Software Requirements Specification (SRS)
Tap Tap Computation

Team: Team 6
Authors: De'Ondre Allen, Bryan Bean, Noah Mezher, Jack Summers, Rob Terravecchia
Customer: Users interested in elementary mathematics
Instructor: Professor Daly

1 Introduction

This document will outline all the requirements for the educational mathematics game Tap Tap Computation. Specifically, it covers the purpose of Tap Tap Computation, its scope, target demographic, hardware and software requirements, as well as definitions, acronyms, abbreviations, functions relevant to its implementation. It will also cover development constraints and an outline of the Tap Tap Computation prototype.

1 Purpose

The purpose of this document is to list all the requirements that Tap Tap Computation must fulfill. The intended audience for this document is the developers of this software and our professor.

2 Scope

Tap Tap Computation will allow users to study mathematics in a game setting. The user will be able to choose from 5 different stages (addition, subtraction, multiplication, division, and algebra) and a difficulty level for that stage from 1 to 3, where 1 is the lowest difficulty and 3 is the highest. The benefits of this system include providing a way for users to study math in a fun way and follows the Common Core Math standard for the U.S. public school system.[1] It incentivizes play by providing players with a scoring system so that they may keep track of how well they do, and quantify it. The goal of this product is to teach users how to compute math quickly in their head and have the coordination to find the correct answer. Tap Tap Computation will keep track of the user's score, which will increase when the user picks correct answers will unlock the next level when the score is high enough. It will also keep track of incorrect answers and will force the student to exit the level if they get too many answers wrong. Tap Tap Computation will also store the user's high score so they can see if they have improved. Specific mechanics are covered in further detail in Section 3.

3 Definitions, acronyms, and abbreviations

Program: The software application that contains Tap Tap Computation and all associated functionality.

Game: The sub-section of the program that contains only the game loop for Tap Tap Computation.

Bucket: A single, user-controlled object that catches orbs.

Orb: An object containing a single possible answer.

Player: The user of the program.

TTC is the abbreviation for Tap Tap Computation.

Stage: An instance of the game focused on a specific section of content.

[1] http://www.corestandards.org/wp-content/uploads/Math_Standards1.pdf

Level: a subset of a stage each with its own difficulty. There are three levels per stage, level 1 being the easiest and level 3 being the hardest.

4 Organization

The rest of the SRS contains the context of TTC, a list of the functions to be used in TTC, an outline of our target demographic, developer constraints, developer assumptions, requirements outside the scope of the project, the requirements of the game, modeling requirements, information on the prototype, references and points of contact. The SRS is divided into 7 categories. The main category is an overview of a certain part of the project. Some categories are broken into subcategories and will go into more detail on different aspects of that category.

2 Overall Description

This section will cover constraints on the developers, context of TTC, explain the different interfaces, list the game's functions, assumptions made about the user's characteristics, assumptions made about the product interactions with users as well as the hardware and the environment it is run on. It also covers some requirements decided to be outside the scope of TTC.

2.1 Product Perspective

TTC is being developed by a team of 5 developers for third grade students that wish to practice and hone their mathematics skills in a fun and interactive manner. The scope of this project involves designing a game prototype for Linux, macOS, and Windows environments mimicking the mechanics of a "catch the beat" game, akin to osu!catch, in a mathematical context. In osu!catch, the player has control of an object, a bucket, which moves along a horizontal axis located near the bottom of their screen. Various objects fall from the top of the screen towards the bottom of the screen. The player can move their bucket left and right to try and catch falling objects.[2] This product is being developed in the Godot engine. Godot is an open source game development engine that utilizes a proprietary scripting language called GDScript, which is similar to Python and allows for rapid development. This product will be able to be executed on any computer that supports Godot. Most modern computers have the hardware capability for this. The full compilation requirements for the Godot engine can be found here, on Godot's documentation website.[3]

[2] https://osu.ppy.sh/help/wiki/Game_Modes/osu!catch
[3] https://godotengine.org/features

2.2 Product Functions

updateScreen– Refreshes screen to show current location of all objects

checkCollision- Checks for collision between buckets and orbs or orbs and the bottom of the screen

generateOrb- Spawns an orb object

clearScreen- Clears the screen of orbs

saveGame- Saves the user's score if it is a new high score

processPlayerInput- Reads the user's inputs

moveBucket- Allows the user to control the position of the bucket by using processPlayerInput

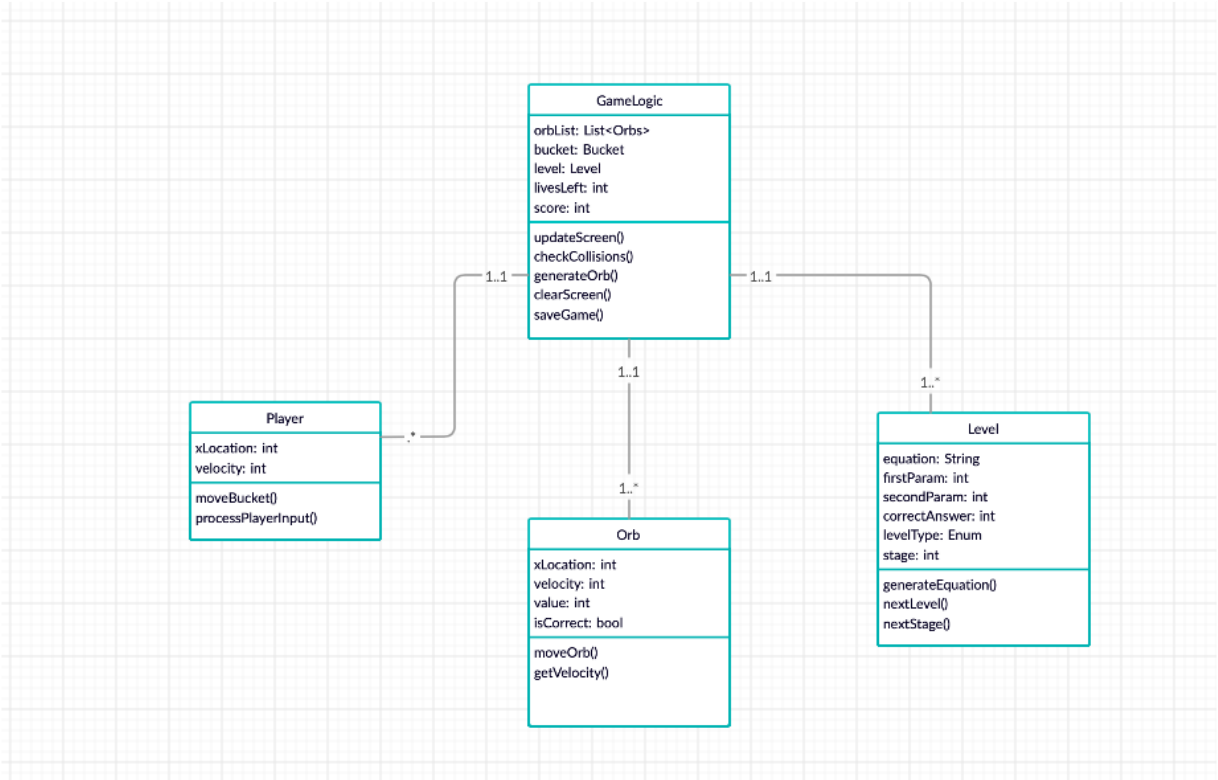
moveOrb- Moves the orb object down the screen

getVelocity- Returns the velocity of the orb object

generateEquation- Creates a random equation based on the chosen stage at the top of the screen

nextLevel– Moves the user to the next highest difficulty level

nextStage- Moves the user to a different stage or their choice



2.3 User Characteristics

The user of the TTC is expected to be a person in the process of learning third grade level mathematics. It is expected that the user is proficient at operating a keyboard enough they will not have to focus on moving more than they focus on solving the equation displayed. The user is expected to know how to solve any of the equations displayed on screen, but not comfortably enough to be able to solve the equation correctly every time when placed under time constraints.

2.4 Constraints

TTC's graphics and development are handled in Godot. TTC only stores the player's high score. It will not require a real name only a username. This eliminates any liability regarding a user's personal data.

2.5 Assumptions and Dependencies

TTC will be run on either Windows, Mac OS X or Linux. The game will be developed using the opensource game environment Godot, so the system hardware must be able to run Godot's engine. This product will come in an executable package, either an .exe file or the UNIX equivalent to be run through the command line.

2.6 Apportioning of Requirements

Requirements determined to be beyond the scope of TTC are extra game modes and challenges that allow the user to continue after level after losing.

3 Specific Requirements

Overall Description

1. Launch Screen
    i. The launch screen will contain buttons for a new game, options, and quitting the program.
    ii. Upon quitting the program, a new dialogue box appears asking the user if they are sure, with options being “Yes” or “Back.”
    iii. If “Yes” is pressed, the program exits.
    iv. If “Back” is pressed, the program returns to the title screen.
2. Start
    i. The Start button will allow the user to create a new profile or automatically load an existing one if already created.
    ii. The user is then transferred to the stage selection screen, where each stage corresponds to 5 options.
        a. The options are addition, subtraction, multiplication, division, and algebra.
    iii. Once a stage is selected, the user then moves to the level select screen, where they may pick one of three levels. The user’s score is displayed for each level.
    iv. Once a level is selected, the user then moves to the game screen.
3. Game UI
    i. The game screen will launch with a user-controlled bucket on the bottom of the screen and an equation in the top-center of the screen.
    ii. The new game will have randomly generated equations.
    iii. The user starts with a score of 0 and three hearts.
    iv. A timer will count down from 3 and the program will execute Section 4.
4. Gameplay
    i. Every few seconds an orb is dropped containing a value.
    ii. Up to 5 orbs can be on screen at the same time.
    iii. Only 1 correct answer can be on screen at once.
    iv. The orbs fall with a fixed velocity towards the bottom of the screen
    v. If the user’s bucket (Section 6) at the bottom of the screen catches the orb with the correct answer the other orbs pop and disappear, a new equation is generated, and 100 points are added to the user’s score.
    vi. If the user’s score reaches 1,000 the user has successfully completed this level and unlocked the next level in its stage. (Section 5)
    vii. If the user catches the incorrect orb, the user loses a heart.
    viii. If the user lets a correct answer fall through, the user loses a heart.
    ix. If either condition vii or condition viii are met, a new equation is generated.
    x. If the user has no hearts left the game concludes.

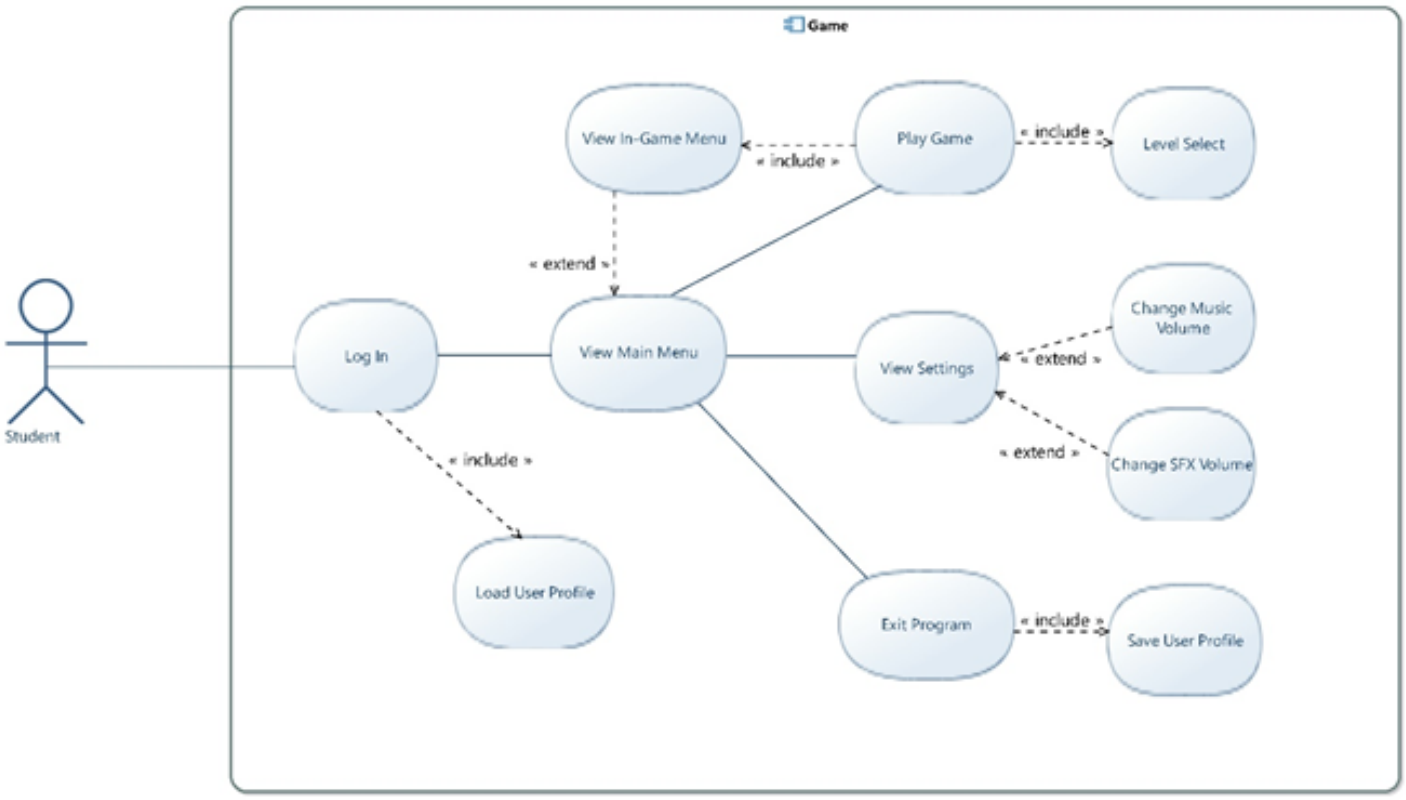
5. Summary Screen
    i. Upon completion of a level, the summary screen presents the number of correct answers and incorrect answers the user had.
    ii. If the user has passed the level successfully, the next level unlocks.
    iii. If the user has not passed the level successfully only an exit button will appear and it will lead the user to the main screen.
6. User
    i. The user bucket is the user-controlled portion of the game.
    ii. The user uses the left and right arrows on the keyboard.
    iii. The bucket responds accordingly, left arrow pushes bucket left, right arrow pushes bucket right.
7. Pause Menu
    i. The Pause Menu can be accessed while in the game by pressing the ‘esc’ key.
    ii. The user will be shown three options, “Resume,” “Options,” and “Return to Title”
    iii. The “Resume” button will resume the current game where the player left off.
    iv. The “Options” button allows the player to access the volume sliders for Music and SFX. Once these are set, the user may press the Back button in the Options screen to return to the Pause Menu
    v. The “Return to Title” button returns the player to the title screen.

4 Modeling Requirements

The game is a single-player experience meant to supplement existing the Common Core curriculum. As such, the use cases for our game are focused on the individual student's interaction with the game, with the goal of providing an engaging experience for incorporation into existing curricula or Individual Education Program. The use-case diagram for this game is found below.

The use case diagram has been constructed using the Universal Modeling Language using the following key:

- A solid line marks an Association, denoting accessibility of individual use cases by the Actor in relation to the current use case.
- A dotted arrow with the < include > tag marks an Include, denoting nested functionality of the use case they are being included from.
- A dotted arrow with the < extend > tag marks an Extend, denoting nested functionality that expands the functionality of the use case they extend to.



Use cases are itemized based on the IEEE standard:

Use Case Name:	Log In
Actors:	Student
Description:	The Student will Enter their username. The Game will then load the corresponding User Profile associated with that Username.
Type:	Primary, Essential
Includes:	Load User Profile
Extends:	N/A
Cross-refs:	Load User Profile

Use Case Name:	Load User Profile
Actors:	N/A
Description:	The Game loads user setting and level information for a given Username.
Type:	Secondary
Includes:	N/A
Extends:	N/A

Cross-refs:	Log In

Use Case Name:	View Main Menu
Actors:	Student
Description:	The Game displays the Main Menu.
Type:	Primary, Essential
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A

Use Case Name:	Play Game
Actors:	Student
Description:	The Student plays the game
Type:	Primary Essential
Includes:	Level Select, View In-Game Menu
Extends:	N/A
Cross-refs:	Level Select, View In-Game Menu

Use Case Name:	Level Select
Actors:	Student
Description:	The Student Selects which level they will play
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A

Use Case Name:	View In-Game Menu
Actors:	Student
Description:	The Student pauses the Game and views the In-Game Menu
Type:	Secondary, Essential
Includes:	N/A
Extends:	View Main Menu
Cross-refs:	View Main Menu

Use Case Name:	View Settings

Actors:	Student
Description:	The Student views the name and value of the Game's Settings
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A

Use Case Name:	Change Music Volume
Actors:	Student
Description:	The Student changes the playback volume of the in-game Music
Type:	Secondary
Includes:	N/A
Extends:	View Settings
Cross-refs:	View Settings

Use Case Name:	Change SFX Volume
Actors:	Student

Description:	The Student changes the playback volume of the in-game sound effects (SFX).
Type:	Secondary
Includes:	N/A
Extends:	View Settings
Cross-refs:	View Settings

Use Case Name:	Exit Program
Actors:	Student
Description:	The Student exits the program
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	N/A

Use Case Name:	Save User Profile

Actors:	N/A
Description:	The Program writes the User Data to a save file and associates the entry containing that data with the Username that was entered during Log In
Type:	Secondary
Includes:	N/A
Extends:	Exit Program
Cross-refs:	Exit Program

5 Prototype

The UI Prototype will demonstrate flow between pages within our program as represented by the use case diagram. All relevant menus will be operational with stand-in values, but must be able to be interacted with, including movement to and from each menu. The UI Prototype will not implement any Gameplay features, or the ability for a Student to Log In. “Load User Profile” and “Save User Profile” will not be implemented as they both require the “Log In” use case to be implemented beforehand.

5.1 How to Run Prototype

V1

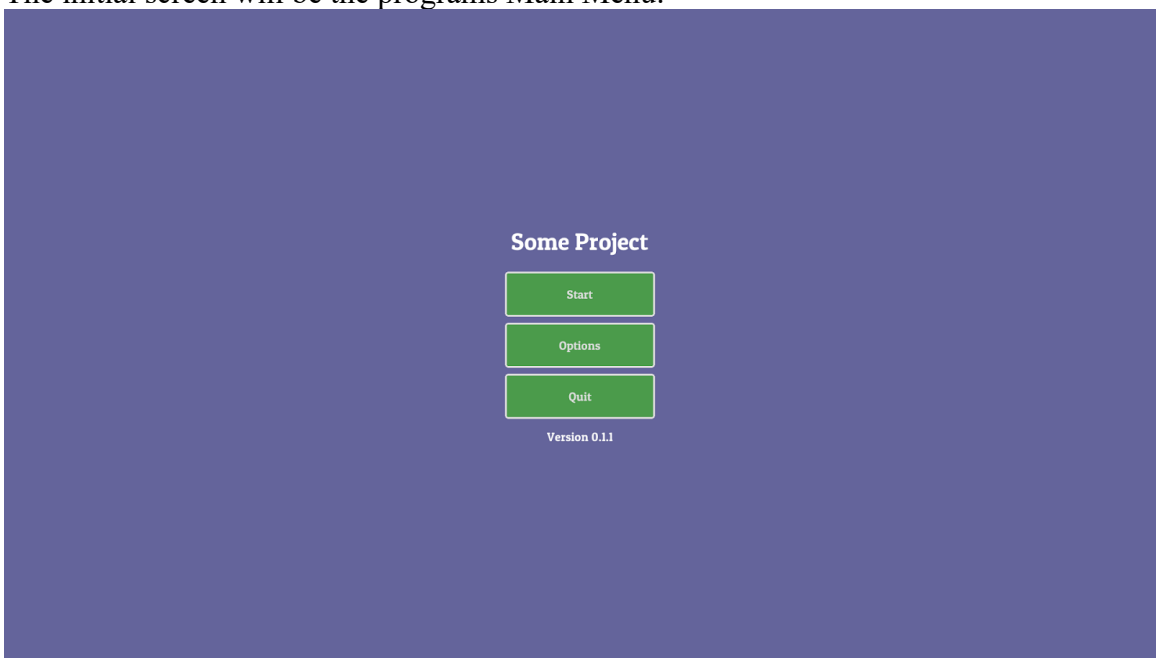
- System Requirements:
  - The Prototype will be made available as an executable compatible for 64-bit systems running Windows, Linux, or Mac operating systems.
- Dependencies:
  - Dependencies for the Prototype are packaged in a .pkg file that will be made available alongside the relevant .exe.
- Additional Configuration:
  - No other changes needed to be made
- Executing the prototype
  - The executable should be run from the same directory as the .zip file downloaded from our site

The Prototype is available for Download from the Team 6 website: http://cs.uml.edu/~bbean/resources.php

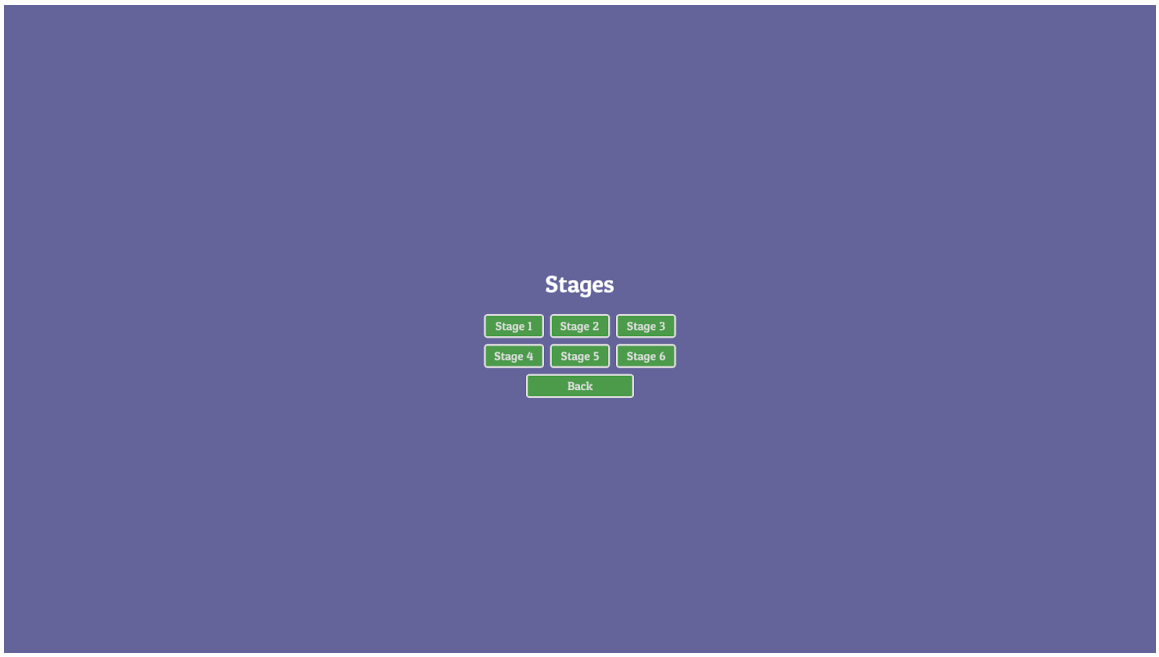
5.2 Sample Scenarios

Give a sample scenario of using your system. Use real data and problem scenarios. Include screen captures illustrating what your prototype produces. As always, be sure to describe all figures.

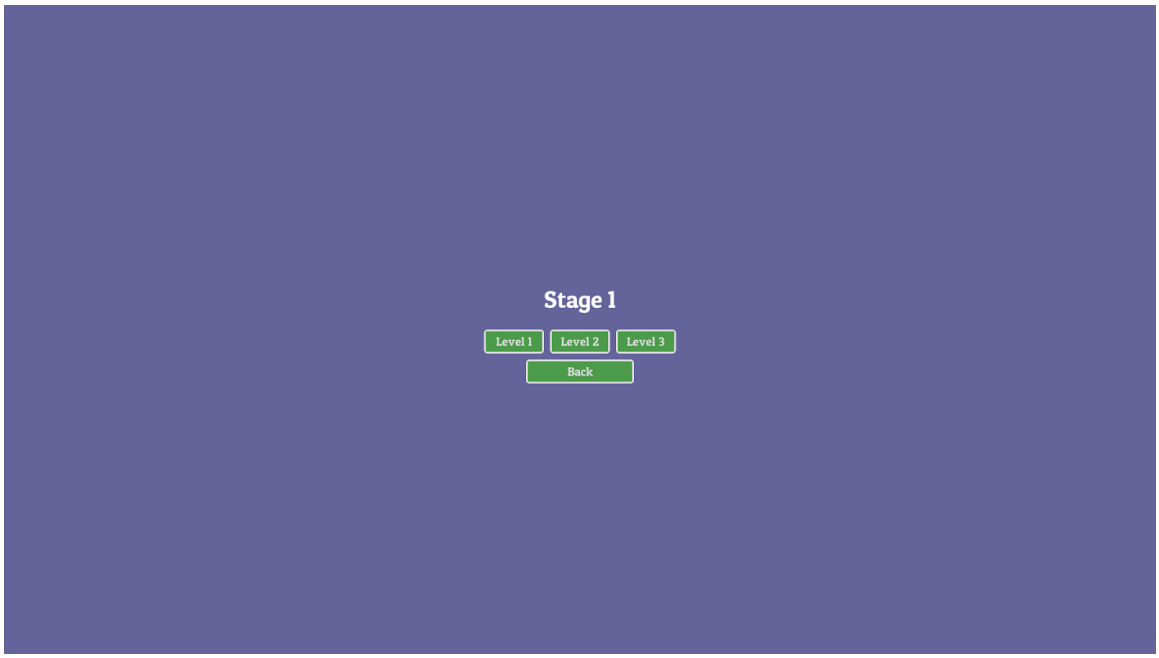
The initial screen will be the programs Main Menu:



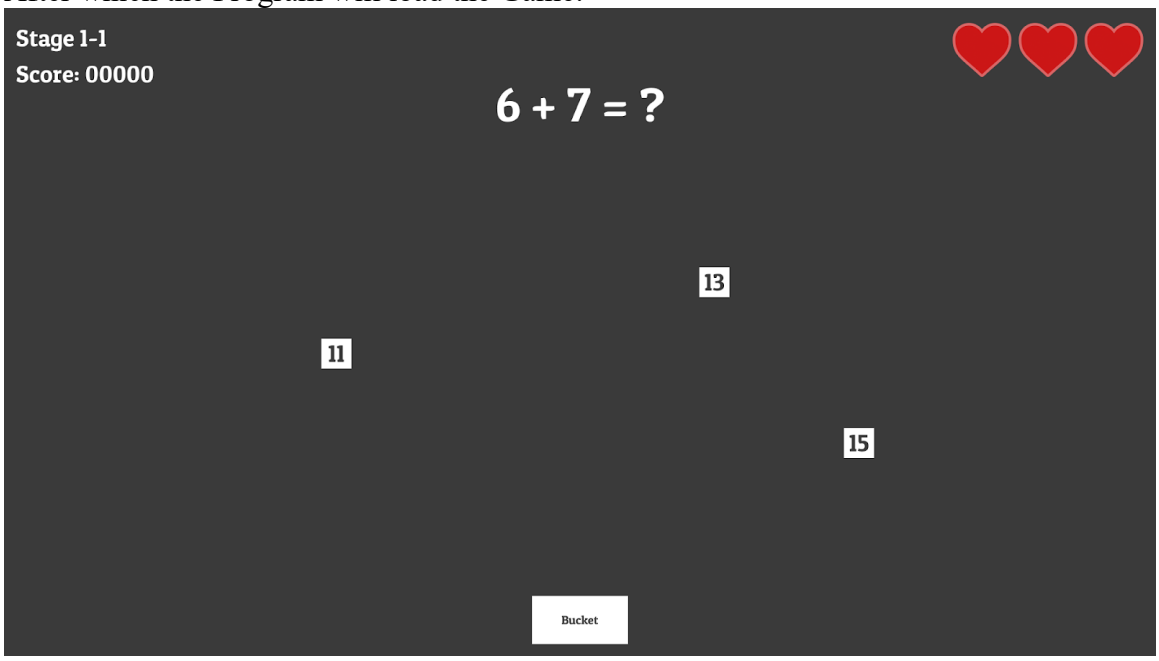
There Student may then choose to click “Start” to begin playing. This will bring up the Level Select Menu



From there the student will select which sub-stage they would like to play:



After which the Program will load the Game:



6 References

Provide list of all documents referenced in the SRS

Identify each document by title, report number, date, and publishing organization.

Specify the sources from which the references can be obtained.

Include an entry for your project website.

Start of your text.

[1] D. Thakore and S. Biswas, "Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks," Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[2] Godot, "Godot Engine – Features." Retrieved from https://godotengine.org/features, November 13th, 2019.

[3] ppy. "Game Modes – osu!catch." Retrieved from https://osu.ppy.sh/help/wiki/Game_Modes/osu!catch, November 13th, 2019.

[4] "Common Core Standards for Mathematics." National Governors Association Center for Best Practices, Council of Chief State School Officers. Retrieved from http://www.corestandards.org/wp-content/uploads/Math_Standards1.pdf, 2010.

[5] Team 6, "Team 6 Website," https://cs.uml.edu/~bbean/, 2019

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.