Software Requirements Specification (SRS) v2.0

Mini-Golf Mania

Team: Team 5

Authors:
Austin Nguyen
Braden Maillet
Edward Alderman
Christopher Lambert

Customer:
Grades 6-8 Instructors

Instructor:
Dr. Daly

Tuesday 10th December, 2024

Contents

1 Introduction

This Software Requirements Specification (SRS) document provides a comprehensive overview of the system requirements and design for Mini-Golf Mania, an educational game that reinforces algebra concepts in a fun and interactive way. The document is structured in a hierarchical approach, starting with a high-level overview and gradually diving into detailed technical aspects. The objective is to guarantee clarity and alignment with the project objectives while maintaining logical progression. The topics covered in this document include the purpose of the product, functional and nonfunctional requirements, system constraints, and user characteristics. This document also describes the modeling requirements, constraints, and a prototype that will help stakeholders visualize the game.

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the core features, functionality, and constraints of the Mini-Golf Mania game. Provide a clear and detailed description of the objectives of the system to ensure that all collaborators have a shared understanding of what the game will achieve and how it will work. This document serves as a foundation for the design, development and testing phases of the game, while guiding the project team through the implementation and validation processes. The intended audience for this document includes developers, project managers, clients who commission the product, customers who purchase the product, and users who will utilize the product. Developers will use this document to guide the design and implementation of the game, while project managers will rely on it to ensure that the project is completed in full and within scope. Clients will use the document to verify that the game meets their expectations, customers will refer to it to understand the features they are purchasing, and end users will interact with the features outlined in the document.

1.2 Scope

The software product to be produced is a 2D edutainment game developed using Unity and WebGL. The game is designed to combine the engaging mechanics of mini-golf with educational content focused on algebra concepts, specifically linear equations. The application domain of this software is the edutainment sector, focusing on both educational and entertainment value. The main objective of the game is to reinforce algebraic concepts through practical and hands-on problem solving within the context of a game. It will allow players to interact with algebraic expressions and observe the impact of their modifications through the trajectory of the golf ball, effectively visualizing the relationship between equations and graphical representations such as a graphing calculator.

The software will focus on game play, where players will enter algebraic expressions to determine the trajectory of a golf ball. Each level will present unique challenges designed to reinforce algebraic concepts. The game will include a user interface with input fields for algebraic expressions to control the trajectory of the ball, a slider to determine how far the ball will travel, and a progression system of levels that tracks the progress and attempts of the player. There are three levels in the game for each of the forms of linear equations: Standard, Point-Slope, and Point-Intercept.

In contrast, the software will not focus on advanced algebraic concepts beyond linear equations, nor will it feature multiplayer functionality or in-depth lessons. The game will be designed for solely for individual learning.

1.3 Definitions, Acronyms, and Abbreviations

MGM: Mini-Golf Mania. The name of the edutainment game designed to integrate STEM concepts into a fun mini-golf experience.

UI: User Interface. The visual and interactive components of *Mini-Golf Mania* that allow users to navigate and play the game.

Mini Golf: A recreational sport where players aim to complete courses by striking a ball into a hole using the fewest strokes.

Golf Ball: The ball used in *Mini-Golf Mania*, which players manipulate to achieve the goal of the game through calculated trajectories.

Course: The play area in *Mini-Golf Mania*, structured around a linear equation.

Expression: A mathematical equation or formula entered by the player to calculate the trajectory of the golf ball.

Input Field: UI element where users can enter mathematical expressions, will only accept expressions.

Trajectory: The path that the golf ball follows when hit, influenced by player input alone and shown on screen as an expression is typed into the input field.

Power Slider: UI element that lets players control the shot strength, determining how far the ball travels along a pre-determined trajectory based on their input equation.

Drag: Game mechanic involving the ball's physics where it will experience friction while traveling on the course .

Walls: The limit of the course and level, in practice they are the bounds of the level. If the ball interacts with these walls it will bounce.

Linear Equation: A mathematical equation with three forms: Standard Form, Point-Slope Form, and Slope-Intercept Form. Used in the game to simulate and visualize straight-line trajectories or other game play elements.

Standard Form: A linear equation written as $Ax + By = C$, where A, B, and C are constants, and A and B are not both zero.

Point-Slope Form: A linear equation expressed as $y - y_1 = m(x - x_1)$, where m is the slope, and (x_1, y_1) is a point on the line.

Slope-Intercept Form: A linear equation written as $y = mx + b$, where m is the slope, and b is the y-intercept

SRS: Software Requirements Specification. A document that defines the software requirements for *Mini-Golf Mania*, including functionality, constraints, and design considerations.

UML: Unified Modeling Language. A standardized modeling language used to represent the system design of *Mini-Golf Mania*, including use case diagrams and sequence diagrams.

Unity: A cross-platform game development engine used to create the mechanics and visuals of *Mini-Golf Mania*.

WebGL: Web Graphics Library. A JavaScript API used for rendering interactive 2D graphics in the browser, enabling *Mini-Golf Mania* to run without additional software installations.

Button: UI component that allows users to interact with the application by triggering specific actions when clicked or tapped.

Tilemap: Internal Unity grid-based system used for creating 2D levels by placing tiles on a grid.

TilemapRenderer: Renderer for the tile portions of Tilemap system to be displayed onto the course.

SpriteRenderer: Renderer for single sprite in the Tilemap system, used when handling dynamic object such as the goal hole of the course and the moving golfball that need to be seperate from the course itself.

Collider2D: Unity Engine component that defines the shape of a 2D object to detects collisions, trigger events, or define boundaries for objects. Can come in different forms to define shapes such as CircleCollider2D which defines a circle.

Rigidbody2D: Unity Engine component that simulates collision responses, within the 2D physics system.

1.4 Organization

This SRS document is structured to provide a clear and detailed description of the requirements for Mini-Golf Mania. The document is organized into sections that progressively describe the system's purpose, scope, functionality, and other critical aspects.
The rest of the document is organized as follows:

Section 2: Overall Description
An overview of the product, its context, major functions, user characteristics, and any constraints or assumptions related to the system. It also covers the system's product perspective, including interfaces with other systems and hardware.

Section 3: Specific Requirements
Detailed functional and non-functional requirements for the game, organized in a hierarchical manner. Each requirement will be described and referenced with corresponding use cases and other specifications.

Section 4: Modeling Requirements
Details regarding the modeling of system components, including diagrams such as use case, class, sequence, and state diagrams, to provide a graphical representation of the system's structure and behavior.

Section 5: Prototype
Discussion on the prototype of the game, including the system functionality, scenarios, and screenshots that demonstrate how the prototype will represent the final product.

Section 6: References
Lists of all documents and resources referenced in the SRS, including any external sources for definitions, libraries, or standards used.

Section 7: Point of Contact

Contact details of the person or team responsible for the project, allowing participants to reach out for further clarification or inquiries

2 Overall Description

This section provides a comprehensive overview of the game's development and operational context. It describes how the game fits within its intended environment, including the interfaces it relies on and any constraints on its design and implementation. There are key features and functions outlined that focus on how the game will interact with users and provide educational value. Characteristics of the target audience are mentioned, along with assumptions about their environment and algebraic proficiency. This section also addresses the constraints impacting development, such as hardware requirements and legal considerations, and highlights potential features that could be included in future versions of this document.

2.1 Product Perspective

MGM (Mini Golf Mania) is an interactive, web-based application designed to teach algebra concepts through engaging gameplay. It is intended as a standalone educational product, leveraging mini golf gameplay to create a fun and interactive learning experience. By solving mathematical equations, players influence the trajectory of a golf ball, merging gameplay with core algebraic principles. MGM is accessible via standard web browsers and is compatible across desktop and mobile platforms, making it a versatile educational tool for students.

System Interfaces:
MGM interface with web browsers, supporting both desktop and mobile environments, with basic HTML5 and WebGL support required.

User Interfaces:
The primary user interface will consist of input fields for equations as well as visual representation of the game, so the user will need a screen, keyboard, and mouse or a mobile device capable of performing the same functions.

Hardware Interfaces:
MGM will require minimal hardware, working on any standard desktop or mobile device capable of running WebGL enabled browsers.

Software Interfaces:
MGM will rely on Unity for game mechanics and WebGL for rendering within the browser, so users will need a device capable of running any popular web browser.

Operations: MGM will operate offline once loaded, requiring no constant internet connection otherwise the initial connection to download the contents of the game.

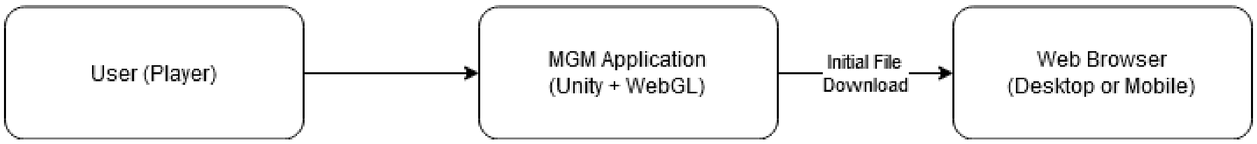
User (Player)
MGM Application (Unity + WebGL)
Initial File Download
Web Browser (Desktop or Mobile)

Figure 1: Pictorial Representation of Game System

2.2 Product Functions

MGM will feature several key functions to achieve its goal of teaching algebra through interactive game play. Players input algebraic expressions, specifically linear equations, to determine the trajectory of a golf ball, which is visually represented in real time. The game incorporates physics to simulate the ball's movement to create an authentic mini-golf experience.

MGM includes multiple levels with increasing difficulty, reinforcing algebra concepts, while a scoring system rewards accuracy and efficiency. A clean, intuitive user interface ensures seamless interaction, allowing players to focus on game play and learning. Additionally, instructional feedback provides hints and guidance, supporting players as they refine their understanding of algebra.

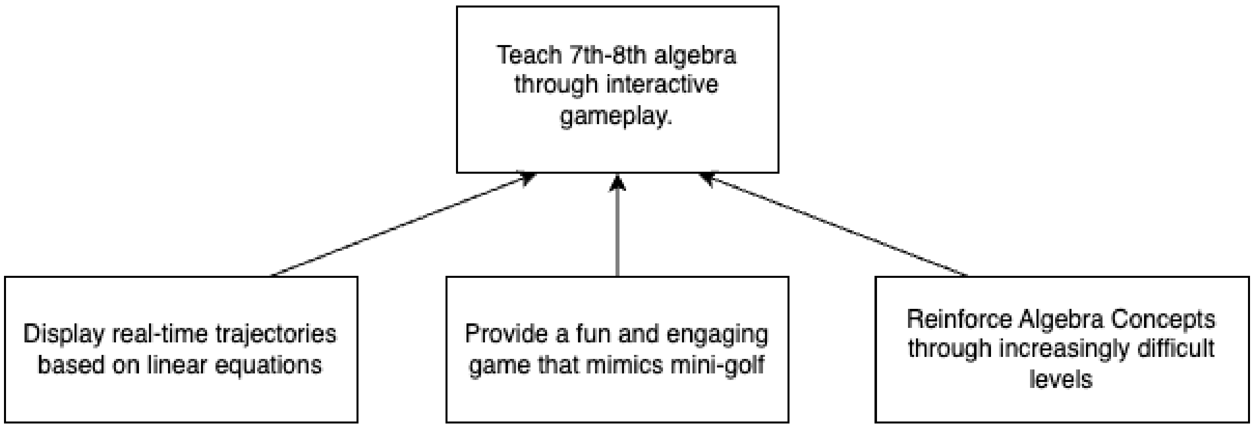
Teach 7th-8th algebra through interactive gameplay.
Display real-time trajectories based on linear equations
Provide a fun and engaging game that mimics mini-golf
Reinforce Algebra Concepts through increasingly difficult levels

Figure 2: High Level Goals

2.3 User Characteristics

The intended users for MGM are students primarily in grades 7 and 8, who are learning Algebra I. These users are expected to have a foundational understanding of algebraic principles, including linear equations and expressions, which are typically introduced in 7th grade. Users should be comfortable working with linear equations in point-slope, slope-intercept, and standard form. By 8th grade, students are expected to have developed proficiency with linear relationships, equations and expressions as recommended by the Massachusetts Department of Education's advisory on the Mathematics Curriculum Framework. The game is designed to challenge and engage users who are proficient in algebraic reasoning therefore MGM targets higher-level middle school students who can apply mathematical concepts to solve interactive challenges.

2.4 Constraints

Developmental Constraints: The development of MGM is constrained by the limited time and resources available to the project. Developers must adhere to a strict schedule and agile method to deliver a functional prototype and subsequent versions. Also, the use of Unity as the development platform imposes a dependency on team members having familiarity with the engine and its scripting environment, potentially limiting design flexibility.

Legal Constraints: The game must comply with the educational clauses stated in Section 110 of U.S. copyright law, which are applicable to all resources compiled for the game. This includes ensuring that third-party assets such as texture packs are used in accordance with their licensing terms and conditions.

Other Considerations: While safety and security are not primary concerns for this game, the project must ensure that any hosted web-based deployment adheres to basic web security standards, such as securing the hosting platform against vulnerabilities.

2.5 Assumptions and Dependencies

Assumptions: The game assumes that end users have access to devices with standard hardware capabilities, including a CPU and graphics card sufficient for running modern web browsers. Users will use browsers such as Chrome, Firefox, Safari, or Edge, all of which must support WebGL for rendering. Users should be able to interact with the game using a mouse or touchscreen, depending on the platform. The game also assumes users have a stable internet connection to access the game through a web browser. Lastly, it assumes that the user has a good foundation in algebraic concepts, particularly linear equations, as the game requires these skills to solve in game challenges related to trajectory and ball movement.

Dependencies: Key dependencies include WebGL support in the user's browser engine to render the game's graphics and Unity WebGL compatibility for developers to build and test the game. As the game targets WebGL as its deployment platform, it requires compatibility with modern web browsers and operating systems. However, devices with outdated hardware or limited graphics capabilities may not perform optimally, which could limit accessibility for some users.

2.6 Apportioning of Requirements

The current prototype focuses on core functionality essential for demonstrating the basic concept of the game. This includes a main menu with operational "Start" and "Quit" buttons, functional input fields for equations, a "Shoot" button, level progression, and a basic win condition. Upon winning, a window prompts the player to either continue or quit the game, ensuring a functional game flow. Functionality that will be implemented later in development includes a ball sprite for visual representation, basic physics found in mini-golf, an equation validation system to ensure input accuracy, a trajectory calculator to compute the ball's path based on user input, a built-in coordinate system for mechanics, and problem sets for each level.

3 Requirements

1. Each level will take the form of an overhead view of a mini-golf course.

   1.1 The view of the mini-golf course will be overlaid with a labeled grid representing the first quadrant of the Cartesian plane.

   1.2 The course will have a set starting point for the ball.

   1.3 The course will feature a hole for the destination of the ball.

   1.4 The course will be surrounded by walls to keep the golf ball within gameplay bounds of the level..

2. Players will decide the trajectory of a golf ball by entering or augmenting a mathematical expression.

   2.1 Once an expression has been entered, an overlay of the projected trajectory of the ball will be shown and will be rendered over the golf course. This overlay will update whenever the expression is altered.

   2.2 The ideal trajectories for each level will be designed to reinforce specific concepts from Algebra 1.

3. Once Players have decided a trajectory, they will decide the distance the ball travels along that trajectory using the Power Slider.

   3.1 The Power Slider will allow players to set the strength of their shot by adjusting its positions. The parity and value of of the power set will determine how far the ball will travel up or down the slope

    3.2 Slider will reset after every shot to the left.

4. Upon determining power of the slider and pressing the shoot button, the ball will move along the trajectory to a final position predetermined by the Power Slider.
    4.1 If the ball's final position is within a specified distance from the hole, it will enter the hole.
    4.2 If the ball enters the hole, a celebratory message will be displayed, and a "Next Level?" button appears to take the player to the next level.
    4.3 On the final level, the "Next Level" button will instead return the player to the main menu.
    4.4 If the ball misses the hole, the ball will be playable from wherever it lands.
5. The game will have a main menu consisting of 3 options: Start Game, Quit Game, and Play Tutorial.
    5.1 If a player is in the level, they may return at any time to this menu screen.
    5.2 Start game will bring the player to the first level.
    5.3 Quit Game will exit the application.
    5.4 Play Tutorial will load a screen that shows how each UI element works.
        5.4.1 To exit the Tutorial, either Start Game or Back to Menu can be selected.
6. Upon beating the final level, a short fanfare will play to congratulate the player.

4 Modeling Requirements

4.1 Use Case Diagram

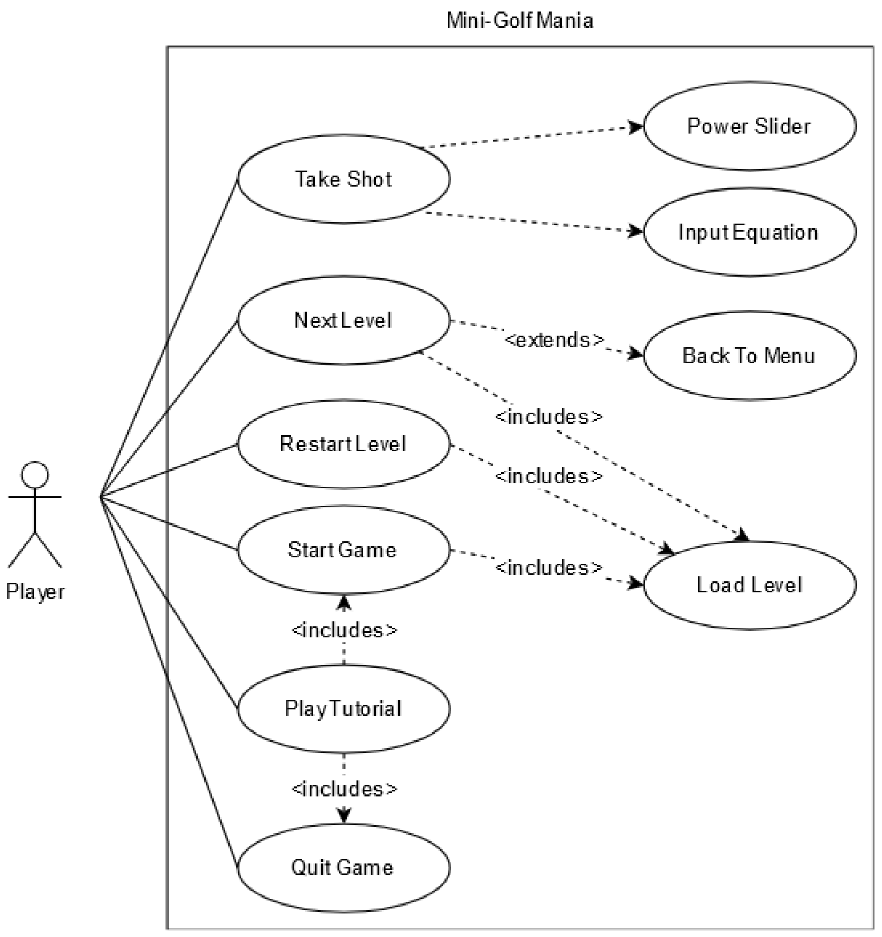

Figure 3: Use-Case Diagram for Mini Golf Mania

Figure 3 shows the primary actor on the left, the Player, interacts with the core features of the game, such as starting the game, progressing through levels, taking shots, and quitting the game. Standard UML notion is used, with each use case of the features represented with an oval circle using `includes` and `extends` as needed.

The Start Game and Next Level cases rely on the shared functionality of loading levels with the Load Level use case. The Next Level has an option to allow the player to return to the main menu. The Take Shot use case incorporates the player's ability to adjust shot parameters like power (via the Power Slider) and input algebraic equations. Quit Game use case allows for the player to end the game session whenever desired.

4.1.1 Use-Case Data Dictionary

Use Case Name:	Start Game
Actors:	Player
Description:	The button to click to start the game present on the main menu, player will then load into the first level of the game. In the current build, if the player returns to the main menu progress will not be saved and ”Start Game” will load back into the first level.
Type:	Primary and essential
Includes:	Load Level
Extends:	N/A
Cross-refs:	Requirement 5.1
Use cases:	Player Tutorial

Use Case Name:	Take Shot
Actors:	Player
Description:	Main shooter button player will have access to within every level. Requires that player set the power using the Power Slider and input a valid linear equation in the form of $y = mx+b$. After shot, the input form will be cleared.
Type:	Primary and essential
Includes:	Power Slider, Input Equation
Extends:	N/A
Cross-refs:	Requirement 4
Use cases:	N/A

Use Case Name:	Quit Game
Actors:	Player
Description:	The button to exit the game, will be available from the main menu. When the button is clicked the user will immediately close the game.
Type:	Primary and essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 5.2
Use cases:	Play Tutorial

Use-Case Data Dictionary Continued

Use Case Name:	Next Level
Actors:	Player
Description:	The button to click to enter the next level after beating the current level. Usable after the player has sunk the ball into the hole and show up in button with the text "Next Level?". If no more levels remain, it will send the user back to the main menu.
Type:	Secondary and essential
Includes:	Load Level
Extends:	Back to Menu
Cross-refs:	Requirement 4.2 & 4.3
Use cases:	N/A

Use Case Name:	Play Tutorial
Actors:	Player
Description:	Will show a tutorial screen that explains all UI elements of the game, similar to a help button. Note that this is a screen and not a level so the only actions the player can take is to quit or start the game .
Type:	Primary and essential
Includes:	Start Game, Quit Game
Extends:	N/A
Cross-refs:	Requirement 5.4
Use cases:	N/A

Use Case Name:	Load Level
Actors:	Player
Description:	Will load any level as needed. If use case was triggered by Start Game then it will load the first level (Hole 1). If triggered by Next Level then it shall
Type:	Secondary and essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 5.2 & 4.2
Use cases:	Start Game, Next Level

Use-Case Data Dictionary Continued

Use Case Name:	Input Equation
Actors:	Player
Description:	Players will input a linear equation that will determine the trajectory of the golf ball.
Type:	Primary and essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2
Use cases:	Take Shot

Use Case Name:	Power Slider
Actors:	Player
Description:	UI Element that will allow the player to select the power for every shot they take, thereby determining how far it will go.
Type:	Secondary and essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 3.1
Use cases:	Take Shot

Use Case Name:	Back to Menu
Actors:	Player
Description:	Button present after every level is loaded to return to the main menu. This will happen automatically if the play selects ”Next Level” on the last level.
Type:	Secondary and essential
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 4.3
Use cases:	Next Level

4.2 Class Diagram

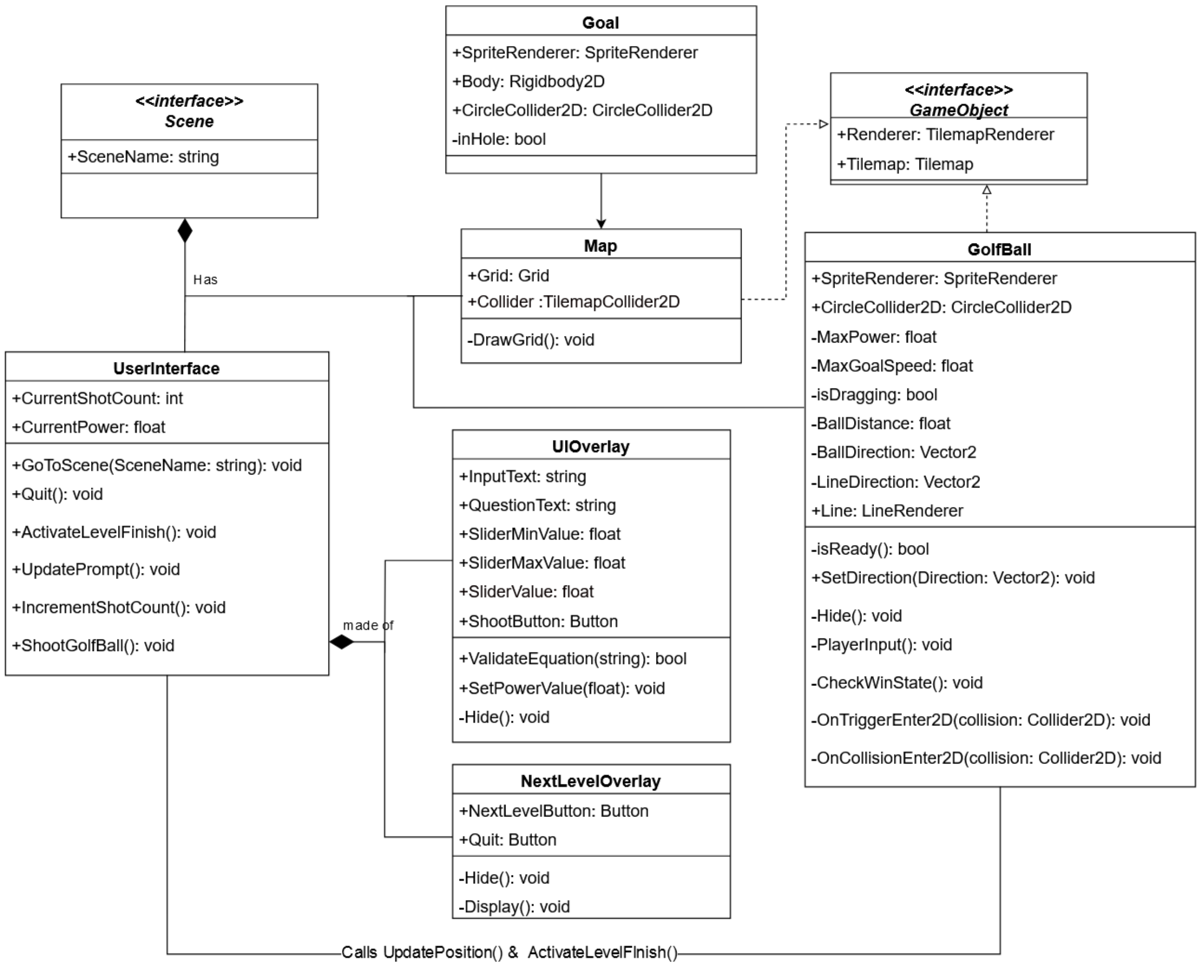

Figure 4: Class Diagram for Mini-Golf Mania

Figure 4 represents the components of MGM where all classes inherit from Unity's built-in GameObject interface. The UserInterface class handles user input, such as scene transitions and shot count, while also interacting with the class to display text and buttons. The Map class is responsible for rendering the game grid using a Tilemap, while the GolfBall class controls the ball's movement, physics, and collision detection. The Goal class checks if the ball reaches the hole, and the NextLevelOverlay manages the transition to the next level. These components work together to simulate the game's physics and display the interface based on user interactions.

4.2.1 Class Diagram Data Dictionary

Element Name	**Description**
Goal	Represents the goal for the level, displayed with a flag in a hole on the course.
Attributes	
SpriteRenderer: SpriteRenderer	Displays the flag sprite onto the level.
Body: Rigidbody2D	Enables physics-based collision and response for the goal object in the 2D world.
CircleCollider: TilemapCollider2D	Defines the shape of collision of the goal object.
inHole: bool	Simple check for if the golf ball is in the hole
Operations	N/A
Relationships	One Goal makes make up a Map.
UML Extensions	N/A

Element Name	**Description**
Map	Defines the playable area of the level and ensures ball does not move outside of the level.
Attributes	
Grid: Grid	Handles rendering the visual representation of a Cartesian Graph, displayed onto the course with single units.
Collider: TilemapCollider2D	Provides collision detection for the so the ball cannot pass through the border.
Operations	
DrawGrid(): void	Draws the grid on the map in order to better show that the starting point and the goal are two points on a graph.
Relationships	One Goal makes up a Map.
UML Extensions	N/A

Class Diagram Data Dictionary Continued

Element Name	**Description**
UIOverLay	Unity's UI Overlay System that handles the Power Slider, Input Field, and the shoot button for each level.
Attributes	
InputText: string	Stores the user's equation if it is valid
QuestionText: string	Question of the level, stored here as it exists on the same level and manged by UpdatePrompt().
SliderMinValue: float	The lowest value the Power Slider can go .
SliderMaxValue: float	The highest value the Power Slider can go.
SliderValue: float	Value the user inputs into the PowerSlider.
ShootButton: Button	Unity Button that triggers the ShootGolfBall() in UserInterface .
Operations	
ValidateEquation(string): bool	Ensures that the expression the player types into the textbox is a valid linear equation, will return true if so.
SetPowerValue(float): void	Lets player either type or slide the value for the Power Slider.
Hide(): void	Hides the UI Overlay when game is beaten.
Relationships	GolfBall, Map, UIOverlay, and NextLevelOverlay make up a UserInterface.
UML Extensions	N/A

Class Diagram Data Dictionary Continued

Element Name	**Description**
GolfBall	Game Object that represents the golf ball within the game.
Attributes	
SpriteRenderer: SpriteRenderer	Tilemap component that renders the sprite for the goal ball.
CircleCollider2D : CircleCollider2D	Gives the golf ball the shape of circle when handling collision with other Game Objects
MaxPower: float	Question of the level, stored here as it exists on the same level and manged by UpdatePrompt().
MaxGoalSpeed: float	A limit on the speed of the golf ball should it make contact with the goal, as the ball can fly over the hole in regular mini golf if going too fast.
isDragging: bool	Flag used to indicate if the player is currently aiming, modified from a state when movement was controlled like dragging a mouse .
BallDistance: float	Current distance the ball is primed to travel at.
BallDirection: Vector2	Current direction the ball is primed to travel at .
LineDirection: Vector2	Current direction the ball's trajectory is set to travel at.
Line: LineRenderer	Displays the ball's trajectory to the screen.

...

Please see next page for the remainder of GolfBall

Class Diagram Data Dictionary Continued

Operations	
isReady(): bool	Check on if the ball's velocity has has stopped so it may be shot again.
SetDirection (Direction: Vector2D): void	Sets direction of the ball, translated from the player's equation.
Hide(): void	Hides the ball when game is beaten.
PlayerInput(): void	Gets player input after it has been validated by the UIOverlay.
CheckWinState(): void	After OnTriggerEnter2D validates that the ball collided with the goal, checks if ball was under max speed and if so inHole is set to true and
OnTriggerEnter2D (collision : Collider2D): void	Checks if the ball collided with the goal object specifically, triggers CheckWinState().
OnCollisionEnter2D (collision : Collider2D): void	Checks if ball collided with the walls of the game, plays a sound of the hitting one of the walls.
Relationships	GolfBall, Map, UserInterface, and make up a Scene. GolfBall is a game object.
UML Extensions	N/A

Element Name	**Description**
GameObject	Class representing in game objects and serves as a container for game components, behaviors, and appearances in the engine.
Attributes	
Tilemap: Tilemap	Collection of tiles within the Tilemap system used to display sprites and levels.
Renderer: TilemapRender	Renderer for the tile portions of Tilemap system to be displayed onto the course, each object will have their own renderer.
Operations	
Relationships	Map and GolfBall inherit from GameObject.
UML Extensions	N/A

Class Diagram Data Dictionary Continued

Element Name	**Description**
NextLevelDisplay	UI that appears after the player has hit the ball into the hole.
Attributes	
NextLevelButton: Button	UI Button that when clicked will call GoToScene() in UserInterface to go to the next level.
QuitButton: Button	PUI Button that when clicked will call Quit() in UserInterface to quit the game entirely.
Operations	
DrawGrid(): void	Draws the grid on the map in order to better show that the starting point and the goal are two points on a graph.
Relationships	One Goal makes up a Map.
UML Extensions	N/A

Element Name	**Description**
Scene	Main Level container that houses all the gameplay and UI elements. Each Scene is effectively its own level that houses its own Map, GolfBall, and UserInterface.
Attributes	
SceneName: string	Name of the scene so that GoToScene() can travel in-between Scenes when called.
Operations	N/A.
Relationships	Each Scene has a Map, GolfBall, and UserInterface.
UML Extensions	N/A

Class Diagram Data Dictionary Continued

Element Name	**Description**
UserInterface	UserInterface that encapsulates the overlays for the UI and next level screen. Contains several functions to handle actions like scene transaction, textboxes,and interactions between the UI and Game Objects.
Attributes	
CurrentShotCount: int	Holds the nubmer of shots the player has taken
CurrentPower: float	Shows the current power of the slider
Operations	
GoToScene (SceneName : string) : void	Goes to the next scene when called, and each scene has this function set up so that only the correct scene may be accessed. For example Hole 1 is set so that only the scene named "Hole 2" can be accessed.
Quit() : void	Quit action that immediately closes the upon when called.
ActivateLevelFinish() : void	Function that calls the Hide() functions in UIOverlay and GolfBall and Display() in NextLevelOverlay as level win screen.
UpdatePrompt() : void	Sets the QuestionText prompt window displaying the coordinates of the ball and hole.
IncrimentShotCount() : void	Increments the CurrentShotCount value by one every time a shot is sent out.
ShootGolfBall() : void	Called when the Shoot Button in UIOverlay is clicked. Will shoot the golf based on its set trajectory and power set by the Input Field and Power Slider respectively.
Relationships	UserInterface is made of a UIOverlay and NextLevelOverlay. Every Scene has a UserInterface
UML Extensions	N/A

Template based on IEEE Std 830-1998 for SRS. Modifications (content and ordering of information) 23 Revised: Tuesday 10$^{\text{th}}$ December, 2024 03:52

4.3 Sequence Diagrams

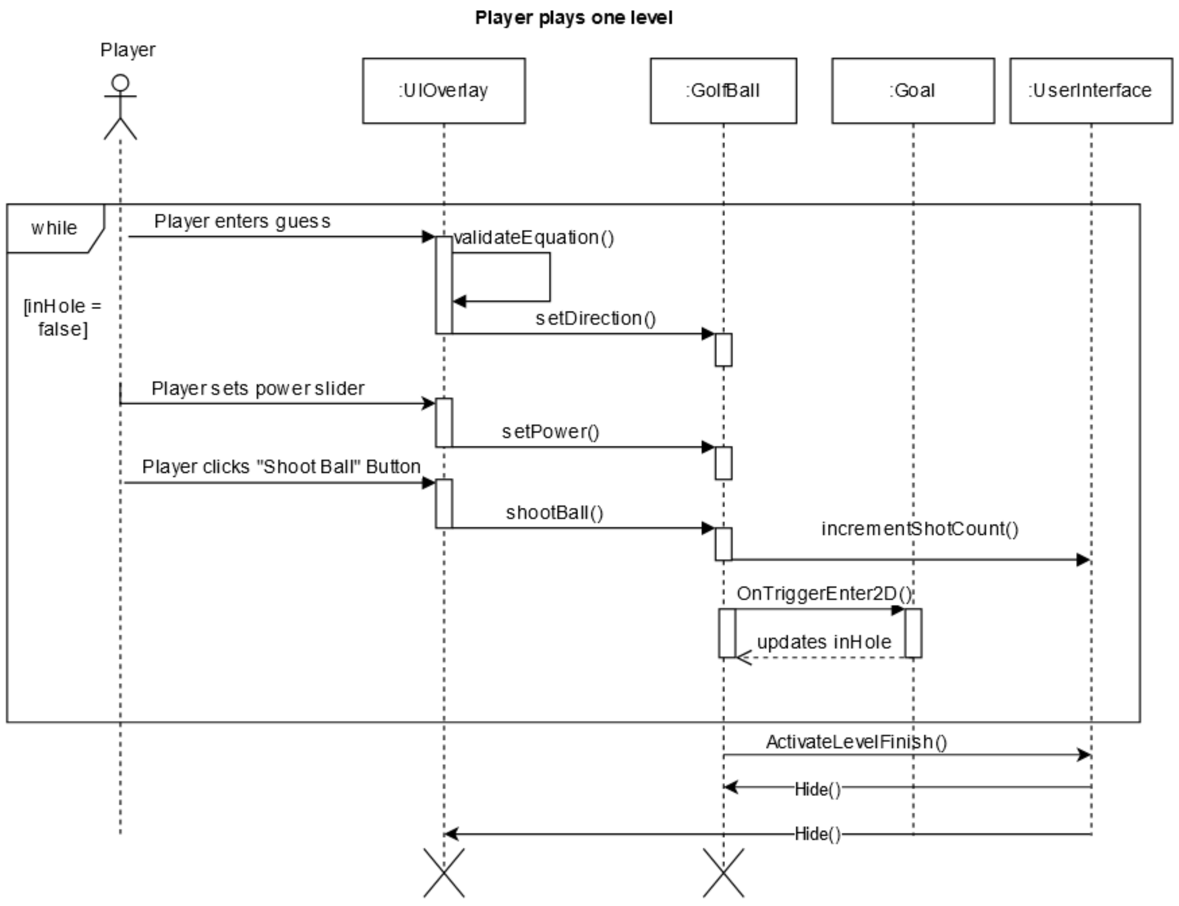

Figure 5: Player Plays One Level

Figure 5 illustrates the process of a player completing one level in the game. The player starts by entering a guess into an input field, which is validated by the UIOverlay component. Once validated, the player sets the direction and power for the ball using the interface, with these inputs being passed to the GolfBall object. The player then clicks the ”Shoot Ball” button, triggering the ball’s motion, while the UserInterface component increments the shot count. As the ball moves, its position is updated, and the system continuously checks for collisions with the goal, concluding the sequence when the ball successfully reaches its target or the player decides to retry.

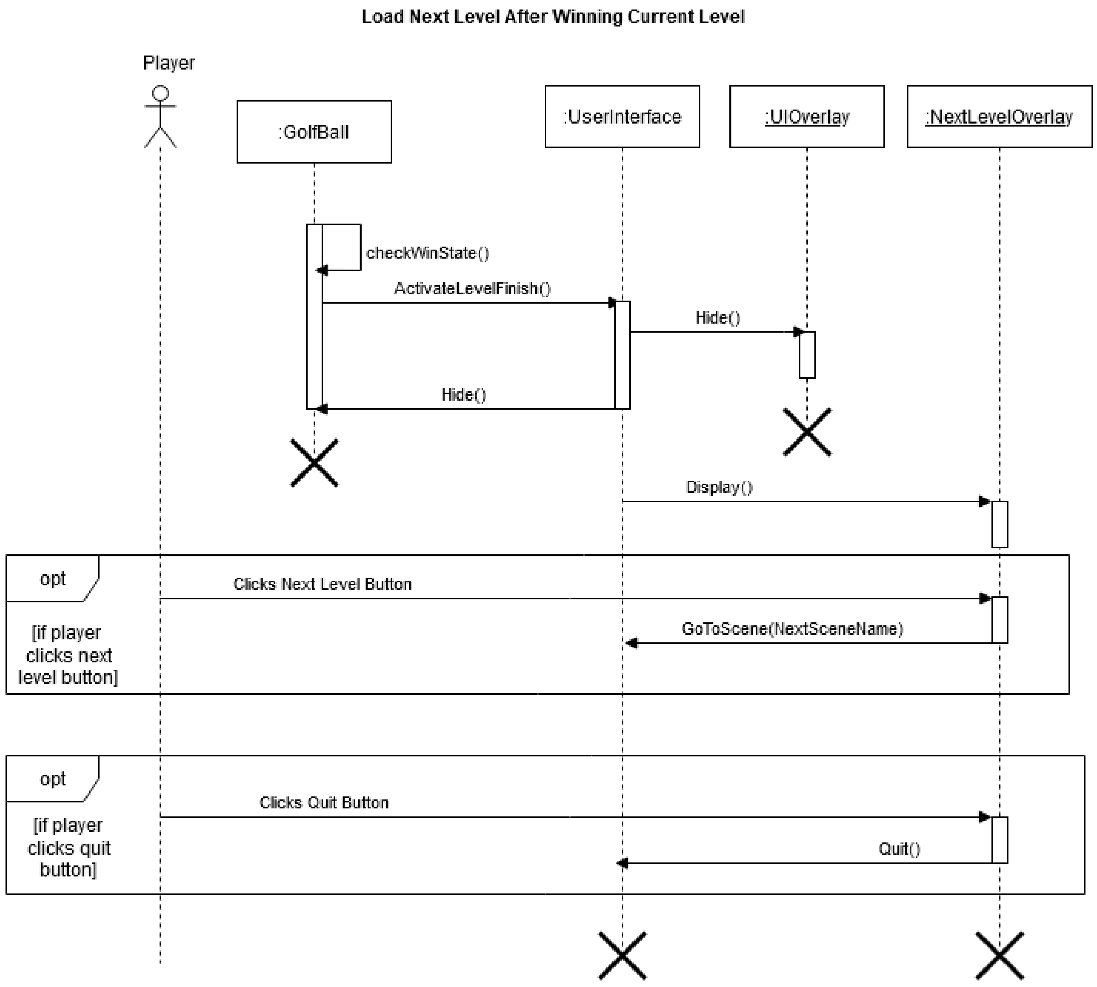

Figure 6: Load Next Level After Winning Current Level

Figure 6 illustrates the process of loading the next level after the player wins the current level in the game. Once the player completes the level, the system checks the win state and activates the level finish sequence. The UI and UserInterface components hide the current level overlays, and the lifelines of the current level's objects are destroyed before rendering the new level. The NextLevelOverlay component then displays the interface for proceeding to the next level. The player can choose to click the "Next Level" button, which transitions the game to the next scene, or click the "Quit" button to exit the game, concluding the sequence.

4.4 State Diagrams

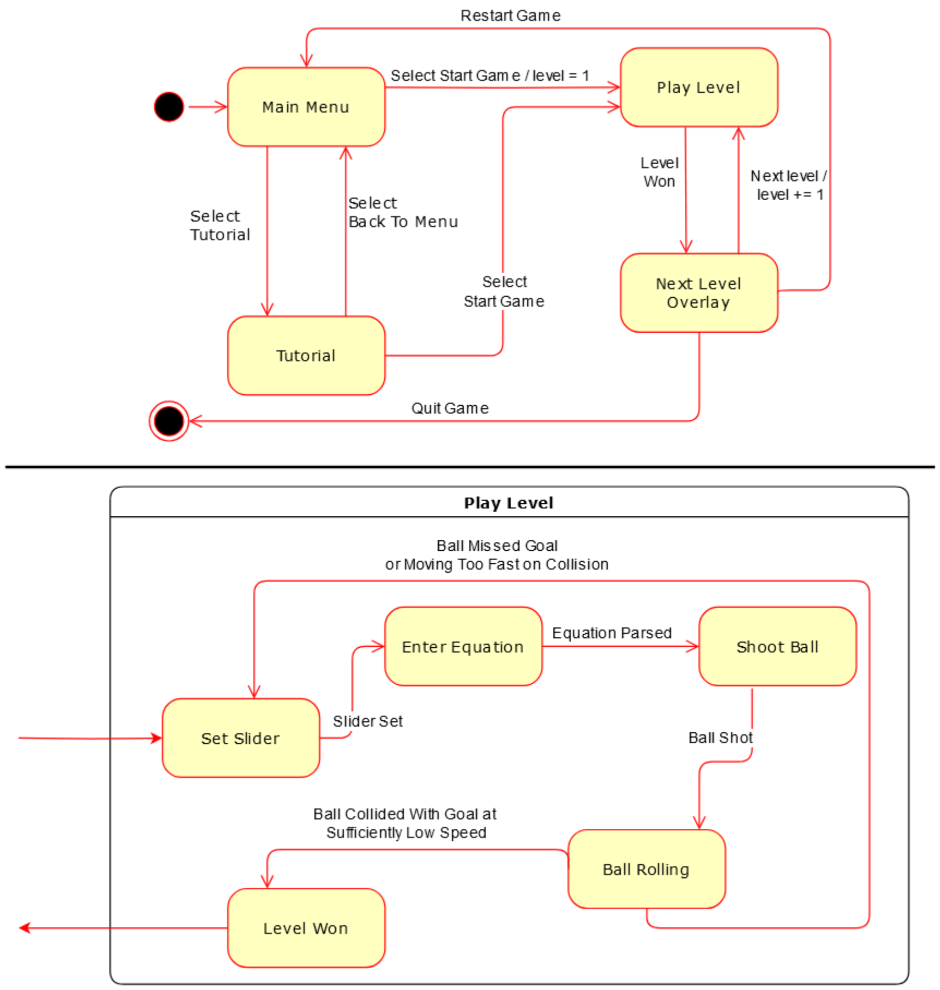

Figure 7: State Diagram showing Overall Gameplay Loop of Mini-Golf Mania

Figure 7 provides an overview of the game flow, depicting the major states and transitions that occur as the player progresses. The game begins at the Main Menu where the player can select to start the game to initiate the first level, or choose to view the tutorial. In the tutorial, players can learn the game mechanics and return to the Main Menu upon completion. Once the game is started, the player enters the "Play Level" state. During the level, players input equations, set the slider, and shoot the ball. If the ball collides with the goal at the correct speed, the level is won, transitioning the game to the Next Level Overlay where the player can advance to the next level. If the ball misses the goal or moves too fast over it, the player remains in the "Play Level" state to retry. At any time, the player can opt to return to the Main Menu or quit the game, ending the session.

5 Prototype

Version 2 of the prototype is deployed and can be played from the project website: `https://ealderman-uml.github.io/EduGame-Website/prototype`.

5.1 How to Run Prototype

To run the MGM prototype, the following steps are required:

1. Setup Unity:
    - Download and install Unity Hub from Unity's official website.
    - Once Unity Hub is installed, use it to download and install the Unity Editor.
    - Ensure that you are using a version of Unity compatible with the project.
2. Clone the GitHub Repository:
    - Clone the project repository from GitHub using the following link: https://github.com/chrislambert3/EduGame.git.
    - You can use GitHub Desktop or any Git client for this purpose.
3. Import Project into Unity Hub:
    - After cloning the repository, open Unity Hub.
    - Click on "Add Project" from Disk and navigate to the location where the repository
    - Select the project folder and add it to Unity Hub.
4. Open the Project:
    - Once the project is added to Unity Hub, click on the project to open it in the Unity Editor.
5. Configure Game Window:
    - In the Unity Editor, navigate to the Game window and set the resolution from Free Aspect to Full HD (1920x1080).
    - Scale the view back out to ensure the entire game screen is visible.
6. Running the Game:

- Press the Play button located at the top center of the Unity Editor window to run the game. The prototype will start running, and you can interact with the game as intended.

System Configuration

Operating System:

- The prototype is compatible with Windows, macOS, and Linux. Ensure your operating system meets the minimum requirements for running Unity projects.

Hardware Requirements:

- A PC or laptop with at least 8GB of RAM, a modern multi-core CPU, and a GPU with OpenGL 3.0 or higher is recommended for optimal performance.

Web Accessibility (for Prototype V2):

- Prototype V2 will be hosted as a WebGL build and should be accessible via a web browser (e.g., Chrome, Firefox).
- Ensure your internet connection is stable when accessing the prototype online.

Plugins and Dependencies:

- Ensure you have the required Unity plugins and packages installed. Any necessary packages will be included in the Unity project and should automatically be configured upon opening the project.

5.2 Sample Scenarios

5.2.1 Main Menu

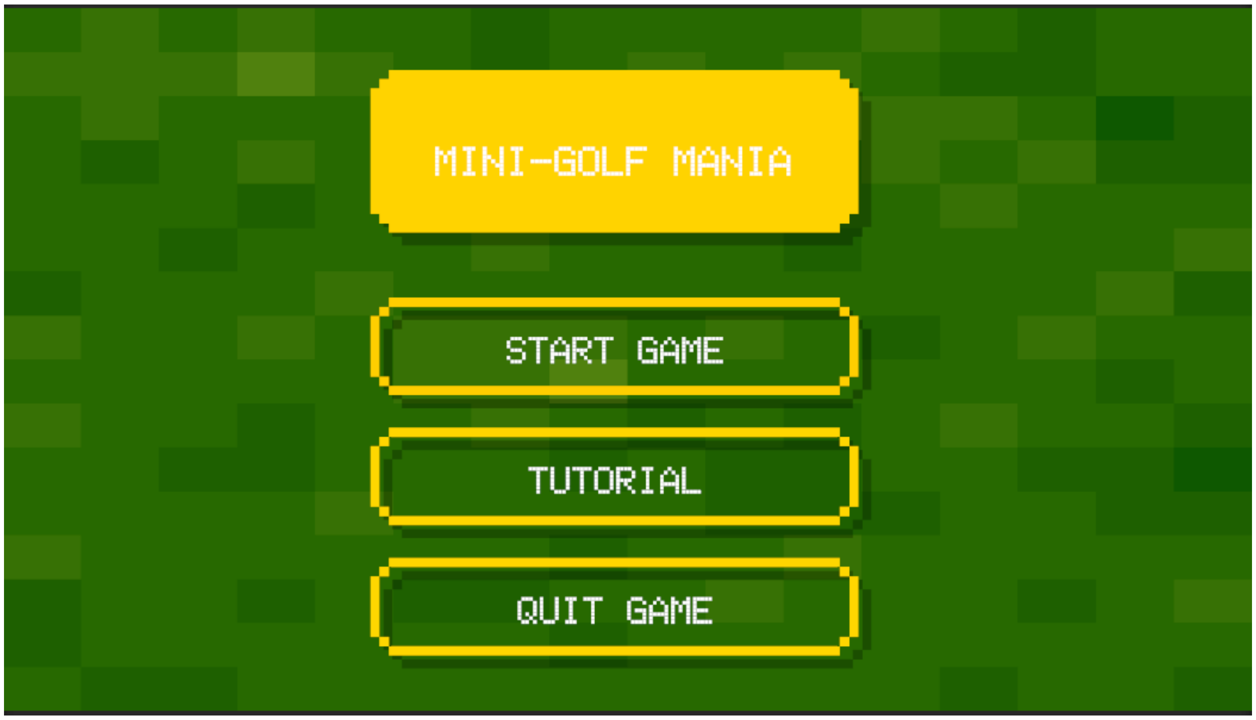

Figure 8: Main Menu Screen

When a player starts the game, they will be prompted with the screen shown in Figure 8, the main menu. From this screen the user will have the ability to start, see the tutorial, or quit the game.

5.2.2 Tutorial Screen

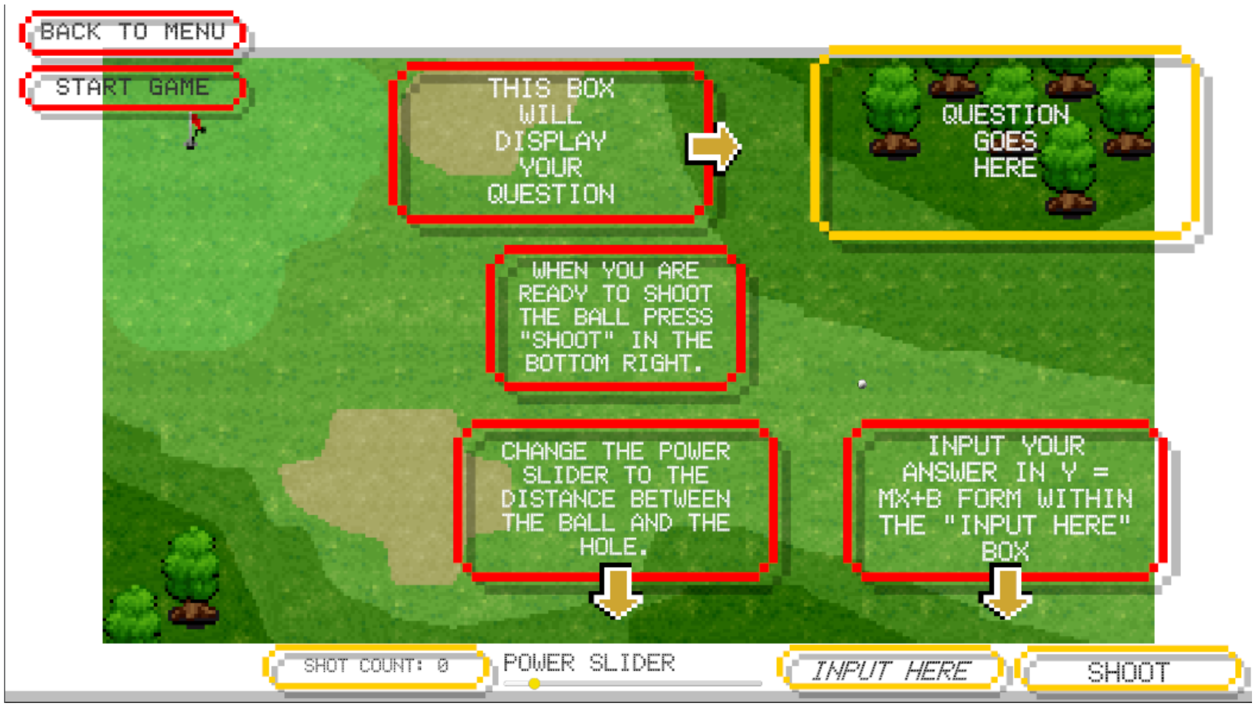
BACK TO MENU
START GAME
THIS BOX WILL DISPLAY YOUR QUESTION
QUESTION GOES HERE
WHEN YOU ARE READY TO SHOOT THE BALL PRESS "SHOOT" IN THE BOTTOM RIGHT.
CHANGE THE POWER SLIDER TO THE DISTANCE BETWEEN THE BALL AND THE HOLE.
INPUT YOUR ANSWER IN Y = MX+B FORM WITHIN THE "INPUT HERE" BOX
SHOT COUNT: 0
POWER SLIDER
INPUT HERE
SHOOT

Figure 9: Tutorial Screen

Figure 9 is a tutorial screen that explains the game's mechanics. It guides the player to input their answer in the form $y = mx + b$ in the "Input Here" box. The screen also instructs the player to adjust the power slider based on the distance between the ball and the hole. The player can then press the "Shoot" button to launch the ball, with the shot count displayed at the bottom. The tutorial helps users understand how to interact with the game and make their shot using the power slider and the inputted equation.

5.2.3 Game Hole

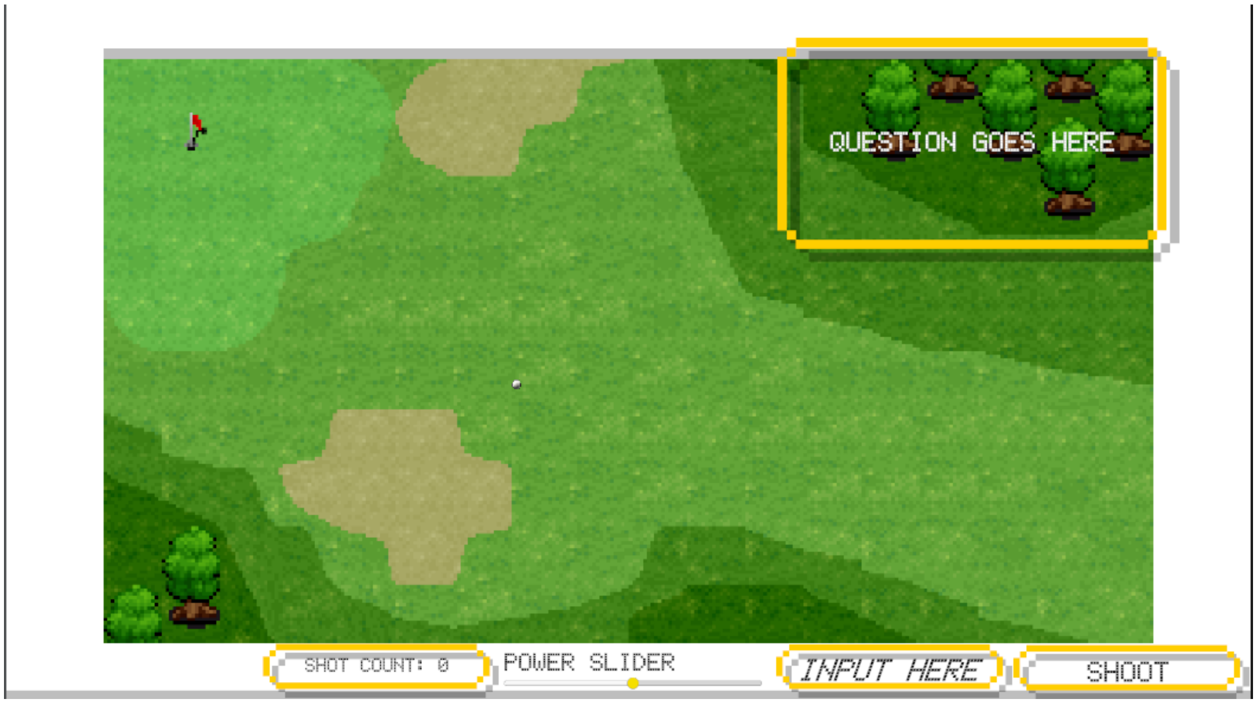

Figure 10: Game Hole

Figure 10 depicts when player chooses to start the game, they will be brought to one of the levels of the course. Here they will be prompted with a question as you can see in the bottom left. They will be able to input an answer within the “input” box and power slider. Once the player is confident with their answer, they will click the shoot button to enter their answer. If the ball misses the hole, the shot count will go up and the user will have to input another equation.

5.2.4 Win Screen

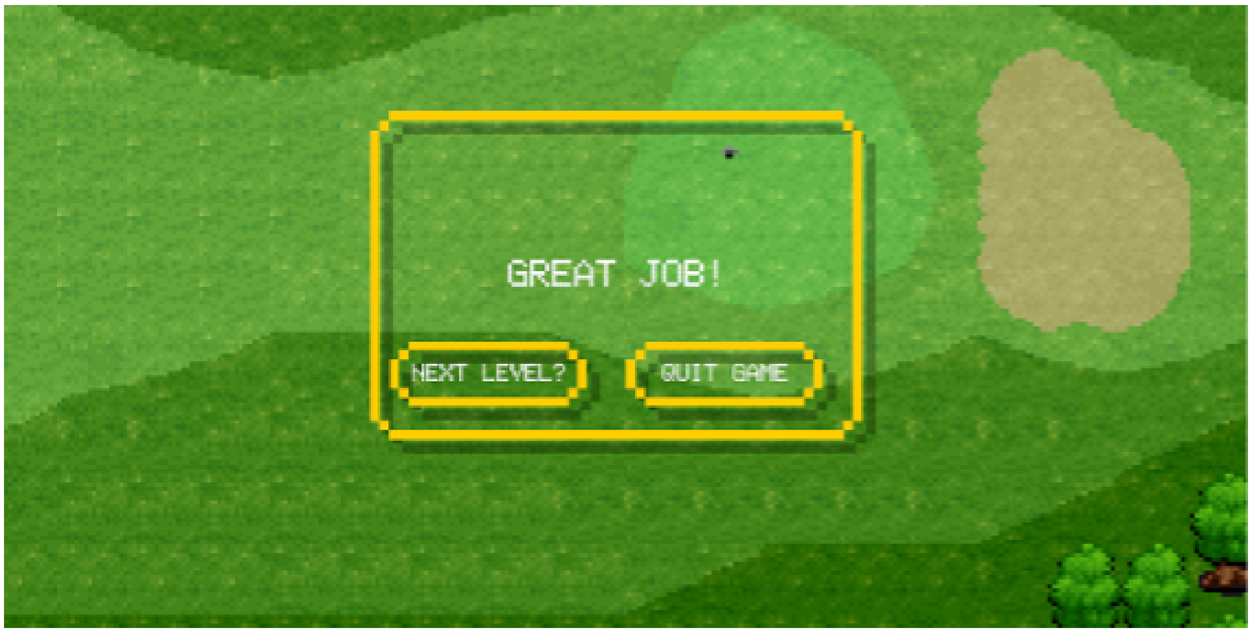

Figure 11: Win Screen

In the case that the players answer is correct, they will be prompted with this screen. Here they will be able to continue to the next hole or quit the game.

6 References

[1] U. S. C. Office, "Chapter 11: Subject matter and scope of copyright," Chapter 1 - Circular 92 — U.S. Copyright Office, `https://www.copyright.gov/title17/92chap1.html#110` (accessed Nov. 19, 2024).

[2] "Copyright in the classroom," Copyright in the classroom — UC Copyright, `https://copyright.universityofcalifornia.edu/use/teaching.html#:~:text=Teachers%20and%20students%20have%20certain,108%20of%20US%20Copyright%20Law` (accessed Nov. 19, 2024).

[3] "Dinky Tiny Golf Asset Pack - free by Mike," itch.io, `https://pixelbitsnbytes.itch.io/dinky-tiny-golf-free` (accessed Nov. 19, 2024).

[4] A. Nguyen, E. Alderman, C. Lambert, and B. Maillet, Mini-golf mania, `https://ealderman-uml.github.io/EduGame-Website/` (accessed Nov. 19, 2024).

7 Point of Contact

For further information regarding this document and project, please contact Prof. Daly at University of Massachusetts Lowell (james_daly@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.