# 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, a 2-D platformer that helps users practice their math skills, 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, 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, as well as the development team behind it.

## 1.2 Scope

The video game that is being created is titled "Super Math Maker". Super Math Maker will be an educational video game that allows users to complete levels that are inspired from platformer games at difficulties dependent on their math skills. Users will be quizzed on math topics in the 4th to 8th grade range in order to craft their own path to the objective of a level. Users will be motivated to sharpen their math skills in order to unlock new possibilities for traversing levels. The application domain will be STEM education in schools, as teachers can present the game to their students as a fun way to practice their skills.

## 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: A 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

The rest of the SRS is broken down as follows:

Section 2 contains an overview of what Super Math Maker actually is, and how the game is played. Section 2.1 outlines the context for the product and how the user and software interfaces are laid out. 2.2 specifies the functions– both the back-end functions which the user doesn't notice and the functions which the user directly interacts with. Section 2.3 specifies the expectations of the user, and 2.4 demonstrates the software, hardware, design and regulatory constraints. 2.5 states the assumptions of the software and of the user, and 2.6 displays the diagrams for each of the functions.

Section 3 lists the complete requirements for the project. It mentions all of the expectations of the user and game, some of which may not be implemented.

The modeling requirements make up section 4. This consists of all of the use case diagrams, class diagrams and sequence diagrams.

Section 5 contains the information for the prototype. It contains a preview of what the user should expect when playing the game and an example playthrough of the game.

Lastly, section 6 has all of the references and section 7 is the 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.

In the rest of this section, the product perspective and its interfaces will be described, followed by the product functions. Here, the inner workings of the project and its goals, along with how they are accomplished will be detailed. Next, the characteristics of the intended users will be covered in order to specify the intended demographic. Then, the constraints of the project such as regulatory, design, hardware and software constraints will be described. That will be followed by the assumptions of the digital environment that the software is used in and what the software depends on to function. Finally, the section will wrap up with a description of features that are not present in the current iteration of the game.

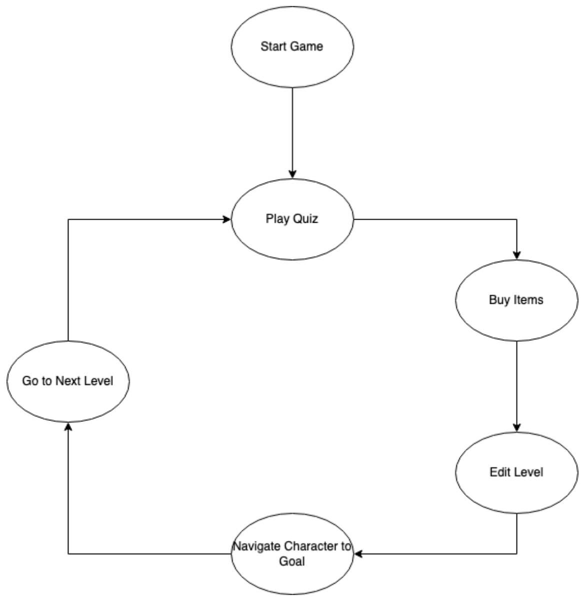
## 2.1 Product Perspective

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 8th grade mathematics, including fractions, pre-algebra and geometry. The educational content follows age-appropriate guidance from the Massachusetts Department of Elementary and Secondary Education, including fractions, pre-algebra, and geometry.

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.

The diagram below depicts how the gameplay loop in Super Math Maker works.



## 2.2 Product Functions

The goal of Super Math Maker is to provide students with a way to practice their math skills in a form that is more engaging than traditional homework assignments. When the game is launched the user is brought to the main menu that 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.

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 problem 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 in order to make each quiz unique. The solution 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. Users are motivated to learn the math used in these quizzes by the opportunity to modify the game level further with the use of shop points.

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. Once they're ready the user can click the "Play Game" button to start the game.

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 entertainment that the gameplay provides, along with the creative opportunities that good quiz scores allow for, create a system where the user is inspired to replay levels and get better quiz scores so that they can craft their optimal solution to each level.

## 2.3 User Characteristics

Super Math Maker 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 that at least allow for fraction arithmetic, and optionally ranging up to geometry and pre-algebra. 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

### 2.4.1 Regulatory Constraints:

The math topics that are covered in this game must fall between the 4th and 8th grade range of the Massachusetts Mathematics Curriculum Framework. This means that elementary topics such as addition and subtraction are too basic, and algebra or calculus are too complex.

### 2.4.2 Design Constraints:

The game must have an educational component that revolves around mathematics. While certain sections of the game may provide breaks from education, gameplay can not solely focus on platforming. The game must also be easily navigated by at least elementary school students in the 4th and 5th grade, meaning that menu navigation may not be overly complex and cannot use sophisticated vocabulary.

### 2.4.3 Hardware Constraints:

The game must be able to run on low end hardware, which is defined as computer parts dated as old as 2015 to 2018. The game may not use intensive graphics techniques such as ray tracing and 4k resolution textures. Additionally, users must control the game using a keyboard and mouse.

### 2.4.5 Software Constraints:

The game must be created using the Godot game engine and be able to run on Windows, Mac, and Linux operating systems.

## 2.5 Assumptions and Dependencies

### 2.5.1 Assumptions

It is assumed that the user will have an operational computer that runs hardware that was released no later than the year 2015. Users are also assumed to have a keyboard and mouse that is connected to their machine somehow. It is assumed that users' machines will be running modern operating systems (e.g. Windows 10, MacOS Sonoma).

### 2.5.2 Dependencies

The game utilizes the Godot API for its libraries and functionality. These are implemented into the game mechanics in the game's infrastructure.

## 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 probability. Finally, a future release will include a greater quantity of levels that users can build upon to reach the goal.

## 3 Specific Requirements

1. Keyboard and mouse will be required.
2. The game will export as an executable file.
3. The game will output to a display with a minimum resolution of 800x600
4. The user will be able to choose between playing the game, changing the settings or quitting the application.
5. A Graphical User Interface (GUI) will exist to display options within the game to the user.
6. There will be an educational aspect to the game that is focused on mathematics.
    - 6.1. Each level will begin with a mathematics quiz which contains questions based on user selected math topics such as geometry, fractions, and pre-algebra.
        - 6.1.1. 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.
        - 6.1.2. Questions will be chosen randomly at runtime from a list of equations, depending on the chosen topic.
        - 6.1.3. A point system will exist that contains a count of the number of points owned by the user.
            - 6.1.3.1. Points will be awarded for each correctly solved question, with difficult questions granting more points.
            - 6.1.3.2. An incorrect solution to a question will deduct points.
        - 6.1.4. The quiz will feature an input parsing mechanism.
            - 6.1.4.1. 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).
7. Gameplay will exist that utilizes 2-D platformer mechanics.
    - 7.1. Levels will exist for the user to progress through.
        - 7.1.1. Platforms will exist in each level that create a path through the level.
        - 7.1.2. The user will be able to reset each level.
        - 7.1.3. The user will be able to pause the game during each level.
        - 7.1.4. The completion of a level will allow for navigation to a subsequent level until all levels are complete.
    - 7.2. Each level will contain a user controlled entity known as the character.
        - 7.2.1. The character will be able to move left, right, and can jump.
        - 7.2.2. The character will be able to wall jump.
            - 7.2.2.1. Wall jumping will occur when the airborne character jumps while in a collision with a wall.
        - 7.2.3. All movements by the character will be animated.
        - 7.2.4. The character will have the ability to melee attack, dealing damage to enemies within a short range.

7.2.4.1. Melee attacks will be able to be directed up, down, left and right.

7.2.5. The character will spawn with a certain amount of lives.

7.2.5.1. Death will occur through contact with enemies and hazards on a level.

7.2.5.1.1. Enemies will exist in each level of the game.

7.2.5.1.1.1. Enemies will be able to move and attack.

7.2.5.1.1.1.1. There will be various types of enemies that each behave differently.

7.2.5.1.1.1.2. Enemies will change their behavior when in the presence of the character.

7.2.5.1.2. Hazards will exist in each level of the game.

7.2.5.1.2.1. Hazards will be static environmental features in levels that can harm the character.

7.2.5.2. Death in a level will decrease the life count by one.

7.2.5.2.1. When the character dies with zero lives remaining, the game will be over.

7.2.5.2.2. When the character dies with more than zero lives remaining, the game will return the character to the beginning of the same level.

7.2.5.3. Lives will carry over between each completed level.

7.3. A goal will exist in each level to signify that the character has completed the level.

7.3.1. Some levels will have secret alternative goals that can be found through exploration.

8. Assists will be able to be placed on each level by the user.

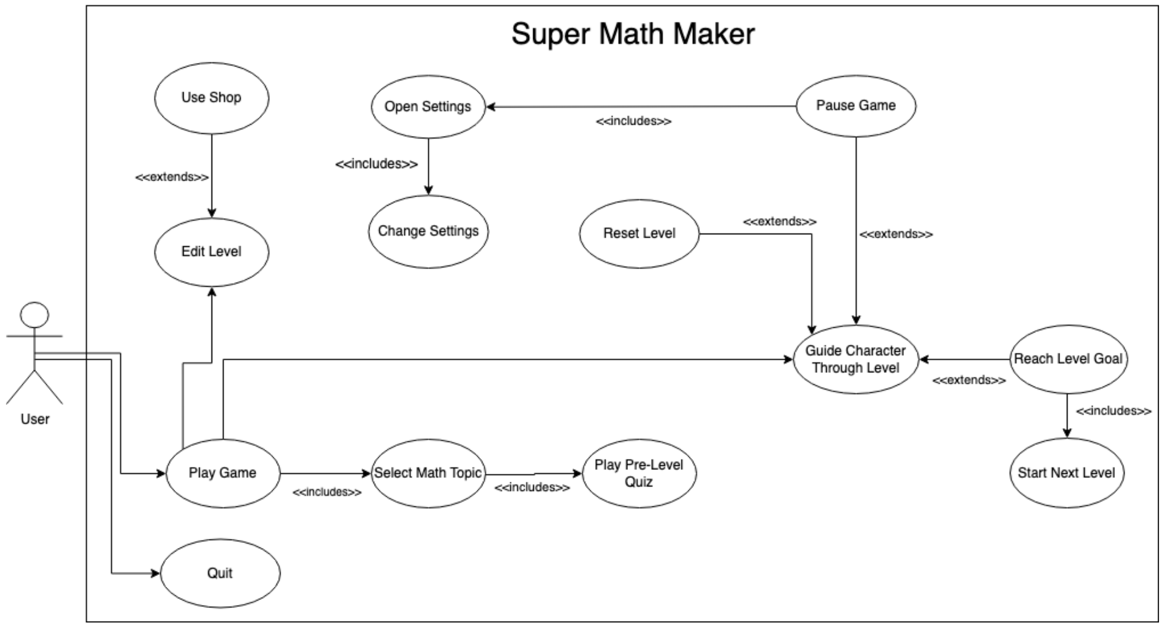
8.1.1. Assists will consist of springs, teleporters, extra platforms, power-ups, etc..

9. 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 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. In the diagram below, the user has two primary use cases: Play Game and Quit. When the user chooses to play the game, they will have several goals starting with selecting a math topic to play with. The user will then have the goal of completing a quiz based on the selected topic. The user will also be able to edit the level that they are in, using items purchased from a shop to do so. For guiding the character through the level, the user will be able to pause the game, reach the level goal, or reset the level. Upon pausing the game, the user will be able to access the settings in order to change settings that they feel necessary. When the user navigates the character to the goal, they will then be able to navigate to the next level.



*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: | Requirement 4 |
| Uses cases: | Select Math Topic, Edit Level, Guide Character Through Level |

| | |
|---|---|
| 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: | Requirement 4 |
| 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: | Requirement 6.1 |
| Uses cases: | Play Pre-Level Quiz |

| | |
|---|---|
| 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: | Requirement 6.1 |
| 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: | Requirement 8 |
| Uses cases: | Use Shop |

| | |
|---|---|
| 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: | Requirement 9 |
| 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 |
| Cross-refs: | Requirements 7.1, 7.2, 7.3 |
| Uses cases: | Reset Level, Pause Game, Reach Level Goal |

| | |
|---|---|
| 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: | Start Next Level |
| Extends: | Guide Character Through Level |
| Cross-refs: | Requirement 7.3 |
| Uses cases: | Start Next Level |

| | |
|---|---|
| Use Case Name: | Start Next Level |
| Actors: | None |
| Description: | The user selects the next level for the game to load. |
| Type: | Secondary |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 7.1.4 |
| 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: | Requirement 7.1.2 |
| Uses cases: | None |

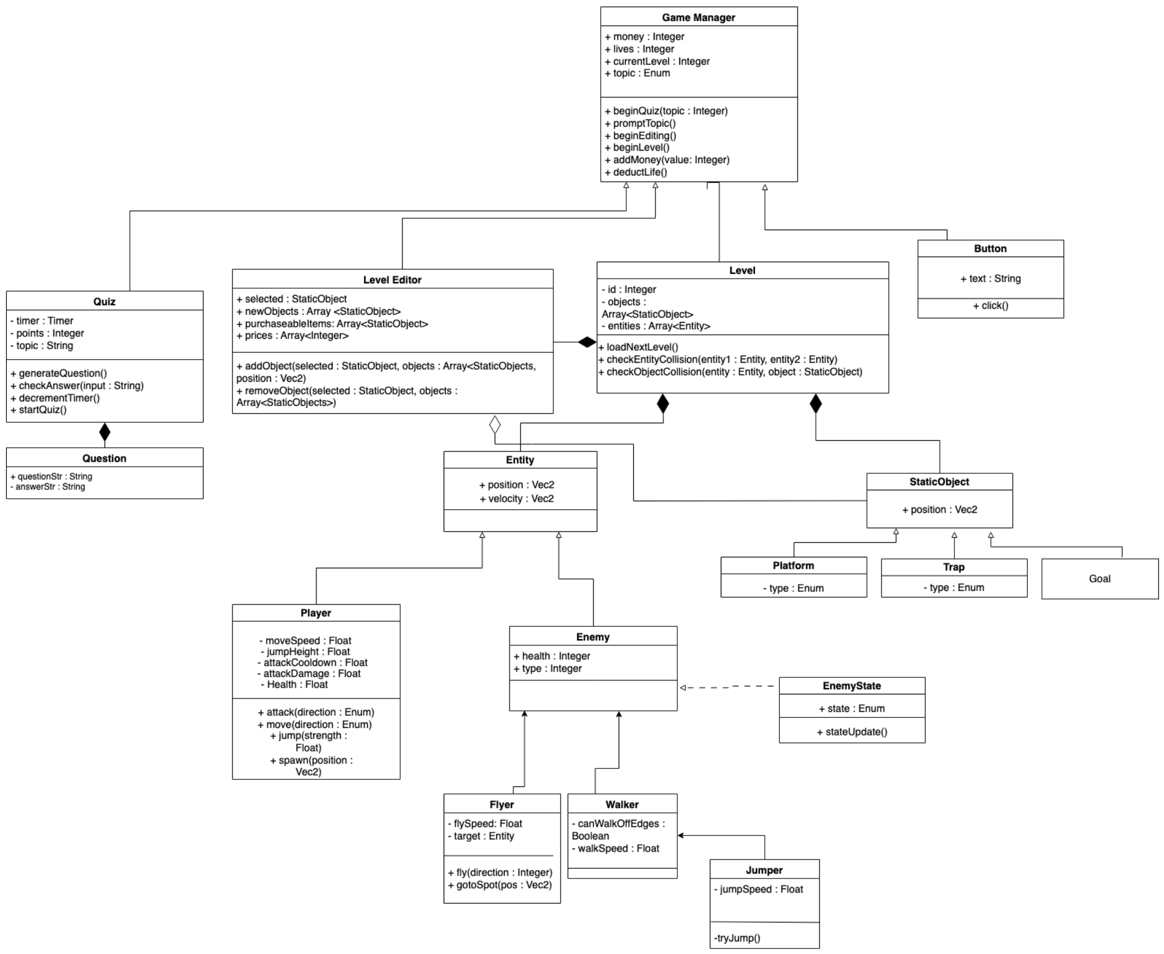
| | |
|---|---|
| Use Case Name: | Pause Game |
| Actors: | None |
| Description: | When activated, the level will no longer proceed until the user deactivates this function.  This will also bring up a menu for the user to either quit the game or edit the settings. |
| Type: | Secondary |
| Includes: | Open Settings |
| Extends: | Guide Character Through Level |
| Cross-refs: | Requirement 7.1.3 |
| Uses cases: | Open Settings |

| | |
|---|---|
| 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 |
| Cross-refs: | Requirement 4 |
| Uses cases: | Change Settings |

| | |
|---|---|
| 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: | Requirement 4 |
| Uses cases: | None |

## 4.2 Class Diagram

The class diagram below contains descriptions of each class that will make up Super Math Maker, by outlining the title, fields, and methods of each class. The class that exists above all others is the Game Manager. This class contains high level information regarding the state of the game, such as the amount of money and lives remaining that the user has, the current level number, and the selected math topic. It communicates with other classes like the Quiz class, Level class, and Level Editor class throughout gameplay. The Quiz class contains the timer for the quiz, the amount of points that the user has earned, and the topic of the quiz. The quiz is composed of the question class, since it contains a set of questions that the user answers. The questions contain both a prompt and a solution which are stored as strings. The Level class contains the variety of objects and entities that exist as well as the level number. It is composed of the Entity and StaticObject class. The Entity class stores the position and velocity of entities like the Player and Enemy classes that it inherits. The Player class contains information such as the speed, jump height, attack damage and health of the character. It contains methods that allow for modification of all of these attributes. The enemy class holds the health and type of enemy and inherits classes for each type of enemy. The Flyer class contains their speed of flight and current target. The Walker class also contains speed as well as a boolean for the ability to walk over edges. The Jumper class contains the speed that they can jump at and a method to attempt to jump. The Level Editor class contains information about how the user is choosing to modify each level, like the selected object, a list of user placed objects, a list of potential objects and the price for each object. It contains methods that allow the user to add and remove certain objects from the level. It is aggregated by the StaticObject class, which contains a position variable. The StaticObject class inherits three classes for different types of objects. The Plaform and Trap objects contain an enumerated type, while the Goal class simply represents the objective of the level.



*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<StaticObject> | All new items to be added to the level |
| | purchaseableItems | List<StaticObject> | What the user can buy |
| | prices | List<Integer> | How much the elements of purchaseableItems cost |
| Operations | AddObject(StaticObject obj, Vec2 pos) | Takes an object and adds it to the list of new items | |
| | RemoveObject(StaticObject 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 | loadNextLevel() | Increments and loads the next level | |
| | checkEntityCollision(Entity ,Entity) : Bool | 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 | Timer | 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() : Question | Creates a question of desired topic | |
| | checkAnswer(string) : Boolean | 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 | String | 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 | 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 | |

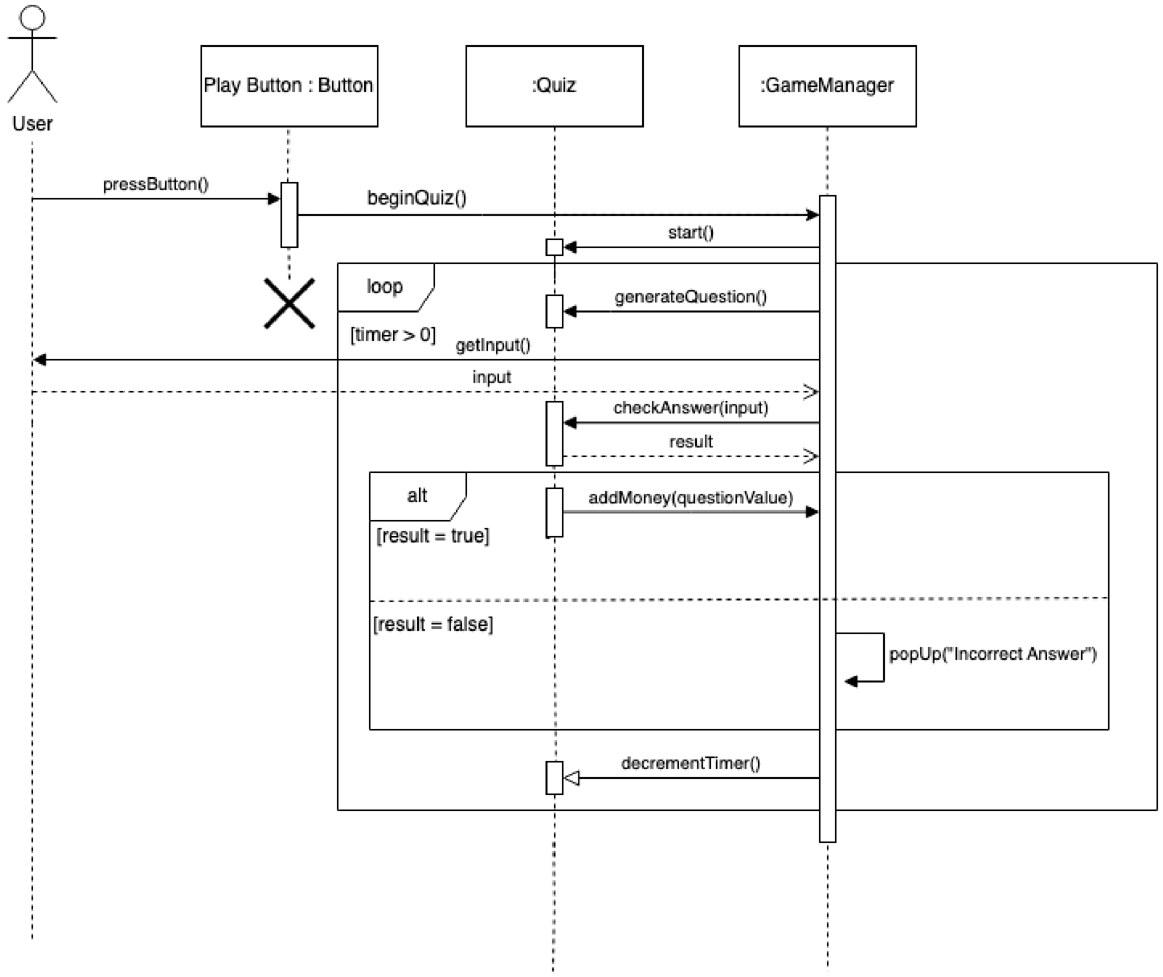
| | | | |
|---|---|---|---|
| Class Name | Platform | | |
| Description | Something the character can stand on. | | |
| Extends | StaticObject | | |
| Attributes | type | Enum | What type is it |

| | | | |
|---|---|---|---|
| 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 |

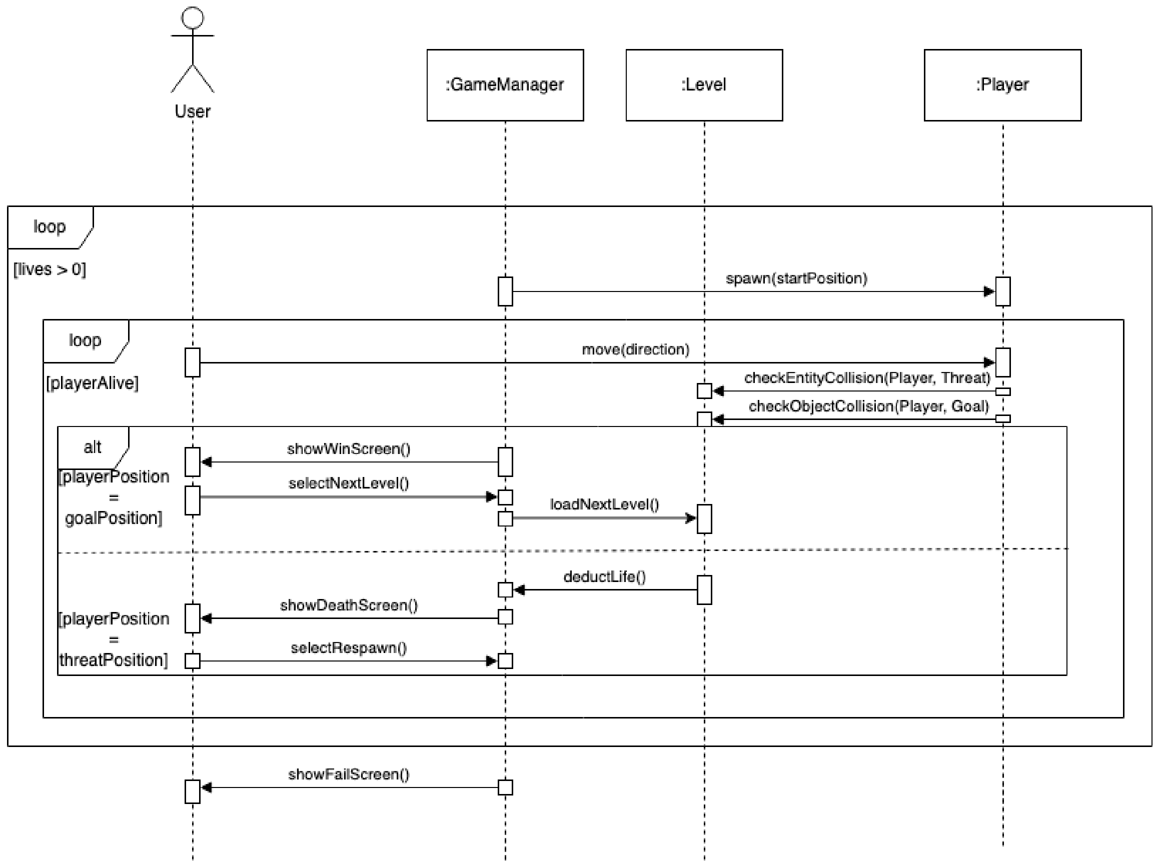
## 4.3 Sequence Diagrams

The following sequence diagrams outline how the game will work through certain actions taken by the user. In the first diagram below, the sequence of the user playing through a pre-level quiz is described. The user presses the start button, which then tells the game manager to begin the quiz. A loop is then entered where the game managers tells the quiz to generate a question and then presents the question to the user and gets input. The game manager then passes this input back to the quiz, which returns the result of the question (right or wrong). If the answer is right, then the user is rewarded with money. If the answer is wrong, the user is notified that they provided an incorrect answer. This loop continues until the quiz timer reaches 0.



*Pre-Level Quiz sequence diagram.*

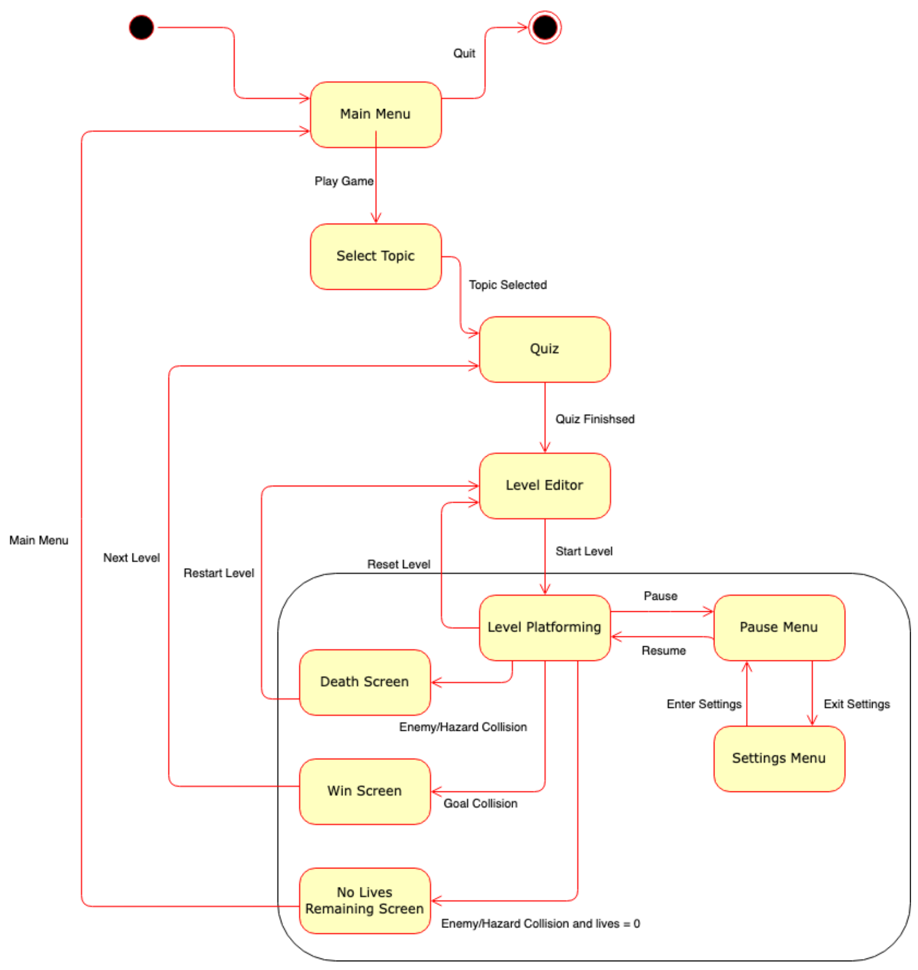
The diagram below depicts the sequence of the user navigating the character through a level. The game manager spawns the player at the beginning of the level, and then a loop is entered where the user navigates through the level until they run out of lives. While the player is alive, the user navigates them through the level, and the level constantly checks to see if entity collision has occurred with the player. If the player is at the same position as the goal, then they win the level and are navigated to the next one. If the player is at the same position as an enemy or hazard, they are deducted a life and must respawn at the beginning of the level. If the player runs out of lives, they are shown a failure screen.



*Guide Character through Level sequence diagram.*

## 4.3 State Diagram

The state diagram below depicts the various states that Super Math Maker has and how they are reached. The game starts in the main menu state, which can exit the game through the quit button. From there, the select topic state can be reached through the play game button. Once a topic is selected, the quiz state is reached. The quiz state goes to the level editor state when the quiz is finished. Once the user starts the level, the level platforming state is entered. From there, the user can reach the pause menu state through the pause button. The pause menu can go to the settings menu or go back to the level platforming state. From the level platforming state, the death screen state, win screen state, or no lives remaining screen state can be reached depending on if the character collides with enemies, hazards, or the goal. The death screen state goes to the level editor when the user restarts the level. The win screen state goes to the quiz state when the user selects the next level. Finally, from the no lives remaining screen state, the user can select to go back to the main menu state.



*State diagram.*

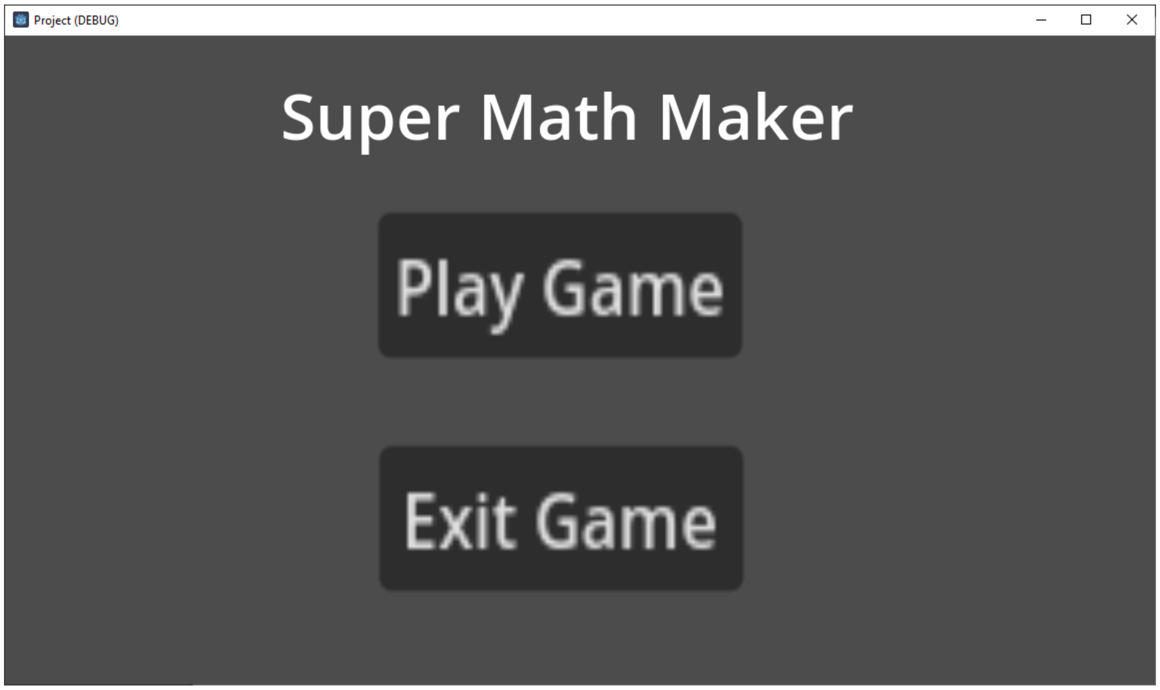
## 5 Prototype

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.

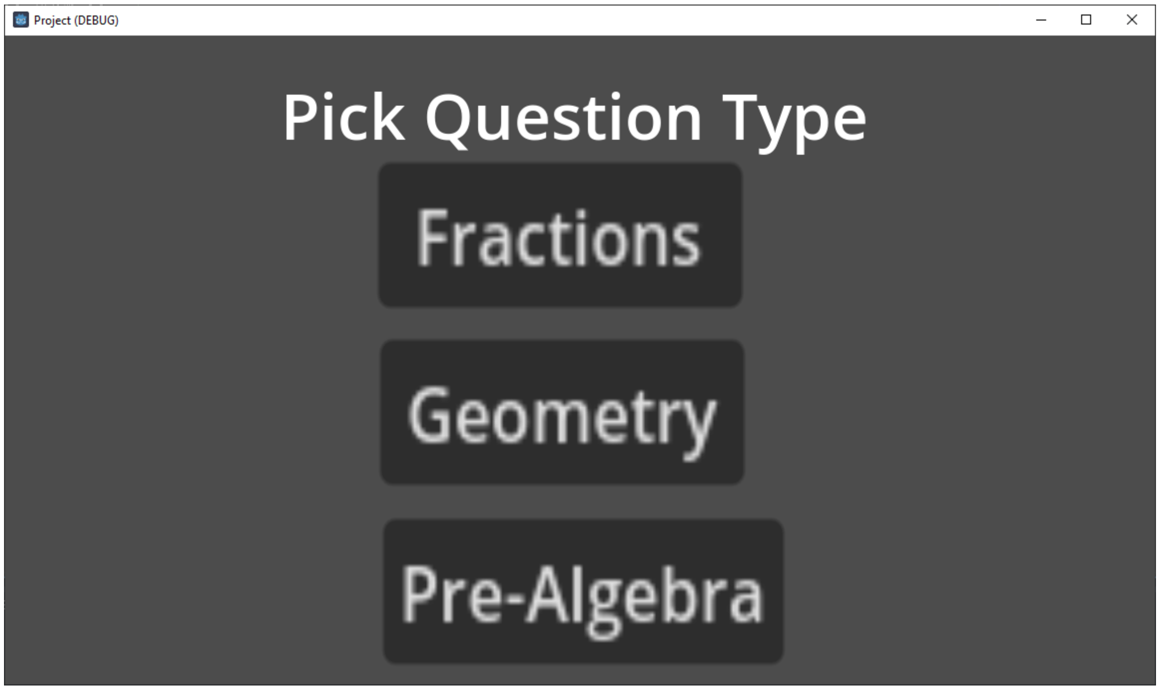
## 5.1 How to Run Prototype

In order to access and run the prototype, a Windows computer with a modern web browser is required. The Prototype can be accessed from the project's website. In order to run it, download the desired file from the website[1], 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.

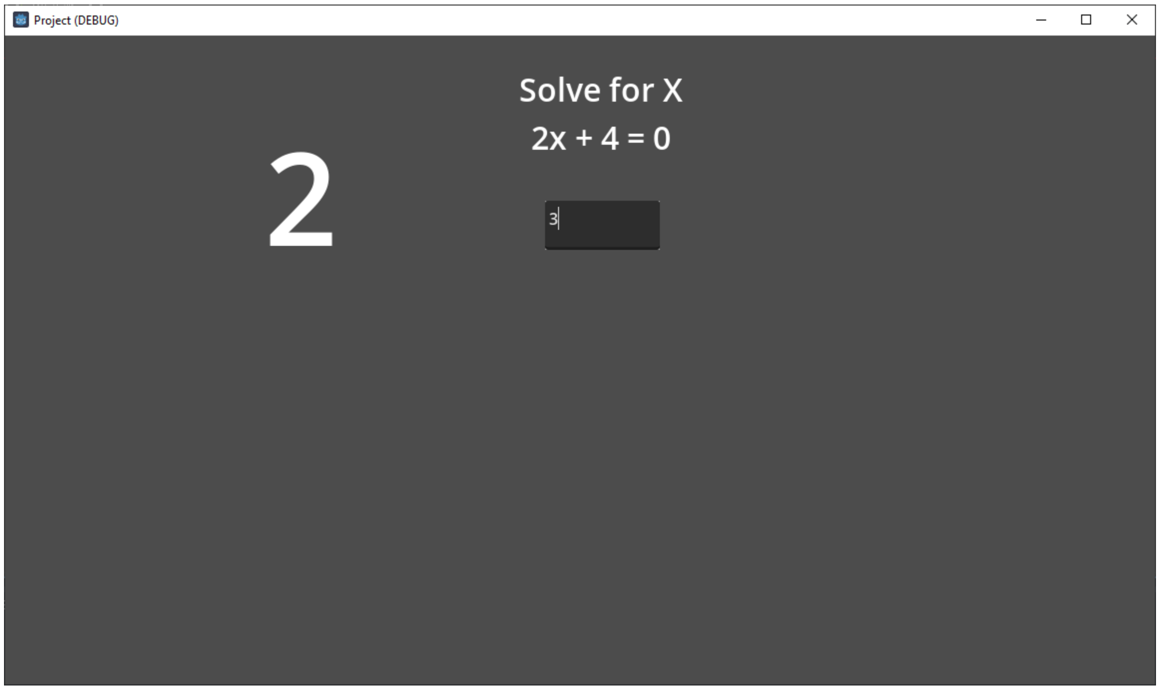
## 5.2 Sample Scenarios



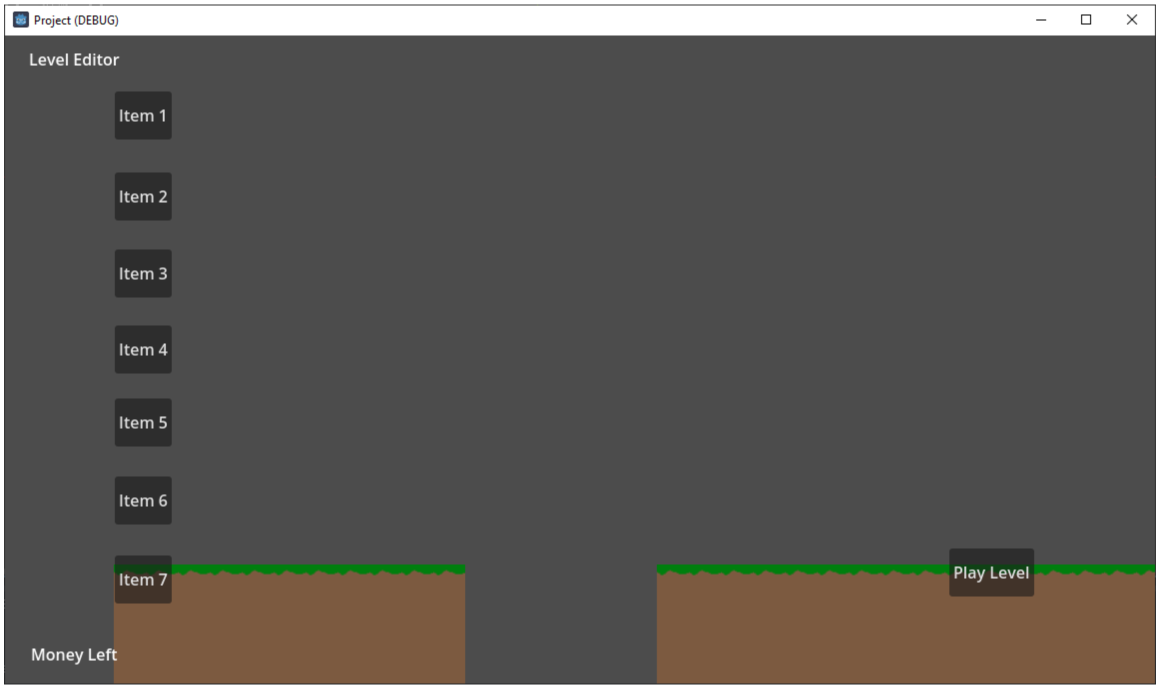
When the user launches the game they will be brought to the main menu screen.



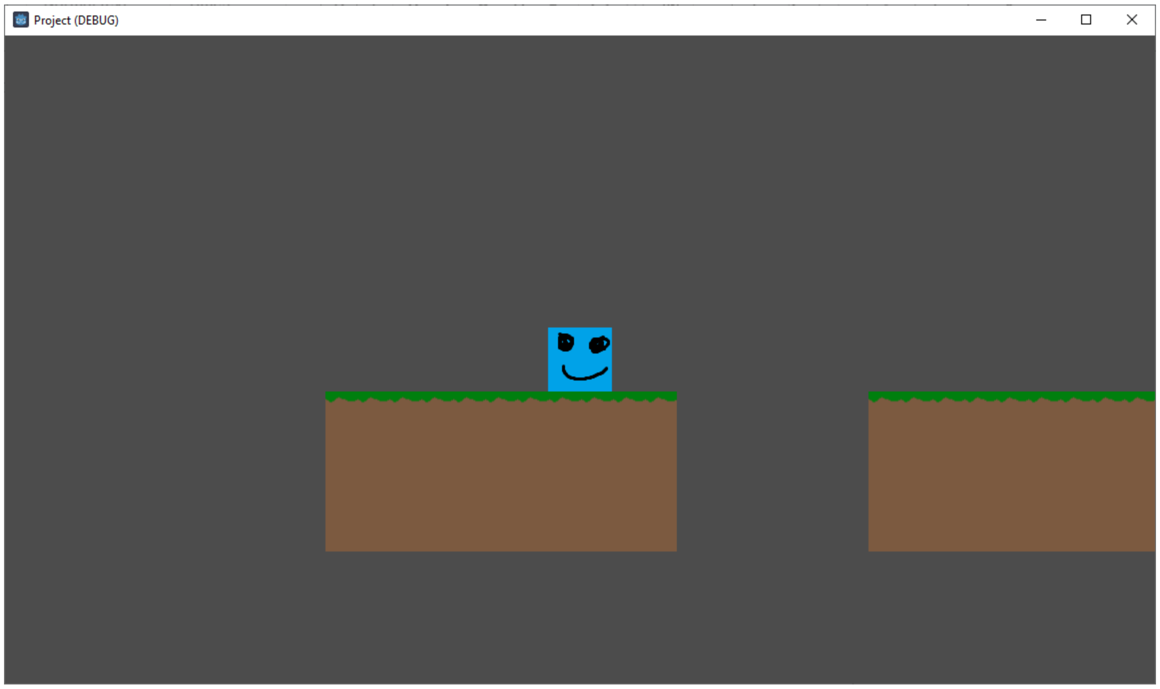
When the user then chooses play and then they can choose between pre-algebra, geometry and fractions for their problem set.



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] D. Stark, G. Higgins, N. Haselton, A. Corkhum, and C. Jimenez, Super Math Maker, 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.