Software Requirements Specification (SRS)
Super Math Maker

Team: 4
Authors: Nick Haselton, Geoffrey Higgins, Adam Corkhum, Daniel Stark, Christopher Jimenez
Customer: 4th-8th Graders
Instructor: Dr. Daly

1 Introduction

This document is a software requirement specification (SRS) for Super Math Maker which describes what the software does and how it operates.

1.1 Purpose

The purpose of this software requirements specification document, is to provide information on the requirements and functionality of this project, Super Math Maker (not final), and how it will play. This document will contain a requirements list, diagrams outlining its functionality, and descriptions on the game's purpose, features, and constraints. This document is intended for the developers of the project to have a concise blueprint and guide for how the game shall function and what it shall contain. This document is also for the clients (Dr. Daly) so that they may review and understand how the work is being completed and that they understand the workings and design of the project.

1.2 Scope

The scope of Super Math Maker is to have 3 different types of questions for the user to choose from. Once chosen they will get a quiz of randomly assigned questions that fit into their desired category. With the money gained from the quiz, the user can buy multiple types of assists. Once they are in the gameplay stage, there will be 3 types of enemies and 2 types of traps in their way of completing the 3 levels ahead of them by getting to the goal.

1.3 Definitions, acronyms, and abbreviations

Super Math Maker: The title of the 2D platformer game that reinforces math skills

User: The person interacting with the game

Character: The User's character in game

Level: A course which the character begins in the far left and must reach the end to progress

Life: Keeps track of how many times the character is able to take a hit before the game over screen

Hitbox: An object that checks for collision with the character and if so a life is taken away

Platform: An object which the character can stand on

Pitfall: An gap which results in the character dying if they fall in

Assist: A placeable object from the character which helps them complete the level

Trap: A Stationary object that kills the character if collided with

Enemy: Dynamic Entities that the character must avoid

Goal: Marks the end of a level

Power-up: Bonus effects that help the character such as increased jump or movement speed

Points: Currency earned from the user answering questions correctly on the math quiz that are spent in the shop

Shop: Place where the user can purchase assists or power-ups using their points

Level Editor: Phase of the level where the user can purchase power-ups or assists from the shop and place them on the level

UI: User Interface

1.4 Organization

1. Introduction - Overview of the SRS document
    - 1.1 Purpose - Objective of the document
    - 1.2 Scope - Intended function of the application
    - 1.3 Definitions - Describes terms used in the document
    - 1.4 Organization - Lays out the structure of the document
2. Overall Description - Overview of Section 2
    - 2.1 Product Perspective
    - 2.2 Product Functions
    - 2.3 User Characteristics
    - 2.4 Constraints - Restrictions of the project
    - 2.5 Assumptions and Dependencies - Assumptions of the software and user
    - 2.6 Apportioning of Requirements - Diagrams of how project functions
3. Specific Requirements - Requirements of the project
4. Modeling Requirements
    - 4.1 Use Case Diagram
    - 4.2 Class Diagram
    - 4.3 Sequence Diagrams
5. Prototype
    - 5.1 How to Run Prototype
    - 5.2 Sample Scenarios

6. References
7. Point of Contact

2 Overall Description

Super Math Maker is an educational video game centered around teaching math to students in upper elementary school. The game focuses on math topics such as fractions, multiplication, division and simple geometry. Super Math Maker is a 2-dimensional platformer where the user gets to add elements to the level in order to make it easier to complete.

The user will select a topic upon launching the game and be put into a math quiz where they will be given a question and for every correct response they submit within a time limit they will be given in-game money which can be spent in the edit phase of the level. The user can spend their money in the shop which contains objects that support the character such as springs and platforms. They can also spend their money on extra lives or decide to keep money for the next level. Each level contains a goal where the character must reach before running out of lives. They must navigate through a number of obstacles including different types of enemies and hazards such as lava and pitfalls. Upon completing the level the user will be put into another math quiz where they can earn money to spend on the next level.

2.1 Product Perspective

Super Math Maker is an educational platform game targeted towards late elementary to early middle school students with a desire to learn math while still doing something they enjoy, gaming. This game works by quizzing the students on various math concepts which rewards them with points they can spend to add objects to each subsequent level in the platformer. This makes the game replayable and open to creative, out of the box solutions to each level.

The game is designed to run on low-end computers using the Microsoft Windows operating system. Its wide range of hardware accessibility will ensure that most school computers will be able to run the game. The game only requires a keyboard and mouse in order to play. After launching the game, users will be greeted by a minimalist user interface that emphasizes getting right to the action. Users will then select from various topics ranging from 4th to 6th grade mathematics, following age-appropriate guidance from the Massachusetts Department of Elementary and Secondary Education.

Super Math Maker emphasizes learning – when users get an answer wrong, they will be shown the correct answer, as well as how to arrive at said answer. The satisfying gameplay loop ensures kids will try to achieve the highest score possible in order to maximize their chances at succeeding in each level. At the beginning of each level, they will start with three lives and whatever upgrades they were able to attain from the shop, which they will then be able to use accordingly. Upon completion, they will be shown their total score, which will give users a sense of pride and accomplishment if they are able to beat their peers.

2.2 Product Functions

Main Menu

- The user is brought to the main menu when the application is launched.
- The main menu contains a button to start the game and one that exits the application.
- When the start button is pressed the user selects one of three math topics; pre-algebra, geometry or fractions.

Math Quiz

- The quiz will begin when the user selects a math topic. The user is given a set amount of time to answer as many questions correctly as possible.
- Questions will be selected based on the topic the user has chosen on the menu.
- Each question will be one of a few possible formats (ex. a geometry question may be one asking to calculate area or perimeter).
- The values of the question will be randomly generated.
- The solution to the question will be entered in by the user and the program will check if the value is correct and then another question appears.
- If the answer is correct then the user gains points to be used on the shop, otherwise they gain no points and the next question is presented.
- The quiz will continue to present questions until the timer on the quiz reaches zero at which point the game proceeds to the level editor and shop section, where the user can use the points they earned on the quiz.

Level Editor

- Upon finishing the quiz the user will enter the level editor phase.
- This phase consists of a shop where the user can purchase placeable objects or other power-ups along with the level which the user can inspect and look through.
- The user can drag objects from the shop onto the screen, spending points and placing the new objects on to the level they will soon play.

Level Gameplay

- The character spawns on the far left side of the level and moves around using the WASD keys and the spacebar key to jump.
- The character must reach the goal at the end of the level to win the level.
- If the character collides with an enemy or hazard they will lose a life and be reset back to the beginning of the level, where they may make changes to the level.
- If the character runs out of lives they are presented with a game over and returned to the main menu.
- The escape key can be pressed by the user to pause the level at any time and will allow the user to quit the game, return to the main menu, or resume the game.

2.3 User Characteristics

The game will target kids in the school grades of 4th through 8th who have an understanding of the math topics that they've been taught. The user should have a basic understanding of using a keyboard and mouse as well as using WASD keys to move. They should have mathematical skills of at least an elementary level. Users do not need to be skilled at platformers in order to complete this game. The ability for the user to add assists to each level means that they can place them in areas that they may not be able to complete otherwise.

2.4 Constraints

Regulatory Constraints:

- The math topics in the game must fall into the 4th and 8th grade range of the Massachusetts Mathematics Curriculum Framework.

Design Constraints:

- The game must have an educational component that revolves around mathematics.
- The game must be easily accessed and navigated by elementary school students.

Hardware Constraints:

- The game is required to support low end hardware
- The game must not use intensive graphics techniques.
- The game must be played using a keyboard and mouse.

Software Constraints:

- The game must be made using the Godot game engine.
- The game must be played on Windows, Mac, or Linux.

2.5 Assumptions and Dependencies

Assumptions of the user's hardware:

- The user has a functional and modern computer
- The user has keyboard and mouse peripherals connected to their computer

Assumptions of the user's software:

- The user runs a modern operating system on their computer (e.g. Windows 10, MacOS Sonoma)

Assumptions of the user:

- The user has math skills taught in at least the 4th grade in accordance with the Massachusetts Common Core.
- The user is familiar with navigating software menus.
- The user is familiar with the objective of platformer games.

2.6 Apportioning of Requirements

There are features that would serve to benefit the game, but are out of the scope of this project and will be scheduled for a future release. The features include customizability for the character in the game, such as different outfits and accessories such as hats. A wider variety of assists such as sidekicks will be added that protect the character from threats. New math topics will be implemented to test the user's knowledge further like Algebra and Geometry. Finally, a future release will include a greater quantity of levels that users can build upon to reach the goal.

3 Specific Requirements

1. The game will consist of hardware requirements for what machines can run it.
   1.1. Keyboard and mouse will be required.
   1.2. The game will export to Windows as an executable file.
   1.3. The game will output to a display with a minimum resolution of 800x600
2. The game will have a series of menus that navigate through the game screens.
   2.1. The game will have a main menu screen.
      2.1.1. The user will automatically load into the main menu when the game is launched.
      2.1.2. The user will be able to choose between playing the game, changing the settings or quitting the application.
      2.1.3. When the user plays the game they will be required to choose what math topic to practice.
   2.2. A Graphical User Interface (GUI) will exist to display options within the game to the user.
      2.2.1. The GUI will allow the selection of one mathematical topic, from a list of mathematical topics, based on their grade level.
      2.2.2. The GUI will allow scrolling to see different areas of the level.
   2.3. The game will have a settings menu.
      2.3.1. The settings menu will allow adjustment of SFX and music volume.
3. There will be an educational aspect to the game that is focused on mathematics.
   3.1. Each level will begin with a mathematics quiz which contains questions based off of the selected mathematical topic.
      3.1.1. Equations will be used for each quiz question that tests user's on mathematical topics in the 4th grade to 8th grade range, according to the Massachusetts Mathematics Curriculum Framework.
      3.1.2. Questions will be chosen randomly at runtime from a database of equations, depending on the chosen topic.
      3.1.3. Points will be awarded for each correctly solved question, with difficult questions granting more points.
      3.1.4. An incorrect solution to a question will deduct points.
      3.1.5. The quiz will feature an input parsing mechanism.
         3.1.5.1. Answers to quiz questions will be parsed in such a way as to avoid correct answers being marked incorrect (i.e. inputting "3.5" instead of "3.50" will not be counted as an incorrect answer).
4. A series of gameplay features will exist that assemble to create a two-dimensional platformer.
   4.1. Platforms will exist in each level that create a path through the level.
   4.2. Each level will contain a user controlled entity known as the character.
      4.2.1. The character will have the ability to perform a series of movement options.
         4.2.1.1. The character will be able to move left, right, and can jump.

4.2.1.2. The character will be able to wall slide.
4.2.1.2.1. Wall Sliding will occur when the character is airborne after a jump and comes in contact with a platform wall.
4.2.1.2.2. The character enters a wall jump state when airborne and pushing into a wall
4.2.1.2.2.1. Wall jumping will occur when the character jumps while in a wall slide state.
4.2.1.3. The character will be able to crouch to make the hitbox half size.
4.2.1.4. All movements by the character will be animated.
4.2.2. The character will have the ability to melee attack, dealing damage to enemies within a short range.
4.2.2.1. Melee attacks will be able to be directed up, down, left and right.
4.2.3. The character will spawn with a certain amount of lives.
4.2.3.1. The character will be able to die in a level.
4.2.3.1.1. Death will occur through contact with threats on a level.
4.2.3.1.2. Death in a level will decrease the life count by one.
4.2.3.1.2.1. When the character dies with zero lives remaining, the game will return to the main menu.
4.2.3.1.2.2. When the character dies with more than zero lives remaining, the game will return the character to the beginning of the same level.
4.2.3.2. Lives will carry over between each completed level.
4.3. A goal will exist in each level to signify that the character has completed the level.
4.3.1. Some levels will have secret alternative goals that can be found through exploration.
4.4. Threats will exist in each level that will deal damage to the character when their hitboxes collide.
4.4.1. Threats will exist in each level of the game.
4.4.1.1. Enemy threats will be a type of threat that are able to move and attack.
4.4.1.1.1. There will be various types of enemies that each behave differently.
4.4.1.1.2. Enemies will change their behavior when in the presence of the character.
4.4.1.2. Environmental threats will exist in each level of the game.
4.4.1.2.1. Environmental threats will be able to be static.
4.4.1.2.2. Environmental threats that move will consistently follow a predetermined path.
4.5. Assists will be able to be placed on the level by the user.

4.5.1. Assists will consist of springs, teleporters, extra platforms, power-ups, etc..

4.6. A point system will exist that contains a count of the number of points owned by the user.

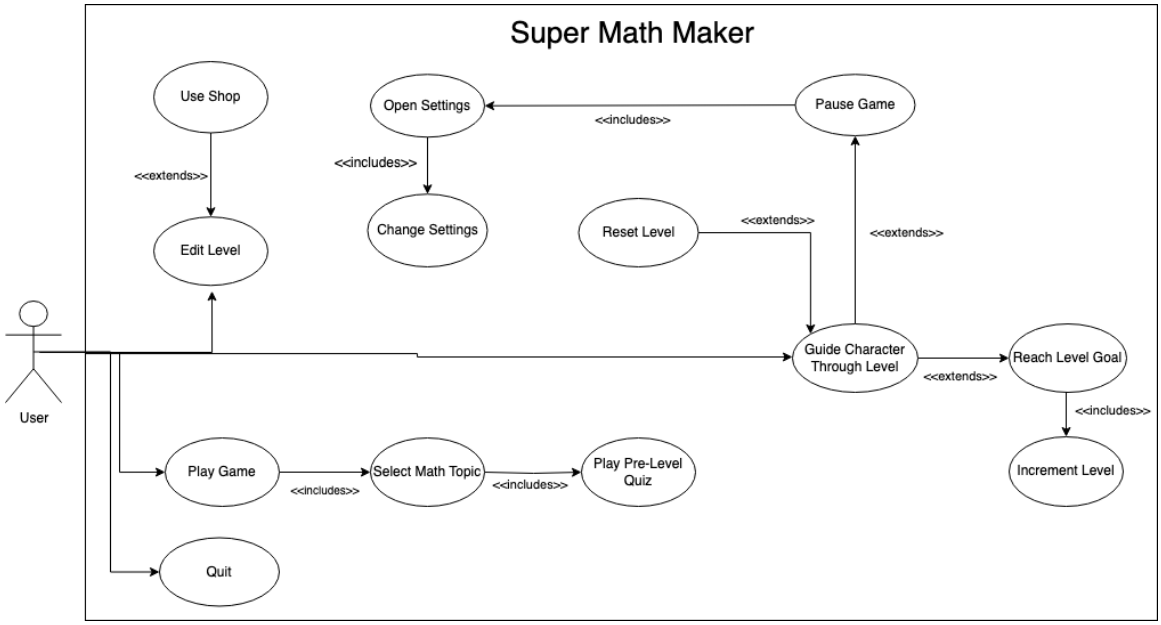
4.6.1. Points will be awarded on equation completion.

4.7. A shop will be available to the user to spend points on assists and extra lives.

4 Modeling Requirements

4.1 Use Case Diagram

The use case diagram contains cases that will occur within the game, Super Math Maker and their relation to each other. Each use case is shown within its own oval and relations are shown between use cases with arrows and descriptors, includes and extends. Each use case is then outlined within the charts below the diagram.



Use case diagram.

Use Case Name:	Play Game
Actors:	User
Description:	Initiates the gameplay loop by pressing play game on the menu.
Type:	Primary
Includes:	Select Math Topic
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Quit
Actors:	User
Description:	Closes the game when this option is chosen from the main menu.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Select Math Topic
Actors:	User
Description:	User chooses a topic for which the game's timed quizzes will focus on.
Type:	Secondary
Includes:	Play Pre-Level Quiz
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Play Pre-Level Quiz
Actors:	None
Description:	A timed quiz will begin where questions will be presented and the user will attempt to answer them until the timer reaches zero, at which time the quiz will close and reward points for questions completed.
Type:	Secondary
Includes:	Edit Level
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Edit Level
Actors:	None
Description:	The user will drag and drop various assets onto the platforming level to create their own solutions for the level.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Use Shop
Actors:	None
Description:	The user will be presented with a shop full of assets with prices for each. The user will purchase assets from this shop that can then be dragged and dropped onto the level via the edit level case.
Type:	Secondary
Includes:	None
Extends:	Edit Level
Cross-refs:	None
Uses cases:	None

Use Case Name:	Guide Character Through Level
Actors:	None
Description:	The user will now control their character and attempt to reach the goal by using the controls allocated to the character, while avoiding threats on the level.
Type:	Primary
Includes:	Death via Threat Collision
Extends:	None

Use Case Name:	Guide Character Through Level
Actors:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Reach Level Goal
Actors:	None
Description:	When the character collides with the goal object the level will complete and the user will have won the level.
Type:	Secondary
Includes:	Increment Level
Extends:	Guide Character Through Level
Cross-refs:	None
Uses cases:	None

Use Case Name:	Increment Level
Actors:	None
Description:	The user selects the next level for the game to load.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Reset Level
Actors:	None
Description:	This will reset the character back to the start of the level to ensure the character cannot get indefinitely stuck or if the user would like to rearrange placed assets from the edit phase.
Type:	Secondary
Includes:	None
Extends:	Guide Character Through Level
Cross-refs:	None
Uses cases:	None

Use Case Name:	Pause Game
Actors:	None
Description:	When activated, the level will no longer proceed until the user deactivates this function. This will also bring up a menu for the user to either quit the game or edit the settings.
Type:	Secondary
Includes:	Open Settings
Extends:	Guide Character Through Level
Cross-refs:	None
Uses cases:	None

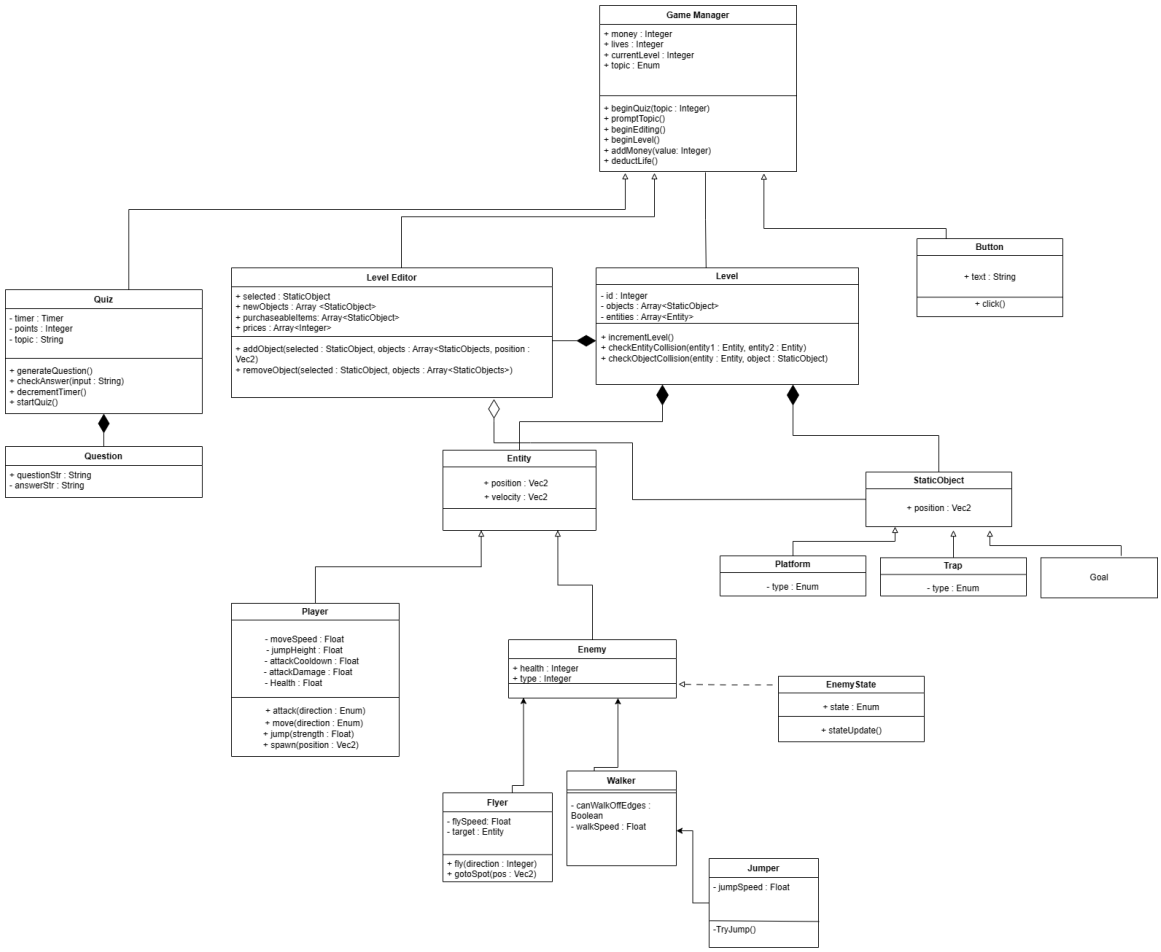
Use Case Name:	Open Settings
Actors:	None
Description:	When activated a settings menu will open with options to edit various game settings.
Type:	Secondary
Includes:	Change Settings
Extends:	None

Use Case Name:	Open Settings
Actors:	None
Cross-refs:	None
Uses cases:	None

Use Case Name:	Change Settings
Actors:	None
Description:	When the user changes settings the game will change accordingly to these changes.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None

4.2 Class Diagram

The class diagram contains descriptions of each class that will make up Super Math Maker, by outlining the title, fields, and methods of each class. The relationships of these classes to one another are also shown through various arrows connecting each class. Black diamond arrows show composition relations, white diamond arrows show aggregation relations, and traditional arrows show an inheritance relationship.



Class diagram.

Class Name	Game Manager		
Description	Singleton that handles all persistent data. Changes and keeps track of game state.		
Extends	None		
Attributes	Money	Integer	Currency used for buying assists.
	Lives	Integer	How many times the character can take damage before having to restart the level.
	CurrentLevel	Integer	Index into levels to find where they currently are
	Topic	Enum	What the question looks at to decide what to make
Operations	beginQuiz()		Starts a timer and creates questions for the user to answer.
	promptTopic()		shows a menu of types of questions.
	beginEditing()		start the level editor phase.
	beginLevel()		Starts the gameplay part of the level. Spawns the character and parents the camera to him.
	addMoney(amount)		increments money based off type of question

Class Name	Level Editor		
Description	Allows placement of items on a level before it starts.		
Extends	None		
Attributes	selected	Object ptr	The current item being manipulated
	newObjects	List<Objects>	All new items to be added to the level
	purchaseableItems	List<StaticObjects>	What the user can buy
	prices	List<Integer>	How much the elements of purchaseableItems cost
Operations	AddObject(Vec2 obj, Vec2 pos)		Takes an object and adds it to the list of new items
	RemoveObject(Object obj)		Removes item from list of new items

Class Name	Level		
Description	Game level data such as entities, character, tiles.		
Extends	None		
Attributes	id	Integer	Index into levels
	staticObjects	Tilemap<StaticObjects>	List of all static objects such as platforms
	entities	List<Entities>	All dynamic entities such as character, enemies
Operations	incrementLevel()	Increments and loads the next level	
	checkEntityCollision(Entity ,Entity)	Checks for hitbox collision between entities in the level.	

Class Name	Quiz		
Description	Creates, displays, and verifies user answers to questions.		
Extends	None		
Attributes	timer	Float	How long left to answer questions.
	money	Int	How much money have they made
	topic	Enum	What type of questions should the quiz create
Operations	generateQuestion()	Creates a question of desired topic	
	checkAnswer(string)	Verify string inputted matches with desired result.	
	decrementTimer()	update timer GUI widget	
	startQuiz()	initialize quiz assets, set timer to desired time	

Class Name	Question		
Description	Generated from Quiz, contains the question and answer.		
Extends	None		
Attributes	questionStr	String	Question in readable form
	answerStr	Int	Answer that user must match

Class Name	Entity		
Description	A dynamic object that will move around in levels.		
Extends	None		
Attributes	Position	Vec2	Where is the entity in the world
	Velocity	Vec2	How is the entities movement changing

Class Name	StaticObject		
Description	Something in the world that never moves.		
Extends	None		
Attributes	Position	Vec2	Where is the entity in the world

Class Name	Flyer		
Description	Enemy the freely flies around the map and chases the character.		
Extends	Entity		
Attributes	flySpeed	Float	how fast can it move in the air
	target	Entity	who if anyone should it chase
Operations	fly(direction)	Flies in a specific direction	
	gotoSpot(Vec2 pos)	Go to a specific spot	

Class Name	Walker		
Description	A basic enemy that walks back and forth.		
Extends	Entity		
Attributes	walkSpeed	Float	how fast can it move
	canWalkOffEdges	Bool	Should it walk right off an edge or should it reverse directions
Operations	changeDirection()	Flips direction	

Class Name	Walker		
Description	A basic enemy that walks back and forth.		
Extends	Entity		
Attributes	walkSpeed	Float	how fast can it move
	canWalkOffEdges	Bool	Should it walk right off an edge or should it reverse directions
Operations	changeDirection()	Flips direction	

Class Name	Jumper		
Description	Enemy that jumps in the air to hit the character.		
Extends	Walker		
Attributes	jumpSpeed	Float	How fast does it go after jumping
Operations	TryJump()	Sees if it can jump and if so jump	

Class Name	Platform		
Description	Something the character can stand on.		
Extends	StaticObject		
Attributes	type	Enum	What type is it

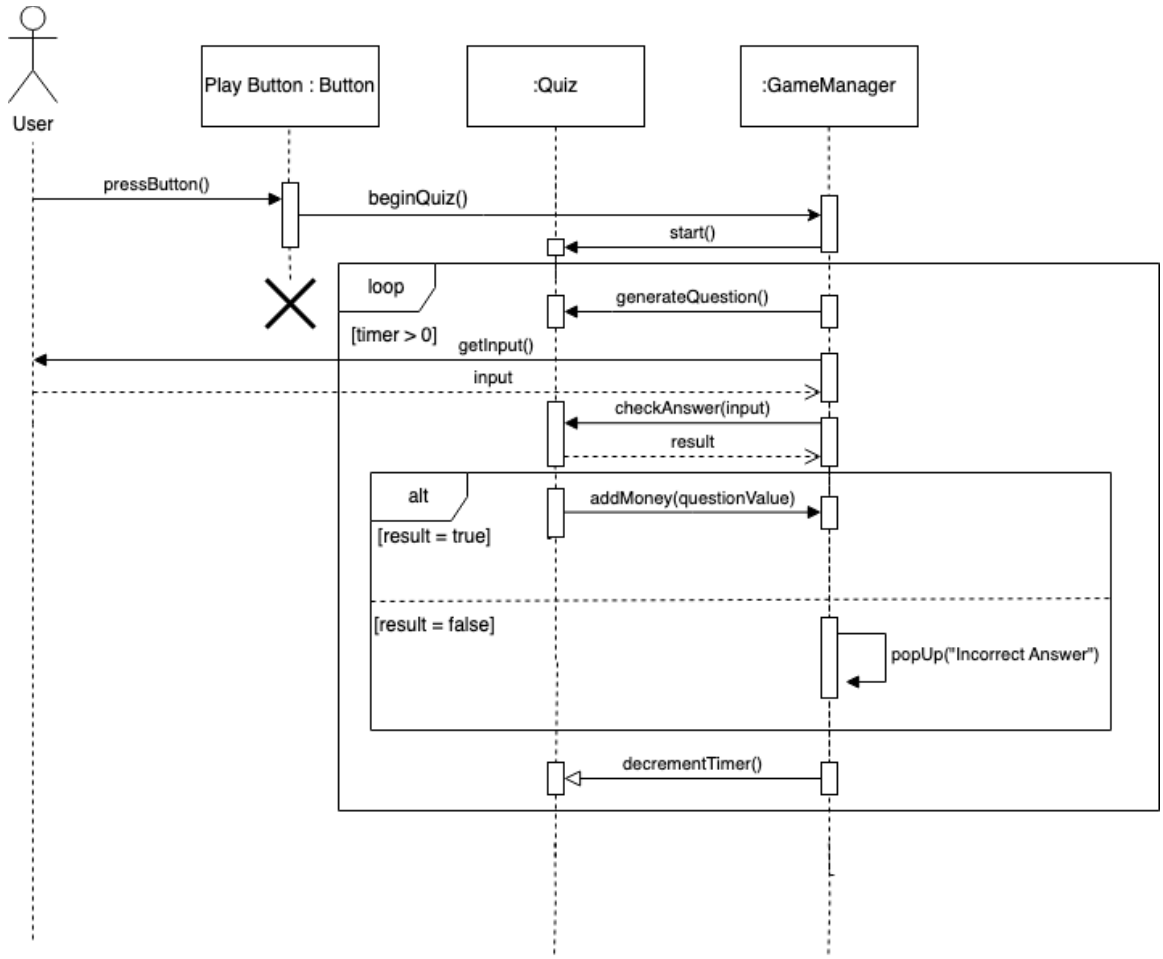
Class Name	Trap		
Description	Something static that can harm the character.		
Extends	StaticObject		
Attributes	type	Enum	What type is it

Class Name	Goal
Description	What the character must touch to win the game.
Extends	StaticObject

4.3 Sequence Diagrams

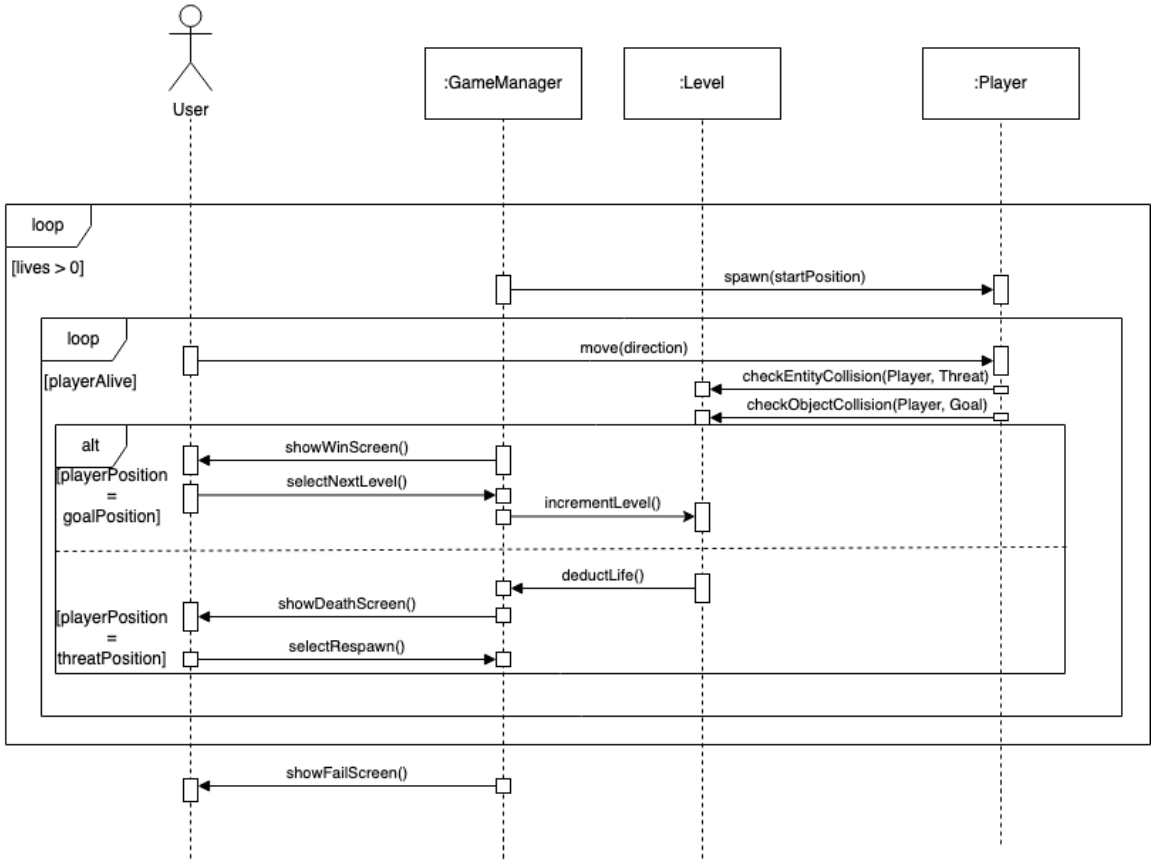
The following sequence diagrams outline how the game will work through certain actions taken by the user. The two below show how the pre-level quiz will traverse as well as the actual playing of a level. The diagrams depict a user and applicable objects at the top with rectangles to show when each of those are actively in use. It then shows class methods being called and returning via horizontal arrows between objects. Lastly, there are boxes that outline optional (if statements) actions as well as loops with the conditional statements for their activations listed below their titles.

Use Case Name:	Play Pre-Level Quiz
Actors:	User
Description:	After selecting a topic, the user plays through a math quiz on their selected topics.
Type:	Secondary
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	None



Pre-Level Quiz sequence diagram.

Use Case Name:	Guide Character Through Level
Actors:	User
Description:	After completing the pre-level quiz, the user guides the character to the goal of the level.
Type:	Primary
Includes:	Reach Level Goal, Pause Game, Reset Level
Extends:	None
Cross-refs:	None
Uses cases:	None



Guide Character through Level sequence diagram.

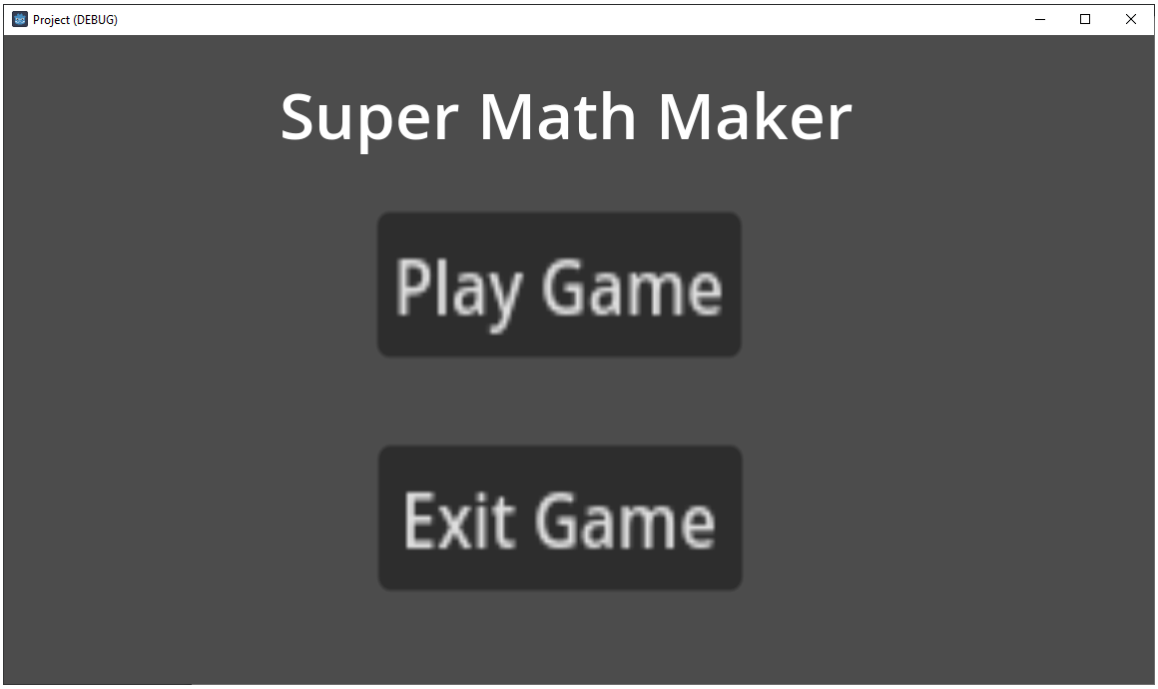
5 Prototype

The prototype consists of a main menu where the user can select between playing and closing the game. If the user chooses to play the game, they will be presented with a series of math topics to choose practicing their math skills on. These topics are fractions, geometry, and pre-algebra. After selecting their desired topic, the user will be quizzed. The user will earn points depending on how many questions they answer correctly. These points can be redeemed for objects that will assist them in completing the next level, such as springs, platforms, and powerups. Each level consists of platforms the character must jump through and enemies the character must either avoid or defeat. There are several enemy types, such as walkers, jumpers, and flyers.

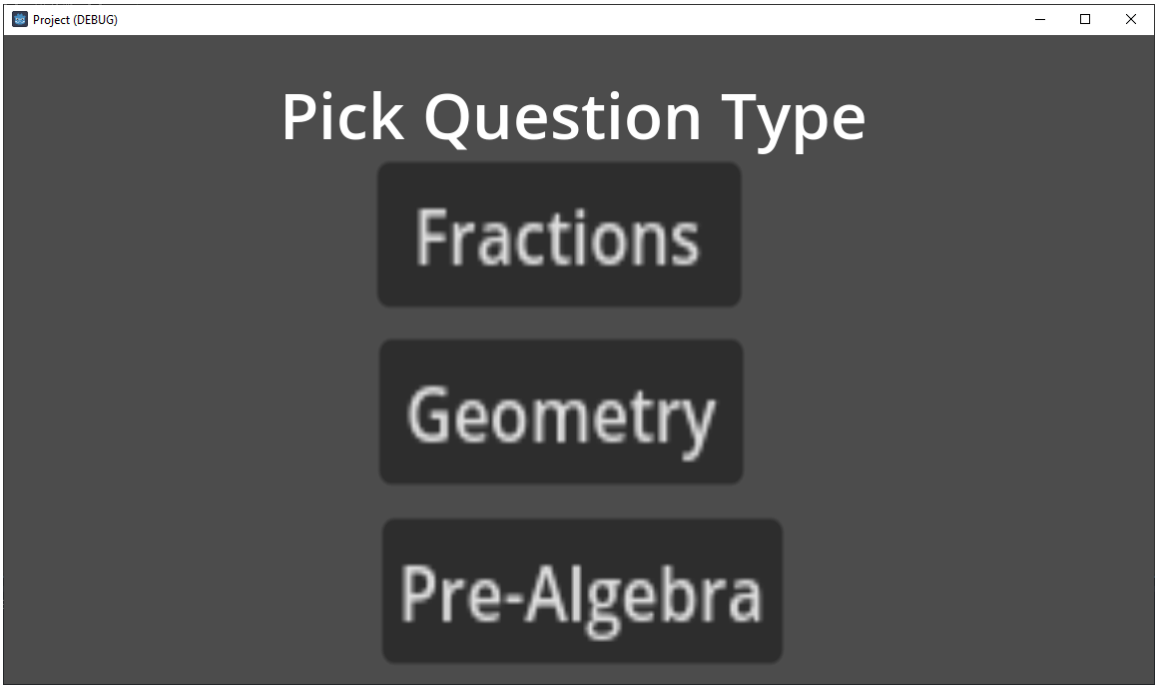
5.1 How to Run Prototype

In order to access and run the Prototype and Prototype UI's, a Windows computer with a modern web browser is required.. The Prototype and Prototype UI can be accessed from the project's website. In order to run the prototype, download the desired file from the website by clicking on the "Prototype" link and unzipping the downloaded folder. Both the "Game.exe" and "Game.pck" files must be present for the game to function correctly. Once inside the game, the user can navigate through the UI elements and end up at a very basic platformer game.

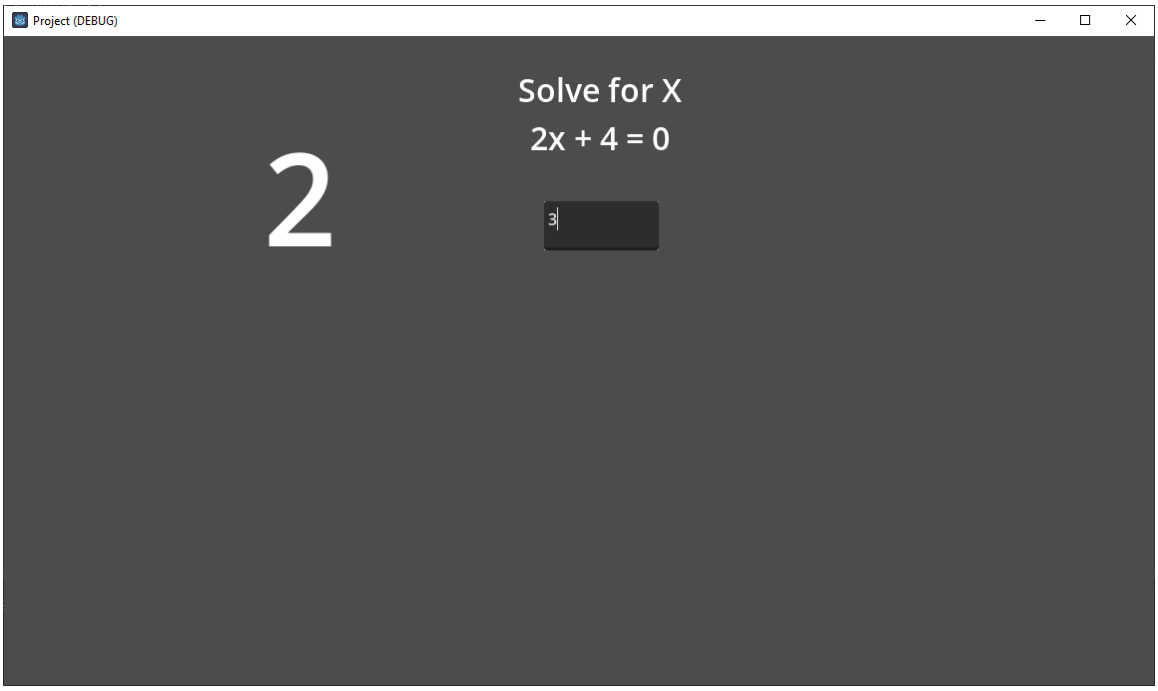
5.2 Sample Scenarios



When the user launches the game they will be brought to the main menu screen.

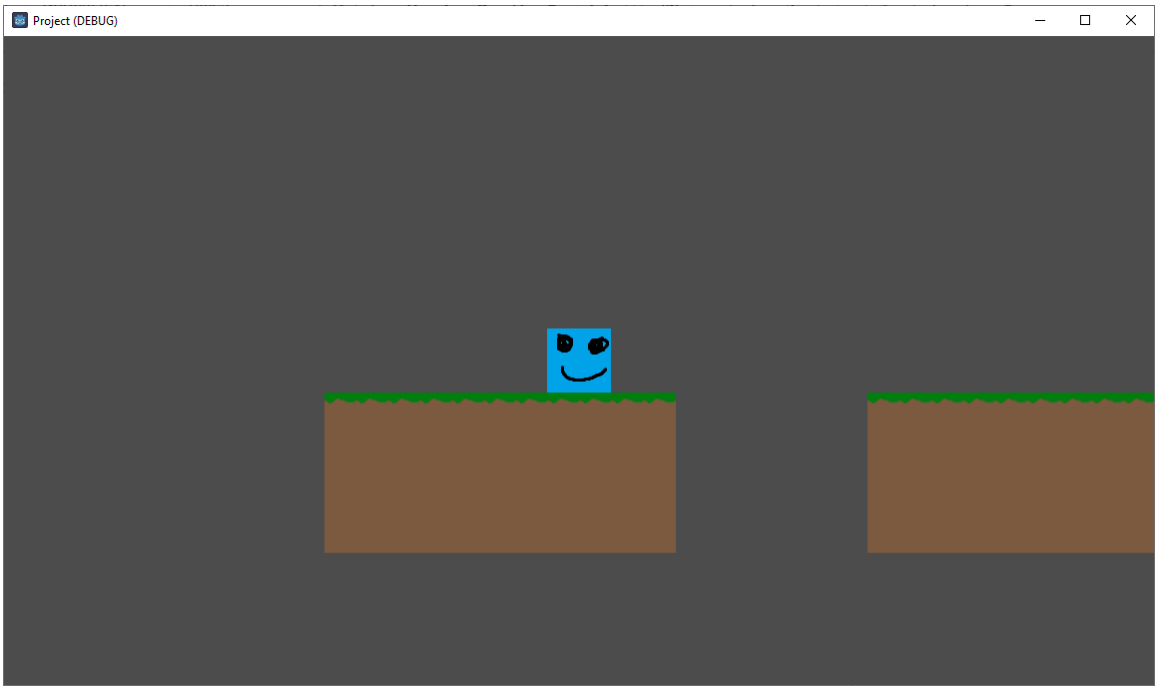


When the user then chooses play and then they can choose between pre-algebra, geometry and fractions for their problem set.



For this example, the user will choose pre-algebra. They will then be given a timed quiz which they have about a minute to answer as many questions as possible.

After this quiz they will be sent to the level editor with buttons that will allow them to buy items. They can also move their mouse to an edge of the screen to scroll. They are allowed to view the entire level to plan for what's ahead.



The user is then taken to the game part. They are free to run around and test out the controls. This is where the bulk of the game will take place. They can also press escape to get to the pause menu.

6 References

[1] "Massachusetts Curriculum Framework for Mathematics," Massachusetts Department of Elementary and Secondary Education, 2017. [Online]. Available: https://www.doe.mass.edu/frameworks/math/2017-06.pdf.

[2] Project website: https://super-math-maker.github.io/Super-Math-Maker/

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.