumptions that we as the developers made in order to create this software. The Apportioning of Requirements section will go over requirements made by the client but have not been implemented yet in the current system.

2.1 Product Perspective

Learning to code can be a difficult thing for students of all ages. Programming is a new concept that contains many unique ideas and ways of thinking. If we are able to implement a good way to learn some of the basic concepts of programming. While being able to teach kids at a young age, it could make learning programming a lot easier down the line.



Figure 2.1.1: A screenshot of the start screen of the game. Artwork done by Kristina Russell.

Our game, Pixel Path: The Road to Code, is aimed at young kids to be able to start the process of learning to code and with the aim of getting them excited and involved with programming. The main goal of this is of course to teach them very basic concepts like variables and conditions, in fun and interactive ways. The interface offers a

map of different games and activities you can complete in order to advance to harder topics.



Figure 2.1.2: A screenshot of the spawn point on the game map

The hardware limitations that come along with this game are the need for at least 4 GBs of RAM and a Windows operating system. The software limitations are that the player should only be able to control, customize the character, select, and complete the lesson.

2.2 Product Functions

The major features of the game will include being able to customize your character, play different kinds of mini-games in order to advance and unlock new paths. The user should be able to go from the start screen to the customization page (which then leads to the game map), the options menu, or exit the game. From the game map, the user should be able to select and get information on each level, play any unlocked level, or pause the game. The pause screen should have a resume, options, and exit features. The mini-games include matching, fill in the blank, asteroids, puzzle slides, and more. All of the mini-games can be paused at any time and will show the pause screen. There is also a return feature in every mini-game that will return the user to the map.

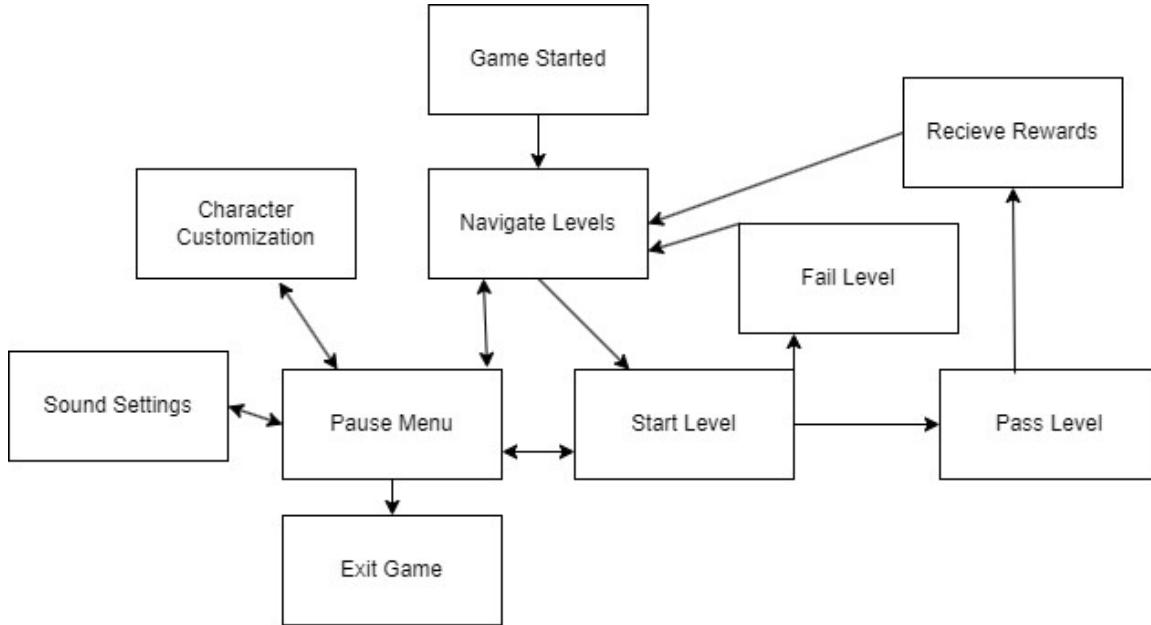


Figure 2.2.1: A High-Level Goal Diagram

2.3 User Characteristics

The target audience is younger kids in grades 4th-8th. The game is to be designed in a way that you start with easy concepts, like variables or conditions, and slowly make your way to harder concepts like stacks or recursion. The user will be able to go at their own pace and review when they need to. Because of this the game could be played by almost anyone.

The user doesn't need much prior knowledge in order to be successful in this game. As long as they know how to use a mouse and keyboard with basic knowledge of how computers work, such as navigating the web or installing programs, they can find success with our game.

2.4 Constraints

The constraints for this project consist mostly of constraints from the Unity Engine. These constraints include needing to code it in C#, or JavaScript which is a language that most of our group members haven't worked with before. We also needed to learn how the engine itself works and how to implement the plans that we had. Overall, though, there were no major constraints on what we were able to implement and most of what we wanted to implement is in the game. The biggest constraint we had was the time. Anything that was put in the requirement sheet but isn't in the game is due to the time constraint that we had while making the game.

2.5 Assumptions and Dependencies

We assume that the user is running the game on a computer with 4GBs of RAM and has a Windows operating system. These are the fundamental requirements to run the game. It's not going to be a super intensive experience on the computer and the game itself will be pretty simple. The user will only be able to input basic movement with the WASD keys along with clicking on the mouse. The user is going to be able to navigate the map and select levels.

2.6 Apportioning of Requirements

As of the first prototype, we have implemented the main map screen, the pause menu, a couple of games, and basic customization features. The map itself is designed to be simple and easy to navigate with the controls placed in large font on the map itself for easy access. There are three levels included in the prototype. The first level is a game where you take values and you sort them into the correct variable bins. The second level is a conditional game where you take a customer's order and break it down into an if else statement. And the third level is going to teach loops where they need to measure how many planks are needed to cross a bridge and create a for loop for it. The pause menu is also implemented where you can access sound settings, save game, and character customization. Right now the only way that you are able to customize your character is by changing its color.

3 Specific Requirements

1. Game Map
 - 1.1. Maps will be set up so there are multiple paths to choose.
 - 1.2. The player will be able to:
 - 1.2.1. Navigate the map using the WASD keys
 - 1.2.2. Select an unlocked level by pressing ‘e’ on its tile
 - 1.2.2.1. This brings them to a mini-game focused on the particular topic
 - 1.2.3. View future levels and their topics
 - 1.2.4. View what is required to unlock the boss levels
 - 1.2.5. Go back and replay previous lessons
 - 1.3. The Escape key will bring up the Pause Screen (See Pause screen section)
 - 1.4. Game is organized by concept
 - 1.4.1. Variables
 - 1.4.2. If / Conditional statements
 - 1.4.3. Loop
 - 1.4.4. Functions
 - 1.4.5. What libraries are used for (later grades)
 - 1.4.6. Stack/Queue
 - 1.4.7. Recursion
2. Mini-games
 - 2.1. Mini-games will be played in order to complete each section in a level.
 - 2.1.1. Coins will be rewarded for completing levels, these coins can be used to customize characters and progress through levels
 - 2.2. The mini-game itself will vary depending on the topic at hand.
 - 2.2.1. Asteroids
 - 2.2.2. Puzzle slide
 - 2.2.3. Sorting Mini-game (strings, ints etc)
 - 2.3. The player will be able to interact with games assets using the mouse and the WASD keys along with the arrow keys
 - 2.4. Once the mini-game has been finished:
 - 2.4.1. An autosave will occur
 - 2.4.2. If won
 - 2.4.2.1. The lesson will be marked as completed on the map and the path will update.
 - 2.4.3. If failed
 - 2.4.3.1. The lesson can be redone in order to move on
 - 2.4.3.2. Can potentially offer supplemental material to aid in understanding
 - 2.5. Mini-games will be able to be replayed after they are completed
3. Start Game Screen

- 3.1. Player can choose to:
 - 3.1.1. begin the game via the start button
 - 3.1.2. Change their sound settings via the options button
 - 3.1.3. Exit the game
4. Boss Level
 - 4.1. Unlockable once corresponding pieces have been collected via winning levels
 - 4.2. Will show what topics are relevant
 - 4.3. Once the boss level is cleared, a new section of the map will unlock
5. Pause Screen
 - 5.1. Edit sound settings in the options screen
 - 5.2. Save game to be continued later
 - 5.3. Exit game
6. Character selection
 - 6.1. Pick a player model
 - 6.2. Change color scheme of the character
 - 6.3. Add/remove any accessories they have unlocked
 - 6.4. Name their character
7. Users can adjust the sound settings from the options screen and return to the previous screen.

4 Modeling Requirements

These use cases are a description of how our game works and how each process is able to interact with one another. Each case represents an ability that the user has when they are playing a game, and the pathway that the program itself takes while it is running.

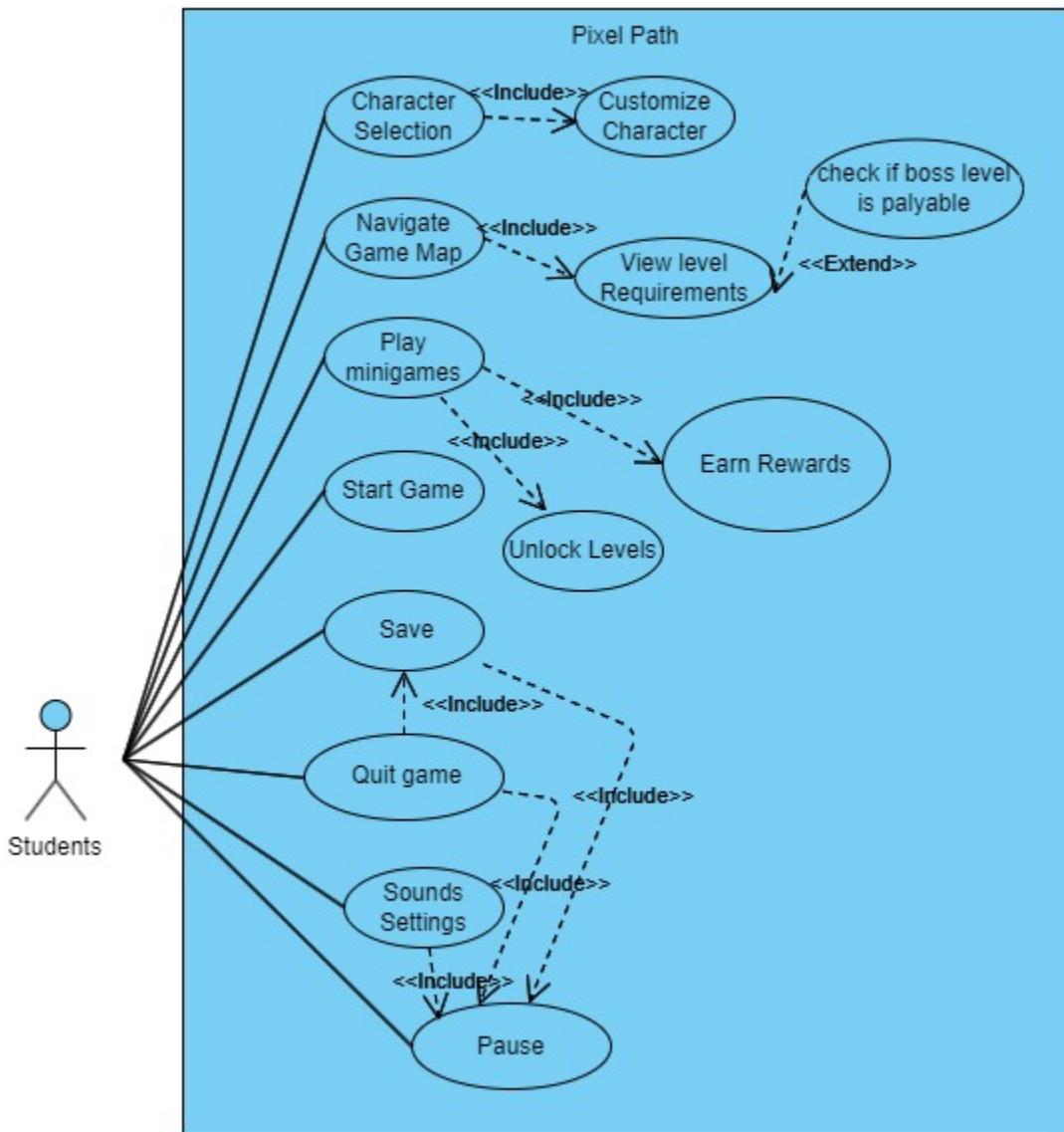


Figure 4.1: A screenshot of the use case diagram

Use Case Name:	Start Game
Actors:	Student
Description:	This is the user starting the game and running the program. This will open the startup screen
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 3
Uses cases:	This is used whenever you want to open and play the game. The game must be started before you can play

Use Case Name:	Character Selection
Actors:	Student
Description:	This happens when you start a new game. you are able to select from a multitude of different characters for the game.
Type:	Primary
Includes:	Character Customization
Extends:	None
Cross-refs:	Requirement 6
Uses cases:	This is used whenever you want to start a new game. the player is able to select which character they would like to play as.

Use Case Name:	Character Customization
Actors:	Student
Description:	This is a customization feature that will allow you to customize your character that you have already selected with accessories that you have already unlocked.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirements 6.1.2 - 6.1.4

Uses cases:	When the user unlocks new accessories for their character they can go to the pause menu and to the character selection screen in order to add or get rid of things off of their character.
-------------	--

Use Case Name:	Navigate the Game Map
Actors:	Student
Description:	The user is able to navigate all around the game map and to multiple anywhere on the map
Type:	Primary
Includes:	View Level Requirements
Extends:	None
Cross-refs:	Requirement 1.2.1
Uses cases:	The user wants to make it from lesson one to lesson two. They are able to control the character using the WASD keys to get there.

Use Case Name:	View Level Requirements
Actors:	Student, system
Description:	The User is able to view what is required to view any given lesson at any given time.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	Requirements 1.2.3, and 1.2.4
Uses cases:	The user wants to move to a new topic but is unsure what is needed inorder to go there. They are able to navigate to the lesson itself and view the requirements to be able to access the level.

Use Case Name:	Check if Boss is Playable
Actors:	System
Description:	The system must be constantly checking if the boss level is playable for the user, if it is then it must be unlocked.
Type:	Primary

Includes:	None
Extends:	View Level Requirements
Cross-refs:	Requirements 4
Uses cases:	When the user collects the required material to unlock the boss then the system must check that and then unlock the level for them.

Use Case Name:	Play Mini-game
Actors:	Student
Description:	The student is able to go and play a mini-game that is currently unlocked, they are also able to replay any mini-game that they want to.
Type:	primary
Includes:	Unlock level
Extends:	Earn Rewards
Cross-refs:	Requirements 1.1, 2
Uses cases:	The user is able to navigate themselves towards any game they would like, if the game is unlocked then they are able to play it as many times as they would like to.

Use Case Name:	Earn Rewards
Actors:	Student
Description:	Once a game has been completed for the first time the User will receive rewards, including currency and material needed for the boss at the end of the level.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	Requirements 2.5.2
Uses cases:	The user will collect the currency in order to purchase accessories for the character, and can collect material in order to unlock future bosses

Use Case Name:	Unlock level
----------------	--------------

Actors:	System
Description:	When a lesson is complete the system will automatically go and unlock the next lesson, unless the next lesson is a boss then it will use that process instead
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	Requirements 1.1
Uses cases:	The system needs to unlock these levels for the user so that they are able to continue on their journey.

Use Case Name:	Pause Game
Actors:	Student
Description:	The User is able to pause the game at any point on the map screen or any lesson or boss. this will bring up a button to access saving the game, options screen, or exit game.
Type:	Primary
Includes:	None
Extends:	Quit, Save, Settings
Cross-refs:	Requirements 5
Uses cases:	The user needs to be able to pause the game and access these settings at any point during their gaming session.

Use Case Name:	Quit
Actors:	Student
Description:	The option to quit the game is accessible through the pause menu. this option will auto save the game and close the application
Type:	Primary
Includes:	Save
Extends:	None
Cross-refs:	Requirements 5.3

Uses cases:	The user can quit the game at any time, the game will automatically save and the application will be exited
-------------	---

Use Case Name:	Save
Actors:	Student
Description:	The User is able to save the current point that they are at in the game to be able to be accessed later on.
Type:	Primary
Includes:	None
Extends:	None
Cross-ref:	Requirements 5.2
Uses cases:	The user needs to be able to save their progress in order to access it later.

Use Case Name:	View Level Requirements
Actors:	Student, system
Description:	The User is able to view what is required to view any given lesson at any given time.
Type:	Secondary
Includes:	None
Extends:	Check if Boss is Playable
Cross-ref:	Requirements 1.2.3, and 1.2.4
Uses cases:	The user wants to move to a new topic but is unsure what is needed in order to go there. They are able to navigate to the lesson itself and view the requirements to be able to access the level

Use Case Name:	Options Screen
Actors:	Student, system
Description:	The User is able to edit settings like the volume of the game
Type:	Primary
Includes:	None
Extends:	None

Cross-references:	Requirements 7
Uses cases:	The user is able to change the volume of certain objects and the main game through the options menu.

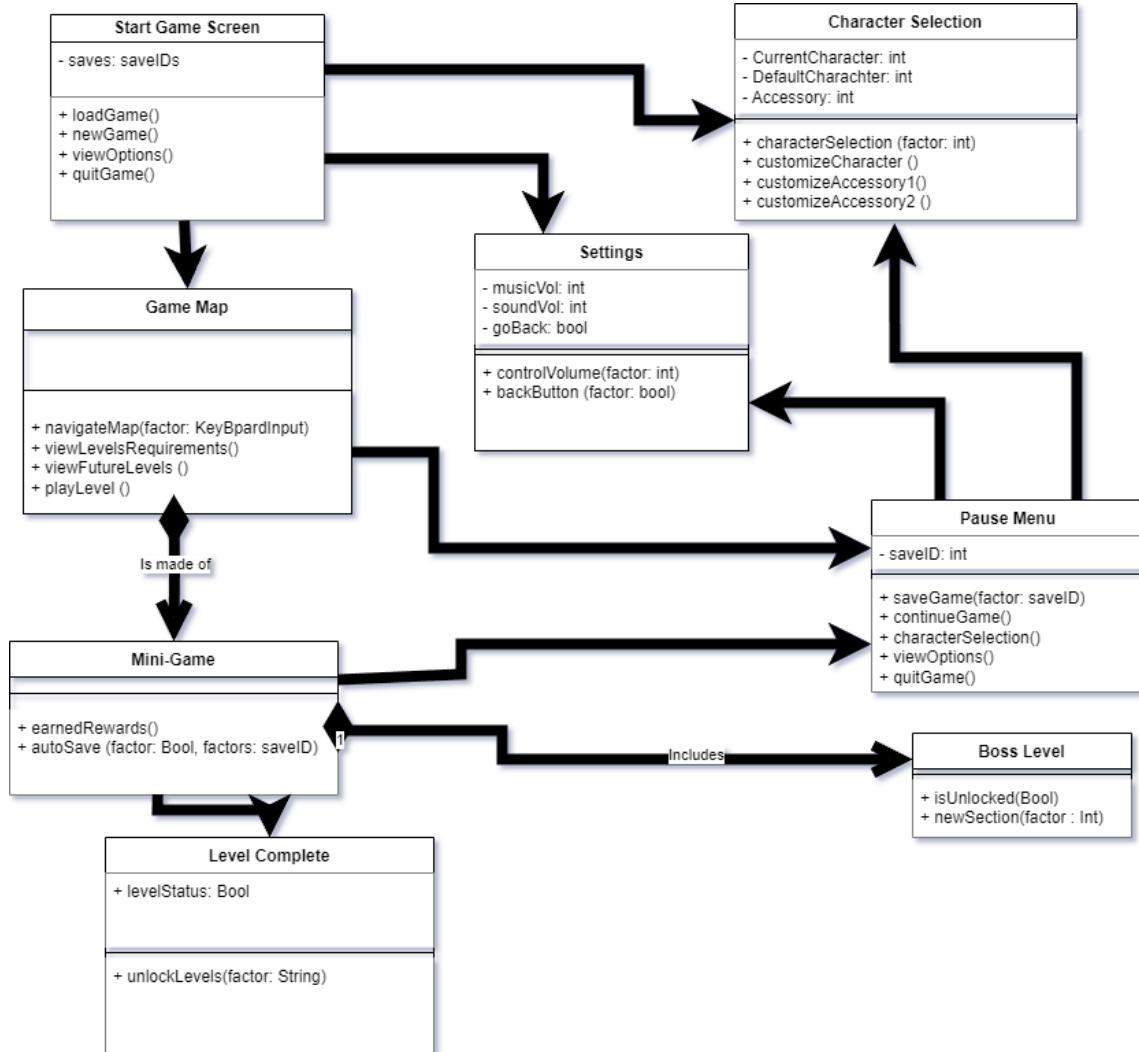


Figure 4.2: A screenshot of the class diagram

The class diagram above is a description of all the classes that we feel is necessary for the system to be able to accomplish all of the use cases that we described above. Each class contains a list of local variables in functions that are vital for the function as a whole.

Start Game Screen	
-saves:saveIDs	The start screen is able to access previously saved games so save files get passed in if it's needed.
+loadGame(factor: saveID)	This function would take in the saveID and load the saved game for the user
+newGame()	This function starts a brand new game if selected and it would bring the user to the start game screen
+viewOptions()	This function would bring the user to the options screen in order to edit the settings
+quitGame()	This function would quit the game when it is used.
Class description: This class is the starting screen when the game is opened. it will display options to load, and save games, also to edit the settings and to quit the game	

Game Map	
+navigateMap()	The User uses the WASD keys in order to navigate around the map and get to different lessons in the level.
+viewLevelRequirements()	This will display the current requirements needed for the selected level.
+viewFutureLevels()	This function would allow the user to be able to preview each future lesson that they would have to complete.
+playLevel()	This function will allow the user to start the mini-game that currently selected
Class description: This class is the game map itself. the user can navigate around the map and interact with the lessons as they go around. Game Map is made of Mini-Games	

Mini-Games

+earnRewards()	This function decides what rewards you receive upon completion of the level.
+autoSave(factor:bool, factors: SaveID)	This function will auto save the game once the mini-game is complete
<p>Class description: This class is the mini-game class that controls what happens after a mini-game is completed for the player. Class includes Boss Level</p>	

Boss Level	
+isUnlocked(factor:Bool)	This function checks if the requirements to enter the boss level have been met.
+newSection	This function unlocks a new path after the boss has been defeated.
<p>Class description: This class is a special level that features a boss fight at the end of a path and unlocks a new path once completed.</p>	

Level Complete	
+levelStatus: Bool	Whether the mini-game was completed successfully or not.
+unlockLevels(factor:String)	This function unlocks the next level if the current one is completed.
<p>Class description: This class checks if the mini-game was completed and allows the user to progress to the next level.</p>	

Pause Menu	
-saveID: int	ID of current game state to be stored.
+saveGame()	This function saves the current game state to one of the save slots.
+continueGame()	This function closes the pause menu and returns the user to the game state before

	pausing.
+characterSelection()	This function allows the user to change to a different character.
+viewOptions()	This function would bring the user to the options screen in order to edit the settings.
+quitGame()	This function would exit the user back to the game map and autosave the current game state.
Class description: This class is a menu that allows the player to stop in the middle of a mini-game to save, continue, or quit the game in the current state and also allows the player to access the settings.	

Settings	
-musicVol: int	Volume of background music.
-soundVol: int	Volume of other sound effects.
-goBack: bool	Whether the player wants to return to the previous menu or not.
+controlVolume(factor:int)	This function allows the user to increase or decrease the volume with a slider.
+backButton(factor:bool)	This function returns the user to the previous screen, either the start game screen or pause menu.
Class description: This class is the settings menu that allows the user to set the volume to the desired level and applies the volume setting to the game.	

Character Selection	
-CurrentCharacter:int	ID for current character model
-DefaultCharacter:int	ID for default character model
-Accessory: int	Additional cosmetics applied to the base character model.

+characterSelection(factor:int)	This function lets the user select which character model to use with arrow buttons and sets it as the current character.
+customizeCharacter()	This function allows the user to view available accessories.
+customizeAccessory1()	This function allows the user to set the accessory in slot 1.
+customizeAccessory2()	This function allows the user to set the accessory in slot 2.
Class description: This class controls the character that will be displayed and allows the user to change the character or equip accessories.	

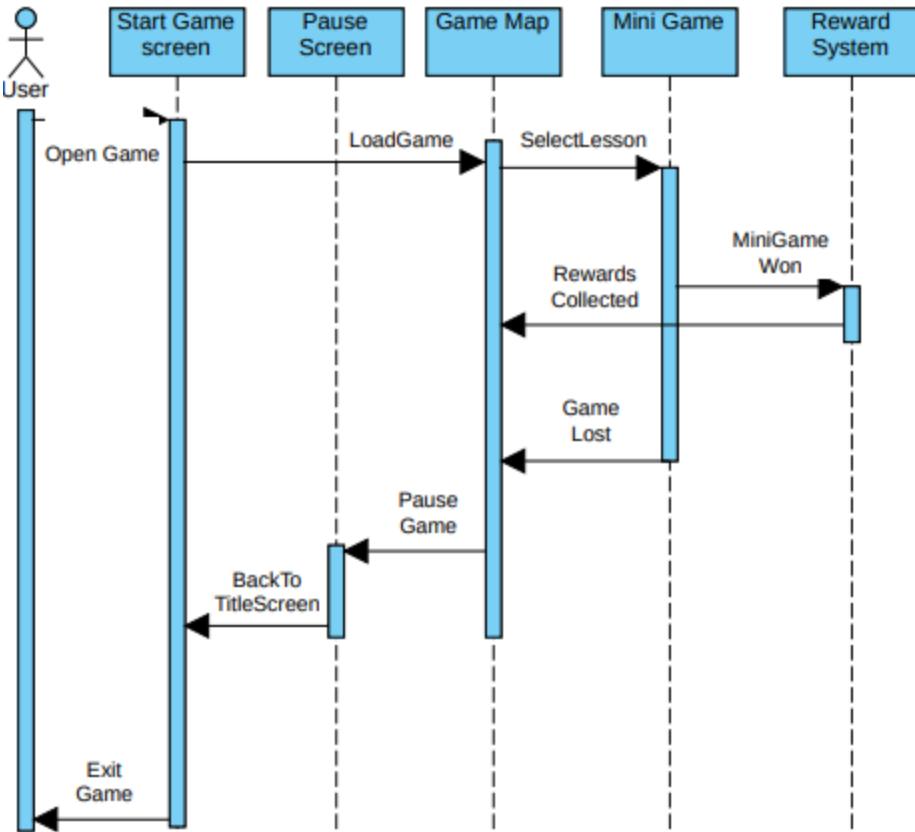


Figure 4.3: A screenshot of the sequence diagram

This Sequence diagram shows the path a user could take if they wanted to play a level in our game. They would start by launching the game where they would be brought to the start screen of the game. Then they would select the new game option on the screen. From there it will bring them to the game map which can be navigated using the WASD keys on the keyboard where they find their first level. Once they decide to play that level they will go through it. If they won the game they will collect their rewards then be sent back to the main map. If they lost then they would get sent back to the map. Then they press escape to pause the game, select back to the main menu then exit the game.

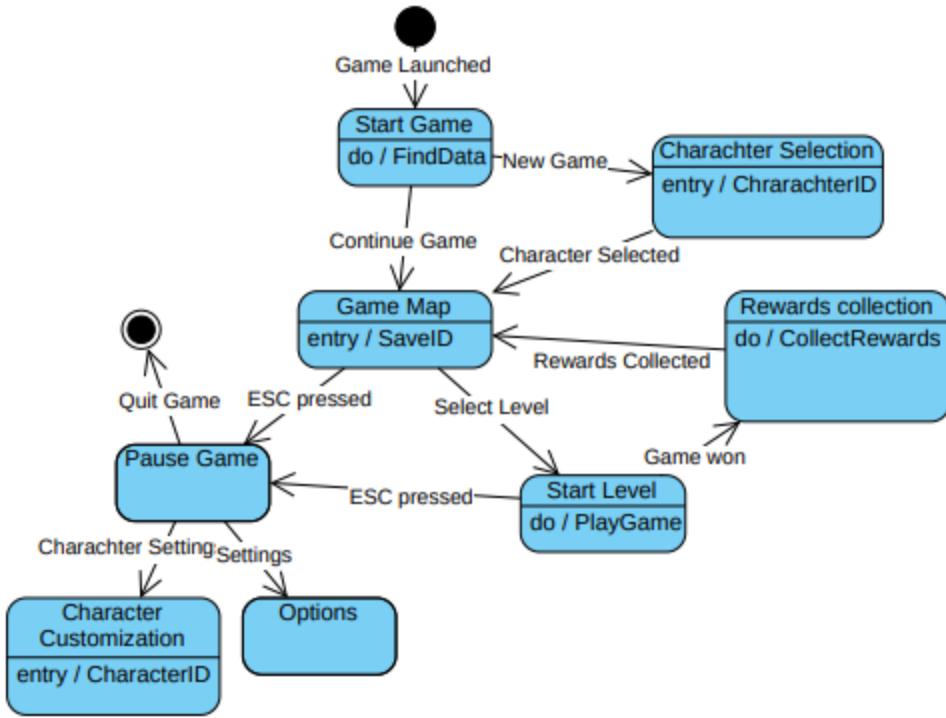


Figure 4.4: State Diagram for Pixel Path

This state diagram represents the possible states of our game and how they interact. First on launch you get put to the start screen. From the start screen you can load the game or start a new game. If you start a new game then you get sent to select your character then you get sent to the map after it has been selected. If you decide to load the game then you will spawn in at your most recent save point. From there you can play a level. You either win or lose the level, if you win you get your rewards then get sent back to the map to keep playing. If you lose then you will just be sent to the map to try again. From either the game map or a mini game you can enter the pause menu by hitting the ESC key on the keyboard. Once it has been brought up you are able to go to options, character customization, or exit games. With character customizations you are able to change the color of your character. Once the game has exited it will close completely.

5 Prototype

Our prototype will be able to show the user navigating through the map. By default only the first path is unlocked. The user can enter levels by interacting with level tiles. A placeholder level is shown in the prototype which will be replaced with a mini-game. The user will be able to quit, save the game, or open the settings. From there the user would be able to do things such as manage volume or set the display to fullscreen or windowed.

5.1 How to Run Prototype

The prototype of the game needs to be run on a Windows computer with 4 GBs of RAM. The game is not super intensive and should be able to run on almost any computer that you have available. It is in the form of a .exe that the user will need to run in order to start the game. The game can be found on our website and can be downloaded from there to be played.

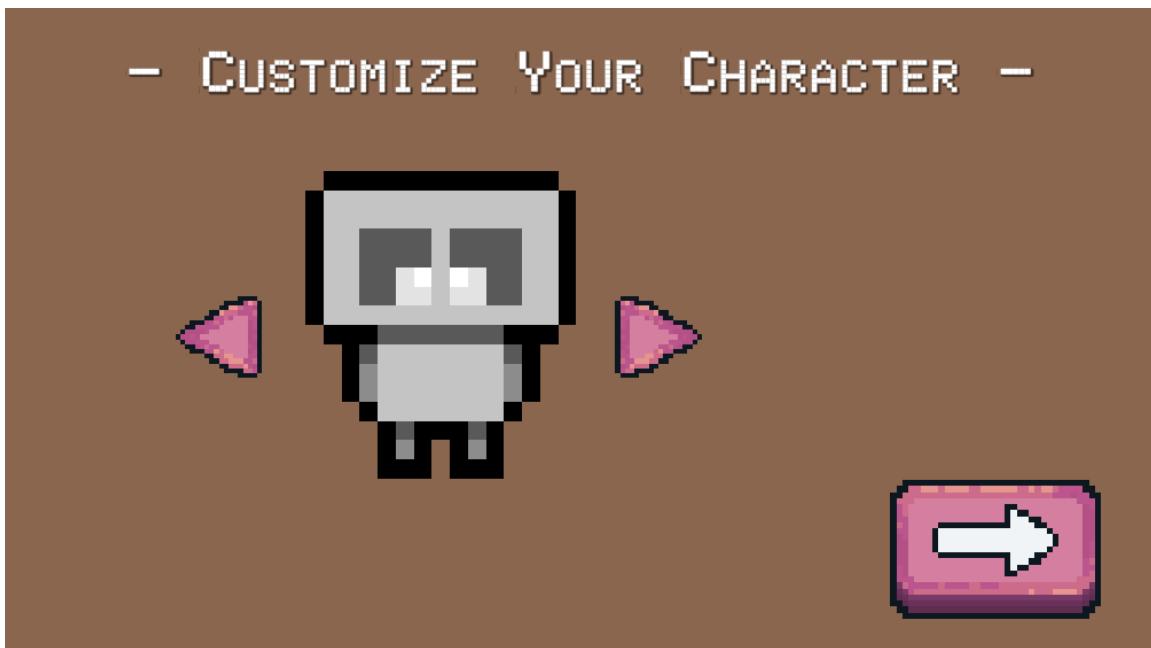
5.2 Sample Scenarios

A student wants to learn a specific programming concept such as variables. The student will click Start from the start screen and customize their character. The student will then be shown an open world map with the character. The student will navigate their character using the WASD keys to the Variables path. In the prototype the path is represented as placeholder level 1. Moving along the path, the student will encounter level tiles. The student will move the character to an available level space that they want to play and press ‘e’ to start the level.

The student is shown this start screen upon launching the program.



After pressing start, the user will customize their character.



After customizing their character, the student will be brought to the world map that shows a character controllable with WASD.



The student will move their character along the path and encounter several levels along the way.



Levels are represented by red blocks. The student will move their character over the block and press 'e' to interact with it. Pressing 'e' will start the level if it is unlocked.



The student will be able to start a lesson or play a mini-game upon starting the level. Currently a placeholder level is shown below.



6 References

- [1] “Coding Games for Kids by Kidlo: Learn Programming Online.” Kidlocoding.Com, IDZ Digital, www.kidlocoding.com/. Accessed 17 Nov. 2023.
- [2] Technologies, Unity. “Creating a 2D Game.” Unity, Unity, docs.unity3d.com/Manual/Quickstart2DCreate.html. Accessed 17 Nov. 2023.
- [3] Unity. “Beginner Scripting.” Unity Learn, learn.unity.com/project/beginner-gameplay-scripting?tab=overview. Accessed 17 Nov. 2023.
- [4] Kottke, Jason Aleksandr. “Silkscreen.” Silkscreen Font, Dafont, 2005, www.dafont.com/silkscreen.font.
- [5] <https://dylans1lk.github.io/WebsiteSkeletonSWE/index.html>

Note about game assets: All game assets were created by Kristina unless otherwise noted

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.