Assessing the Educational Value of an Equation Solving Mobile Game

COMP4905 – Honours Project

Jamie Paisley
jamiepaisley@cmail.carleton.ca

Supervised by: Robert Collier, PhD

Carleton University
School of Computer Science
5302 Herzberg Laboratories
1125 Colonel By Drive
Ottawa, ON, K1S 5B6
Abstract — The objective of my project was to create an app that would be able to improve the user’s ability to solve math equations quickly and accurately. To test this, I first created a game. I then created two quizzes, Quiz 1 and Quiz 2, that I used to test participants. I recruited student participants via email, then split them up into two groups. I had my variable group take Quiz 1, then play my game for 10 minutes, then complete Quiz 2. I had my control group do something unrelated to math for 10 minutes between taking the two quizzes. To check the validity of my hypothesis, I compared the time taken to complete and the amount of correct answers of the two quizzes. I found that the use of my game did indeed decrease the time taken on Quiz 2 by almost a minute on average. The control group only had a decrease of about 1 second on average. Therefore, my hypothesis that my game would improve the user’s ability to solve math equations quickly was correct. Unfortunately, there was no statistically significant difference between the control group and the variable group when comparing the quiz scores between the two quizzes. Therefore, my hypothesis about the accuracy improvement was incorrect.
**ACKNOWLEDGEMENTS**

I would like to thank all of my participants who graciously volunteered, during this very busy time. Without them this project would be worthless. As well as my supervisor, Robert Collier, who definitely kept me busy these past months, but ultimately helped me succeed in making the best project I could.

**TABLE OF CONTENTS**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>6</td>
</tr>
<tr>
<td>Background</td>
<td>7</td>
</tr>
<tr>
<td>Development Process</td>
<td>10</td>
</tr>
<tr>
<td>Experimental Design</td>
<td>23</td>
</tr>
<tr>
<td>Results Analysis</td>
<td>26</td>
</tr>
<tr>
<td>Quantitative Analysis</td>
<td>26</td>
</tr>
<tr>
<td>Qualitative Analysis</td>
<td>36</td>
</tr>
<tr>
<td>Future Work</td>
<td>36</td>
</tr>
<tr>
<td>Conclusions</td>
<td>37</td>
</tr>
<tr>
<td>References</td>
<td>39</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1.............................................................................................................. 7
Figure 2.2............................................................................................................... 8
Figure 2.3............................................................................................................... 9
Figure 3.1............................................................................................................. 10
Figure 3.2............................................................................................................. 11
Figure 3.3............................................................................................................. 11
Figure 3.4............................................................................................................. 12
Figure 3.5............................................................................................................. 13
Figure 3.6............................................................................................................. 14
Figure 3.7............................................................................................................. 14
Figure 3.8............................................................................................................. 15
Figure 3.9............................................................................................................. 16
Figure 3.10......................................................................................................... 16
Figure 3.11......................................................................................................... 18
Figure 3.12......................................................................................................... 20
Figure 3.13......................................................................................................... 21
Figure 3.14......................................................................................................... 21
Figure 3.15......................................................................................................... 22
Figure 4.1............................................................................................................. 23
Figure 4.2............................................................................................................. 24
Figure 4.3............................................................................................................. 25
Figure 4.4............................................................................................................. 25
LIST OF FIGURES CONTINUED

Figure 5.1 ......................................................................................................................... 27
Figure 5.2 ......................................................................................................................... 27
Figure 5.3 ......................................................................................................................... 28
Figure 5.4 ......................................................................................................................... 28
Figure 5.5 ......................................................................................................................... 29
Figure 5.6 ......................................................................................................................... 30
Figure 5.7 ......................................................................................................................... 31
Figure 5.8 ......................................................................................................................... 32
Figure 5.9 ......................................................................................................................... 32
Figure 5.10 ....................................................................................................................... 33
Figure 5.11 ....................................................................................................................... 34
Figure 5.12 ....................................................................................................................... 34
Figure 5.13 ....................................................................................................................... 35
The objective of my project was to create an app that would be able to improve the user's ability to solve math equations quickly and accurately. I did this by creating a fast-paced game that involves having the user quickly solve math equations that are raining down the screen. Each equation would have the answer or an operand swapped with a “?”. To solve these equations, the user would have to swipe the equation into the column labeled with the missing number for that equation. For example, the equation “6 x ? = 42”, would have to be swiped into the column labeled “7”. I originally had the idea for this game about a year ago. It popped out of nowhere, “Guitar Hero, but with Math”. That is the first thing I visually thought it resembled, but really the only similarity is between the movement of the notes in Guitar Hero and the equations in my game. They both fall down the screen, but instead of having to click the correct button (in Guitar Hero), in my game you have to slide the equation into the correct column before it reaches the bottom of the screen. Preparing for my honours project, I started altering and expanding the previous idea. I tried to figure out how this quick solving math game could benefit people in a fun way.

After brainstorming I started thinking about the educational value of the game and its potential to benefit its users. The problem I noticed even in people my age was their ability to solve basic math equations correctly, in a fashionable time. It isn’t the biggest problem, but it is relevant even in people my age. For the purpose of testing, this project’s current target audience are people my age, and in my case that consisted of 1st to 4th year students anywhere between the age of 17-21 at Carleton University that are majoring in Computer Science. The participants in both group seemed comfortable in solving basic math equations, and seemed like they were all on the same playing field. Even if they did not enjoy doing math, majoring in Computer Science means having to take some mandatory algebra and calculus courses. Given that the amount of math courses taken may change based on what year the participant is in, it seemed fair since these were only basic math equations. The idea is that the use of my game will improve the player’s speed in completing an equation and its correctness.

Math is something that everybody is going to have to do for the rest of their life. If we could save some time by accurately solving these equations faster, that could help a lot of people out, especially those majoring in Computer Science. My original thought was for this game's main demographic to be people my age., and for the purpose of this project it was. In the future I would like to either add a new
game mode, or create a whole new game altogether, that would be suitable for children. I believe this game is perfect for kids, because they are always learning and increasing their knowledge of math.

II. BACKGROUND

In preparation for this report I did some research to find games similar to mine in an effort to find a suitable competitor. Quickly, to my surprise, I found many math games where the object was to type in the answer of a math equation as quick as possible. There is one game that has been around for many years, although it isn’t a math game, it has managed to successfully add a fast-paced educational value to a rather fun game, and that is *The Typing of the Dead*. It started off as a cabinet arcade machine made in 1999, Japan. It became so popular that it branched out to many different platforms and had multiple games in the series, with the latest one being released in 2013. This game places you in a zombie outbreak, and the only way to kill the zombies is to type a word that is hovering over the zombie before they reach you. Below is a gameplay screenshot of the original game, as you can see the player must type the words “Struggle” and “Bodies” before it is too late.

![Screenshot from the original The Typing of the Dead](image)

*Figure 2.1: Screenshot from the original The Typing of the Dead*
Even though *The Typing of the Dead* has nothing to do with math, it trains the player in a similar way to my game. Prompting the user and having them complete a task in a limited amount of time is the base gameplay of both games. Ideally overtime each game will help you complete a certain task. In the case of *The Typing of the Dead*, it helps players improve their typing speed, whereas my game helps players improve their math solving speed. Both games also encourage reflexive ability and the correctness of the task, because if you can’t type the word correctly in one try, the zombies are more likely to get to you before you regain your bearings and finish typing the word. My game encourages correctness because if you put an equation in the wrong column, then you will instantly lose once the equation gets to the bottom of the screen. Seeing that there is clearly a market for this type of fun and educational game, is really inspiring to further develop my game. One thing my game needs before release is a more fun gameplay. This task can be difficult considering the definition of “fun” differs between players, but will greatly improve once I add visuals and sound effects.

The game I found that was most similar to mine was *Math Games (Fast)* by Eğlenceli Bilgi Oyunları. *Math Games (Fast)* is a math solving puzzle game where the user is prompted with an equation and they have to solve it by clicking one of the 4 given answers. The game seems to feature 15 levels, where you can only play the next level by completing its predecessor. Each level consists of completing 4 questions, an addition, a subtraction, a multiplication, and a division equation. You have 3 seconds to complete each question. If you do not succeed to select the correct answer in 3 seconds, you fail, and have to restart that specific level. I barely made it to level eight since it requires you to read, compute, find, and click the answer to a pretty complex equation in only 3 seconds. Here are some examples of equations found on level 8:

![Figure 2.2: Screenshots from Math Games (Fast) gameplay](image)
As you can see they are quite difficult equations to solve in 3 seconds, if not impossible to solve on your first try. *Math Games (Fast)* definitely tests you in speed, but I believe my game tests you in a more efficient way. Instead of challenging the user with almost impossible equations to solve in the given time, I give the user multiple simpler equations to solve in a greater amount of time. Here is a screenshot of my game to better understand what I am talking about.

![Figure 2.3: Screenshot of my game in the Unity Engine](image)

As you can see there are about 10 equations on the screen at once so you have to manage your time wisely to correctly place every equation before it reaches the white bar, which triggers the answer check. There is definitely something to learn from this game considering it has a 4/5 star rating and 100,000-500,000 downloads on google play.
III. DEVELOPMENT PROCESS

The second I had the idea for my game, I drew a sketch. A lot has changed but I still have the same initial concept. Below is a picture of the actual first sketch I ever made for this game.

Figure 3.1: Image of the first sketch I ever made for this game

My original idea was to have the math equations coming down conveyor belts being packed into boxes and shipped out. The name was going to be “Math Packing”. After some research I quickly realized that the animations that would be required to make a game like this would be too taxing, so I packed it away and forgot about it for months. Come August 2017 I had to start thinking about my honors project, so I revisited the idea of Math Packing. I thought of having a more modern style of gameplay like the games you often see made by Ketchapp.
Even though the current state of the art for my game is lacking entirely, I want it to look like similar to those Ketchapp games when it is finished. They have a great way of making a game look beautiful with such a minimalist design.

The reason I want it to be a minimalist design is because it allows me to achieve a great looking product in a really cheap way. It takes less time, it takes less power to run versus full 3D-animated graphics, it is clean and beautiful, and is very popular right now in mobile games.
The first challenge that I came across was learning how to use the Unity game engine. I’ve never used a game engine before and this was the first real game I have ever made. That resulted in me frustratingly tinkering with Unity non-stop for about 2 months. The first issue I had was wrapping my mind around the concept of building this game. I started off by trying to create the conveyor. I knew I had to figure out a cost effective way of having math equations constantly raining down the screen. I couldn’t just spawn an infinite amount of rows that scrolled down because if the user got a high enough score the game could eventually crash with too many objects to render. I came up with the idea of making a conveyor that would endlessly loop, similar to a treadmill. “Conveyor” is the name I gave to the boxed grid that is constantly scrolling down the screen. I needed to figure out a way to have the conveyor have an endless amount of rows scrolling down. I first started this game as a 2D project but then quickly realized that I had to switch to 3D to achieve the visuals I wanted. I wanted it to look similar to Guitar Hero in the way that the equations at the top of the screen would appear smaller and gradually get larger as they approach the bottom of the screen.

![Figure 3.4: Screenshot from Guitar Hero by Red Octane and Harmonix](image)

As you can see in Figure 3.4 that the angle of the camera in respect to the neck of the guitar is pretty shallow. I first started with a similar camera angle, but it was too hard to read the equations near the top of the screen so I went for a steeper angle, closer to a top down view. This allows the user to more effectively see all equations on screen, while still having that perspective camera effect.

To create the conveyor, I started off using 3D cubes as individual cells of the grid. This this was my first experience building in a 3D engine with its own built-in physics engine. I quickly learned to turn off all friction, gravity, and collision properties for all my game objects. I spawned about 16 rows holding three cubes each and wrote a script to make them move down the screen. My idea was to have enough rows so that when a row of cubes went off screen it would spawn back at the top of the conveyor. I did this by using a specific y-position inside of a FixedUpdate() function which calls the
update function every frame. This calls it at a fixed rate to account for lag. Even though I was using `FixedUpdate()` it seemed that as time went on there would be gaps between rows.

![Figure 3.5: First stages of equation movement](image)

I unfortunately don’t have a picture of this happening to the actual rows themselves, but the exact same thing happened when I started spawning in the equations at a fixed y-position, while they had a movement script attached to them. I know it may be difficult to see in this picture, but if you look at the “8 - ? = 2” equation, you can see that the gap below it seems to be smaller than the gap above it. I knew this wouldn’t do, so instead of using a specific y-position, I had to figure out a way to get a reference to the very top row. Whatever row that is, I knew if I could store its y-value somewhere then when a row needs to loop back to the top it can do so by setting its y-value to the y-value of the top row plus the height of one row. Then set the “top row” to be the row you just moved to the top so the next row that loops has a reference. This was my first big obstacle in development. I eventually solved the problem by first setting the variable “tempRow” to Row1. Row1 was a game object that I built in the Unity editor and not in script. It was the initial “top row”.
Then, when the first row had to loop back to the top of the screen, it used tempRow as a reference. This allowed me to move the row to the exact position it needed to be in. After that, I set tempRow to equal the row that was just moved. Once I had the conveyor belt working the way I wanted it to, I got working on the creation of randomized equations.

I wanted everything in my game to be randomized. Everything from the operators (+, -, ×, ÷) and operands (2-9, inclusive) being used in the equations, to the other numbers in the equation being used, to the position of equations and the answers chosen for each round. For my operands and answers I randomly chose a number between 2 and 9 inclusive. For my equations I wanted every operator, operand and the spawning position to be randomized. Overall, getting this feature to work and getting those equations on screen was definitely one of the biggest challenges throughout this project. It took me 7-8 days, working anywhere from 4-6 hours every day to get all randomization up and running.
This is an example of a bug in a feature I added, that would randomly switch the position of the second operand and the answer. I decided that my division equations needed to add some versatility to the way they were being created, because I originally had it so the second operand was always the hidden value. Adding this function was a feature I wanted to add so I could spice up the gameplay a little bit. In Figure 3.8, the function “12 ÷ ? = 3” is supposed to actually look like “12 ÷ 4 = ?”, because the answer options are 3, 5, and 9. The reason I know the missing column is 9 is because of the equation above it “5 + ? = 14”. The issue was occurring in the way I was setting the text of the TextMesh Pro object. Since the equation “12 ÷ 4 = ?” was already created I needed to hide the hidden value. The problem was occurring when I was looking for the hidden value, since before implementing the division equation, the hidden value could only be in the position of the first or second operand. Figure 3.9 shows a code snippet of what was going on.
As you can see in Figure 3.9, if the hidden value isn’t in the first operand’s position then I would automatically set the “?” to be the second operand. To fix this I had to specifically check if the equation was a division equation, then I had to check whether it was in the second operand’s position or the third operand’s position. Here is a snippet of the code that fixed the issue:

```csharp
if (ans.Length == 2) {
    if (div == true && this.formula.Substring(5, 1) == string.Format("{0}", this.hiddenValue)) {
    } else if (div == true && this.formula.Substring(9, 1) == string.Format("{0}", this.hiddenValue)) {
        sb[9] = System.Convert.ToChar("?");
    } else {
    }
} else if (div == true && this.formula.Substring(4, 1) == string.Format("{0}", this.hiddenValue)) {
} else if (div == true && this.formula.Substring(8, 1) == string.Format("{0}", this.hiddenValue)) {
    sb[8] = System.Convert.ToChar("?");
} else {
}
this.cardText.text = sb.ToString();
```

**Figure 3.10: Division equation “?” fix**

As you can see in Figure 3.10, I also had to adjust my checks for the length of the answer because I was getting equations like “25 ÷ ?5 = 5”. So for equations with an answer that was 2 digits long, the second and third operand positions were moved over one index.

My original vision for this game had the number of equations that spawned per row increase as score increased. I created a function that would randomly grab a number between one and three. Based on your score, the chances of getting 3 increased the higher your score was. In theory if the
player got good enough there could be a possibility of having 30 equations on the screen at once, although that would be very unlikely. When writing this code, I didn't realize how hectic that would actually be, but I stopped trying to implement this feature because I just wanted to focus on getting anything on the screen. I set it to the side so I could go back later and work on it. After making the spawned number per row always one, and finishing the random generation I finally got the equations to print on screen. I did so using the built-in text label. I was extremely happy to finally accomplish this but there was just something not right about the way the equations looked. They were very grainy and pixelated, and didn't match the sleek style I was going for. I did some research on some Unity forums and found that there is some amazing software that has recently been bought out by Unity and made free. The software is called TextMesh Pro and allows users to easily create great looking 3D text.

The next step was figuring out how I was going to be able to track the user's touches. With my lack of knowledge in Unity I wasn't sure if I was able to simply have an event that would track whether or not the TextMesh Pro object was interacted with. After some research, I decided to create a trigger box that was attached to every equation. This trigger box's event is triggered if the user's initial touch is within 10 pixels vertically and 20 pixels horizontally of the center of each trigger box. When testing it out I realized that this box size is quite small so I tried to make it bigger. After testing the bigger sized box, I realized there was a sweet spot between two equations where there trigger boxes would overlap. If you would swipe directly between them, you would activate both of their trigger boxes causing both equations to move at once. This meant that I had to shrink the box down again. Throughout my testing, it was mentioned many times by the participants that the trigger box for the swiping should be increased. At times they would find it difficult to swipe. This led me to the idea that I should have a card spawn every second row, that way the player would have more time to solve each equation and I could also increase the size of the trigger box, resulting in more responsive gameplay. I originally had the idea the of swiping equations all based off the cubes inside the conveyor and I would attach a sprite to each Cube that would have the image of the equation. I quickly realized this was a bad approach and continued with the TextMesh Pro option.

To actual do the physical testing required for my game I had to install Unity Remote on my phone and connect it to my computer. This was another obstacle that required hours of researching and playing around with the software to get my phone to basically have a live stream of my game that was being played in the Unity editor. That way I could use my phone as a controller for swipes that I couldn’t do using my mouse. Once I figured this out I was able to begin testing my swipe.
Next, I implemented a score that would increase when an equation, in the correct column, reached the answer boxes at the bottom of the screen. This was easy, since the hidden value variable was attached to every equation. I could just increment the score or have the player lose if the hidden value attached to the equation was not the same the number in the answer box at bottom of the screen. After this I basically finished the game. The only thing left to do was game states.

Creating the game states was very difficult. I began by setting up a canvas object. In the canvas object, I had four different canvas groups which were Main Menu, Playing, Pause, and Game Over. I started off with the main menu by creating a very basic play button. And I set up different Boolean variables to keep track of what game states were active. I started off by setting them all to false except for main menu and instead of using start functions for all of my classes no functions were called until the play button was clicked. Once the play button was clicked I set main menu to false and playing to true. There are many update functions throughout the classes that are constantly checking the activeness of states. In the Figure 3.11 you can see that Movement() and InBoundary() will never get called unless playing is true. This makes it so the rows won’t scroll down when the game isn’t playing, and it won’t continuously make unnecessary InBoundary() checks to see if a card has reached the bottom of the screen.

```csharp
void FixedUpdate () {
    if (myGameState.playing == true) {
        Movement ();
        InBoundary ();
    }
}
```

*Figure 3.11: Conveyor.cs FixedUpdate() function*

Clicking the play button also changes the alpha value on the main menu canvas group and the play canvas group. It sets the main menu canvas group alpha value to 0, which renders it invisible and it sets the playing canvas group alpha value to one, which reveals the pause button. After the play button is clicked I set everything up as if it was the same as clicking the Run button in Unity before I added these games states, so everything was working as intended.
When creating the different menus, one thing I was having issues with was being able to click a button. Since this is a 3D game that involves interacting with game objects using touch, I have to use something called RayCasts. At first I tried simply using x & y-coordinates of the game screen. After hours of debugging I finally realized what the problem was. My game is in a 3D space and the camera is looking at my scene on an angle. The solution to this problem were the RayCasts. RayCasts are basically lasers that shoot onto your game scene that can interact with game objects. This way if I ever touched the screen, an invisible laser would hit the any object in its path and an invisible point in 3D space would be plotted. I could then check if this point is within the bounds of that objects trigger. The canvas groups that were affecting gameplay were the Playing group, the Pause group, and the Game Over group. Each one had an issue at some point when there were equations on the screen. Since canvases interact with RayCasts, I had to tinker with them to get it so I can still block RayCasts when I wanted to hit the pause button but I didn’t want the canvas to block the RayCasts when I was swiping an equation.

For the implementation of the pause button I changed Time.timeScale to be set to zero which essentially stops time. All of my movement functions are based on time. The only movement that isn’t affected by time is the swiping action, so whenever the Time.timeScale is set to zero you are still able to swipe equations. This is clearly an issue and is present in the pause menu and the game over screen. Another issue with having a pause function is that the user could pause the game and quickly figure out where every equation has to go that is currently on the screen. Even if they wouldn’t be able to swipe it in the pause menu, they would still be at a great advantage. I decided to get rid of the pause menu all together because it gives my game better flow and I feel like this type of game would be better without it.

After my game was complete and I had no more bugs to deal with, I decided to revisit my original project proposal to see what my challenges were. There I found that I had mentioned, “I would also like to randomly change the 3 bottom labels every 30 seconds or so to offer another challenging game mechanic and I believe that this will be biggest challenge.” In my proposition letter I assumed that getting the column switch was going to be one of the most challenging parts, it took a lot of trial and error but, it took nowhere near the amount of time it did to get the base game play down. The randomization of everything was by far the greatest killer of time. To implement the column change, I created a counter variable that would call the column change function every 30 equations.
Once the 30th equation would spawn I would stop spawning new equations. This way when the switch happens, there is a buffer where the player gets time to gather themselves and compute what the new answers are. Once the player answers the 30th equation correctly, the equations change randomly to any number (2-9, inclusive). The equations will start coming down from the top of the screen with their hidden values being one of three new answers.

![Game Over Screen](image)

*Figure 3.12: Screenshot of my Game Over screen right before reaching the answer swap*

As you can see in Figure 3.12, the answer swapping buffer started and would have swapped answers if I reached a score of 30.
I had a really big visual issue after building and porting onto my phone. It was as if the camera was zoomed out for the canvases but still at the same position for all the game objects.

![Figure 3.13: Screenshot on mobile](image-url) ![Figure 3.14: Screenshot in Unity](image-url)

It also affected the visuals for the top 1/3 of the screen. Figure 3.14 is what the game is supposed to look like, and is how it looks in the Unity editor. As you can see when comparing Figure 3.13 and Figure 3.14, there is a big difference in the top 1/3 of the screen.

After many days of searching I finally found someone suggest to “un-check the pixel perfect box” which lead me to find that it was something wrong with the canvas itself and not the options I had setup in my Player Settings. After a lot of tinkering, I found that If I change the UI Scale Mode from “Constant Pixel Size” to “Scale With Screen Size” then my UI no longer gets proportionally weird on my phone. My score wasn’t part of the canvas, and it was also being affected by this bug and looked really small, as you can see in Figure 3.13 in the top left. To fix this I created a new TextMesh Pro object and placed it inside the canvas hierarchy under Playing. This way it will only show when the game is running and scale to the proper size when I port it over to mobile.
Figure 3.15: Screenshot of end result

Figure 3.15 shows what the end result of my game looks like. I was able to fix the UI shrinking bug and then I also fixed the score by adding it to the UI so it will scale properly. Unfortunately, I was not able to fix the visuals on the top 1/3 of the screen, but this is definitely something I have to fix in the future.
IV. Experimental Design

To clarify, my hypothesis is that the use of my game will improve the user's ability to solve math equations quickly and accurately. Rather than just being a self-randomizing math test, my game will effectively have all the benefits from that type of test, but execute the test in a more fun and interactive way. This approach may result in more overall use of the game as well as active players, which could often be the downfall of a boring math test. My game is the bridge that will connect the benefits of a boring test with the fast-paced and addictive manner of a mobile game. The experiment I devised to test this hypothesis required me to create two quizzes, one to give the users before using my app, and another that I would give them after. Each quiz is made up of 144 different simple math equations similar to the ones found in my game. Here are some questions from one of my quizzes:

\[
\begin{align*}
8 + 7 &= \underline{15} \\
9 - 2 &= \underline{7} \\
5 \times 4 &= \underline{20} \\
16 \div 4 &= \underline{4}
\end{align*}
\]

\[
\begin{align*}
2 + 4 &= \underline{6} \\
8 - 4 &= \underline{4} \\
6 \times 7 &= \underline{42} \\
15 \div 3 &= \underline{5}
\end{align*}
\]

\[
\begin{align*}
3 + 9 &= \underline{12} \\
5 - 2 &= \underline{3} \\
2 \times 9 &= \underline{18} \\
6 \div 3 &= \underline{2}
\end{align*}
\]

\[
\begin{align*}
6 + 7 &= \underline{13} \\
7 - 4 &= \underline{3} \\
3 \times 9 &= \underline{27} \\
18 \div 6 &= \underline{3}
\end{align*}
\]

\textit{Figure 4.1: Questions from Quiz 1}

The idea is that I will compare the amount of time taken to complete each quiz, and the amount of correct answers they get between the two quizzes. I also created 16 survey questions to give to the participants after they complete the second quiz. This way I can get a better idea of how the participants felt about my game.

I recruited my participants via email. My mentor, Robert Collier, gained permission from the Carleton Research Ethics Board to send a recruitment email to all students majoring in Computer Science. I had to write up a recruitment email, for the Ethics Board to send out to all the students. In my recruitment email, I first introduced myself then had a description of my software and the purpose of my project. It then specified that their participation was entirely voluntary and all data was going to be anonymized. If they were interested in participating, I asked them to please contact me at my Carleton email, where they would be provided with more information on the study. I received about 8 emails in total out of all students studying Computer Science at Carleton. I then scheduled meetup times across the next 2 weeks to get all testing finished in due time. Only 2 students failed to show up to their
assigned meeting, so I would say recruitment and testing overall went well considering the due dates of end-of-year projects and exams were nearing.

Upon meeting with a participant I would begin with giving them the consent form to which they would read through and sign. They would also gain a copy for their own records, in case they ever needed to contact me or had any questions about the study. After they signed, I would give them the first quiz and tell them to complete it as fast as they can and that I would be timing them. Once they’ve completed the first quiz, if the participant was part of my variable group, I would then give them my app for them to test. They would then play my game multiple times for 10 minutes straight. During this time, I would occasionally take down any suggestions they had during gameplay. If the participant was part of the control group, I would instead have them do something unrelated to math for 10 minutes between taking the second quiz. For example, watch a YouTube video, go on Facebook, et cetera. After those 10 minutes the participants would then be given a second quiz and again I told them to complete it as fast as they can and that I would be timing them. If the participant was part of the variable group, the idea is that the time to complete second quiz will be less than that of the first, and the amount of correct answers would be higher. Since I have done no prior research I was not sure what the results of the control group would be. I hypothesized that speed and accuracy of the control group would slightly improve when taking the second quiz, but not as much of an improvement as the variable group. Once participants in the variable group completed the second quiz they would be given an online survey that I made using Survey Monkey. The survey involved mostly Likert scale questions that you had to answer using either strongly agree, agree, neutral, disagree, or strongly disagree. My survey asked questions about gameplay like the ones in Figure 4.2 and Figure 4.3.

5. This game was fun
   ○ strongly agree
   ○ agree
   ○ neutral
   ○ disagree
   ○ strongly disagree

*Figure 4.2: Question 5 from my survey*
I also asked a couple of free form questions which provided a textbox for users to get more in depth with their answers and tell me what they really thought. A really useful one was:

10. Finally is there anything that could be added or changed about the current state of the game that would improve it overall?

This questions gave me a lot of insight into the user's perspective, and helped me see problems in my game that I never would have noticed.
V. Results Analysis

I had both my control and variable group complete two quizzes, and I recorded the time taken to complete and the score they got on each quiz. I also kept track of the variable group’s game scores and their answers to the survey. My control group consisted of 3 participants and my variable group originally consisted of 6 participants. I was only able to analyze the data of 5 of them because I had an issue with one of my participants who was unable to complete either of the quizzes and so their test results could not be considered.

Quantitative Results

As I mentioned earlier, I had each participant in the variable group play the game for 10 minutes between the first and second quiz. Below I have included graphs of each participant’s progress over time, with the “Number of Times Played” as the x-axis and the “Score per Run” as the y-axis. Instead of having the x-axis span across the time of 10 minutes, I decided it would be better to go by the number of runs, since that numbered differed between participants. Generally, if a particular person was getting many low scores that would result in more runs, and if they were getting more high scores they would have fewer runs. Below I have a graph for every one of the 5 participants that tested the game. I originally wanted the data on one graph with different coloured lines, but there was too much going on, and you couldn’t see the difference. Each graph shows the scores for the specific participant, along with a trendline showing their average score over time, a formula $y = ax + b$ that represents the average score, and $R^2$ which is the correlation coefficient.
Figure 5.1: Participant V01 Scores

Figure 5.2: Participant V02 Scores
Figure 5.3: Participant V03 Scores

Figure 5.4: Participant V04 Scores
It seems that each person has their own unique path, but the linear trendlines show us that the average slope is positive in all cases except for V02, this indicates good progress. It should be noted that the highest $R^2$ value is approximately 0.57, which means the chance that the linear trendline picks the correct value is only about 57%, which is not great. Although, a great example of improvement is demonstrated with participant V01’s results. They start off getting low scores, but as they play more and get used to the gameplay, their score seems to consistently increase respectively. It was strange seeing the data graphed out like this, because when observing the participants, I would have imagined more graphs looking like participant V01’s graph. I may have been looking at it in the respect of high scores because over all participants it seems that their high score would increase as time goes on. They would also get their highest score overall nearing the end of their playtime. With all of this progress, it’s hard to believe that there are so many low drops throughout many of the participants scores. I noticed this was due to a common mistake that was occurring between many participants. The reason this was happening a lot was usually due to that fact that they thought it was a different operator. For example, if they had and equation like $10 - 5 = ?$ and swiped it to the “2” column (if there was one), they were mistaking the “-” for a “÷”. The score was a really interesting way to view the progress of the participants, let’s see if it correlates with better quiz results.
I originally hypothesized that the participants in the control group were going to have a slight improvement in their Quiz 2 time.

Although I predicted a slight change in the control group’s quiz results, I thought there was going to be a greater difference in the two quiz times.

*Figure 5.6: Control group’s Quiz 1 and Quiz 2 times*
For my variable group, I originally hypothesized that playing 10 minutes of my game between the two quizzes would result in a quicker Quiz 2 time.

![Variable Group Quiz Times](image)

*Figure 5.7: Variable group's Quiz 1 and Quiz 2 times*

As you can see, there is a drastic improvement in the amount of time taken to complete the second quiz then it did the first. This is great to see, especially when comparing it to the control group. Every participant decreased their time anywhere from 15-23%.
Plotting out the Delta times for the 2 groups allows us to better see the actual time difference between the 2 quizzes.

**Figure 5.8: Control group's delta time**

**Figure 5.9: Variable group's delta time**

As you can see, the average delta time for control group is nowhere near as impressive as the variable group. The average delta time for the control group is -1.3 seconds between Quiz 1 and Quiz 2, and the average delta time for variable group is -54 seconds between Quiz 1 and Quiz 2. As you can see my hypothesis for the variable group was correct and highly exceeded my expectations. Every single participant in the variable group improved and on average shaved almost a minute off their time.
Although my hypothesis for the control group was that they would increase their time slightly, I thought it would have been a more significant change.

Figure 5.10: Quiz time confidence interval

Using MathIsFun.com’s Confidence Interval Calculator, I found that the upper bound of the control group’s time is approximately 7.5 seconds and the lower bound is approximately -10 seconds, when using a confidence interval of 95%. I found that the upper bound of the variable groups time is -34 seconds and the lower bound is -74 seconds, when using a confidence interval of 95%.

Therefore, I can confidently say that the use of my game can definitely decrease the amount of time taken to solve math equations. My variable group did much better than my control group with respect to quickness and the results are statistically significant.

I originally hypothesized that the participants in the control group were going to have a slight improvement on their accuracy, and therefore get a slightly better score on Quiz 2. For my variable group, I originally hypothesized that playing 10 minutes of my game between the two quizzes would improve their accuracy, and therefore get a better score on Quiz 2.
Figure 5.11: Control group’s Quiz 1 and Quiz 2 Scores

Figure 5.12: Variable group’s Quiz 1 and Quiz 2 Scores
The data in Figure 5.11 and Figure 5.12 is a little bit harder to visually determine which group did better than the other. The accuracy of the variable group was much lower on Quiz 2 than I thought it would be. I imagined there would be clear improvement across the board.

![Quiz Score Confidence Interval](image)

*Figure 5.13*

Again, after using MathIsFun.com’s Confidence Interval Calculator, and using a confidence interval of 95%, I found that the upper bound of the control group’s accuracy is approximately 3.3 and the lower bound is approximately -1.3. Using a confidence interval of 95%, I found that the upper bound of the variable groups accuracy is 3.7 and the lower bound is -2.1. Surprisingly enough, the sample mean of the control group was slightly higher than the variable group’s. Since the two confidence intervals almost entirely overlap one another, this small advantage in the mean means nothing at all. Unfortunately, my variable did not do better with respect to accuracy but the result was not statistically significant.

It is surprising to me that the two different groups were so close. Although my game mostly focuses on speed, I thought the act of having to solve so many equations, before taking a quiz, would have had a greater impact on the variable group. I will be able to learn more in the future, and get more accurate results if I test with larger groups.
QUALITATIVE RESULTS

Looking over the survey data, a common issue that all participants were able to agree on was that they thought the scroll speed was too fast. Many also thought the gameplay wasn’t responsive enough, and suggested to increase the trigger box size, “too fast, hit box not responsive enough”. Another common suggestion people had for the future was better visuals. One in particular had an extra suggestion, “Sound and colours, keep high score, data improvement overtime”. This feature would allow the user to monitor their overall progress with the game by showing their scores overtime. Data improvement is a fantastic suggestion and I am truly considering it for when I work on this in the future.

One of my favourite answers that was given was to the question “What could I add to this game to make it more engaging/fun?” and they responded with “some kind of multiplayer where you could play with your friend”. I never thought about a multiplayer feature with this game at all, but it brings me back to where this idea originally came from which was Guitar Hero. Guitar Hero had a multiplayer competitive mode where you would battle to see who can get the most notes. I think there is definitely some potential for a multiplayer feature if I change up the rules, so that you won’t lose when you miss an answer, and players will battle to get the most correct answers in a given set of questions. Making this game fun will be very important because that can also lead to users playing more often, and hopefully further affecting the user’s improvement over time. Overall, this survey was a huge help in guiding me in what direction I should take this game.

VI. FUTURE WORK

I am very thankful to my participants for all of their assistance and ideas. They taught me a lot and gave me a new perspective on something I’ve been staring at for months. I am definitely planning to slow the speed down and have it increase with respect to score, as well as try to make the gameplay more responsive, by only spawning an equation every second row. This would allow me to double the size of the equations trigger box, which would hopefully end the responsiveness issues. I will also definitely add better visuals in the future, to give the game a little more life. This might improve the game’s level of fun and maybe entice more players to download and continue playing the game. The reason the visuals are so bare minimum right now is because I wanted to focus on the gameplay and
testing of my game. I would like to test whether or not this game as any long term benefits. I could do this by implementing the data improvement feature that was suggested to me by a participant in conjunction with more hand written quizzes, that the participant would have to complete every week or so after playing my game regularly. The data improvement overtime will help because I could link the scores with Unity Analytics and be able to compare the participant’s progress with how often they play and what scores they are getting, with their quiz results. I think the data improvement over time feature would be really interesting to see how you progress through the life of the app. I do believe it would be good for me to test my game again in the future, but this time with more participants. It was tough to draw to a conclusion whether or not my game helps accuracy from the data I gathered which is a real disappointment. On the Brightside I know it can greatly enhance the speed it takes a person to solve an equation.

VII. Conclusions

The objective of my project was to create an app that would be able to improve the user’s ability to solve math equations quickly and accurately. After gathering 9 participants I was ready to start testing. I originally had 6 participants in my variable and 3 in my control group. That changed to only 5 participants in my variable group since one of my participant’s data was unusable. I tested all of my participants by having them complete a math quiz as fast as they could. I recorded the amount of time it took to complete the quiz. The variable group would then play my game for 10 minutes, and the control group would do something unrelated to math for 10 minutes. Once the 10 minutes was up, both groups would proceed to do a second quiz very similar to the first. Again, I would record the amount of time it takes them to complete the second quiz. I would then have compared the times and the amount of correct answers between the two quizzes. I found that those who were to play my game between taking Quiz 1 and Quiz 2, would complete Quiz 2 significantly faster than Quiz 1, on average almost a minute faster. Those who did not play my game between quizzes only quickened their pace by about 1 second. The change in accuracy between the two groups was not much different after taking both quizzes. Therefore, my variable group is not better than my control group with respect to accuracy with any statistical significance. Meaning, I was incorrect when I thought that the use of my game would improve the user’s ability to solve math equations more accurately. Although, I was correct that the use of my game would improve the user’s ability to solve math equations quickly.
Making games is what I want to do once I graduate, and I plan on making my own game development studio. The first game I plan on making releasing under my new company is indeed the one that I've been working on for this on his project. It's an idea that I've had for so long now and to see it come to fruition is mind-blowing. It's especially great to hear some participants respond in a such positive way after playing my game. Some of them would honestly say they had a great time playing it and wanted to keep playing at once they were done. This was one of the biggest eye-openers for me, and made me realize that I have the power to create things that can make people's lives better, and I can't wait to do so. Although there was some negative feedback from my participants, it is just inspiring me more to tweak and perfect my project. I always knew I could do it, except the motivation from my participants really inspired me to turn this project into a finished product. Another thing this project has taught me is that your original vision for a game might not always be the same as the final product. Throughout my development of this game I realized the potential in developing this game with kids in mind. If I don't change the concept completely I definitely want to make a kid friendly version, because I believe it would truly benefit kids of all ages improve their mathematical skills. Another plus side to marketing this game towards children is that there is a huge market for kid's games on app stores right now. With technology evolving the way it is, it is becoming more and more common for a tablet to be in every child's household. After researching many different apps similar to mine, I learnt that there is a huge market for educational apps targeted towards children. Almost every app had above a million downloads, it ranges anywhere from 100,000 - 50,000,000. Overall, this honours project was eye-opening and so beneficial to my future. I look forward to applying what I have learned from this task to the improvement of my future work on this game and many others.
Stack screenshots:  https://itunes.apple.com/app/stack/id1080487957

Guitar Hero:
https://www.google.ca/imgres?imgurl=https%3A%2F%2Fyi.timg.com%2Fvi%2FWSM5hrLZvEY%2Fmaxresdefault.jpg&imgrefurl=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DWSM5hrLZvEY&docid=KUu6a0GSxka19M&tnid=FPFJRTv0toHRwM%3A&vet=10ahUKEwi0wfK-o-rXAhVq7YMKHe1wB0cQMwj5ASgKMAo..i&w=1280&h=720&client=safari&bih=658&biw=916&q=Guitar%20Hero&ved=0ahUKEwi0wfK-o-rXAhVq7YMKHe1wB0cQMwj5ASgKMAo&iact=mrc&uact=8

The Typing of The Dead:

http://www.wurb.com/stack/archives/345


http://store.steampowered.com/app/246580/The_Typing_of_The_Dead_Overkill/

Made My Surveys with Survey Monkey: https://www.surveymonkey.com

Downloaded Math Games (Fast) on Google Play:

To make my app I used Unity Game Engine: https://unity3d.com

TextMesh Pro: https://assetstore.unity.com/packages/essentials/beta-projects/textmesh-pro-84126