Software Requirements Specification (SRS)
Project Math Moles

Team: 10
Authors: GJ Burlingame, Natalie Resendes, Abdullah Shaheed, Sukhdeep Singh, Tommy Tran
Customer: Students in the 6th-8th grade
Instructor: Dr. James Daly

1 Introduction

This document details the Software Requirements Specifications (SRS) for Math Moles, an educational math game. This SRS will describe the technical requirements, functionality, and purpose of the software.

1.1 Purpose

The purpose of this SRS document is to describe the software's technical requirements, functionality, and purpose. It will contain a general description of the software functions, expectations, and constraints; a list of requirements for the software; and a section describing information regarding the prototype for the game. Diagrams which illustrate aspects of the game will also be provided, including use-case, class, sequence, and state diagrams. All of these details will provide a clear vision and guide for developers to build the game and ensure it meets all of the requirements.

1.2 Scope

Math Moles is an edutainment game intended to reinforce students' equation-solving skills, both in terms of speed and accuracy, in a fun and engaging way. Equations in the game will consist of simple algebraic equations with one variable to solve for. It is designed to be used as a learning tool both in the classroom and individually by students; the design choice to only require the player to input text in order to play allows it to be more accessible for devices that are typically available to students in school environments.

The gameplay will feature moles that pop up from the ground, and each mole will have an equation attached to it. The player will try to answer the equations before each mole's lifetime expires and they disappear back into the ground. Correct answers will whack the mole of the corresponding equation, causing it to disappear and award points. Successive correct answers will increase the player's streak, awarding additional points per correct answer. Incorrect answers are answers that do not match any active equation, and these will reset the player's streak if entered. Each equation, if not answered within the corresponding mole's lifetime, will disappear back into the ground without affecting the player's score or streak. This will encourage the player to develop speed and accuracy in solving basic algebraic equations.

1.3 Definitions, acronyms, and abbreviations

Player: The person using the software and entering input.

Equation(s): Mathematical equation(s) that appear in our game, which are simple algebraic equations that contain one variable to solve for.

Active equation: An equation visible on-screen that is available for the player to answer.

Game session: A single continuous playthrough of the game that starts when the player presses play from the main menu and ends when the game timer expires or the player manually quits to the main menu.

Gameplay screen: The screen displayed while a player is actively playing Math Moles.

Main menu: The menu displayed while a player is not actively playing Math Moles.

Pause menu: The menu displayed while a player has paused their active Math Moles game session.

Answer: Input the player provides during gameplay.

Correct answer: An answer that matches one or more active equations.

Incorrect answer: An answer that does not match any active equation.

Streak: A running count of how many correct answers the player has entered in a row, which increases how many points the player earns per additional correct answer. It is reset upon entry of an incorrect answer.

Moles: Creatures that appear on-screen during a game session and display an equation for the player to solve.

Lifetime: A set amount of time that a mole will remain on-screen and available to have their equation answered.

Active mole: A mole that is visible on the game screen and whose lifetime has not yet expired.

SRS: Software Requirements Specifications

1.4 Organization

The remaining portion of this document contains Sections 2-6, each of which may be further split into subsections. Section 2 consists of an overall description of the product, its product perspective and product functions, user characteristics, constraints, assumptions and dependencies, and finally apportioning of requirements. Section 3 consists of the enumerated list of specific requirements for this product. Section 4 consists of modeling requirements and includes several diagrams with corresponding descriptions. Section 5 consists of details regarding the product's prototype, including a description of what the prototype will accomplish, a description of how to run the prototype and what is required to do so, and sample scenarios of the prototype in action. Section 6 consists of references for resources used in the researching and creation of this product.

2 Overall Description

This section provides an overview of our Math Moles edutainment game. The following sections will describe requirements and constraints of the game, as well as the assumptions made about the hardware and software that users will be using, and the assumptions made about the user themselves. The product's major functions will also be explained, and the section will end with a brief description of additional features that are beyond the scope of this project, but may potentially be addressed in the future.

2.1 Product Perspective

This game is an educational game designed to reinforce middle-school mathematics skills through whack-a-mole-style gameplay. It does not interact with outside applications, but it can be used alongside instruction in a classroom. Figure 1 illustrates how teachers can use the game in their classrooms: after introducing students to algebraic equations, teachers supervise them as they play and identify individual students who are struggling to solve equations, then intervene and provide help. Students who don't require additional help still benefit by getting more opportunity to practice and solidify their skills.

Users playing the game will practice solving simple algebraic equations, developing proficiency and speed. They may also develop mental math skills as they become more familiar with the equations and learn tricks for solving them.

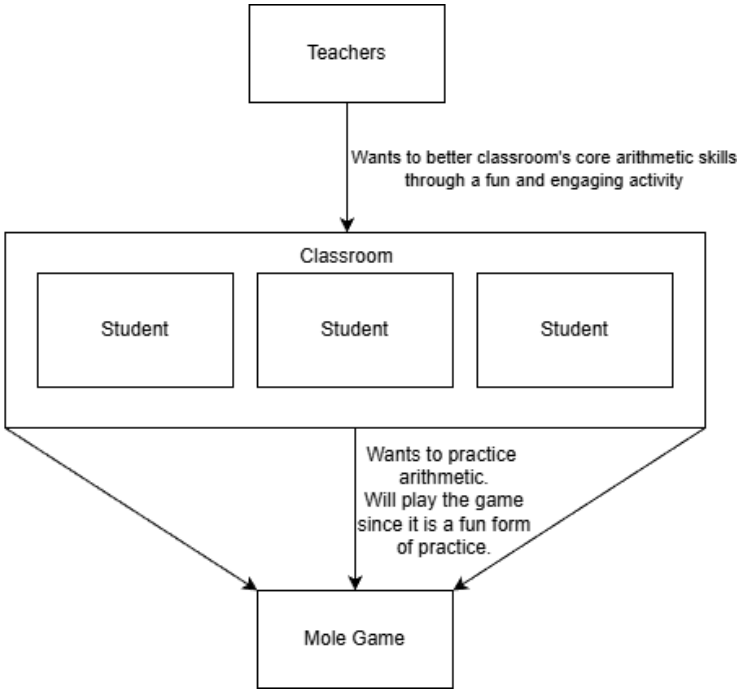

Figure 1: How Math Moles fits in a bigger system

2.2 Product Functions

The game's main functions are in its gameplay and menus. The gameplay revolves around moles with attached equations that pop up from the ground at regular intervals which the player needs to solve before they disappear. There is a single textbox where players enter answers, and each entered answer is compared with every active equation to check for matches. When an answer matches one or more equations on screen, points are awarded to the player and their streak is incremented. Successive correct answers award more points through a streak mechanism. When an entered answer doesn't match any active equation the player's streak resets to zero. The gameplay is timed, and the game automatically ends when the game screen's timer reaches zero. The game will feature a leaderboard where player highscores are stored and displayed alongside an associated username. The game's main menu will feature the option to adjust game settings, access to the game's leaderboard, and a button to begin playing the game.

These gameplay features encourage users to practice mathematical skills, increasing their proficiency and speed by rewarding accuracy and speed with higher point gains.

Figure 2 is a high-level goal diagram for our game, which illustrates the process that the program goes through in order to accomplish the primary goal of helping students improve their math skills.

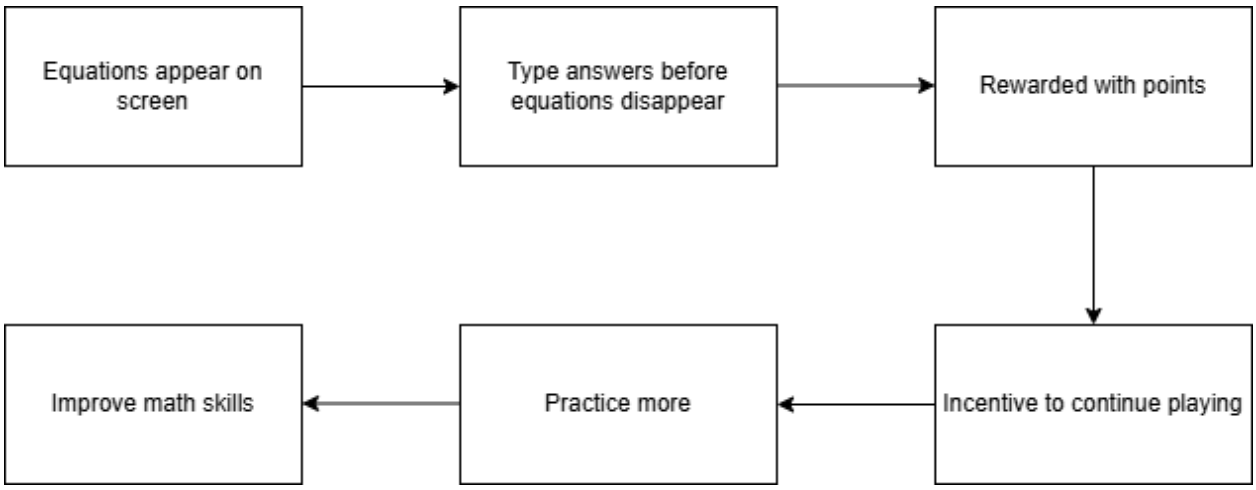

Figure 2: High-Level Goal Diagram

2.3 User Characteristics

The user is expected to be a student in middle school, between sixth and eighth grade, who has been through the previous levels of math classes and is enrolled in their current grade level mathematics class. They are expected to have been introduced to the process of solving simple algebraic equations with one variable, but their solving proficiency can be low. Playing the game is intended to provide practice to help them improve their ability to solve these problems, so their proficiency is expected to increase as they play. The user is expected to have basic motor skills, specifically the ability to type answers using the keyboard and navigate with a cursor through the use of a mouse or trackpad. The user is also expected to be able to read basic English instructions.

2.4 Constraints

The game is intended for students in the sixth to eighth grade, so the featured mathematical content is restricted to basic algebra and below.

It must be possible to run the game on devices available to students and schools, so the game must be compatible with relatively low-powered devices. It must also be accessible to students in the target age range, and so it must contain relatively simple instructions and controls as well as remain age-appropriate in terms of game visuals and content.

2.5 Assumptions and Dependencies

Some assumptions about what hardware the user will be using or have access to. It is assumed that the user has access to a computer with a keyboard and either a screen or monitor that can display graphics. The user also needs to have access to either a mouse or trackpad. In terms of software, it is assumed that the user's computer should be running, or be capable of running, a relatively recent operating system. The user is assumed to be capable of reading English at a middle-school level.

2.6 Apportioning of Requirements

Several additional features would enhance the game but are considered to be out of the scope for this project. An enhanced, interactive tutorial feature could be added that would provide interactive guidance on how to use and play the game, but it is not crucial to the game's function. The ability for players to create accounts to track their statistics and view game history could be helpful, but it would add a large amount of additional work that requires more time than is allowed. Additional game modes could be added to allow the user to add custom equations in order to tailor the game to their practice needs, but this would also push the project past its deadline. These features could be considered in the case of future updates, but are not expected to be included in the base project.

3 Specific Requirements

1. The game will feature game sessions
    1.1. Each game session will feature a timer counting down from 45 seconds
        1.1.1. The game will end when the timer runs out
            1.1.1.1. The game will display the player's score when the game ends
    1.2. Game sessions will feature creatures that appear and display math equations
        1.2.1. The player will be able to answer these equations to gain points
        1.2.2. Entering an answer that matches any equation on screen will award points
            1.2.2.1. Only numerical input is allowed
            1.2.2.2. Entering successive correct answers will build a streak that will increase the amount of points earned per correct answer
            1.2.2.3. If the entered answer matches more than one equation on screen, it will apply to all matching equations
        1.2.3. Entering an answer that does not match any equation on screen will not award points
        1.2.4. Entering an answer that does not match any equation on screen will reset the player's streak to zero
2. The game will feature a main menu
    2.1. The player will be able to start a game session from the main menu
    2.2. The player will be able to exit the game from the main menu
    2.3. The player will be able to access the settings menu from the main menu
    2.4. The player will be able to access a tutorial from the main menu
    2.5. The player will be able to provide a username in the main menu
3. The game will feature a settings menu
    3.1. The player will be able to modify the game's volume from the settings menu
    3.2. The player will be able to return to the main menu from the settings menu
4. The game will feature a pause menu
    4.1. The player will be able to access the pause menu while in a game session
    4.2. The player will be able to adjust the game's volume from the pause menu
    4.3. The player will be able to exit the pause menu and return to their game session
    4.4. The player will be able to quit their current game session and return to the main menu from the pause menu
    4.5. The timer will be paused upon entering the pause screen
    4.6. The timer will be unpaused upon returning to the game session
5. The game will keep track of a list of the highest amount of points earned in a game session
    5.1. The game will display a leaderboard of the top ten highest scores from highest to lowest on the main menu

5.2. If any game session surpasses a top ten score on the leaderboard, a new entry will be added to the leaderboard with the username that was provided, next to the score

6. The game will feature various sounds upon certain actions that take place
   6.1. Correct answers will play a “success” noise
   6.2. Incorrect answers will play a “failure” noise
7. The game will contain a tutorial that teaches the player how to play the game
   7.1. The tutorial will be accessible from the main menu
   7.2. The tutorial will explain how to input answers, gain points, and use the pause menu
   7.3. The tutorial will use clear, easy-to-understand language

4 Modeling Requirements

The following section contains all diagrams created for this game, which consist of two use-case diagrams, one class diagram, two sequence diagrams, and a state diagram. Each diagram will have an accompanying description to provide additional information about what the diagram is illustrating.

4.1 Use-Case Diagrams

Figure 3 and Figure 4 illustrate use-cases for our game, Math Moles, and how they relate and interact with each other. The first diagram represents use-cases present when interacting with the gameplay screen. The second diagram represents use-cases present when interacting with the main menu.

4.1.1 Gameplay Screen Use-Case Diagram

Figure 3 provides a view of the primary actor, the player, and how they interact with the game's gameplay screen through entering answers or pausing the game. It also shows how the System actor, which represents the game's logic, interacts with the gameplay screen, managing what is displayed on the screen and the responses to the primary actor's actions. The Standard UML notion is used, with each use case of the features represented by an oval circle, also using <<includes>> and <<extends>> as needed.

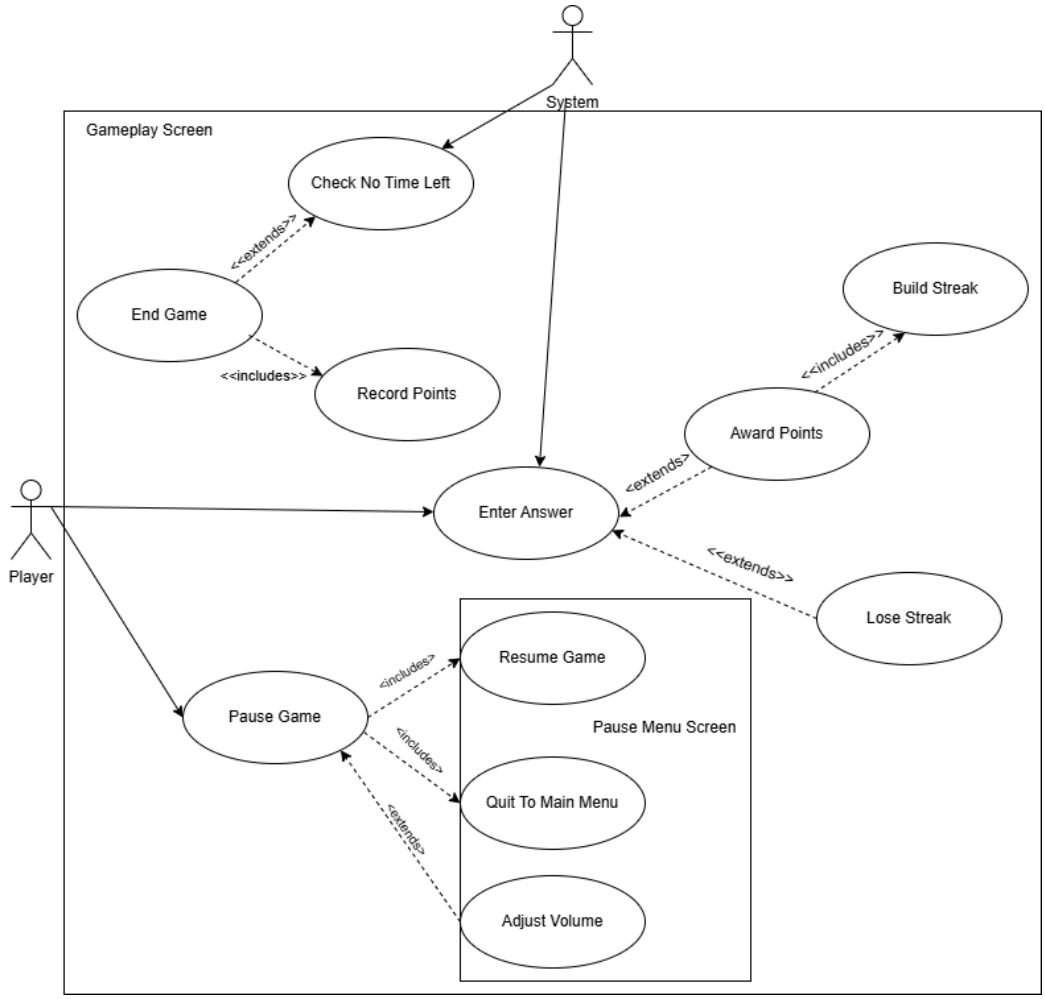

Figure 3: Use-Case Diagram for Gameplay Screen

Use Case Name:	Pause Game
Actors:	Player
Description:	Pauses the game session
Type:	Primary
Includes:	Resume Game, Quit to Main Menu
Extends:	None
Cross-refs:	Requirement 4
Uses cases:	Resume Game, Quit to Main Menu

Use Case Name:	Resume Game
Actors:	Player
Description:	Resumes the game session
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 4.3
Uses cases:	None

Use Case Name:	Quit To Main Menu
Actors:	Player
Description:	Ends the game session and returns the player to the main menu
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 4.4
Uses cases:	None

Use Case Name:	Adjust Volume
Actors:	Player
Description:	Adjusts the game’s volume
Type:	Primary
Includes:	None
Extends:	Pause Game
Cross-refs:	Requirement 4.2
Uses cases:	Pause Game

Use Case Name:	Enter Answer
Actors:	Player
Description:	Enters an answer to one of the equations shown on a creature, this answer will be checked to ensure it is correct
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 1.2.1
Uses cases:	None

Use Case Name:	Award Points
Actors:	System
Description:	Awards points to the player upon a correct answer
Type:	Primary
Includes:	Build Streak
Extends:	None
Cross-refs:	Requirement 1.2.2
Uses cases:	Build Streak

Use Case Name:	Build Streak
Actors:	System
Description:	Adds to the player’s current streak upon a correct answer
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 1.2.2.2
Uses cases:	None

Use Case Name:	Lose Streak
Actors:	System
Description:	Resets the player's current streak to zero upon an incorrect answer
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 1.2.4
Uses cases:	None

Use Case Name:	Check No Time Left
Actors:	System
Description:	Checks to see if the timer hits zero; when zero, the game will end.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	Requirement 1.1
Uses cases:	None

Use Case Name:	End Game
Actors:	System
Description:	Ends the game session and returns the player to the main menu, the player's score will be recorded if it falls within the top ten best scores
Type:	Primary
Includes:	Record Points
Extends:	Check No Time Left
Cross-refs:	Requirement 1.1.1
Uses cases:	Record Points, Check No Time Left

Use Case Name:	Record Points
Actors:	System
Description:	Record the player's score if it falls within the top ten best scores, add the score to the leaderboard if so
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 5.2
Uses cases:	None

4.1.2 Main Menu Screen Use-Case Diagram

Figure 4 provides a view of the primary actor, the player, and how they interact with the game's Main Menu Screen through entering answers or pausing the game. It also shows how the System actor interacts with the gameplay screen, managing what is being displayed and the responses to the primary actor's actions.

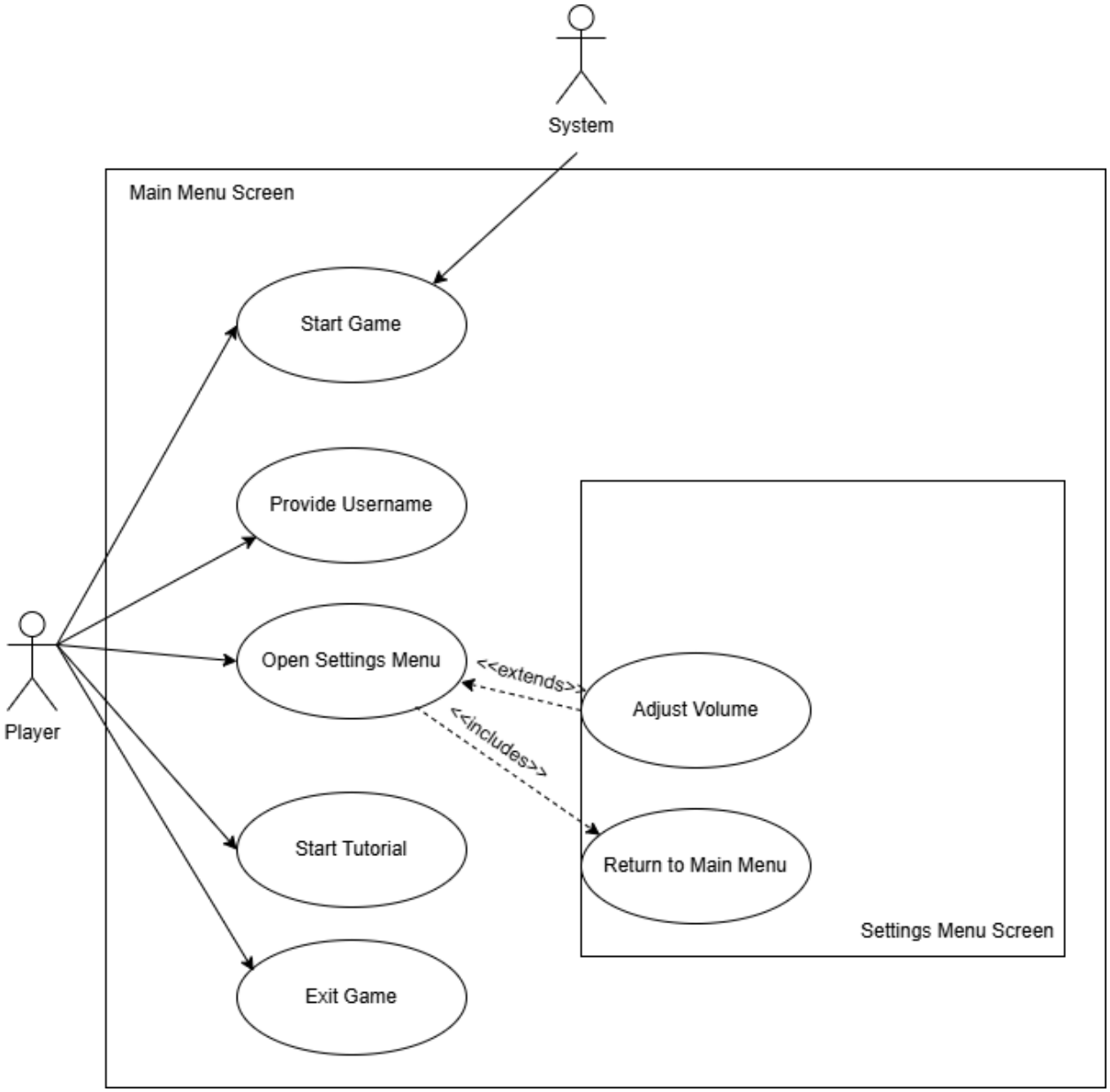

Figure 4: Use-Case Diagram for Main Menu Screen

Use Case Name:	Exit Game
Actors:	Player
Description:	Closes the game window
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 2.2
Uses cases:	None

Use Case Name:	Start Tutorial
Actors:	Player
Description:	Starts the tutorial
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 7
Uses cases:	None

Use Case Name:	Open Settings Menu
Actors:	Player
Description:	Opens the settings menu
Type:	Primary
Includes:	Return to Main Menu
Extends:	None
Cross-refs:	Requirement 3
Uses cases:	Return to Main Menu

Use Case Name:	Return to Main Menu
Actors:	Player
Description:	Returns the player to main menu
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 3.2
Uses cases:	None

Use Case Name:	Adjust Volume
Actors:	Player
Description:	Adjusts the game’s volume
Type:	Primary
Includes:	None
Extends:	Open Settings Menu
Cross-refs:	Requirement 3.1
Uses cases:	Open Settings Menu

Use Case Name:	Provide Username
Actors:	Player
Description:	Provides a username to display on the leaderboard
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 2.5
Uses cases:	None

Use Case Name:	Start Game
Actors:	System
Description:	Starts the game
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 2.2
Uses cases:	Provide Username

4.2 Class Diagram

Figure 5 displays information about all of the different classes that make up our game, Math Moles. Each includes information such as its name, fields, and methods. The GameManager class is the center control of our game, containing other classes which manage specific functions, like the Leaderboard class which contains all fields and methods pertaining to the leaderboard feature that keeps track of player highscores, and the ScoreManager class contains fields and methods pertaining to storing and modifying the game score. Each screen in our game is represented by its own class.

4.2.1 Class Diagram

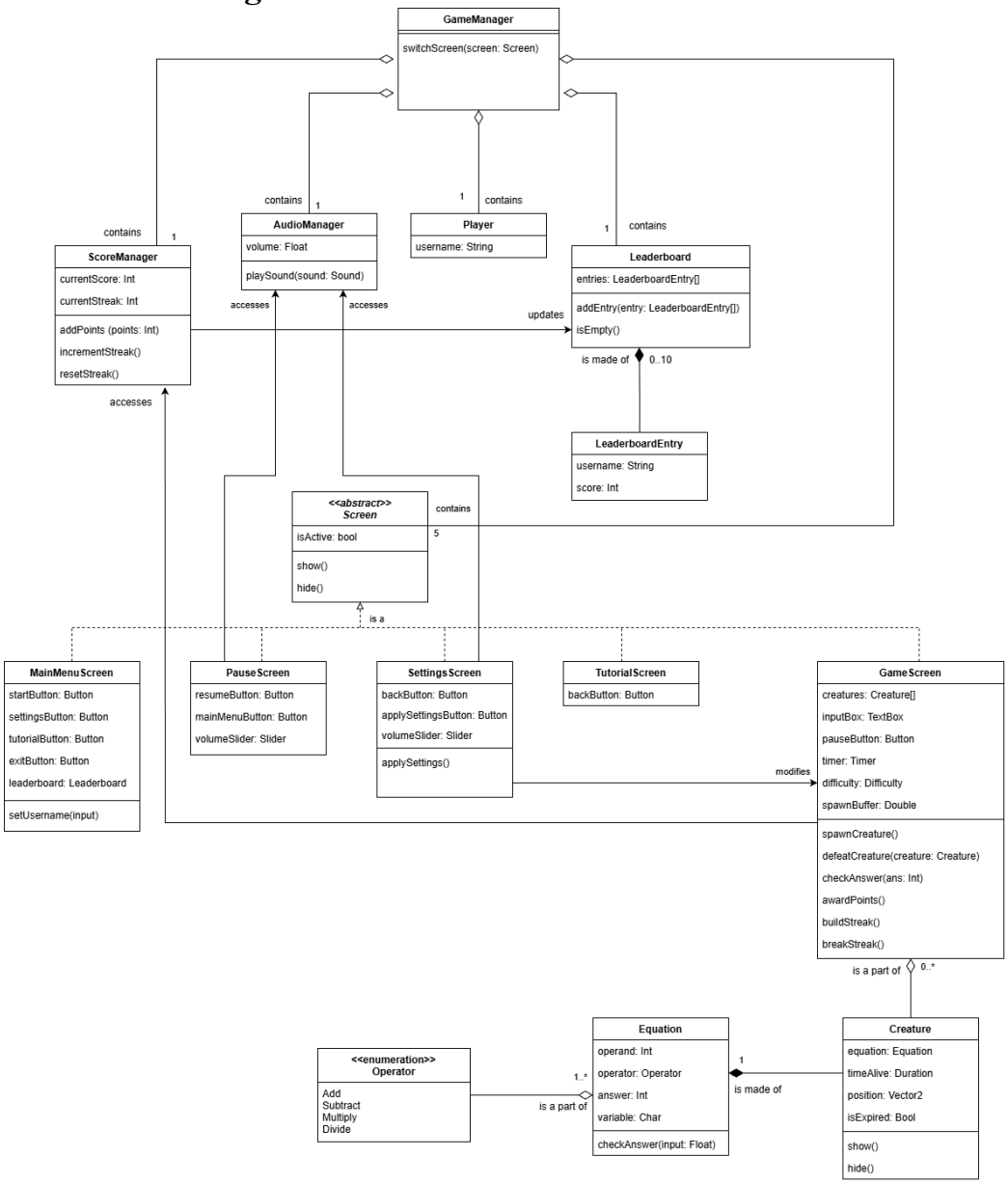

Figure 5: Class Diagram

4.2.2 Class Descriptions

Element Name		**Description**
AudioManager		Contains information related to game audio and also controls game audio.
Attributes		
	volume: Float	The volume at which game sound plays
Operations		
	playSound(sound: Sound)	Plays the sound associated with the given argument
Relationships	AudioManager is contained by GameManager, and is accessed by the PauseScreen and SettingsScreen	
UML Extensions	None	

Element Name		**Description**
Creature		The mole creatures that appear in the game and display math equations for the player to solve.
Attributes		
	equation: Equation	The mathematical equation which the creature will display
	timeAlive: Duration	How long the lifetime of the creature should be
	position: Vector2	The creature's position in the game
	isExpired: Bool	Signals when the creature's lifetime timer has ended
Operations		
	show()	Makes the creature appear above ground

		on the screen
	hide()	Makes the creature disappear below ground on the screen
Relationships	Creature has attributes made of Equation class objects. It is also part of the GameScreen class.	
UML Extensions	None	

Element Name		**Description**
Equation		Represents the equation which appear on screen for the player to solve
Attributes		
	operand: Int	The operand in the equation
	operator: Operator	The operator in the equation
	answer: Int	The answer to the equation
	variable: Char	The variable in the equation
Operations		
	checkAnswer(input: Float)	Checks if the input matches the equation's answer
Relationships	Equation is used as part of the Creature. Equation contains an Operator.	
UML Extensions	None	

Element Name		**Description**
GameManager		Contains information related to all game screens.
Attributes	None	None

Operations		
	switchScreen(screen:Screen)	Switch the display to the specified screen.
Relationships	Contains AudioManager, ScoreManager, Leaderboard, Player, and multiple Screens.	
UML Extensions	None	

Element Name		**Description**
Leaderboard		
Attributes		
	entries: LeaderboardEntry[]	A list of the scores which make up the game's leaderboard
Operations		
	addEntry(entry: LeaderboardEntry	Adds a new entry to the leaderboard
	isEmpty()	Returns true if there no scores have been entered into the leaderboard
Relationships	Leaderboard is contained by the GameManager. It is updated by ScoreManager. It is made of zero or more LeaderboardEntry.	
UML Extensions	None	

Element Name		**Description**
LeaderboardEntry		Class representing the entries of the leaderboard
Attributes		
	username: string	The username related to the current

		leaderboard entry
	score: int	The score of the current leaderboard entry
Relationships	The Leaderboard will contain zero or more LeaderBoardEntry objects.	
UML Extensions	None	

Element Name		**Description**
MainMenuScreen		Class representing the Main Menu Screen of the game.
Attributes		
	startButton: Button	Unity Button that allows the user to start a game session when pressed.
	settingsButton: Button	Unity Button that allows the user to access the settings menu when pressed
	tutorialButton: Button	Unity Button that allows the user to view the tutorial when pressed
	exitButton: Button	Unity Button that allows the user to close the game when pressed
Operations		
	setUsername(input)	Saves the given input as the player's username
Relationships	Inherits from the abstract class Screen.	
UML Extensions	None	

Element Name		**Description**
Operator		Enumeration for the type of arithmetic
Attributes		
	Add	Represents addition arithmetic
	Subtract	Represents subtraction arithmetic
	Multiply	Represents multiplication arithmetic
	Divide	Represents division arithmetic
Relationships	One or more operators are a part of an equation.	
UML Extensions	Is a part of Equation.	

Element Name		**Description**
PauseScreen		Class representing the pause menu screen of the game
Attributes		
	resumeButton: Button	Unity Button that allows the user to resume the game session when pressed.
	mainMenuButton: Button	Unity Button that allows the user to return to the main menu when pressed.
	volumeSlider: Slider	Unity Slider that allows the user to set the volume of the game
Relationships	Inherits from the abstract class Screen.	
UML Extensions	None	

Element Name		**Description**
Player		
Attributes		
	username: String	Stores the username of the player as a string
Relationships	GameManager contains 1 Player class (composition)	
UML Extensions	None	

Element Name		**Description**
ScoreManager		
Attributes		
	currentScore: Int	Value representing the player's current game score
	currentStreak: Int	Value representing the player's current game streak
Operations		
	addPoints(points: Int)	Increments the player's points by the provided amount
	incrementStreak()	Increments the player's streak by 1
	resetStreak()	Resets the player's streak to zero
Relationships	GameManager contains a ScoreManager object. ScoreManager updates the Leaderboard.	
UML Extensions	None	

Element Name		**Description**
Screen		An abstract class from which other screen classes are derived
Attributes		
	isActive: Bool	True if the screen is being shown, False otherwise
Operations		
	show()	Shows the screen
	hide()	Hides the screen
Relationships	Screen is an abstract parent class to all other screen objects. It is contained by GameManager.	
UML Extensions	None	

Element Name		**Description**
SettingsScreen		Contain information and controls pertaining to game settings
Attributes		
	backButton: Button	Exits the settings screen upon being pressed
	applySettingsButton: Button	Clicked when the user wants to save the setting changes they made
	volumeSlider: Slider	Sets the volume value
Operations		
	applySettings()	Updates the settings with new values
Relationships	SettingsScreen is derived from the Screen class. It accesses the AudioManager class.	

UML Extensions	None

Element Name		**Description**
TutorialScreen		
Attributes		
	backButton: Button	Exits the tutorial
Operations		
	handleInput()	Manages input from the user while navigating the tutorial
Relationships	TutorialScreen is derived from the Screen class.	
UML Extensions	None	

4.3 Sequence Diagrams

Figure 6 and Figure 7 describe two different scenarios that may occur while using our game. The first scenario shows a user playing the game by entering answers, and the second shows the game asking the user for a username. In the following subsections, these diagrams will be accompanied by a description of the processes they are displaying.

4.3.1 User Playing the Game

Figure 6 shows the process of the user playing a game session. It begins with the timer starting to count down at the beginning of the game, and after it does, moles with attached equations will begin to pop up on screen each time the game's spawn buffer time elapses. Moles continue appearing until the game timer runs out, and their lifetime timer begins counting down as soon as they appear. The player can enter answers into the answer textbox at any time while the game has not ended. Each time the player enters an answer into the textbox, the game will check if the answer matches any active mole's equations. If the entered answer matches one or more active mole's equations, those moles will be defeated and the game will award points to the player and increase their streak. If the entered answer does not match any active mole's equation, the game will reset the player's streak and no points will be awarded. Each time a mole's lifetime

timer reaches its end, the mole will disappear with no effect on player score or streak. When the timer runs out, the game will automatically end.

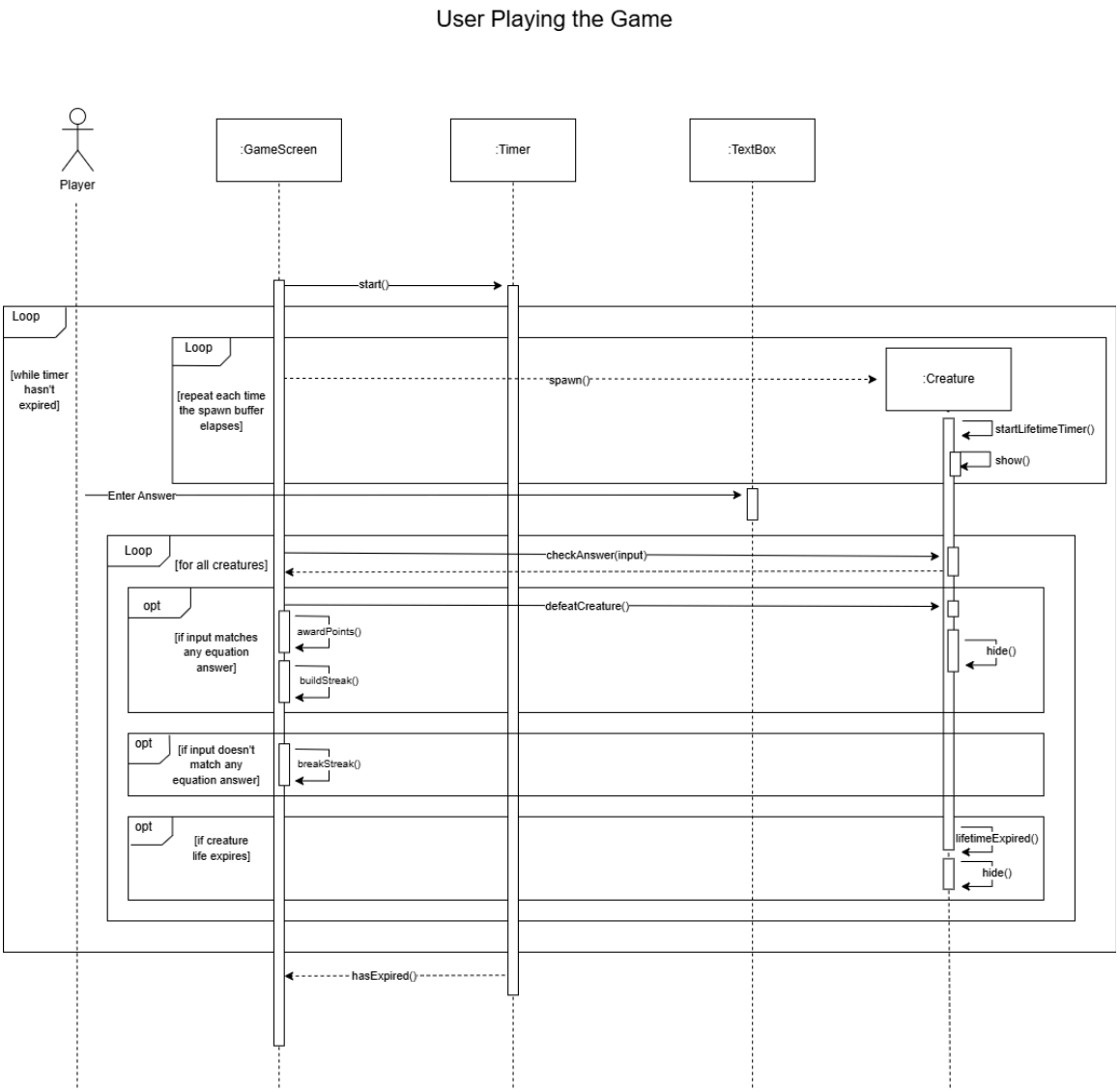

Figure 6: User Playing the Game Sequence Diagram

4.3.2 Game Asking for Player Username

Figure 7 shows the process of the player entering a username on the Main Menu screen. The player enters text into the provided space on the screen, and the game saves and uses the provided name for new entries on the leaderboard.

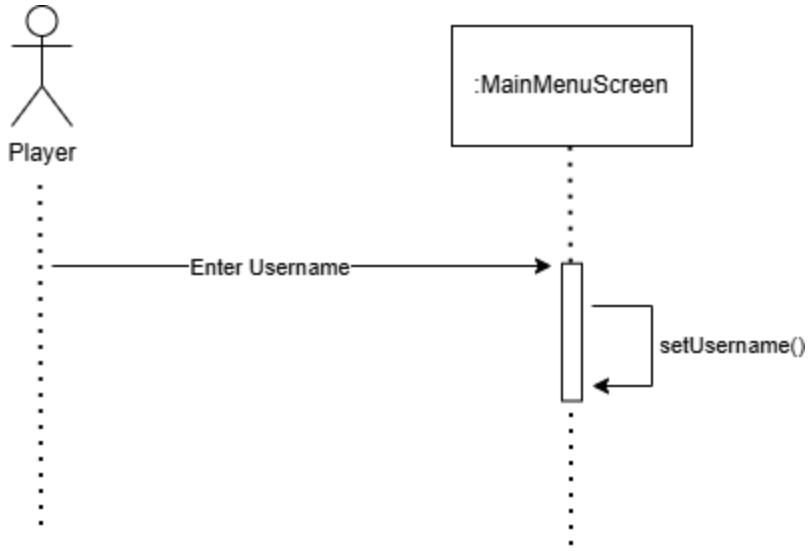

Figure 7: Game Asking for Player Username Sequence Diagram

4.4 State Diagram

Figure 8 illustrates the major operational states of the game and the transitions that occur as the player interacts with the game.. The system begins at the Start state, which immediately transitions to the Main Menu. From the Main Menu, the player may choose to start a new game session, access the tutorial, open the settings menu, or quit the game entirely.

When the player selects Play Game, the system transitions into the Equations/Moles Pop Up state. In this state, moles appear with mathematical equations, and the player enters answers. The entered answer moves the system to the Check Answer state, where the system evaluates correctness. If the answer is correct, the system transitions to the Mole Whacked / Points Awarded state, updates the score and streak, and then returns to the Equations state. If the answer is incorrect, the system returns to the gameplay loop with a reset streak.

During gameplay, the player may choose to pause, which transitions the system to the Pause Menu. From the Pause Menu, the player may resume the game (returning to the Equations state), return to the Main Menu, or quit the game entirely.

From the Main Menu, the player may also enter the tutorial or settings states. Both states allow the player to navigate back to the Main Menu via a “Back” action. At any point where a “Quit” action exists, the system transitions to the Exit state, ending the application.

Overall, the state diagram shows a cyclic gameplay loop controlled by user input, surrounded by navigable menu states that serve as entry points and configuration areas. It

captures the high-level flow of the entire game experience, from initial startup through gameplay and eventual exit.

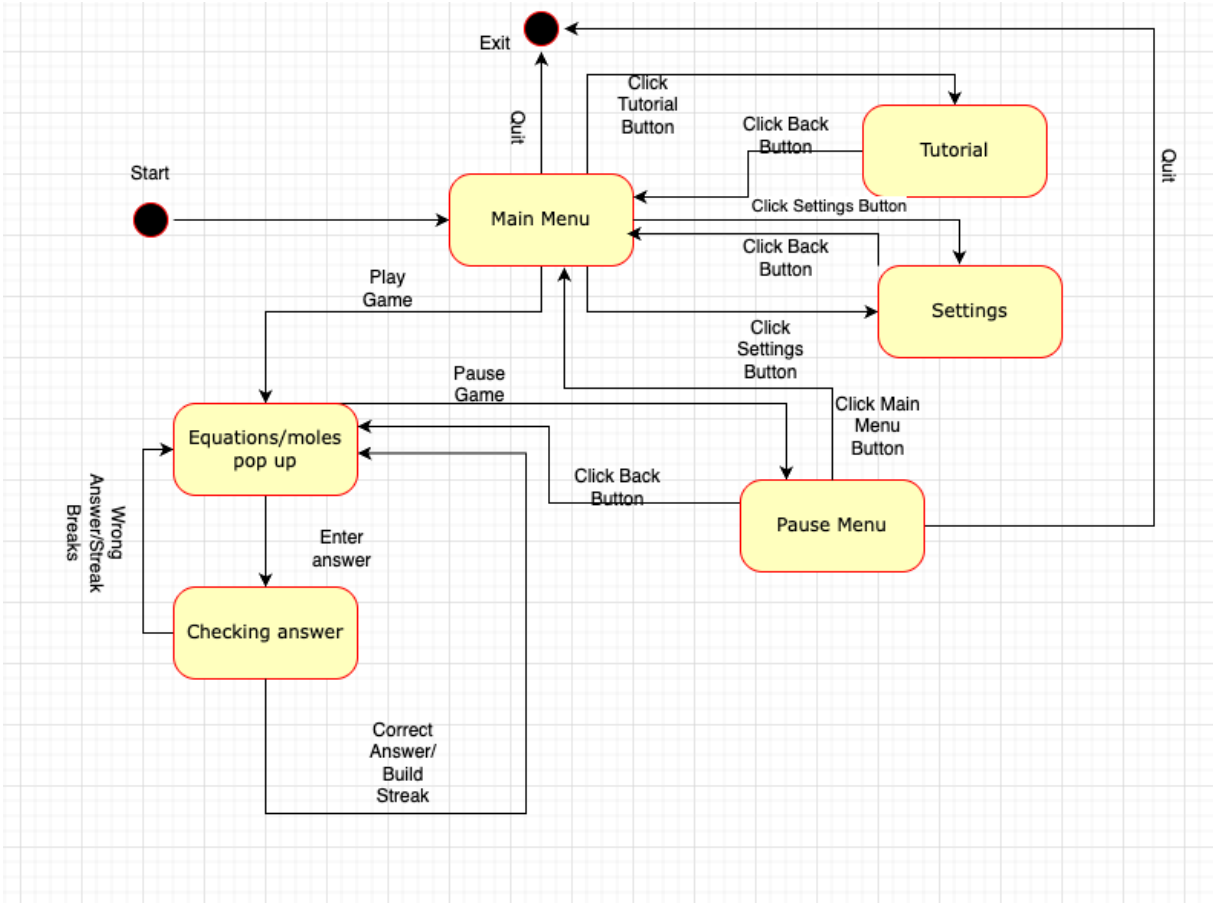

Figure 8: State Diagram

5 Prototype

The prototype features a functioning Main Menu including buttons to play, access settings, view tutorial, and quit. There is also an input box and leaderboard in the Main Menu, allowing the player to enter their desired username before starting the game if they would like to save their high score. The player's scores will be added to the leaderboard after a game session under this entered username.

Upon hitting the Play Button, the player will be brought to the gameplay scene where moles will begin to appear with equations for the player to solve. As the player enters their answer, the numerical input appears on screen. Pressing the Enter key submits the answer, prompting a message to show up to indicate whether any of the on-screen equations were solved correctly.

At any time during gameplay, the player can pause by pressing the "Pause" button in the top-left corner. This opens up a menu with options to unpause, open settings, or return to the Main Menu. The settings menu accessed from the pause menu allows the player to adjust the volume and then return to the pause screen using a back button.

Pressing the "Tutorial" button from the main menu brings the player to a menu consisting of sequentially ordered slides with visuals accompanied by text to explain how the game works.

Choosing Quit from the Main Menu closes the game.

5.1 How to Run Prototype

To run the prototype, the user must have a Windows machine with a keyboard and mouse. The game itself is very small and is not graphically intensive so a lower end PC or laptop can run it smoothly. The game can be downloaded through our website: https://mangorox.github.io/swe1-g10-website/. On the home page, click "Download Latest" to download the game. Otherwise, the user may click "Install Guide" for more details on how to go through the whole installation process if they need to.

5.2 Sample Scenarios

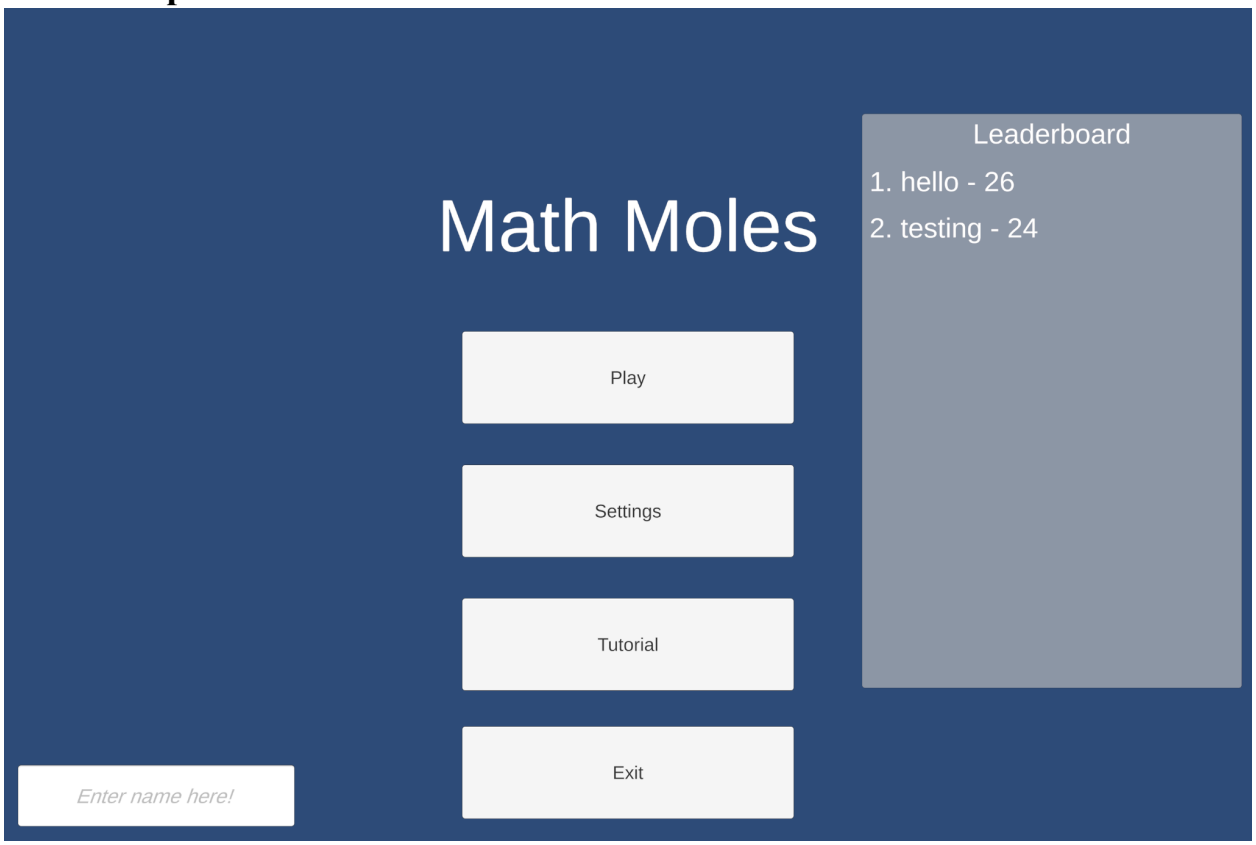

Figure 9: Main Menu

When first opening the game, the player will see a screen similar to the one depicted in Figure 9. From here, they can navigate to the tutorial, settings, or they can begin playing. They may optionally enter a name on the bottom left which the game will remember, saving it on the local leaderboard alongside a highscore, if the player reaches one.

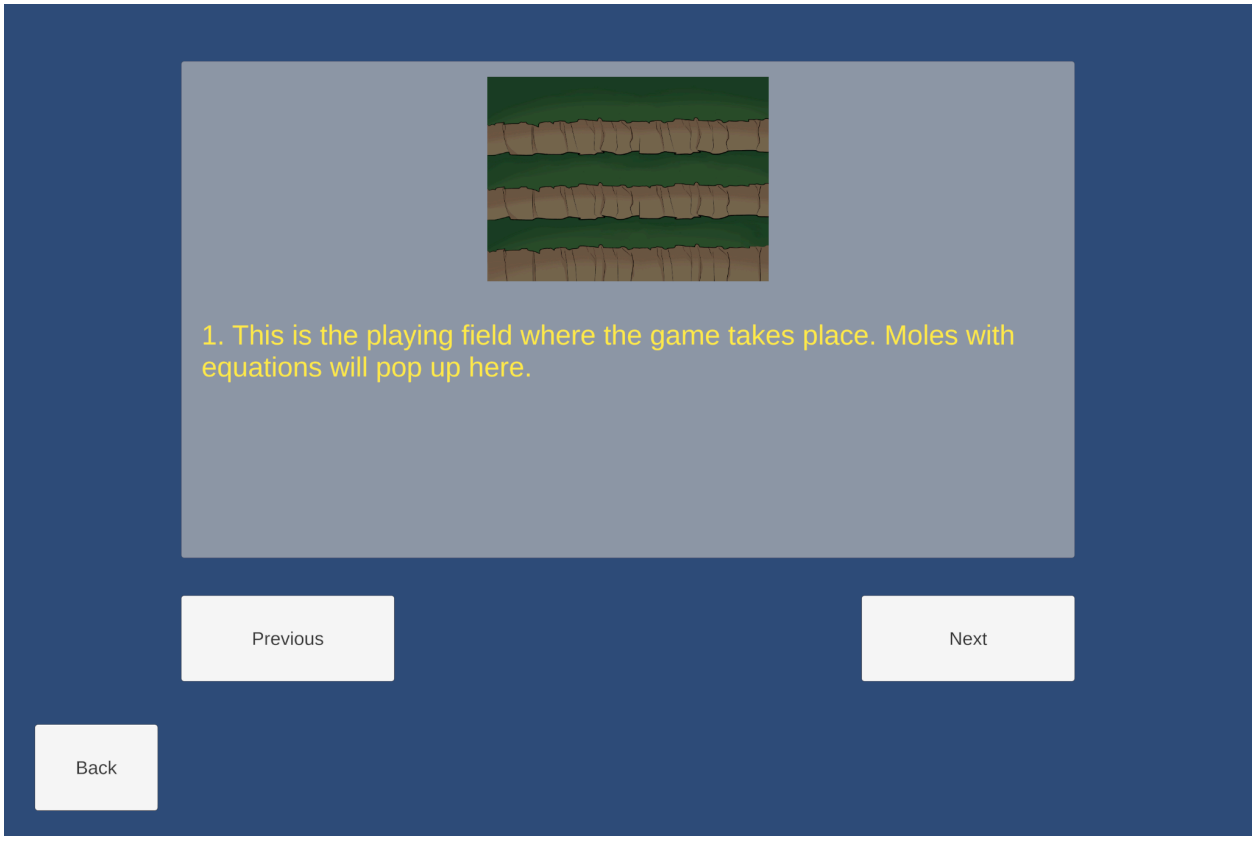

Figure 10: Tutorial

The player can navigate from the Main Menu to the Tutorial to learn how to play, as shown in Figure 10. Clicking the next button will advance to the next instruction.

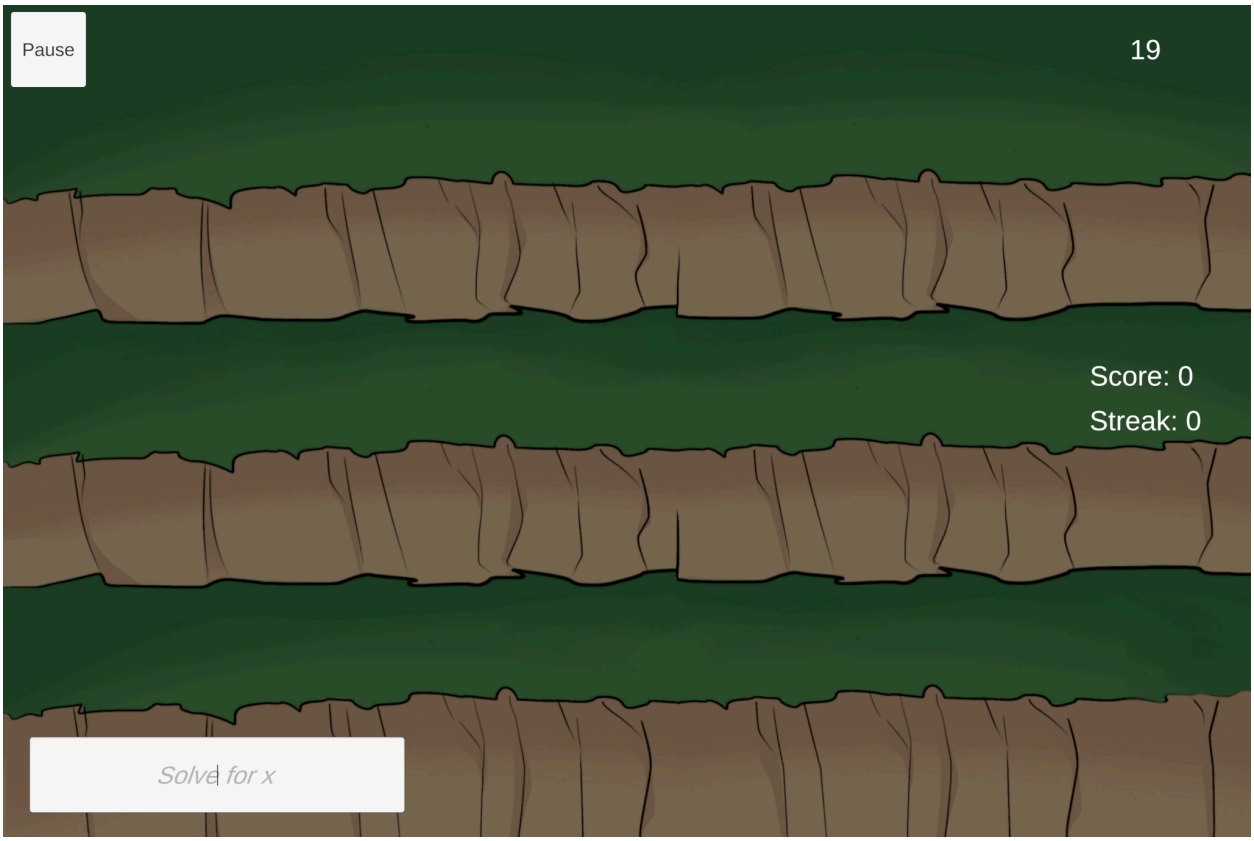

Figure 11: Playing the Game

When first starting the game, the player will see a screen like the one shown in Figure 11. The timer in the top right will begin counting down immediately.

Figure 12: Mole Appears

Once the timer has started, moles will begin to appear with equations over their heads. The player must solve for x and enter the answer into the box in the bottom left corner, as seen in Figure 12.

Figure 13: Correct Answer Entered

When the player enters a correct answer, all the moles with an equation matching the answer will be whacked and then disappear. The player will be awarded points and their streak will increase.

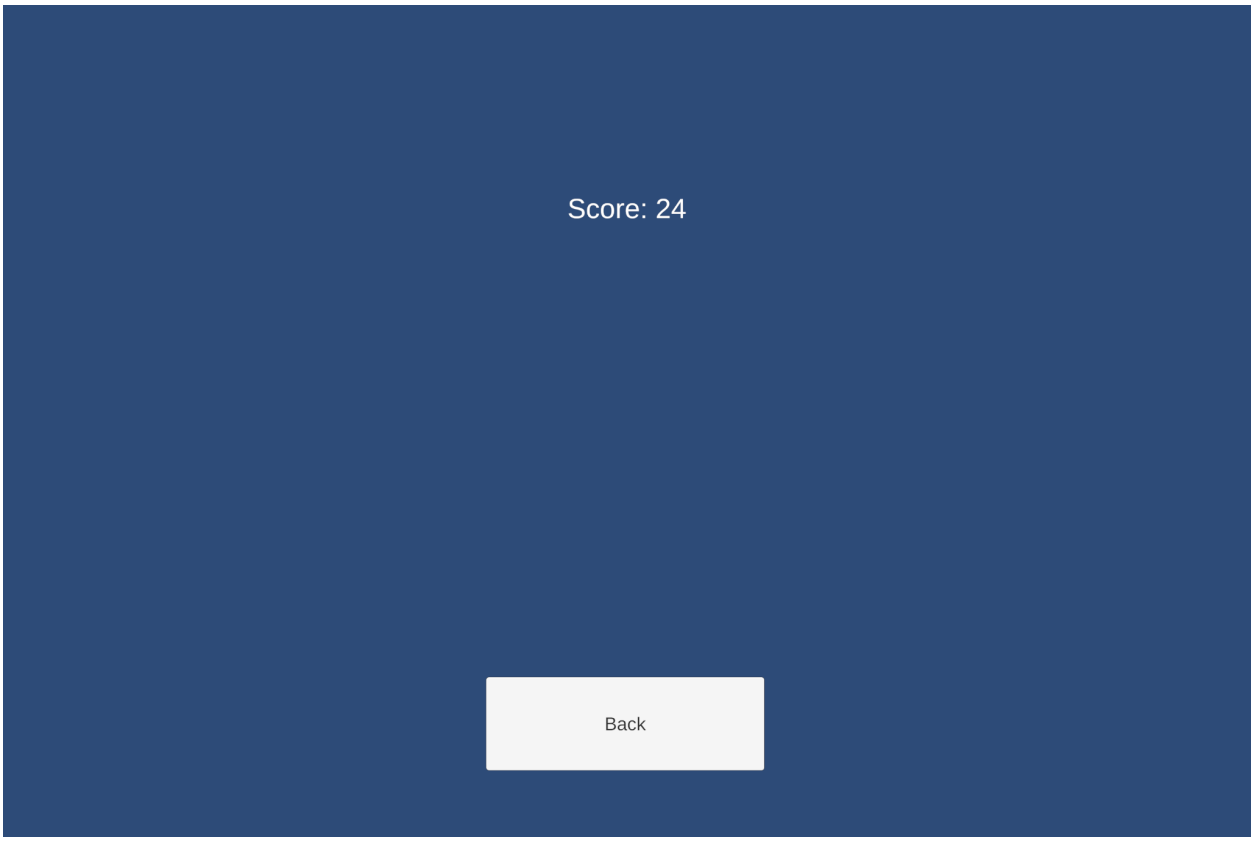

Figure 14: Game Ended

When the timer runs out the game will end, and the player's score will be displayed on screen, as shown in Figure 14. They can return to the Main Menu by clicking the Back button.

6 References

[1] Website “Mole Game”, 2025 https://mangorox.github.io/swe1-g10-website/

[2] Unity Technologies, “Unity (Version 6.2, 6000.2.10f1) [Game Engine],” 2025. https://unity.com/

[3] Unity Technologies, “Unity Documentation,” 2025. https://docs.unity.com/en-us

[4] 2017 Curriculum Framework for Mathematics Detailed Revisions of 2010 Standards for PK-12. https://www.doe.mass.edu/frameworks/math/2017-06revisions.pdf

[5] Game Sprites and Background. Natalie Resendes, natalie_resendes@student.uml.edu

[6] Game sounds are from https://pixabay.com/

a. Hammer hit sound: https://pixabay.com/sound-effects/cartoon-bonk-sfx-424181/

b. Incorrect answer sound: https://pixabay.com/sound-effects/wrong-47985/

c. Correct answer sound: https://pixabay.com/sound-effects/correct-choice-43861/

d. General UI sound: https://pixabay.com/sound-effects/button-09-190435/

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.