Software Requirements Specification (SRS)
Match Match

Team: Group 5
Authors: Eric Carey, Shuhe We, Jacob Hempel, Alex Meier
Customer: Elementary Schools
Instructor: Dr. James Daly

1 Introduction

This Software Requirements Specification document consists of the following subsections:

1. Purpose
2. Scope
3. Definitions
4. Perspective
5. Functions
6. Characteristics
7. Requirements
8. Prototype
9. How to Use
10. User Scenarios

We start by going over why the game is being made, what the scope of the software is, and terms that need to be known to understand the software. We then delve into the context of the software, what requirements and functions are involved in developing and running it, and its characteristics. And finally at the end of the document one can find the prototyping, and instructions on how to run the prototype.

As for the game itself, Math Match is designed to help elementary school students learn and develop basic math skills by computing equivalence between expression on a moveable piece and those in bucket in the main play field. Users can choose between two different levels from the landing screen, easy which only has addition and subtraction, or hard which uses expressions with multiplication division as well. Users will have to drag the piece to the bucket with an equivalent expression quickly so the field does not fill up and the game ends. Once the buckets reach the bottom of the field the game is over and the users score will be displayed.

1.1 Purpose

This document is meant to explain the functions and detailed workings of the software for our Math Match edutainment game. Several sections cover how the software works and what the user will need to know in order to interact with it. Reading this document will give the reader the knowledge to understand exactly what the system does.

The audience of our game are children between the ages of 6 and 9, typically elementary school age, who are starting to learn basic mathematics skills. Since the game is meant to help develop addition, subtraction, multiplication and division skills, the audience should at least have been introduced to these skills and understand the concept of equivalency.

1.2 Scope

The software product we are creating, and describing in throughout this document is Python based application called Math Match. This software will be used to educate young children, specifically from the ages of 6 to 9, basic math skills. They will be able to practice their addition and subtraction skills, and those more experienced can move on to multiplication and division. A benefit of having the focus on equivalent equations and numbers the game will strengthen their math skills as well as their decision making and timing. The objective of the game is to match each piece with either a number or equation on it in the hand with its equivalent number or equation bucket in the game field. The ultimate goal is to fill all of the buckets before they reach the bottom and fill the game field.

The software will facilitate drag and dropping of piece from the hand to the game field. It will also be able to evaluate equivalency and remove pieces and buckets that match up correctly. The buckets will be able to move down from the top to the bottom of the game field, only stopping when matching with a game piece. The software will time the user and generate a score based off of this time. A limitation of Math Match is the skill level it helps develop. At some point the game will become too easy and redundant for children with proficient skills in basic mathematics.

1.3Definitions, acronyms, and abbreviations

Bucket: A rectangular box holding an expression to be matched to and erased by the user

Equation: A mathematical expression consisting of two terms, utilizing either multiplication, Division, Additon, or subtraction (Example: “1+1”, “2*2”, “8/2”)

Value: A whole integer value that is equal to the associated equation. for example, if the equation is “1+1”, the value would be the integer 2.

Hand: This is the box that holds the pieces that are available for the user to drag to the play field

Math Match: An educational game designed to help practice basic math skills by matching equivalent expressions

Piece: The object with the expression the user is trying to match to an expression in a bucket.

Play field: The main area of the game that holds the buckets. This is the area where the user will drag the pieces to and match to a bucket

Pygame: A simple graphics library for python geared towards games

Python: A high-level scripting language designed to be easy to read and simple to implement

Score Counter: A 4-digit number on the top left corner of the prototype screen displaying the user's score

1.4 Organization

The rest of this document will consist of different aspects of the actual software. We begin with Section 2, a description of the system and it's different interfaces, followed by the functions being done. We then move on to sections 3 and 4, the characteristics of the user and any constraints that might be present. Requirements and use cases are also included in these sections. Finally section 5 covers the prototype and how to use it.

2 Overall Description

This section will cover the functionality and characteristics of the software and the user.

2.1 Product Perspective

This product is a stand alone game, meant to be used as a supplementary addition to teaching and testing of elementary children's basic mathematical skills. This product can be found in schools or at home for fun educational purposes. This game will work in the education environment because it will provide a fun and useful platform for students to practice their skills. Students will enjoy the game aspects while the teacher will enjoy the fact that their students are working on the fundamental skills they need to succeed. Teachers can use Math Match during open class times or suggest it as a fun game for students to play at home.

The game will be restricted to on screen controls, selected by a mouse click made by the user. The system interface will read the expression in the piece, outputting whether that is equivalent to the expression in the bucket. While the user interface will consist of mouse clicks and drags controlling the pieces. This will be how the user plays the game. Pygame styled python code will be used to run the system. Pygame allows for quick and simple development allowing the developers to create a seamless and easy to use game for the user. Any computer with python installed will be able to run Math Match. The user will simply need to run the executable that is provided and start playing. This game will take up very little memory. Only a few operations for evaluating equivalency and determining what buckets pieces are in will be needed. User will be able to customize the game by picking what difficulty game to load. This will generate either a level with addition and subtraction or with multiplication and division.

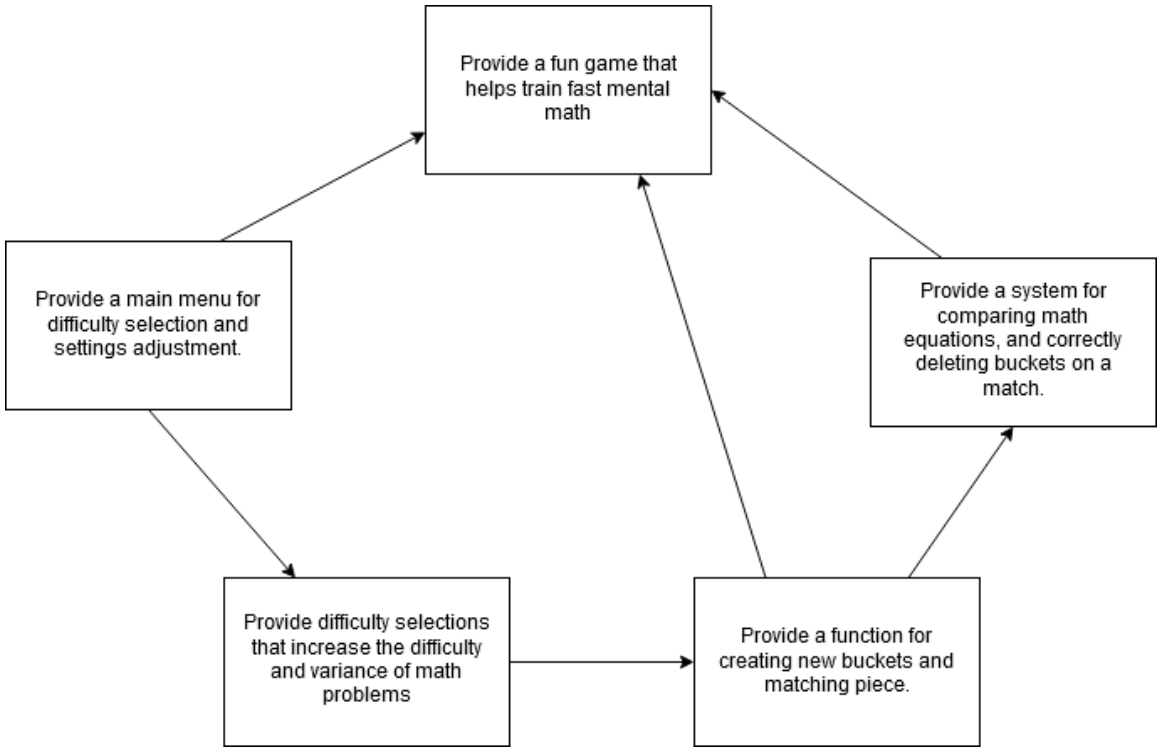
2.2 Product Functions

The major functions of the product include difficulty selection, expression creation, expression comparison, and object removal.

The game will create new equations for the users to solve. This may be done using a big list of valid equations with their solutions. This function will use the output of

the difficulty selection area to determine which equations are valid. The game will also compare equations as the user attempts to match them. This will be used to determine if the two equation blocks should be deleted, of it the user's piece should be returned to the starting area. If the two equations match, then the game will remove both of them from the game world, and award points to the user.

The following diagram gives an overview of the product's high level goals.



2.3 User Characteristics

The intended user for our edutainment game are children between the ages of 6 to 9 years old. These children will likely be in elementary school and just starting to learn basic mathematics. Users background should involve addition, subtraction, multiplication, division, as well as the concept of equality. Mixed operations will not be implemented in this version of the game so knowledge of the order of operations will not be necessary. There are several difficulties to our game, where the younger kids can

practice simple addition and subtraction and the older kids can move on to multiplication and division. The desired audience should have a simple level of knowledge of how a computer works and be able to operate a mouse to drag and drop the pieces into place. Just a small amount of skill will be sufficient enough to operate and play this game.

2.4 Constraints

The application will be limited to the scope of the Python language and the Pygame library as these are the tools used to design the game. Development may also be constrained by the problems that come with collaboration. The developers will need to work together and divide tasks causing some to rely on others to get their parts done correctly. The only safety properties that need to be considered are developers compromising the system when possible additions to the complexity of expressions is made for future levels.

2.4 Assumptions and Dependencies

In order to run this application, a python3 installation is required as well as the pygame graphics library. Likewise, the user either has these installed on their system, or have the permissions required to do so. Additionally, it is assumed that the user will have some basic computer knowledge such as “dragging and dropping” items, and using the mouse to interact with buttons on screen. The user must have a computer running Windows, OSX, or any linux distribution. This application is not compatible with IOS, Android, or other mobile operating systems (including ChromeOS).

3 Apportioning of Requirements

This product will not have score or time based requirements to move on to the next level. The level to be loaded is simply a decision made by the user at the beginning of the game. There will also not be a change in complexity of expressions. The different levels will simply separate just addition and subtraction from all four major operations. Another aspect that will not be included in this version but may be added later is the addition of various sizes and shapes of the buckets. Everything will be symmetrical and move in a static pattern.

3.1 Specific Requirements

1. User should be asked to select either easy mode or hard mode.
    a. Easy mode requires mastery of addition and subtraction.
    b. Hard mode requires mastery of higher level math concepts such as multiplication and division
    c. When the user moves the mouse on the easy mode it should show suggested age 6-7.

    d. When the user moves the mouse on the hard mode it should show suggested age 8-9.

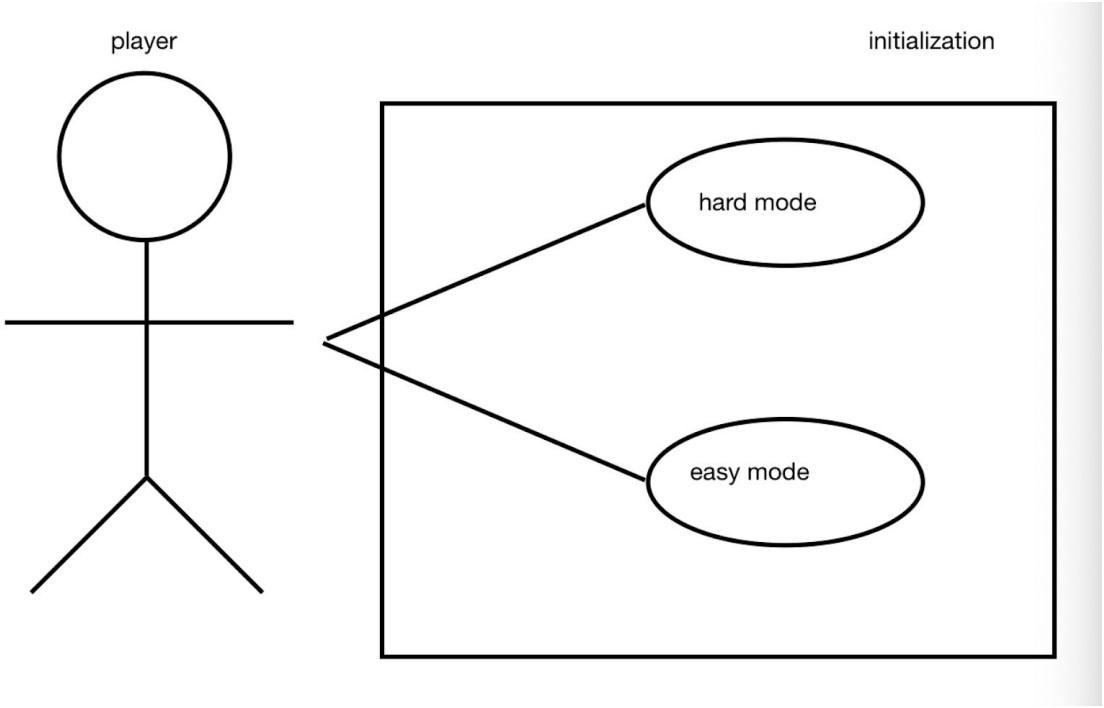
2. Program should generate 3 equations with different values after the user chooses the difficulty of the game.
    a. Each of the equation should be displayed in a bucket on the top of the screen.
    b. Easy mode will include addition and subtraction, and no more than 3 numbers in each equation.
    c. Hard mode will expand on easy mode. In addition to addition and subtraction, some equations will include multiplication and division.
        i. No single equation will contain both additive and multiplicative operators(ie. “2x2+2” will not be a possible equation) in order to avoid order of operations for children of this level.

3. When the game begins there will be 3 buckets in the playfield, and a set of 3 pieces in the hand. each piece in the hand should have a value that corresponds to one of the buckets.
    a. The score awarded for matching a piece with a bucket is equal to the value of the equations multiplied by a combo multiplier (the combo counter increases by one each time the player makes a correct match, and is reset to 1 when the player makes a mistake.
    b. over time, the buckets in the playfield will move from the top of the screen towards the bottom of the screen. New buckets are added in above the first row as it falls down the screen.
    c. The scrolling speed slowly increases over time.

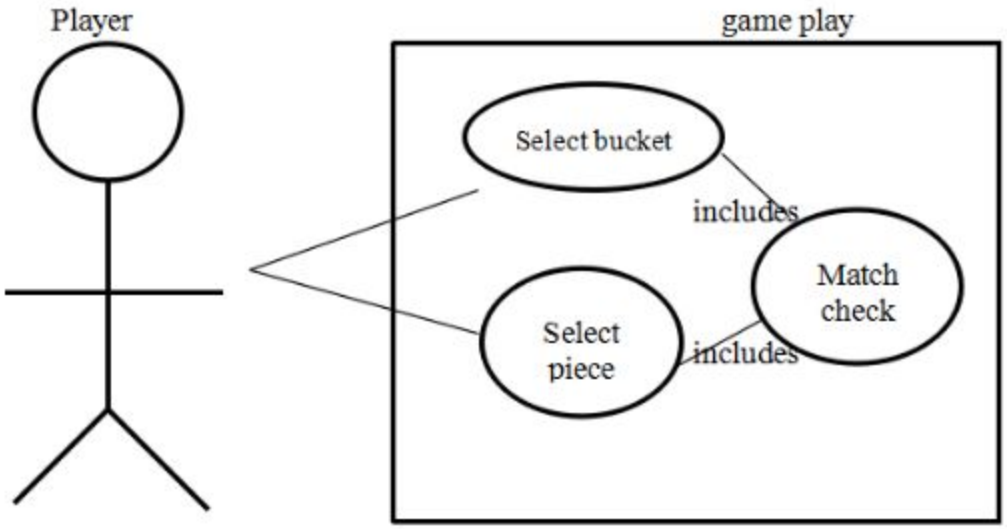
4. When user drags and drops a piece to a bucket with an equivalent value, both will disappear from the screen, and the game will add the appropriate score.
    a. Score is displayed in the upper left corner of the window.
    b. Initial score should be 0
    c. The piece gets deleted only when the value of the function in the piece matches the user’s choice in the top buckets
5. When the user pulls the wrong piece to the wrong bucket then the piece should go back to the original place in the hand.
    a. When the user chooses the wrong function, the combo counter is reset.

6. When the buckets reach the bottom of the screen, the is game over
    a. The score will show on the screen
    b. Total time elapsed will show on screen
    c. The player is given the option to restart the game

4 Modeling Requirements

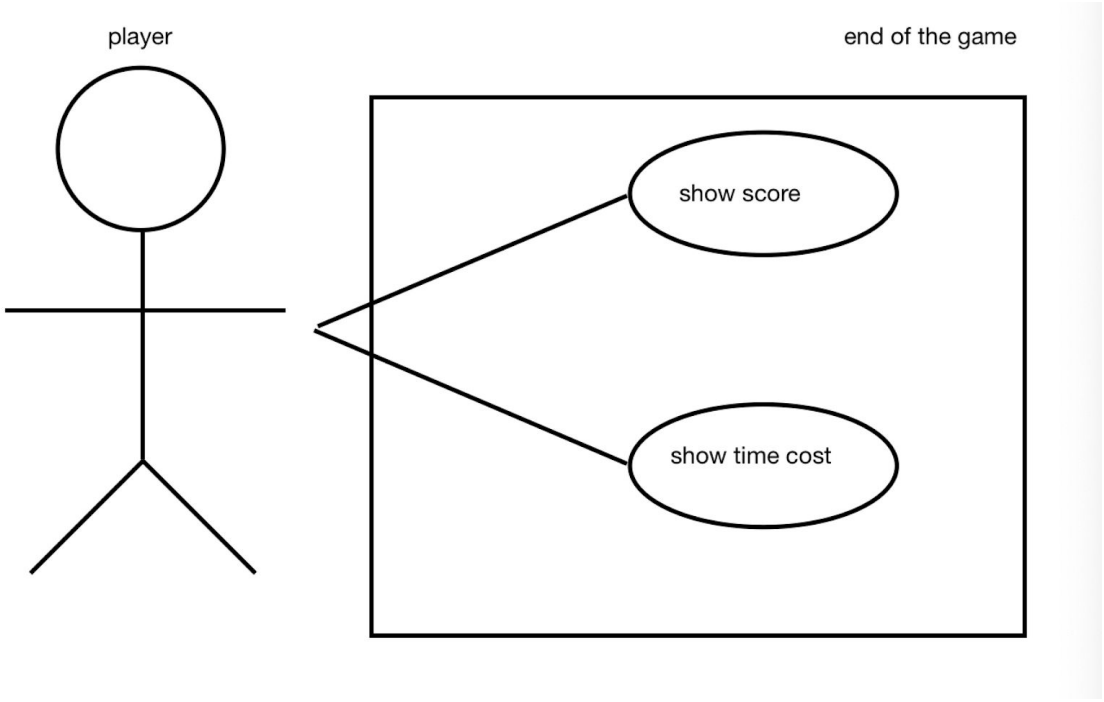


Use Case Name:	Initialization
Actors:	Player
Description:	Before the game starts, the player will be prompted to choose a difficulty setting. A pop up window displays buttons for easy, and hard.
Type:	Primary and essential
Includes:	
Extends:	
Cross-refs:	
Uses cases:	Initialize game



Use Case Name:	Game play
Actors:	Player
Description:	When game begin, under the buckets on the top of the screen, one function that has the same value from one of the function in the bucket should shows and as time goes by there should be more and more function shows in a roll.Pieces can be pulled to bucket. When user select a function in the bucket below and then pull it to the function has same value in the top bucket then perform right match.When user select a function in the bucket below and then pull it to the function has different value in the top bucket then perform wrong match. When the last bucket reach the bottom of the screen then perform End the game and show score.
Type:	Primary and essential
Includes:	Match check

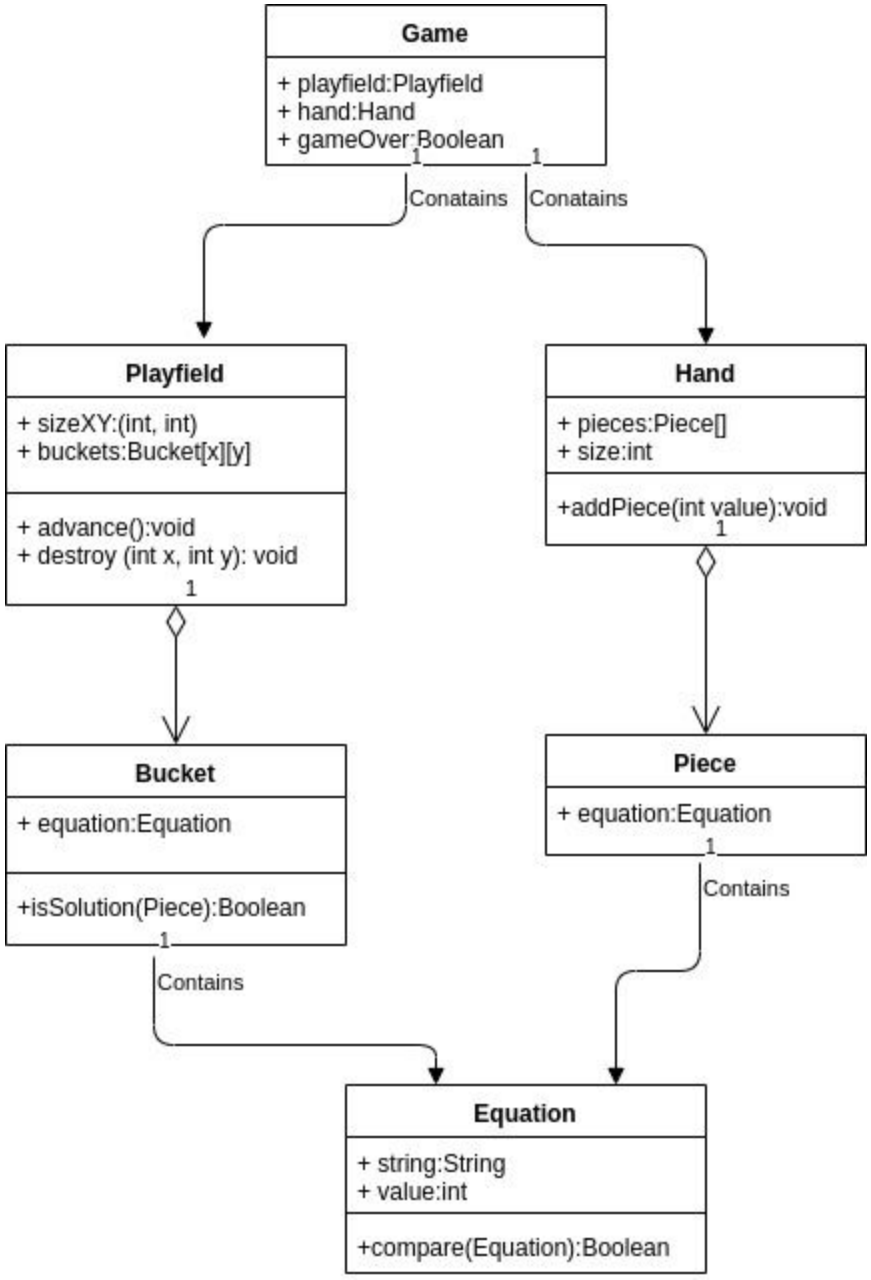
Extends:	
Cross-refs:	
Uses cases:	End the game and show score



Use Case Name:	End of the game
Actors:	Player
Description:	When the last piece reach the bottom of the screen then the game should over. Program will show the score and time cost on the screen.
Type:	Secondary and essential

Includes:	Screen
Extends:	
Cross-refs:	
Uses cases:	None

Class Diagram



Data Dictionary

Element Name	Description

Playfield		Contains Buckets, represents the "Game Board"
Attributes		
	sizeXY:Integer Tuple	Width and height of play field grid
	bucket:Bucket[x][y]	2 dimensional array of bucket objects
Operations		
	advance ():void	Advances the buckets 1 step downward
	destroy (int x, int y): void	Removes the bucket at the given coordinates.
Relationships	Contains a grid of Bucket objects.	
UML Extensions		

Element Name		**Description**
Bucket		Contains an equation
Attributes		
	equation:String	Mathematical equation represented as a string ("3+1", "1 + 1" etc.)
	solution:Int	Integer solution to math problem used to compare with answer Pieces
Operations		
	isSolution(Piece):Boolean	Returns true if the Piece object given is a solution to the Bucket.
		Description4

Relationships	
UML Extensions	

Element Name		**Description**
Hand		Represents the players hand of playable pieces.
Attributes		
	pieces:Piece[]	Array of Piece objects.
	size:int	Maximum hand size
Operations		
	addPiece(int solution):void	Add piece with the given solution to the hand
Relationships		
UML Extensions		

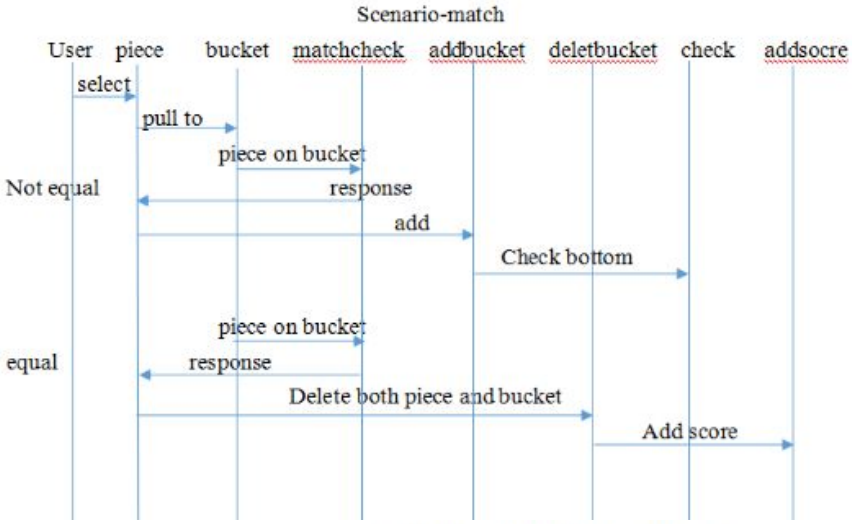
Element Name		**Description**
Piece		Represents a piece that can be played by a player
Attributes		
	equation:String	Mathematical equation string
	solution:int	Integer solution to Equation for comparison with Bucket objects.
Operations		

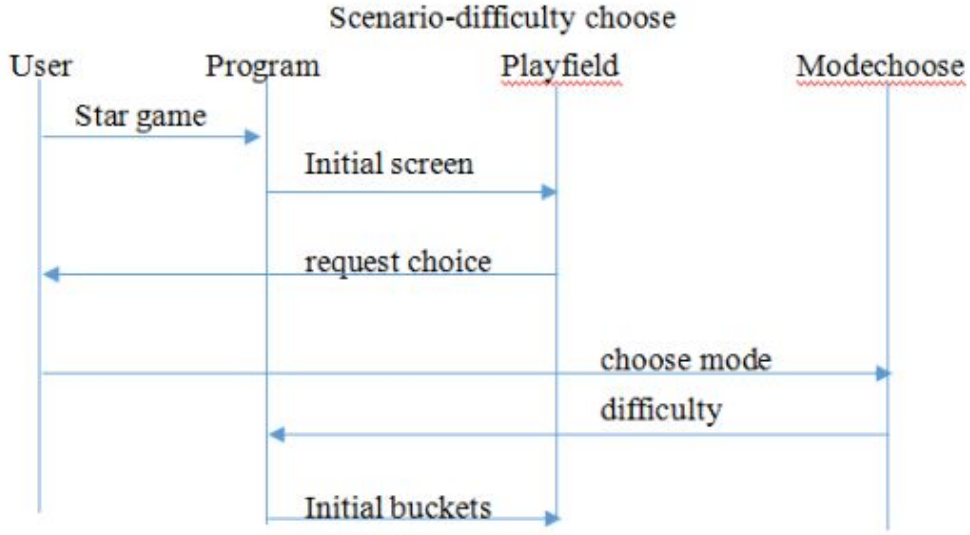
Relationships	
UML Extensions	

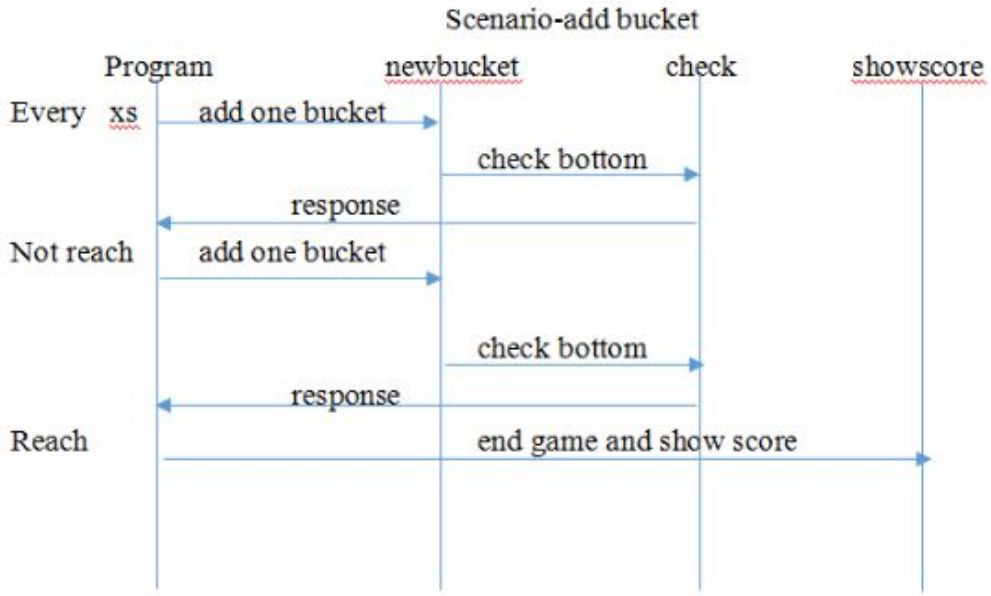
Element Name		**Description**
Equation		Contains a string representation of an equation and an associated
Attributes		
	string:String	Mathematical equation represented as a string (“3+1”, “1 + 1” etc.)
	value:Int	Integer solution to math problem used to compare with answer Pieces
Operations		
	compare(Equation):Boolean	Returns true if the given equation object is a a valid solution.

Element Name		**Description**
Game		Singleton object that keeps track of game state.
Attributes		
	playfield:Playfield	Instance of Playfield Object

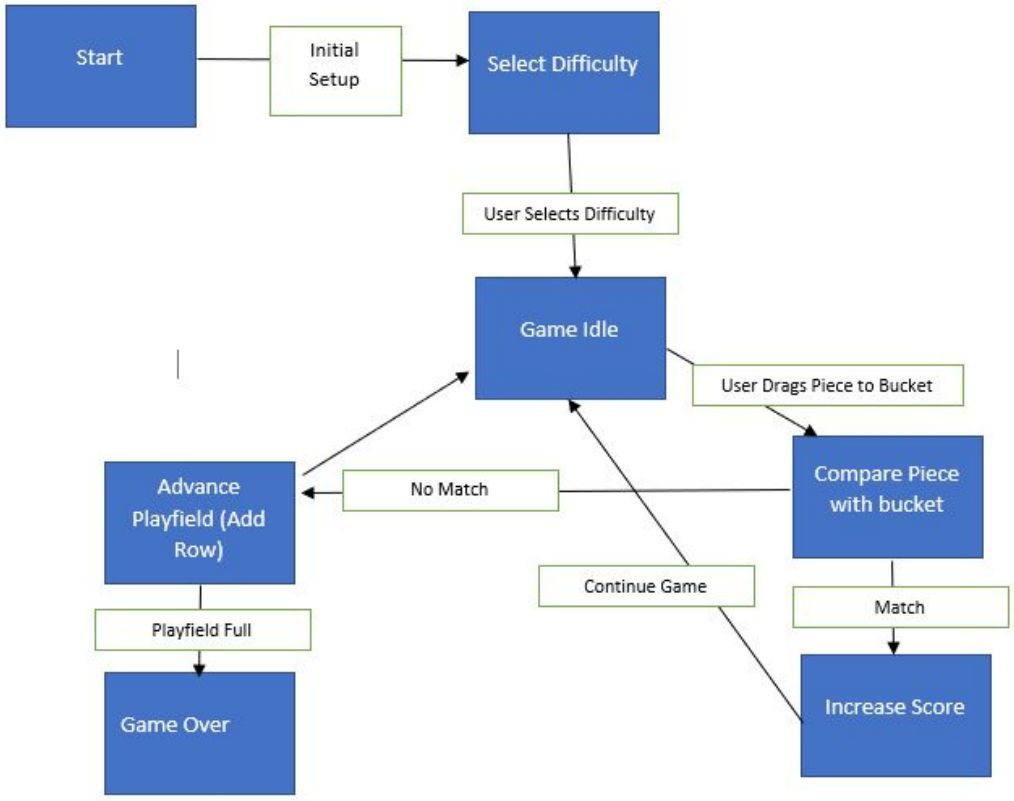
	hand:Hand	Instance of Hand Object
	gameOver:Boolean	Set to true if a game over condition is met.
Operations		
Relationships		
UML Extensions		







State Machine:



5 Prototype

The Prototype shows a basic representation of the UI elements that will be present in the full version of the game. It demonstrates the process flow from starting the application and selecting a difficulty from the startup menu, then entering into the main gameplay loop. When a gameover state is reached (I.E. the buckets reach the bottom of the playfield), a simple game over screen will show in the window, along with the final score and elapsed time. This prototype includes the functionality required to demonstrate the premise of the game, however, the graphics and colors are a placeholder, and are something to be improved upon in the future.

5.1 How to Run Prototype

This prototype requires a Python 3.0 installation. Documentation on installing python and its dependencies can be found on the official python website (https://www.python.org/download/releases/3.0/). Once python is installed, the required dependencies for the prototype can be installed using the pip3 command in your operating systems command interpreter:

Pip3 install pygame

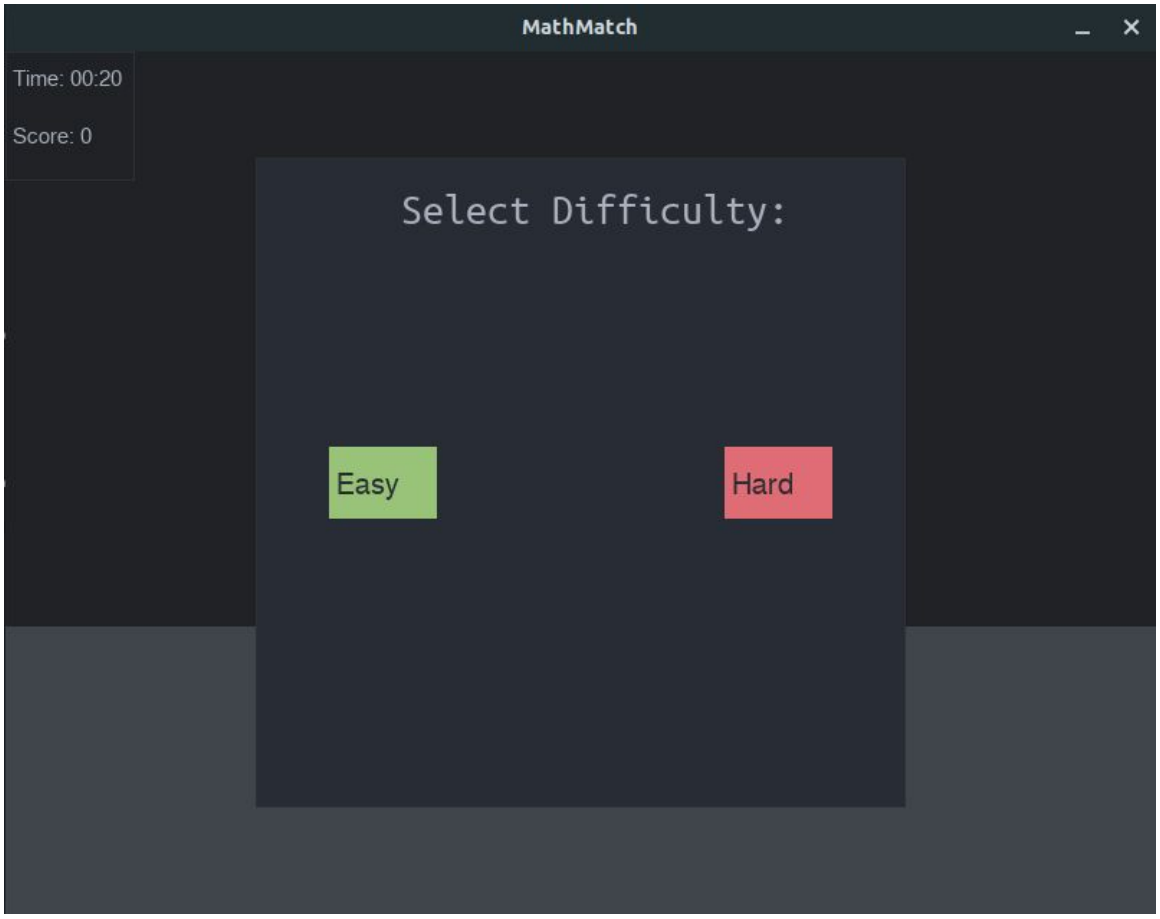
Pygame is compatible with any of the major operating systems (Windows, OSX, Linux). Once the above requirements are met, the prototype can be cloned or downloaded from the following repository:

https://github.com/alexmeier2828/SoftwareEngineering

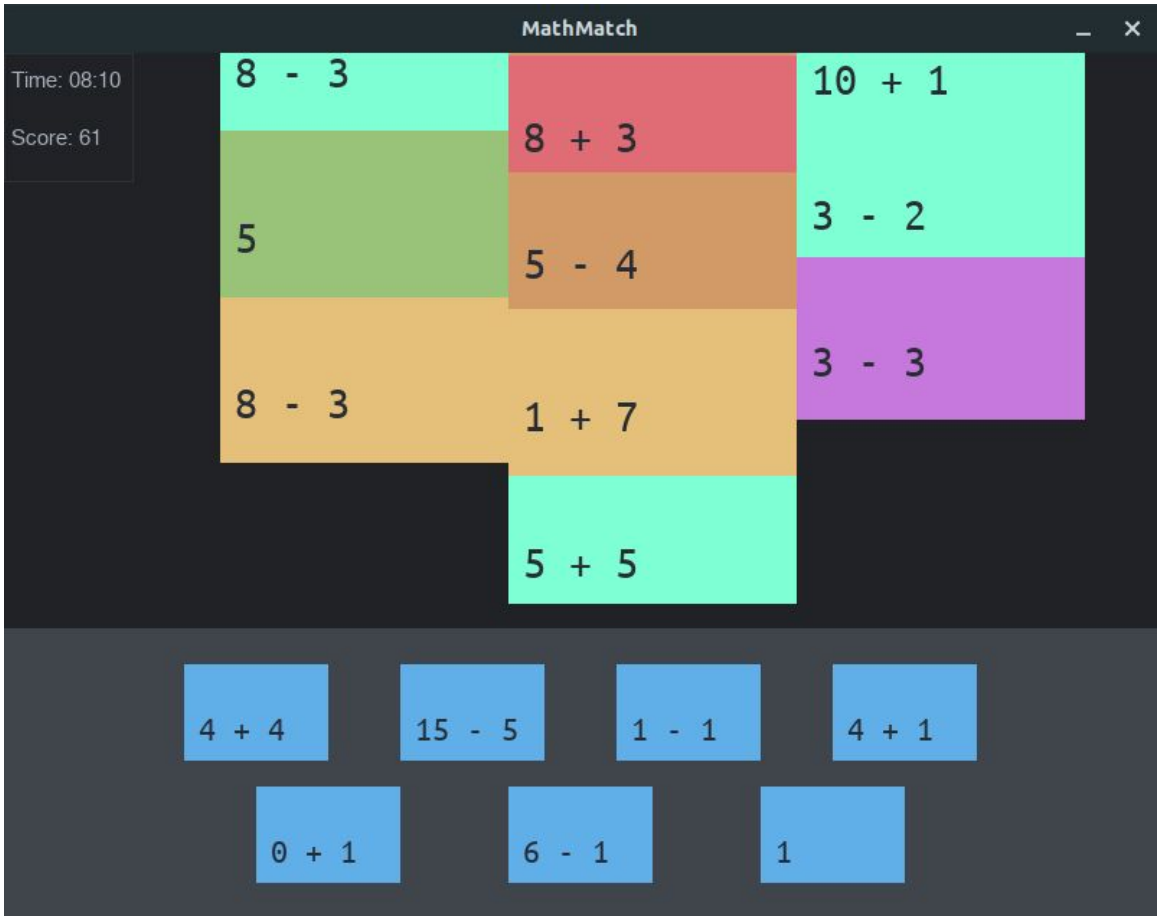
To run the prototype, navigate to the SoftwareEngineering/Prototype folder, then run the following command:

python3 main.py

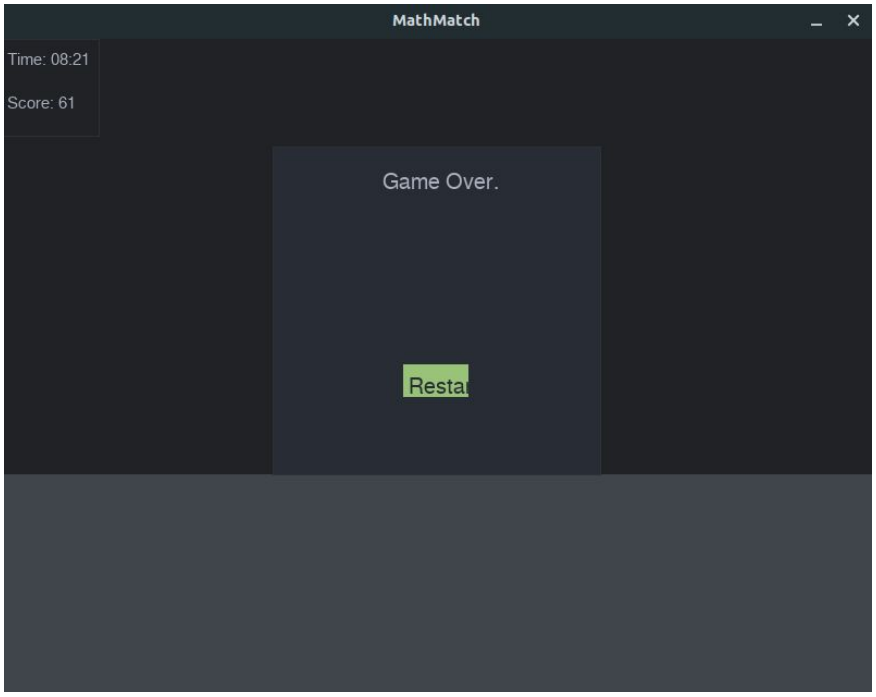
6 Sample Scenarios



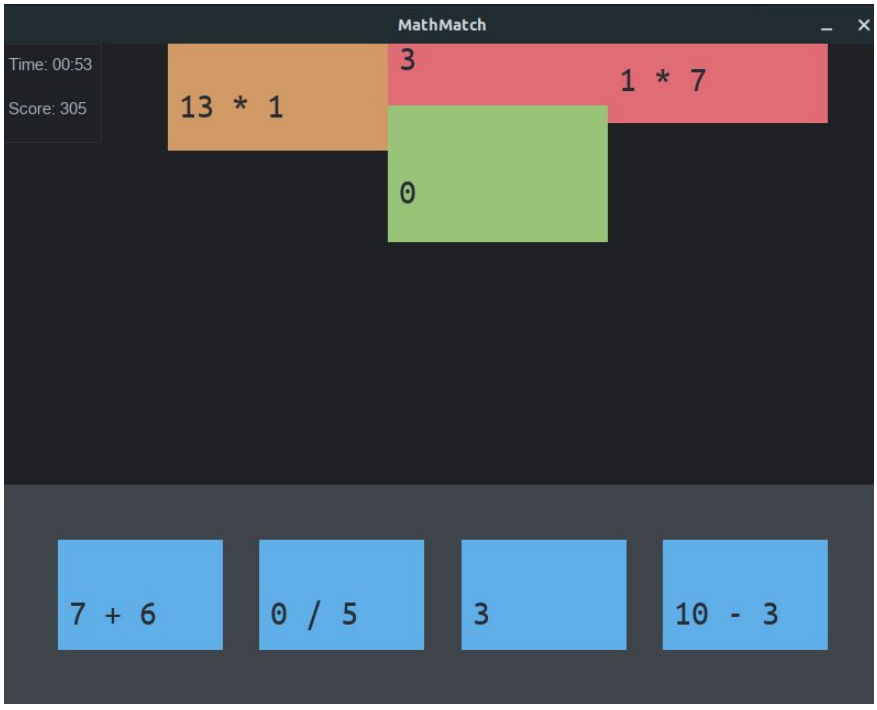
This screenshot shows the first screen that users will see when they launch MathMatch. It provides a difficulty select screen, with two difficulty screens.



This screenshot shows a user playing MathMatch on easy mode, because there are only addition and subtraction equations on the rectangles. You can also see that the “5 + 5” block is about to hit the bottom of the dark gray area. If that happens, the user will see the following screen:



This is the game over screen. The user's final score can be seen in the top left corner. Hitting the restart button will return the user to the main menu, where they could choose a new difficulty.



This screenshot shows the user playing on hard mode, which can be seen because of the multiplication and division equations that appear on the rectangles.

7 References

1. Anonymous “Class Ages & Placement Information | Cobham | ACS International Schools, 2019, www.acs-schools.com/cobham/admissions/grade-placement.

2. Bartłomiej, Burek “Python-Examples”, n/a , n/a, October 2019. www.github.com/furas/python-examples/tree/master/pygame

3. Carey, Eric, et al.“Math Match.” Math Match Game, 2019, www.cs.uml.edu/~ameier/website/website.html

4. Christensson, Per. “Python Definition.” TechTerms. Sharpened Productions, 15 June 2010. Web. 25 November 2019. https://techterms.com/definition/python.

5. Standards, Common. “Grade 3 " Introduction.” *Introduction | Common Core State Standards Initiative*, Common Core State Standards Initiative, 2019, www.corestandards.org/Math/Content/

6. Master, Post. “Rough Guide to US School Grades Compared to English (UK) School Years.” *A Comparison of English (UK) and American (US) Education Systems*, 2019, www.free-for-kids.com/uk-us-education-systems.shtml

7. Meier, Alex, et al. “Math Match Repository .” *Math Match*, GitHub, 2019. https://github.com/alexmeier2828/SoftwareEngineering

8. Learning, Time. “Elementary Math Curriculum.” *Time4Learning*, 2019, www.time4learning.com/elementary-math.shtml

8 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.