5-30-2017

Improving Elementary Math Learning Through iPad Games

Kaya M. Thomas
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/senior_theses

Part of the Computer Sciences Commons

Recommended Citation
https://digitalcommons.dartmouth.edu/senior_theses/122

This Thesis (Undergraduate) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Undergraduate Theses by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.
IMPROVING ELEMENTARY MATH LEARNING THROUGH IPAD GAMES

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Bachelor of Arts
in
Computer Science
by
Kaya Thomas
Dartmouth College Computer Science Technical Report TR2017-825
Hanover, New Hampshire
May 30, 2017

Thesis Advisors:
Professor David Kraemer
Professor Lorie Loeb
Mathematics has proven to be challenging to many from a very young age. Young students are influenced by their teachers on how to feel about math and how well they can perform. Currently many methods of teaching mathematics do not encourage learning, but instead promote memorization which has been shown to increase students’ anxiety about math. Math anxiety affects student performance as well as their ability to understand the material. Fractions are one of the most difficult concepts for young students to learn. Various techniques have been created in order to better instruct students on how to understand fractions. More recently digital learning techniques have become more popular and have been shown to increase student engagement and improve performance. In this paper, we present three iPad mini-games, each of which use a different educational method for fraction instruction. The target user of the games are elementary school age students, so all of the games use illustrations of animals and food to represent the fraction learning material. The goal of the games is to curb math anxiety and improve the student’s understanding of the material.
Preface

I would like to thank both my advisors, Professor David Kraemer and Professor Lorie Loeb for helping me complete this project and guiding me throughout the entire process.
Contents

Abstract ................................................................. ii
Preface ................................................................. iii

1 Introduction .................................................. 1
  1.1 The Problem with Current Math Teaching Techniques .... 1
     1.1.1 Traditional Textbook Learning and Testing .......... 2
     1.1.2 Math Anxiety ........................................... 2
     1.1.3 Digital Math Games .................................... 3
  1.2 Effective Learning Techniques for Fractions ............. 4
     1.2.1 The Difficulty Elementary Students Face when Learning Fractions 4
     1.2.2 Using Number Lines, Shapes for Whole/Part Comparisons, and Non-symbolic Approximations to Teach Fractions 5
     1.2.3 Virtual Manipulatives ................................. 5
  1.3 Educational Content of the iPad Mini-Games ............. 6
     1.3.1 Number Line Game (Monkey Game) .................. 6
     1.3.2 Whole/Part Comparisons with Shapes (Shark Game) .... 7
     1.3.3 Whole/Part Comparisons with Whole Objects (Monster Game) 7

2 Designing the User Interface .............................. 8
Chapter 1

Introduction

This chapter provides an introduction to various educational techniques that are used to teach mathematics and how they affect a student’s understanding of and feelings towards math. The last section of the chapter describes how I used the education research presented in the previous sections to create the three mini-games on the iPad.

1.1 The Problem with Current Math Teaching Techniques

Teaching mathematics is incredibly complex. Teachers are not only influenced by how they have learned math themselves, but also how math makes them feel [5]. If a teacher has had negative experiences with math, they can unintentionally pass those negative feelings to their students [5]. The traditional methods in which math is taught contribute to the negative thoughts people often associate with mathematics
1.1 The Problem with Current Math Teaching Techniques

[5].

1.1.1 Traditional Textbook Learning and Testing

Children begin learning math before they enter school. Although they do not know the formal terminology, they begin to construct basic ideas about math through their relationship to the surrounding environment and people they interact with [5]. Once they enter school, the constructive way they have begun using to understand math is interrupting by traditional teaching techniques.

These traditional techniques include using textbooks and worksheets to focus on learning through repetition. Teachers who use these techniques are often focusing on getting their students to memorize the material. These teachers will then use timed test to measure how well their students have memorized the material. Timing activities has been shown to increase anxiety and decrease student performance which leads to a negative attitude towards the material [5].

1.1.2 Math Anxiety

Students who suffer from math anxiety has been shown to exhibit poorer understanding of the material and lower grades in the subject [11]. The age group we are focusing on in this report are elementary school students. The first study on math anxiety in elementary school students was done in 2013 at the University of Chicago. This study found that math anxiety affects a student’s working memory, which is crucial for solving math problems. Working memory is a limited cognitive system responsible for holding and processing information. When students with high levels of math anxiety are worrying about doing the math, their worries take up valuable working
1.1 The Problem with Current Math Teaching Techniques

memory resources which prevents them from using their working memory to actually complete the math problems effectively [11]. Therefore young students who heavily rely on their working memory to do math and have math anxiety are more severely affected [11]. Math anxiety in elementary students is especially detrimental because it could lead to a snowball effect in which students continually avoid math and their overall ability to solve difficult math problems decreases [11].

1.1.3 Digital Math Games

Researchers have conducted several studies to compare the effectiveness learning via paper and pen versus digital games. A study on combating math anxiety divided a classroom into two groups, one group received worksheets to complete and the others were able to use math software on the computer. The group that used the software showed a significant increase in the number of problems answered correctly whereas the group who used paper worksheets did not [5]. Other research has shown that digital games can enhance visual short-term memory, multitasking and spatial cognition [3]. Unlike traditional techniques, which often focus on explanation, digital games focus on action. Games also make room for students who learn differently whereas when teacher use textbooks and worksheets, they are forcing every student to try and learn in one way [9]. Instead of reinforcing the material through pressurized tests, games reinforce knowledge interactively while creating a personal motivation for the student [9]. Teaching math with games could potentially lower math anxiety because of how games can enhance cognitive ability.
1.2 Effective Learning Techniques for Fractions

Fractions have been proved to be one of the most difficult math skills for children to master [10]. Even though fractions are taught beginning in elementary school, the National Mathematics Advisory Panel has shown that by middle school at least 40 percent of students showed difficulty with fractions. Many children may understand fractions intuitively, but do not connect that intuition to the formal learning they receive [10]. Research suggests that using manipulatives is one of the most effective ways of teaching fractions [2]. Manipulatives are physical objects used to teach concepts, for example, circles, counters and cuisenaire rods.

1.2.1 The Difficulty Elementary Students Face when Learning Fractions

Many elementary students think the rules of whole numbers apply to all numbers, so fractions confuse their understanding of numbers [4]. Fractions also have more than one meaning. They can be ratios, quotient division, operators, part-whole or measure. The multifaceted nature of fractions is a key reason children have a hard time conceptualizing them. Unfortunately, many teachers focus on teaching fractions through repetition. Repetition often leads to memorization instead of conceptual understanding. Without the conceptual understanding the student is more likely to make computation errors [4].
1.2 Effective Learning Techniques for Fractions

1.2.2 Using Number Lines, Shapes for Whole/Part Comparisons, and Non-symbolic Approximations to Teach Fractions

Various representative models can be used to represent fractions. The models I chose to focus on are number lines and shapes for whole/part comparisons. Number lines can be used to teach fractions in order to show students the relation between whole numbers and fractions, the different ways in which a number can be named, and how there are infinitely many of rational numbers between two whole numbers [6]. Using shapes for whole/part comparisons is a common teaching practice with many advantages, but also disadvantages. Students are often familiar when you use an object, for example: a pizza, to instruct them on how fractions work in the real world. Unfortunately if using familiar shapes can cause students to just count the parts of the whole instead of conceptually understanding how the part relates to the whole [6]. Non-symbolic approximation uses anything other than numbers or mathematical symbols to represent a math problem. Research shows that non-symbolic approximation used for addition and subtraction has improved the math skills of children as young as three years old [7]. I incorporated all of these instructional methods in the mini-games to ensure the students can be exposed to various styles of how fractions can be represented.

1.2.3 Virtual Manipulatives

Manipulatives are common practice in mathematical instruction. Often times these manipulatives are physical objects or on paper. Now, with the introduction of new
1.3 Educational Content of the iPad Mini-Games

 technologies in the classroom the use of virtual manipulatives has been introduced. One of the advantages of using virtual manipulatives is the ability to connect dynamic visual images to abstract symbols [8]. Studies have shown that combining the use of virtual manipulatives and physical manipulatives resulted in a positive gains in regards to student achievement and conceptual understanding. Students have also shown a positive reaction to the use of virtual manipulatives in the classroom. On one study, students stated that [virtual manipulatives] made fractions easier for them because it was faster than pen/paper and helped them understand better [8].

1.3 Educational Content of the iPad Mini-Games

The mini-games I have developed have incorporated all of the previous research in hopes to increase student understanding of fractions as well as curb math anxiety. There are three different mini-games, each which focus on a different method of representing fractions.

1.3.1 Number Line Game (Monkey Game)

The first mini-game using a number line to represent fractions between 0 and 1. The number line gets more complex as the levels increase. For example, the first level the fractions on the number line are 1/4, 2/4, 3/4. The last level uses fractions that represent whole numbers: 0/7, 8/8, 10/5, 9/3, 16/4, 30/6, 6/1, 14/2, 24/3, 18/2, and 50/5. The objective of the game is to collect bananas on different spots on the number line. The instructions tell the student where on the number to collect the bananas, for example, Collect 10 bananas on a point greater than 1/2.
1.3 Educational Content of the iPad Mini-Games

1.3.2 Whole/Part Comparisons with Shapes (Shark Game)

The second game uses shapes to represent fractions, specifically food. The objective of the game is to select the portion(s) of food that is the biggest portion less than a particular fraction. For example, the student would see four different portions of pizza that represent the following fractions: \(\frac{4}{8}, \frac{2}{8}, \frac{1}{8}, \text{ and } \frac{3}{8}\). They also have access to visual to whole pizza to be able to relate the portions to the whole. The instructions could say Jo cannot eat more than \(\frac{1}{3}\) of the pizza, feed him the most amount he can eat. The correct answer is \(\frac{2}{8}\) of the pizza. The student would then have to compare \(\frac{1}{3}\) to the of the portions of food they see which means they have to know which fractions the portions represent in relation to the whole food.

1.3.3 Whole/Part Comparisons with Whole Objects (Monster Game)

The third game uses non-symbolic approximation to represent fraction based problems. The student is shown a full bag of candy for a few seconds and then they are shown a candy bag that is missing some candy. The student then has to select from three candy piles which pile would complete the bag that is missing candy. For example, the bag they are shown at first has 27 pieces of candy in it. After they are shown the full bag, they are shown a bag that has 11 candies as well as three different piles with the following amount of candy as an example: 16 candies, 8 candies and 6 candies. They have to use approximation to understand that the bag of 11 candies is missing more than half so they need to select the larger pile (16 candies) and not the other smaller piles.
Chapter 2

Designing the User Interface

The focus during designing the user interface of the games was to ensure that the students can be engaged. Each of the games has a story associated with the main character to make the user feel that the objective goes beyond solving math problems.

2.1 Game Scenes and Characters for User Engagement

Each game tutorial reads like a picture book in hopes to make it accessible as well as makes it clear what the user’s role is in the story which gives them incentive to play. The first game’s main character is Suzie, a monkey. Suzie moved to a new part of a tropical forest and needs help collecting bananas on the correct part of the path (Figure 1,2). If she collects bananas on the wrong part of the path, she would be stealing from other monkeys.
2.1 Game Scenes and Characters for User Engagement

The second game’s main character is Jo, a shark. Jo is shown in his underwater kitchen (Figure 3). His story is that he loves to cook, but he is not good at portioning his food. He often eats too much so the user has to help him portion correctly (Figure 4, 5).

The third game’s main character is Toni, a fictional monster who lives in a candyland (Figure 6). Toni only eats candy and goes throughout the land collecting candy each day. On Toni’s walk home from collecting candy she drops her bag of candy and some spills out (Figure 7). The user has to help her put back the right amount of candy in her bag (Figure 8).
2.2 Translating Educational Content into an Understandable User Interface

Each of the games represents a different method of representing fractions. I designed all the virtual manipulatives so that they would be familiar to the users. The manipulative used throughout all the games was food. For the first game the food is not what is in fraction form but it is used to instruct the user where to go on the number line. The number line is represented on a path in the forest. The path is an everyday object that the user can relate to, similar to a sidewalk or street crossing. By putting the number line on the path, the user can relate the symbolic math concept to the real-world. The second game using round foods to represent the part/whole method of teaching fractions. The user has to select the foods that are greater than or less than a specific number but they have to relate it to the whole food. The third game’s UI is partly inspired by a study on non-symbolic approximate arithmetic [1]. In the study they used dots to represent what the user was suppose to add. They covered up the dots after exposing the user to them for a short period of time and then asked to them to approximate the answer. Instead of dots, I designed the third game to use candy as the non-symbolic representation.
2.3 User Interactions

All of the games have three consistent interactions. The back button which allows the user to go back to the home screen where they can choose which game they want to play (Figure 9). The start button which allows the user to start the game after they have finished reading the instructions. Lastly, the tutorial button which allows the user to read the game tutorial. All three of these buttons are seen on Figure 10.
2.3 User Interactions

The first game has two unique interactions, the first is the ability to control the main character’s movement and the second is the ability to collect bananas through moving the main character (Figure 11). The user drags the monkey across the screen with their finger and as they move it, the monkey’s arms swing from left to right. Once the monkey collides with the banana it disappears.

The second game’s main interaction is with the food portions. According to whichever instructions the user receives they have to select one of the foods. The plate will vibrate to indicate to the user that it has been selected and then they will receive a message on whether or not their selection is correct (Figure 12, 13).
2.3 User Interactions

The last game has two different scenes. The first scene shown in Figure 14, is the initial full candy bag. The user sees the bag for a few seconds and then is brought to the second scene that shows them a candy bag (that is missing candy) and three different piles of candy spread across the mountain floor (Figure 15). The user can then select the which pile they believe will equate to the original bag of candy if that pile were put back into the bag that is missing candy.
Chapter 3

Development

In this chapter we demonstrate the development process of the games which were written in Swift.

3.1 Swift, SpriteKit and UIKit

The app which contains all three mini-games was written in Swift. Swift was released in 2014 by Apple. The app uses three main frameworks: SpriteKit, UIKit, and Firebase. Spritekit allows users to create games by providing an entire graphics rendering and animation infrastructure. One of the key functionalities I used from Spritekit is the physics simulation which allows you to create a physics world which you characters can act upon. UIKit allows developers to construct and manage an app’s user interface through user interactions and system events. Combining UIKit and SpriteKit allowed me to create a more powerful app that harnesses the functionality of both frameworks in order to create a better user experience. Any class mentioned that begins with SK comes from SpriteKit or that begins with UI comes
3.1 Swift, SpriteKit and UIKit

from UIKit.

3.1.1 Connecting SKScenes and UIViewController

A SKScene is a game scene that contains all the nodes and objects displayed in that current view. A UIViewController allows you manage the view of your current app. Originally when I create the games there were only SKScenes and I was not using UIKit. This made it extremely difficult to implement features that were not involved in gameplay like a home screen where you can select which game you want to play and buttons to start the game. When I implemented UIKit in the app, I created three main viewcontrollers as shown in Figure 16. The startViewController contains view which includes three UIButtons each of which are images of the main characters (Figure 9). There is a UILabel on the view that instructs the user to select one of the characters. The UITextField on the view is off to the side because it is for me to input a UserID that will be used to store their game resulted, how the data is stored will be covered in section 3.3. Once the user selects one of the buttons a segue is triggered and a sceneViewController is created, information is sent between the segue so the sceneViewController knows which game was selected. The sceneViewController contains three buttons which are presented as text to the user. The start button triggers the start of the game as shown in Figure 17. According to the information sent during the segue the correct SKScene is created and the correct start function of that SKScene is called when start is selected. Each of the SKScenes contains a reference to a sceneViewController in order to send information back to the controller to store the data, to hide or show the start button depending on if the user has selected it or not, and to tell the controller whether the game
3.1 Swift, SpriteKit and UIKit

Figure 16

has started or not. The back button sends the user back to the home screen, the startViewController. The tutorial button triggers a segue to a pageViewController and sends which game is currently selected to the controller. The pageViewController is a UIPageViewController data source and delegate which means it controls the data presented in the view (the tutorial pages) and also handles the user interaction with that data (swiping between pages). The pageViewController allows for the storybook experience when the user is reading the tutorial for that current game.

Figure 17
3.1 Swift, SpriteKit and UIKit

3.1.2 Using SpriteKit Built-in Physics Functionality

SpriteKit’s physics functionality is a powerful tool that allows developers to create a simulated physical world for their characters and objects within a game scene. The first mini-game depends heavily on the physics functionality for the user’s gameplay. In the game the two physical objects in the simulated physics world are the monkey and the bananas. The physics functionality then allows me to know when these two nodes have collided. Each node has a ColliderType which is a UInt32 that is the unique identifier for that node. The SKScene is a SKPhysicsContactDelegate which allows it to handle contact between any physical objects. The physical world is initiated through a CGVector which is a two dimensional vector from the Core Graphics library. The following functions handle a collision between two objects. The didBegin function detects that two objects made contact and checks for their unique identifiers to ensure it is a collision between the monkey and a banana. If it is the correct collision, the handleCollision function is called. This function checks the position of the banana node that the monkey collided with and checks if that position is greater than or less than the threshold value. The threshold value is the position on the view that denotes where the user is suppose to collect bananas. The function then removes the banana from the scene. If the user moved the monkey to collect a banana that is on the wrong part of the view, the game will end. The game will also end if the user collected 10 bananas on the correct part of the view.

```swift
// checks if nodes collided with each other
func didBegin(_ contact: SKPhysicsContact) {
    let firstNode = contact.bodyA.node
    let secondNode = contact.bodyB.node
```
if (firstNode?.physicsBody!.categoryBitMask > secondNode?.physicsBody?.categoryBitMask){
    handleCollision(firstNode, otherNode: secondNode)
} else {
    NSLog("Unknown collision detected.")
}

func handleCollision(_ node: SKNode!, otherNode: SKNode!) {
    var removeThese: [SKNode] = []

    if (((over && (Int(node.position.x) < thresholdVal)) || (!over && (Int(node.position.x) > thresholdVal))) || seconds == 0){
        let info = "Wrong value selected on level \(level). Threshold pos: \(thresholdVal)\
Selected pos: \(node.position)
Fraction: \(limit) Time elapsed: \(120-seconds)"
        viewController.writeToFile(info)
        gameEnds(false)
    }
3.1 Swift, SpriteKit and UIKit

```swift
if bananasCollected > 8 {
    let info = "Won level \(level). Threshold pos: \(thresholdVal)
    Selected pos: \(node.position)
    Fraction: \(limit)
    Time elapsed: \(120-seconds)"
    viewController.writeToFile(info)
    gameEnds(true)
}

// removes a node upon collision
removeThese.append(node)
removeChildren(in: removeThese)
updateScore()
}
```

### 3.1.3 Gesture Recognizers versus Touch Events

In the first mini-game the user drags the monkey across the scene. In order for the user to do this I implemented gesture recognizers. The UIPanGestureRecognizer specifically recognizes when the user drags their finger(s) across the screen. Then I look for the CGPoints that represent where the user has dragged with the panFor-Translation function shown below. I check for if the position of the monkey is within the bounds of the current view. Then I call the move function which updates the x and y coordinates of the monkey node which is a custom class that extends SKNode.
3.2 Game Mechanics

// moves the main character the user controls
func panForTranslation(_ translation: CGPoint) {
    let position = mainCharacter.position
    if (position.x < 25
        || position.x > 999
        || position.y > 760
        || position.y < 25 ){
        monkey.move(position.x + 10, y: position.y + 10)
    } else {
        monkey.move(position.x + translation.x, y: position.y + translation.y)
    }
}

For the second and third game there is no dragging of any nodes so I did not use a gesture recognizer. Instead I usedUITouch that can detect when the user touched the screen. Both the second and third mini-games focus on the user selecting a node but not moving it across the screen.

3.2 Game Mechanics

Although we have covered some of the game mechanics in the previous section, now we will go over how the game functions through the use of data structures and various algorithms. There were also techniques used consistently through all the apps like randomization and the use of timers.
3.2 Game Mechanics

3.2.1 Data Structures

The use of data structures was essential in getting each of the games to function. For all of the games the use of dictionaries is essential. For the first game, the threshold value, as mentioned in section 3.1.2, represents an x position on the view. According the instructions the user has to either select a banana at an x position less than or greater than the threshold value. At each level the threshold value is presented to the user as a string in a label, i.e. Inside of the scene there is a dictionary that holds the key, pair values for the thresholds, shown below.

```
let thresholds = [["1/2":530, "1/3":364, "1/4":255, "2/3":627, "3/4":783],
                 ["1/2":133, "3/4":220, "2":510, "3":752],
                 ["1/2":160, "3/4":220, "2":510, "3":752],
                 ["1/2":80, "1":130, "5":505, "8":785]]
```

The second game contains a dictionary with a key, value pair of Float and String. When we randomly select from an array (covered further in section 3.2.4) a fraction value to include in the instructions. Then that value is converted into a string through finding it’s value from the dictionary and using it in a label on the view. In the third game a dictionary is used to match an Int to a String, the Int represents the amount of candy and the String represents the name of an image that shows a candy bag or pile with that amount of candy.

3.2.2 Combination Algorithm to Find Winning Sums

For the second game, once the user gets past level 1 they have to select multiple foods that together are less than a specific fraction, but the highest value that is less
3.2 Game Mechanics

than. In order to figure out if they selected the right combination of food (each food represented a fraction value) I have to know what all the possible combinations are. I implemented a recursive function that goes through an array of floats and appends all the possible combinations of the floats from the given array into a new array. The code for this algorithm is shown below.

```swift
func getCombos(_ given: [Float], new: [Float], n: Int, k: Int) {
    var new = new
    if (k == 0) {
        combos.append(new)
        new.removeAll()
        return
    }
    var i = n - k
    while (i < n) {
        new.append(given[i])
        getCombos(given, new: new, n: n, k: k - 1)
        i += 1
    }
}
```

Then according to whether the user was instructed to select the foods with the greatest amount or least amount, I find the correct sum through the functions shown below.
3.2 Game Mechanics

```swift
func getWinningSumList(_ low: Bool) -> [Float] {
    let list = [Float]()
    getCombos(selectedNodes, new: list, n: selectedNodes.count, k: level)
    return combos[getAllSums(combos, low: low)]
}

func getAllSums(_ listOfLists: [[Float]], low: Bool) -> Int {
    var sums = [Float]()
    var i = 0

    while (i < listOfLists.count - 1) {
        sums.append(sum(listOfLists[i]))
        i += 1
    }

    if (low) {
        let lowest = sums.min()
        let index = sums.index(of: lowest!)
        return index!
    } else {
        let highest = sums.max()
        let index = sums.index(of: highest!)
        return index!
    }
```
3.2 Game Mechanics

3.2.3 Offset Algorithm

In third game, in order to increase difficulty throughout the game, there is a specific offset between the amount of candy piles shown to the user. This means that the difference between the correct pile and incorrect piles decreases as the levels increase. For example, on the first level, if the user expected to pick the pile with 18 candies the incorrect candies could be piles of 6 or 8. The following functions are the implementation of this offset algorithm. The node number represents whether the node is the first incorrect pile or second incorrect pile, this is to ensure they have different values. The offset for level one is 10, for level two it is 6 or 10, for level three it is 3 or 6 and for level four it is 3. By level four it is difficult to tell the difference between all the piles. The selectPileByLevel function keeps track of which values have already been chosen and subtracts amount of the offset value from value of winning pile.

```swift
func getOffsetByLevel(_ chosenPile: Int, nodeNumber: Int, chosenValues: [Int]) -> Int {
    var offset = Int()

    if (level == 1) {
        offset = 10
    }

    } else if (level == 2) {
        offset = nodeNumber == 1 ? 6 : 10
    } else if (level == 3) {
```
3.2 Game Mechanics

```swift
offset = nodeNumber == 1 ? 3 : 6

} else if (level == 4) {
    offset = 3
}
return selectPileByLevel(offset, chosenPile: chosenPile,
chosenValues: chosenValues)
}

func selectPileByLevel(_ offset: Int, chosenPile: Int,
chosenValues: [Int]) -> Int {
    for num in numberOpts {
        if (num >= chosenPile-offset && !chosenValues.contains(num)){
            return num
        }
    }
return 0
}

3.2.4 Using Randomization and Timers to Trigger User Interactions

Randomization is often an essential component of gameplay. In order to keep the user engaged I made sure that each time they played the game, they would be challenging with a different problem. In order to do this I implemented a random number function
3.2 Game Mechanics

does a given range to return a random integer between that range. Whenever I choose a value to that is essential to the problem the user is supposed to solve, it is randomized.

```swift
func randomNumber(_ range: Range<Int>) -> Int {
    let min = range.lowerBound
    let max = range.upperBound
    return Int(arc4random_uniform(UInt32(max - min))) + min
}
```

All of the potential values are stored in arrays, the range I pass into the random number function is from 0 to the length of the array. I then use the return value to index into the array and use that value. For example, in the monkey game on level 3 the points on the number line are: 0/2, 1/2, 2/2, 3/2, 4/2, 5/2, 6/2, 7/2, 8/2. There is an array with all of those values, and using the randomNumber function I will pick one of those values as the threshold for the number line. Timers are used throughout the game to trigger actions. For every game, a scheduled timer is used to show the user how much time has passed in the game. The user has 120 seconds to play until the game will end. The game timer starts when start button is pressed. The label for the timer is updated through a selector that is set when the timer is created. For the shark game and candy monster game various timers are used throughout the game play. When the user selects a food item in the shark game it will move for a few seconds and then the new selection options are presented to the user. The timer is scheduled when the user selects the food and the timer's selector is a reset function that will set up the next game. In the candy monster game a timer is used in the initial view when the user sees the full bag of candy, after a few seconds that bag
3.3 Tracking User Interactions

disappears and the game scene is set. A timer is also used when the user selects a bag of candy, after a few seconds the game scene is reset and they will see full bag of candy again.

3.3 Tracking User Interactions

In order to know if the games are effective, I have to be able to track the users interactions while they play. I used Firebase’s framework to store this data. Firebase hosts a database on their platform which can be easily written by using their framework. After configuring your project with their online platform, you link their framework to the hosted database through adding configuration in the AppDelegate. In the StartViewController (Figure 9) there is a textField with the placeholder USERID. Before the user plays the game I will enter in an userid for them that will allow me to be able to distinguish their gameplay from another user’s. Since each SKScene has a reference to the SceneViewController when they want to send information to the Firebase database they can call writeToDatabase function shown below.

```swift
func writeToDatabase(_ info: String){
    let date = NSDate()
    let rootRef = FIRDatabase.database().reference()
    let dataObj = DataObj(date: String(describing: date),
                          info: info, game: gameSelected)
    let dataRef = rootRef.child(userid)
    dataRef.setValue(dataObj.toAnyObject())
}
```
3.3 Tracking User Interactions

The DataObj is a struct I created shown below that is able to store the information I need to write to the database and convert it into a dictionary that can be processed in JSON by the Firebase database.

```swift
struct DataObj {
    let key: String
    let date: String
    let game: String
    let ref: FIRDatabaseReference?
    var info: String

    init(date: String, info: String, game: String, key: String = "") {
        self.key = key
        self.date = date
        self.game = game
        self.info = info
        self.ref = nil
    }

    init(snapshot: FIRDataSnapshot) {
        key = snapshot.key
        let snapshotValue = snapshot.value as! [String: AnyObject]
        date = snapshotValue["date"] as! String
        game = snapshotValue["game"] as! String
        info = snapshotValue["info"] as! String
    }
}
```
3.3 Tracking User Interactions

```kotlin
def toAnyObject() -> Any {
    return [
        "game": game,
        "date": date,
        "info": info
    ]
}
```

Once the information is sent using the setValue function on the database reference, it shows up on Firebase’s online platform as shown in Figure 18.
Chapter 4

Problems

This chapter goes over the problems that arose during the development process and gameplay.

4.1 Internal vs. External Database for User Tracking

Tracking user interactions can be done in several ways in iOS development. Deciding on which way to track the user interactions in the game was difficult because there are multiple trade-offs. For example, most external databases are costly and often require user authentication for security reasons. Internal databases allow you to store data on the device, but that means you can only test with a few devices at a time. For this project, I decided on using Firebase, a cloud based system made by Google. It is free, but the main issue with security. Right now there is no user login, all the data is being sent to one database and we use a user ID to differentiate between each
4.2 Repetitiveness of the Game Features

User engagement is an essential part of game play especially if the game is educational. Two of the games, the shark game and the candy monster game, both have touch selection as the main user interaction. This can be repetitive for the user and potentially cause boredom instead of engagement. The monkey game allows the user to move the character around and collect items on the screen. The other games only allow the user to click on an item and there is no interaction with the characters. Ways to potentially mitigate this issue could be, allowing the user to feed the shark by dragging the food and then an animation of the shark eating is shown and for the candy monster game, allowing the user to insert the candy into the partially filled bag.

4.3 Glitches Caused By Timer Collisions

Timers are an important part of the game mechanics, but also one of the most challenging. The timers are triggered through user interaction, but if the user does not interact as expected it creates problems within the game. If the user clicks multiple times on an item that triggers a timer, multiple timers will get triggered which causes the game to be reset several times one after another. This causes the user a lot of confusion and frustration. A potential way to solve this issue is to temporarily dis-
4.3 Glitches Caused By Timer Collisions

able user interaction in the game scene after a timer trigger is selected so they cannot select another item that would trigger another timer.
Chapter 5

User Analysis

In March 2017, I held a user study for the iPad games in the Education Department Lab with Professor Kraemer. We had four elementary students aged 8-10, come into the lab to play the games. Since the study was only one hour, it was not enough time to tell if the games had any significant impact on their understanding of fractions or math anxiety. The students did give useful feedback that can be used to improve the games. All of the students were confused by the monkey game. Many mentioned the bananas came up too slow which made the game boring and the number lines were confusing. All of the students enjoyed the candy monster game. It seems that it’s simplicity won them over and one student mentioned it being their favorite because it required estimation. The shark game was also well received, one of the students mentioned it as their favorite because they had different options to choose from. Among the four students there was one who showed a high level of math anxiety according to the math anxiety assessment we gave before gameplay. The student mentioned that they try to avoid math games as much as possible and they do not like it when the math in the game is obvious. This student struggled with
User Analysis

the games and after playing said they were boring. The balance between fun and educational is not easy. The students who played the games and loved math visibly had more fun than the ones who did not. With such a small sample size we cannot make any generally assumptions or statements about the games and how they might affect student learning, but in the future with more user studies we can test for more significant impact.
Chapter 6

Next Steps

As stated in the last chapter, there is a lot more user studies that need to be conducted in order to know if the games will truly improve students’ understanding of fractions. Fortunately, the work will be continued on beyond my senior thesis. The Education Department is continuing this project with another student through the Presidential Scholar Research program. As the project continues, it can be improved through the user feedback from students as well as launched on the App Store so it can be more easily distributed. The ultimate goal is to be able to run long-term studies in classrooms, so we can see in real-time if the games improve the student’s achievement in math classes specifically regarding their fraction coursework.
Bibliography


