Functions Fun: An iPad Educational Game for Teaching Mathematical Functions and Graphs

Xuan Liu

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Osman Balci, Chair
Eli Tilevich
Anderson H. Norton III

May 8, 2019

Blacksburg, Virginia

Keywords and phrases: Educational game, game-based learning, iOS application, iPad game, mathematics education, teaching functions and graphs

Functions Fun: An iPad Educational Game for Teaching Mathematical Functions and Graphs

Xuan Liu

ABSTRACT

Teaching and learning mathematical functions and graphs pose significant challenges for teachers and students. Students often have difficulty in understanding a functional relationship between two quantities such as distance and time, temperature and precipitation, and gas price and number of gallons. Teaching students to have quantitative thinking about functions can help them understand the rate of change for complicated functions and later succeed in learning Calculus. Traditional educational methods such as static graph images and some learning tools usually have some limitations. Teaching students the dynamic changes of quantities within the static picture has serious difficulties. Compared to the learning tools, the game-based learning increases interest when students are learning complicated functions. This thesis presents a game-based learning application called *Functions Fun*, which runs on iPad tablet computers. The game is created to teach / learn the following functions: Linear, Quadratic, Exponential, Logarithmic, Trigonometric, and Polynomial with degrees over four. Each function is covered under a game level. The game setting is a jungle environment. Each game level has its own scene, challenging the player to take an action while teaching a function and its graphical representation. *Functions Fun* enables students to play and learn functions and graphs in a more effective and entertaining manner.

Functions Fun: An iPad Educational Game for Teaching Mathematical Functions and Graphs

Xuan Liu

GENERAL AUDIENCE ABSTRACT

Teaching functions and graphs plays an important role in mathematics education. Teachers and researchers emphasize the need for students to form the quantitative thinking habits when they are learning different function graphs. This thesis presents *Functions Fun*, an iPad educational game that aims to help secondary students to understand function graphs by varying two quantities x and y such as distance and time, temperature and precipitation, and gas price and number of gallons. Adventure game is chosen as the game genre and six real-life models related to the following six functions are created: Linear, Quadratic, Exponential, Logarithmic, Trigonometric, and Polynomial with degrees over four. Each function is covered under a game level. The game setting is a jungle environment. Players are rewarded for their accomplishments. Through game effects and animations, students can have a better understanding of how the value changes of variables x and y influence the shape of graphs.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Osman Balci for his professional guidance for my research. He has provided great advice during the game development process. With his help, I have gained professional knowledge and skills to be a successful software engineer. I am very grateful for his support during my studies at Virginia Tech, both professionally and personally.

I would like to acknowledge Dr. Anderson Norton for his rich experience and knowledge in mathematics education and for providing technical support in education problem specification. I also would like to thank my committee member Dr. Eli Tilevich for his valuable time and guidance.

I thank my friends Tiancheng Ying and Donghan Hu for their support during my graduate studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACROYNMS

2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
DEG	Digital Educational Game
GAMED	Digital Educational Game Development Methodology
GB	Giga Bytes
GPUs	Graphics processors
IDE	Integrated Development Environment
iOS	Apple's mobile Operating System
OOD	Object Oriented Design
plist	Property List
PPI	Pixels Per Inch
QA	Quality Assurance
RAM	Random Access Memory
STEM	Science, Technology, Engineering and Mathematics
UI	User Interface
XML	Extensible Markup Language

Chapter 1: Problem Definition and Overview

1.1 Introduction

Graphing activities of functions for students and teachers remain an important area in mathematics education. The interpretation of the graph for some complicated functions (e.g., inverse trigonometric functions, polynomial functions) provides an easier and more efficient way for students to learn, understand, and remember functions. However, the way students learn functions and their graphs influence their comprehension about the relationship of variables in functions. Students should learn quantitative thinking about graphs. Instead of just using the slope and the global properties of shape to interpret functions, observing the relationship of two co-varying quantities (represented by x and y) within the coordinate system helps students to better understand the rate of change for complicated functions, which is helpful for later education in Calculus.

1.2 Identification of the Education Problem

Graphing activity shows difficulties in mathematics education, especially in teaching students how to understand functions and their graphs. According to the research results [Moore and Thompson 2015], secondary students have the ability to construct organized ways of thinking about functions and their graphs, but they often lack a basis in reasoning about the quantitative relationships between variable values. Moore and Thompson [2015] conducted interviews with middle-school students. The way some of them understand the graph is based on global properties of the shape and perceptual cues. For example, thinking a line falling downward from left to right means negative slopes. They think the slope and the shape represents the graph instead of understanding how quantities change together within axes' orientation. They interpret the graph statically and give the graph meaning by combining learned facts. But without actually understanding about how quantities change to influence the graph, students are constrained to particular labels and orientations, which in the future will be a big stumbling block to their success in mathematics and other STEM fields.

1.3 Significance of the Education Problem

The function names are not names of shape; but names of a covariational relationship. Learning functions with reasoning about quantities, instead of using perceptual cues for the shape of functions, helps secondary students to gain insight into function relationships between quantities and form abstractions and generalizations from their reasoning. Even when the coordinate conventions, systems changes, so that the shape of the graph changes in a visual sense, students could still understand the relationship of functions [Moore et al. 2013].

On the other hand, Thompson [1994] mentioned that learning The Fundamental Theorem of Calculus needs the understanding of the accumulation of a quantity and the rate of change of its accumulation. He also stated that the rate of change imagination involves a schematic coordination of relationships between two co-varying quantities and their accumulation results

[Thompson 1994]. Reasoning with co-varying quantities in function education helps students form the images of the change of rate, which is a steppingstone for the later Calculus education.

1.4 Identification of the state of the art in solving the education problem

There are plenty of tools for teachers and students to use in order to teach and learn functions and their graphs. Different kinds of media types are used in solving the education problem. Images sometimes cannot satisfy teachers' needs in mathematics education due to its static characteristics, while animation is a relatively better media to demonstrate dynamic changes.

GeoGebra, an interactive geometry, algebra, statistics, and calculus application, aims to learn and teach mathematics and science from primary school up to university level with animations [GeoGebra 2018]. However, most of the tutorial examples on GeoGebra mainly focus on showing the big picture of graph properties for functions. For example, Figure 1 shows how to use GeoGebra to learn quadratic functions. The software enables students to change the value of a, b and c by dragging the button, and the corresponding feedback to the graph shows as the change of the shape and the position. Although it provides a direct-viewing impression on how graph changes and graph properties of quadratic functions, it omits the quantitative meaning for x and y variables.

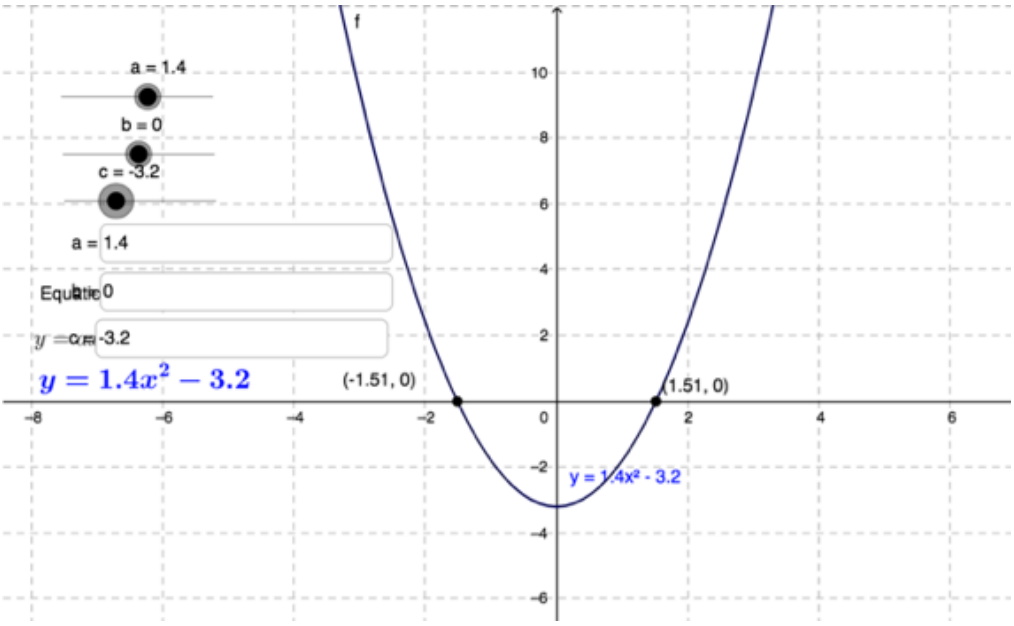

Figure 1. Quadratic Function

Compared to Figure 1, Figure 2 shows a co-variation example that helps students pay more attention to the quantitative relationship between x and y variables.

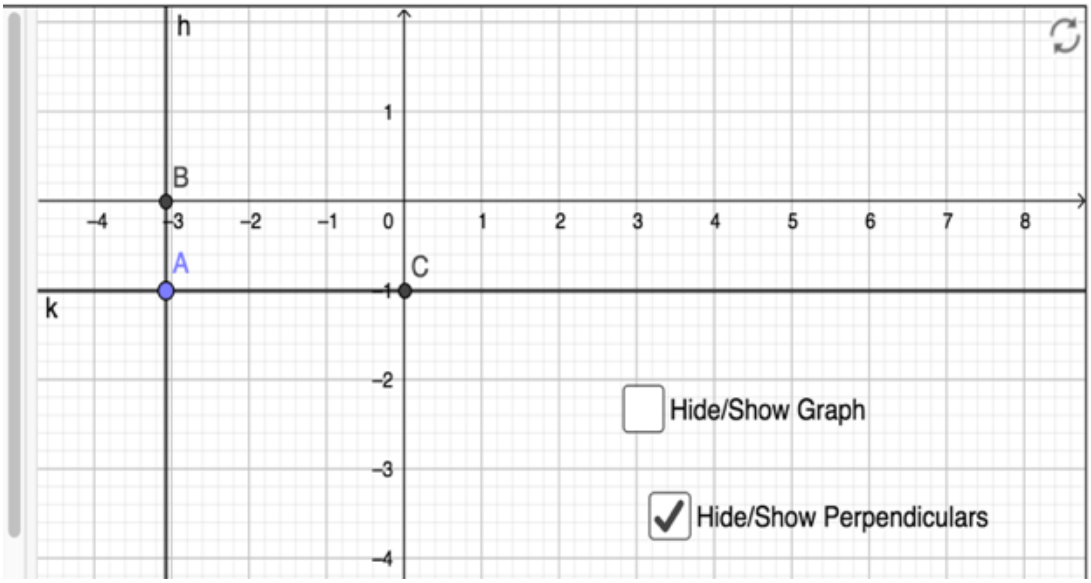

Figure 2. Co-Variation

This stimulates students to think about the question "when x variable increases, would y variable increase or decrease? Increase rapidly or slowly?" Since not many examples are related to the quantitative relationship of co-varied x and y variables, discovering more efficient and interesting educational methods to encourage students to form quantity thinking for functions becomes an important and necessary task in current mathematics education.

1.5 Effectiveness of game-based learning for solving the education problem

Traditional educational methods usually have some limitations. For example, graphs that teachers draw on the blackboard are static, students need to understand the dynamic changes of quantities within the static picture, which shows difficulties in teaching. The game-based learning approach can be a solution to this problem. By using different kinds of media types such as animation, video, etc., game-based learning method is not constrained by static images and text, it can present dynamic changes and provide feedbacks to students. It provides an easy and relaxing way for secondary students to obtain the knowledge during the game playing process and brings more fun than traditional teaching methods.

Compared to current online animation tutorials (GeoGebra), the game-based educational solution allows secondary students to explore the image of the quantitative relationship by themselves during the game, increases the interest in learning complicated functions. By playing the game, students are no longer passive receivers of information, they are results/answers seekers, which makes the learning process efficient, exciting, and fun.

1.6 Definition of the education problem and specification of the game-based learning objectives.

By illustrating the two different ways of function graph thinking above: using the perceptual cues of graph shape to understand the relationship of functions and reasoning about two co-varying quantities in functions, we encourage students to form quantitative thinking of variables in functions. Since the problem is based on the interpretation of functions and their graphs, the boundary of the problem should be restricted by different functions from simple to hard, including linear, quadratic, exponential, logarithmic, sinusoidal, and polynomial. The game-based learning iPad application presented herein is expected to be a useful tool for secondary school students and teachers, who are facing this education problem. *Functions Fun* provides a solution to help students learn functions by considering the quantitative relationship and changes of two variables.

1.7 Overview of the Thesis

This thesis is organized as follows: Chapter 2 describes earlier literature work that was useful for the generation of game ideas. Chapter 3 describes the learning objectives, target audience of the game, and the game ideas for all six game levels. Some iOS game development techniques are presented in Chapter 4. Chapter 5 discusses the game development life cycle and the prototype of the game design. The functionalities of each game level are described in Chapter 6. Chapter 7 introduces the reward system for evaluating the performance of the player. Chapter 8 provides a self-evaluation of the game app with respect to 12 specific quality indicators. Conclusions and future work are presented in Chapter 9.

1.8 Summary of Contributions

The research contributions can be summarized as follows:

1. Created a game-based learning application called *Functions Fun* running on iPad tablet computers to help secondary students learn functions and graphs.
2. Fully applied the GAMED (Digital Educational Game Development Methodology) [Aslan and Balci 2015] in creating the *Functions Fun* game.
3. Identified real-life representations (models) of the following six functions: Linear, Quadratic, Exponential, Logarithmic, Trigonometric, and Polynomial with degrees over four.
4. Created the *Functions Fun* game based on the identified real-life representations, each one under a different game level.
5. Created and implemented a reward system to recognize player's accomplishments.
6. Conducted a self-evaluation of the *Functions Fun* game quality.

Chapter 2: Literature Review

This chapter reviews the work that was useful in the game idea generation process of the GAMED (Digital Educational Game Development Methodology) [Aslan and Balci 2015]. We discuss the importance of applying real-life models to mathematics education and to our game-based learning solution. We also describe some existing applications that import real-life problems to the game design.

2.1 Mathematics Education with Real-Life Models

Teachers and researchers are constantly seeking the best and most efficient way to teach mathematics for secondary students. They found mathematics education is not just teaching students to do calculation and computation, but related to the real-life problems. An experienced math teacher in an interview explained:

"I would hope that it [the functions and graphing unit] does [helps students] in the sense of seeing how, how math problems, in general, are not just simple computation. I think we kind of mislead students into, into saying that here's a sheet of math problems, work these out, because again, that's not real-life... . Uh, nobody walks up when you're 40 years old and hands you a sheet of math problems and says, 'Can you do these?' You know, people just don't do that." [Stein, Baxter and Leinhardt 1990]

More and more teachers are focused on applying mathematics in practical situations and real-life problems. Education researchers emphasize the importance of finding the real-world representations and connections in secondary mathematics teaching, and try to guide students to think about the relationship between mathematical symbol systems like graphs and everyday reality [Gainsburg 2008; Gravemeijer and Doorman 1999].

2.1.1 Why use real-world problems for mathematics education?

Analysis of interviews, reports, and survey results from teachers of middle school classes indicate that real-world problems could encourage students to get involved in learning mathematics in the classroom. Learning just a lot of numbers on the paper without real-life context seldom engages young teenage students and many of them become disinterested or even resentful when they face difficulties. In contrast, mathematics becomes more meaningful when students can combine their real-life experience with mathematical knowledge to reinforce each other [Pierce and Stacey 2006].

Teachers can choose real-life contexts including pleasant sights, sound, and social experiences to increase positive affection of students. This is a good strategy for teachers to create enjoyable and memorable lessons. If the real-world problems are related to students' experience, students will attribute great interest to this general link with past memories.

2.1.2 Apply real-world mathematical models to game-based education solution

Mathematical modeling is a process that tries to represent real-world problems in mathematical terms to solve the problems. Mathematical models can be generated by simplifying complex real-world problems into mathematical forms [Ang 2001]. In our game development, we need to find connections between real-world situations with mathematical function graphs and convert real-world problems into mathematical problem to design the game. For example, modeling the population growth as an exponential function, dirt bike jumps as a quadratic function, and rising of water in a cylinder as a linear function.

The game-based solution integrates mathematics education during the game process. In order to make the game become more attractive to students, and make students become active learners and interested in what they learn in the game, applying real-world mathematical models to games is necessary. In this thesis, I will describe several real-world models (in Chapter 3.3) for 6 mathematics functions (Linear, Quadratic, Sinusoidal, Exponential, Logarithm, Polynomial) and apply them to the proposed game-based solution.

2.2 Existing Game Apps with Real-Life Models

Recently more and more teachers have begun to use math game apps to help their students acquire the required mathematical skills easily while having fun. Many apps use real-world models to attract students' interests in learning math problems. *Pizza Fractions 1* is a math game that offers help for elementary school kids to learn fractions in a simple, visually-based way [Pizza Fractions 1 2015]. Cutting and sharing a pizza with family and friends are common actions in real-life. Within a restaurant background, players are asked to cut and choose a correct fraction of a pizza to serve customers. The evaluation system such as counting the time and the scores of each attempt stimulates students to perform better. *Candy Factory* is another example of using real-life models to design a math game [Ying, Balci and Norton 2019]. These apps provide us examples on how to integrate math games with real-world situations, which could help us to design game levels for different functions and graphs.

Chapter 3: Game Idea Generation

In this chapter, we first define the target audience of the game, describe the learning objectives, and determine some game settings such as the game genre and game story. Then we illustrate the game ideas for each game level and build the real-world models for later game design and implementation.

3.1 Define Target Audience and Learning Objectives

Functions Fun is designed for students and school teachers to learn and teach functions and graphs. The target audience is students from grade 8 to grade 12 when they begin to learn Algebra. Students at this level are at the entrance of recognizing different functions, solving systems of equation problems and graphing the proportional relationship between variable x and y. In order to increase the interest of learning functions and equations and to encourage them to explore the quantitative relation represented in graphs between function variables, *Functions Fun* picks six functions that are very common in secondary education as six game levels: Linear Function, Quadratic Function, Exponential Function, Logarithm Function, Trigonometric Function, and Polynomial Function. *Functions Fun* addresses three learning objectives:

1. Familiarize students with different function equations and function graphs. Distinguish functions according to their graphs.
2. Develop quantitative thinking about the function graphs. For example, how does the value change of x variable influence the y value? How does the value change of the two variables affect the function graph?
3. Train students to think about the rate of change of two co-varying quantities when drawing the graph.

Functions Fun guides secondary students to explore the knowledge of functions and graphs from games by themselves. During playing the game, students are expected to form a better thinking habit to interpret functions and graphs. We hope *Functions Fun* can help students succeed in using functions and graphs to solve real-life application problems.

3.2 Define the Game Genre and Game Story

Before generating the game ideas, we need to define the game genre first. Adventure games, multiplayer online games, puzzle games, strategy games or role-playing games are several genre types to develop game apps. Students are always curious and excited about the adventure game since it's an unknown journey for them. Therefore, we define *Functions Fun* as an adventure game with the jungle theme. Students can use their knowledge to solve all the difficulties in the adventure and receive awards. The game character begins the adventure under the environment of rivers, woods, mountains, and cliffs. The player is asked to complete the challenges to unlock the next level.

3.3 Generate Game Ideas from Real Life

After determining the game theme, finding a real-life model related to each function became a demanding task. Creativity and imagination play important roles in generating game ideas. I started brainstorming with Dr. Osman Balci and some friends who have rich gaming experience to collect game ideas and pick the best among different ideas.

3.3.1 Linear Function

The riverboat problem is a good real-life model for linear functions. As Figure 3 shows, the boat has velocity 3m/s and it aims to reach point A. However, because of the river speed 5m/s to the east, the boat reaches point B. The position of the riverboat with respect to the time is represented as the linear function: $y = 3/5x$. The learning objective is to make students think about how the value changes in two directions influence the shape and the slope of the graph. In the game, changing the speed of the boat and the river speed will change the slope of the linear function. Using riverboat problems to build the game scene is fun for students and matches the game theme.

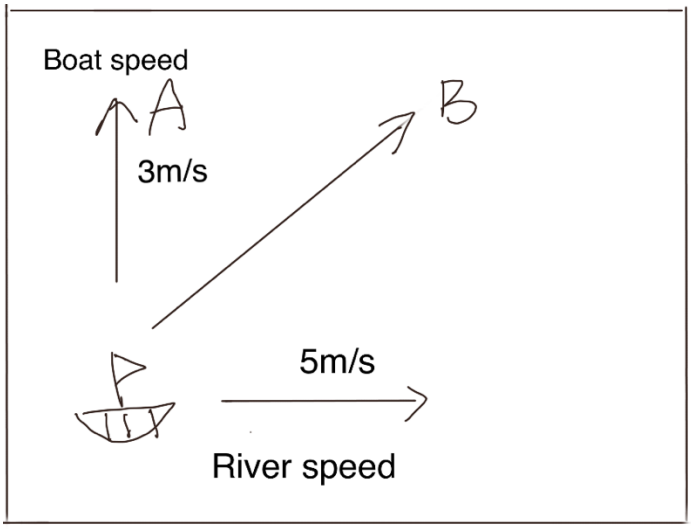

Figure 3. Riverboat Problem

3.3.2 Quadratic Function

In our daily life, many objects behave like a parabola. It's not hard to find real-life models for quadratic functions, for example, throwing a ball, dirt bike jumps, and firing a missile in a mission [MathisFun 2017]. In our game, we choose the dirt bike jump as the real-life model. As Figure 4 shows, the trajectory of the bike is the quadratic function graph. Angry Bird provided us inspiration for designing the game level. The Angry Bird game lets players choose the force and the angle of throwing the bird to strike the target position. In our case, our game allows players to choose the jump angle and the initial speed of the bike to reach the safe area. All landings within the safe area have different jump angles and speed, which generate different quadratic functions.

3.3.3 Exponential Function

Exponential functions are usually related to real-world application problems, such as compound interest, population growth, and student loans. Exponential growth is a critically important aspect of finance, demographics, biology, electronics and many other areas. The most obvious characteristic of the exponential function graph is rapid growth. The game design should focus on how to make students think about the growth results of exponential functions. The game idea is inspired by the games Flappy Bird and Mario. Figure 5 shows the game idea for this level. The ground platforms are placed in the shape of the exponential function graph. Players control the game character to jump and step on the ground platforms just like Mario. This level lets students experience the increasing rate of the exponential function by jumping between ground platforms.

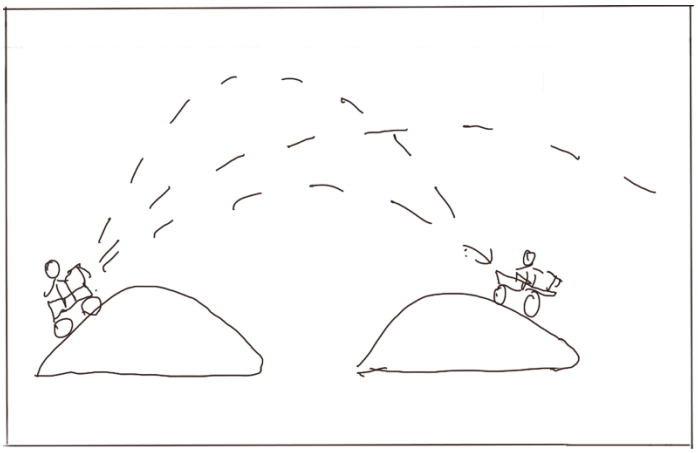

Figure 4. Dirt Bike Jump

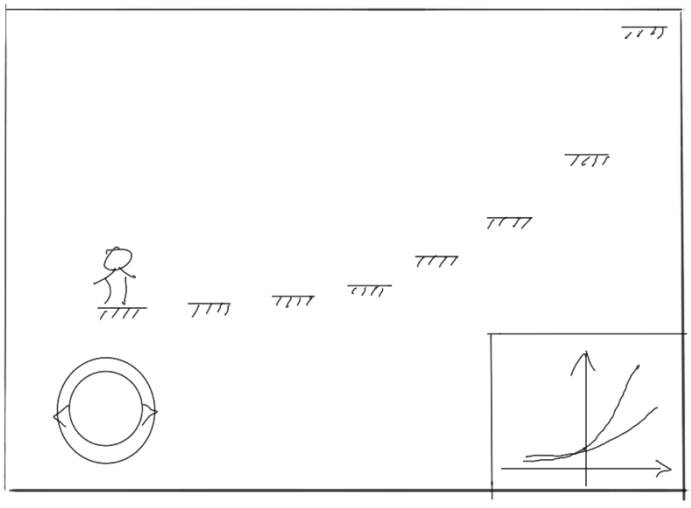

Figure 5. Game Idea of Exponential Function

3.3.4 Logarithm Function

The logarithm function is the inverse of the exponential function. Very similar to the exponential function, the logarithm function has many real-world applications, such as earthquakes (Richter scale) and chemistry (pH balance). The graph of the logarithm function only goes through the positive half of the x axis. With the increase of the x variable, the growth of the y variable becomes slower and slower. According to these features of logarithm graphs, we use a bending springboard to represent the logarithm function graph. Players push down the springboard to change the degree at which the springboard is bent to create different logarithm functions. Figure 6 shows the original idea of combining the springboard with logarithm functions. The game character will jump over a mountain with the help of the bent springboard.

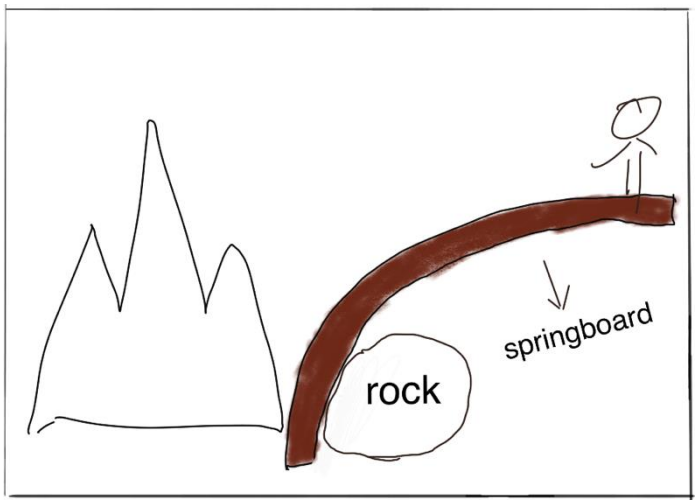

Figure 6. Game Idea for Logarithm Function

3.3.5 Trigonometric Function

Trigonometric functions can be used to model many real-life scenarios: radio waves, tides, musical tones, electrical currents, and the motion of a pendulum. A common sinusoids example is the Ferris wheel problem that asks students to determine the height of a rider with respect to the time (shows in Figure 7) [Mathdemos 2010]. To fit the game theme, we changed the Ferris wheel to a water wheel against the background of a river. Amplitude, Period and Phase Shift are three properties of the trigonometric function. The Amplitude represents the peak value for the trigonometric function graph. The Period goes from one peak to the next (or from any point to the next matching point). The Phase Shift describes the shift in horizontal distance from the usual position [MathisFun 2018]. As for the game design, we use the water wheel size, starting position, and period value to represent amplitude, phase shift, and period in order to let students associate the function equation with the graph by considering these three properties.

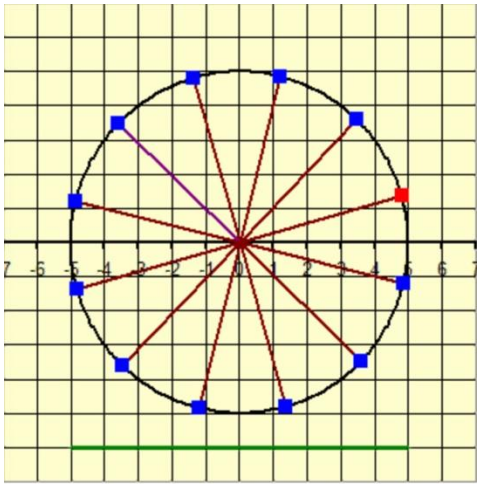

Figure 7. Ferris Wheel Model for Trigonometric Function [Mathdemos 2010]

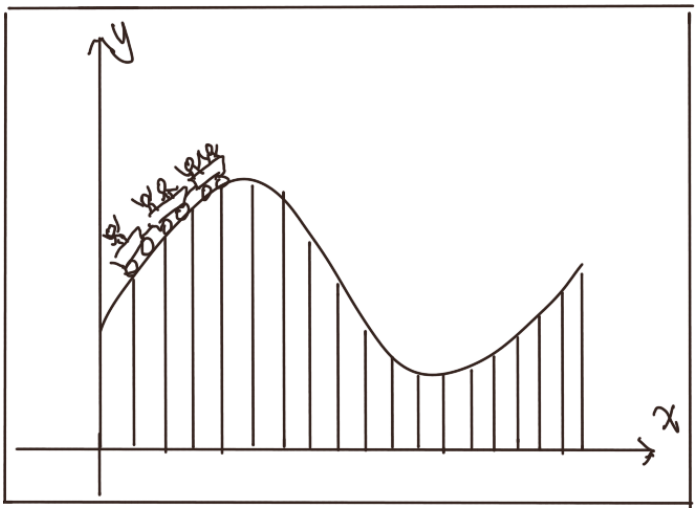

Figure 8. Roller Coaster Model for Polynomial Function

3.3.6 Polynomial Function

The polynomial function is the most complicated function among these six. A polynomial has constants, variables and exponents that can be combined using addition, subtraction, multiplication and division (except division by a variable). Polynomial functions usually have multiple terms and can be written in a standard form that puts the term with the highest degree first. Functions Fun focuses on polynomial functions with degrees over four. The roots of a polynomial function is an important topic when teaching these functions. A polynomial equation of degree n has exactly n roots. Drawing the polynomial function with degrees over five is quite difficult. The understanding of the multiplicity of a root in a polynomial function enables students to sketch the graph of the function [MathCentre 2009]. The learning objective in this

level is to ask student to convert the standard form of a polynomial function to the multiple-root form according to the function graph. Students should know how to read roots information from the graph and estimate the multiplicity of a root according to the quantitative value of x and y variables. The graph of a polynomial function looks like the roller coaster as Figure 8 shows; therefore, we use a roller coaster as the mathematical model for polynomial functions.

Chapter 4: Game Design

In this chapter, we discuss the game development process for *Functions Fun*. We also introduce the DEG development methodology, the spiral design strategy, and some development principles. We then explain how we employ these methods to the design and development of *Functions Fun*.

4.1 Game Development Cycle

According to GAMED: Digital Educational Game Development Methodology [Aslan and Balci 2015], the development of a digital game should follow the DEG life cycle that consists of four phases shown in Figure 9. The phases include the Game Design Phase, Game Software Design Phase, Game Implementation and Publishing Phase, and Game-based Learning and Feedback Phase. We have discussed the education problem and game idea generation in Chapter 1: and Chapter 3:. In this Chapter, we explain the game design of *Functions Fun*. As for the Game Design Phase, we will introduce the techniques that are used in developing the game in Chapter 5:. The game functionalities and evaluation system corresponding to Game Implementation and Publishing Phase and Game-based Learning and Feedback Phase will be described in Chapter 6: and Chapter 7:.

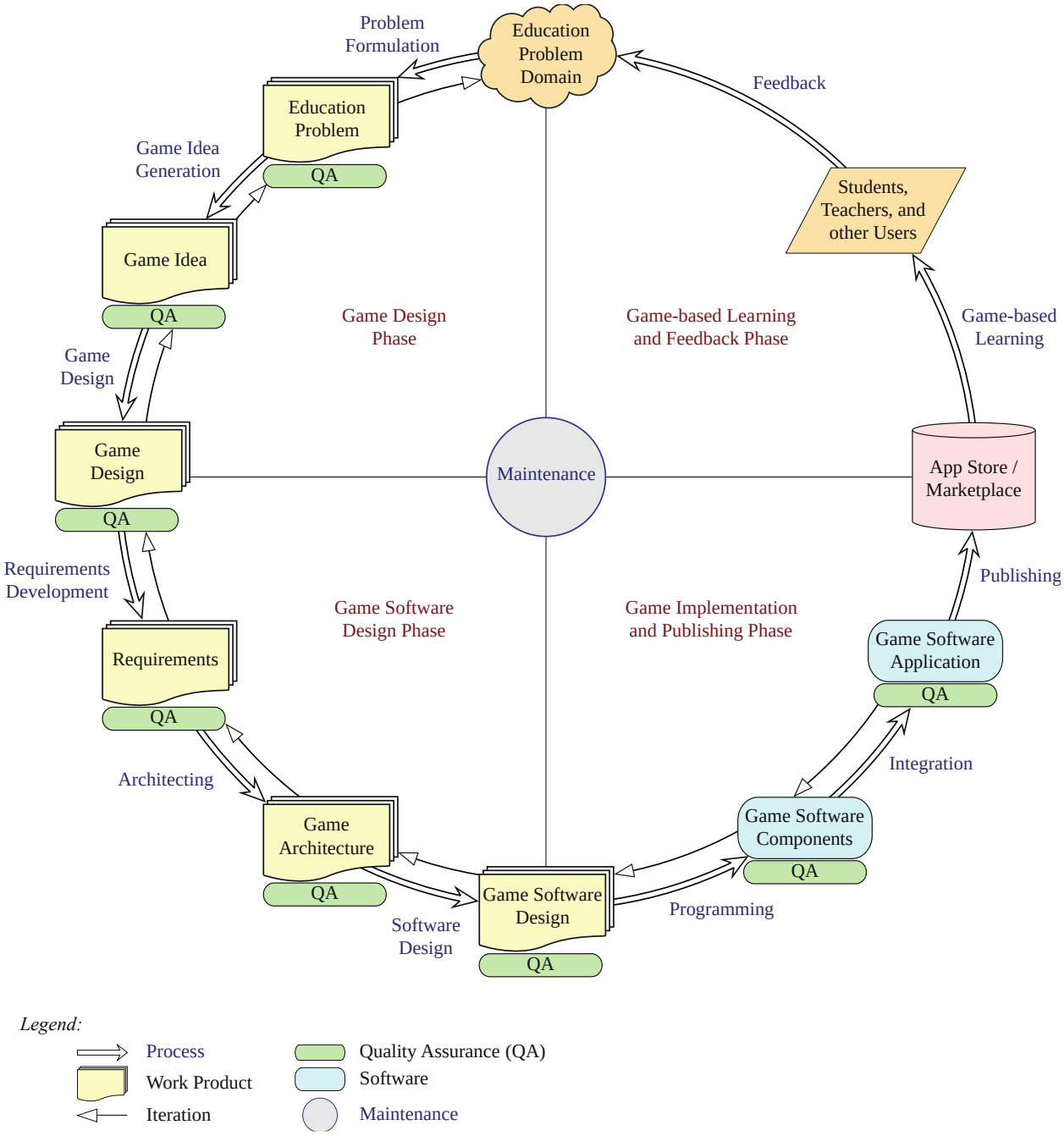

Figure 9. Life Cycle of Digital Educational Game Development [Aslan and Balci 2015]

4.1.1 Spiral Game Design Strategy

With the game ideas mentioned in Chapter 3:, the next step is to design the game user interfaces and the game functionalities to fulfill the learning objectives. The Spiral Game Design strategy [Aslan and Balci 2015; Boehm 1986] shown in Figure 10 is very useful during the game design

process. The game design process consists of four major activities: prototyping, playtesting, evaluation, and risk analysis.

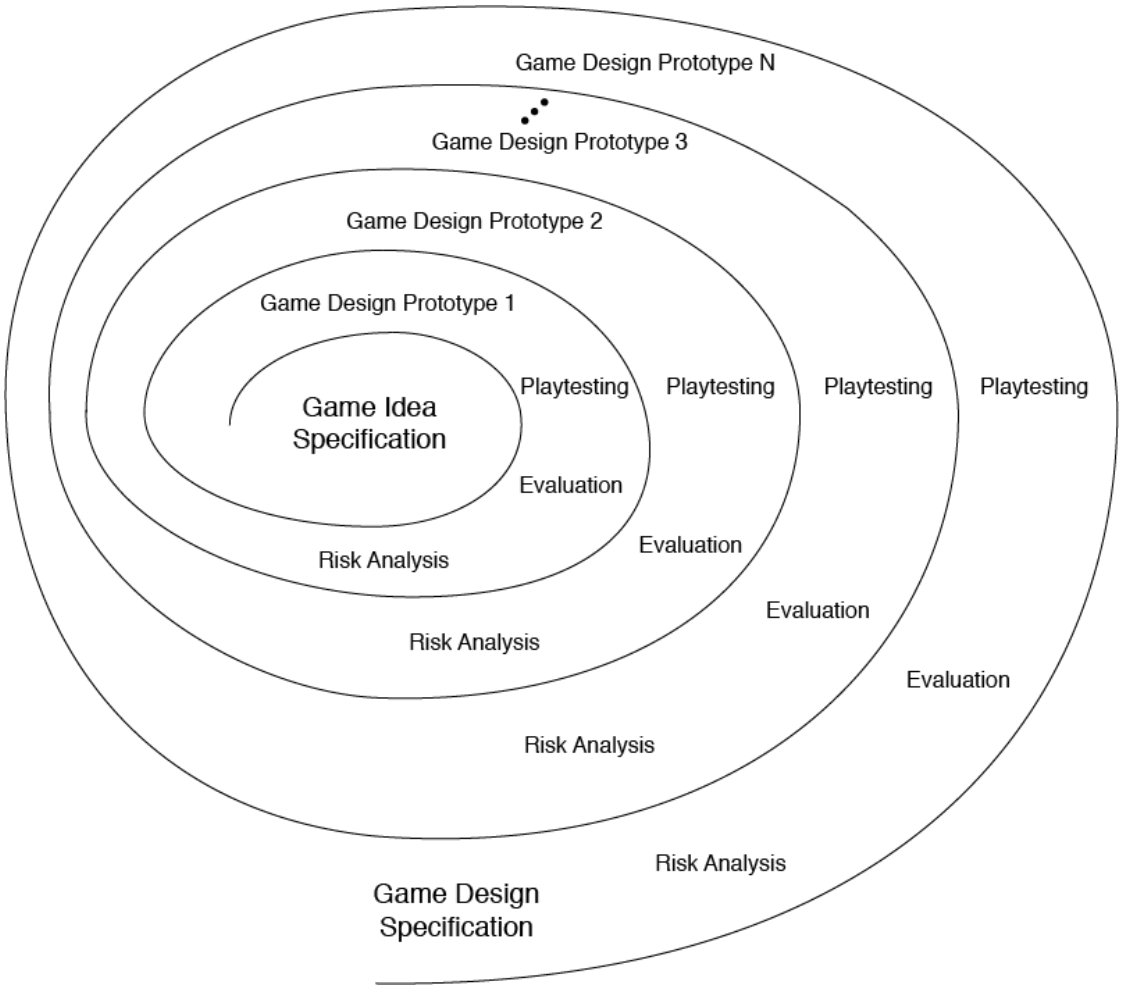

Figure 10. Spiral Game Design [Aslan and Balci 2015]

The prototype of the game represents the game's user interface to enable a player to interact with. It should be built rapidly and easy to modify. Two major prototyping methods exist: Paper Prototyping and Software Prototyping. Compared to Software Prototyping that uses software to put user interfaces together, Paper Prototyping that uses hand-sketched to draw the game graphic UI and update changes is much easier. So, in the early stages of game design, especially in game idea generation stage, we used Paper Prototyping to design graphic UI and interactive activities. We later continued with Software Prototyping to develop the prototype version for playtesting. In section 4.2, we will show the game prototype.

Playtesting, an important step during the process, aims to discover design flaws, provide insight, and give feedback after playing with a prototype of a game design. Dr. Osman Balci and some classmates experimented with the prototype of the game and provided plenty of helpful suggestions. Suggestions were from four major areas: missing interactions between users and

interfaces, lack of instructions for playing the game, unclear learning goals of game levels, and user experience feedback. We listed some design problems that fell into these four areas and the solutions to improve our design. As for the missing interactions part, experimenters mentioned that sometimes they expected to have interactions with buttons but nothing actually happened after they clicked the button. They waited for interaction feedback for some time and considered whether the game had crashed. So we checked all the interactions between users and interfaces to be completed when developed the game. The lack of instructions of the game usually makes users feel confused about how to play the game. Experimenters reported that sometimes they did not know where to start. In order to solve this design drawbacks, we added a question button for users to get help information on how to play the game. Unclear learning goals of game levels was the toughest problem in the educational game design. Redesigning the game is needed if experimenters feel that they do not know the learning objectives of that game level. We redesigned the game to include a specific learning goal for each level and made clear what users should have learned. The last suggestion area is the user experience feedback. Experimenters complained that when they lost a game life, the notification is not obvious so that they did not notice they lost a life. To improve that, we added flashing actions to remind users that they have lost one life and added sound to indicate the failure of the attempt. The position of interfaces should make users feel comfortable and the notification should be obvious.

Evaluation is a self-evaluation step that judges the game quality by 12 quality indicators. The 12 indicators are Acceptability, Challengeability, Clarity, Effectiveness, Engageability, Enjoybility, Interactivity, Localizability, Rewardability, Simplicity, Transformativeness, and Usability. We will discuss game app evaluation in Chapter 8:.

Risk Analysis is expected during the game development project management, which consists of risk identification, risk analysis, risk planning, and risk monitoring. Risks should be identified, prioritized in terms of their estimated probability of occurrence and consequences, monitored during the project, and mitigated under a plan [Pressman and Maxim 2015; Sommerville 2011]. In our case, risk happens when the app runs on different iPad sizes. The UIs look different on a 9.7 inches iPad from a 12.9 inches iPad. All contents should be compatible on all screen sizes. We tested *Functions Fun* on all types of iPads and made sure that it works properly. Instead of using fixed numbers for the size and fixed position to create the UI, we used relative sizes and positions of the screen to make sure that UIs looked the same no matter how the screen sizes changed.

For all six game levels, we repeated the above four activities six times. We employed the Evolutionary Development principle by gradually improving the game interaction and adding new game effects to *Functions Fun* according to the user experience feedback. We also employed the Incremental Development principle in our game development process. Every time we added new features to *Functions Fun*, we identified the application as a new version.

4.2 Prototyping the Game Design

In this section, we show the prototyping of the game design in Figure 11, which is the early design of the game.

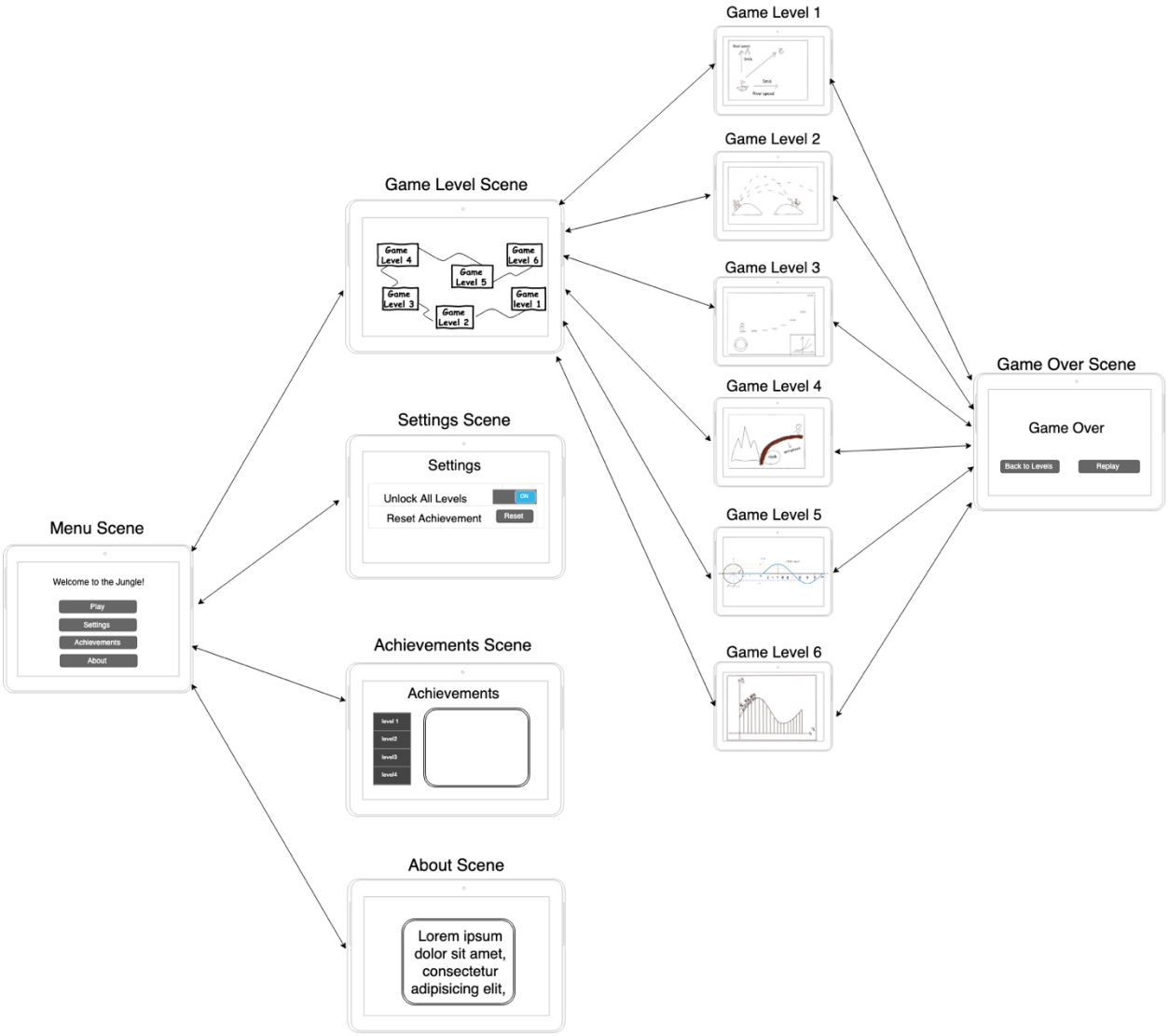

Figure 11. Game Prototyping of *Functions Fun*

Since we did not use a storyboard for developing the game, we created this game prototyping to show all game scenes and their relationships. The Menu Scene is the first scene after users launch *Functions Fun*. The four buttons on the Menu Scene direct to four different scenes: the Game Level Scene, the Settings Scene, the Achievement Scene, and the About Scene. These four scenes can go back to the Menu Scene. The Game Level Scene is the game scene where users can choose which game level they want to play. Users are allowed to switch between the Game Level Scene and each game level. If players fail to accomplish the game challenges for the game scene, *Functions Fun* switches to the Game Over Scene. At this time, users can only choose to replay the current game level (go back to the game scene) or choose a different game level (go back to the Game Level Scene).

Chapter 5: Game Implementation Techniques

In this chapter, we focus on the technical tools such as software, API, and framework that are used for game implementation and explain some specific properties and methods for game animations and interactions. Furthermore, we introduce the physics engine and describe how to make a physic world in the game. We also illustrate how to deal with collisions between objects.

5.1 Software, Programming Language and Game Engine

Functions Fun is an iPad game application So we chose Xcode as the developing software and Swift as the developing language. We decided *Functions Fun* to be a 2D game. In order to achieve animations and physics effects for 2D game development, we applied Spritekit and Scenekit to *Functions Fun* instead of Metalkit which is used in 3D games.

5.1.1 Software

Xcode is an Integrated Development Environment (IDE) for macOS developed by Apple and aimed at developing software for macOS, iOS, watchOS and tvOS applications. The source code editor of Xcode provides an easy way to modify or refactor code. Users can see source control changes with the related line in Xcode. Xcode has a built-in testing engine that dramatically speeds up testing using simulated devices running in parallel [Apple 2019c]. Xcode also allows users to customize the debugging tools. Compared to other developing IDEs, Xcode is the best IDE for developing applications in Apple platforms. We use the latest release version 10.1 to develop the game app on iPad.

We use the latest version iPad Pro with the 11-inch display screen for testing *Functions Fun*'s performance. The memory storage of the testing iPad is 256GB. The display employs 2388-by-1668-pixel resolution at 264 pixels per inch (PPI). The RAM of the iPad is 4GB.

5.1.2 Swift

Swift is a powerful, multi-paradigm, compiled programming language developed by Apple for all Apple platforms. Although Swift follows the Object-Oriented Design (OOD) principles, the syntax of Swift is more concise compared to Java. Features like closures unified with function pointers and multiple return values make codes more expressive [Swift.org 2019]. Swift is also a safe language for use. It eliminates entire classes of unsafe code. Users need to initialize variables before use. Swift will check arrays and integers for overflow and manage memory automatically. Since we only focus on developing the iPad version of the game, we use Swift as the developing language.

5.1.3 SpriteKit

SpriteKit is a framework for drawing shapes, particles, text, images, and video in 2D. SpriteKit is built on Metal (the framework that renders advanced 3D graphics and perform data-parallel computations using GPUs) to take advantage of the high-performance rendering while providing a simple programming interface for developers to create games and other graphics-intensive apps in an easy way. Using SpriteKit, SceneKit, along with advanced new physics effects and animations, we can add force fields, detect collisions, and generate new lighting effects in games to customize the physics world.

In SpriteKit, the graphic shape is created by a SKNode. SKNode is the base class of all SpriteKit nodes. The text is created by a SKLabelNode. After adding these nodes to the scene and setting the size and position of the nodes, the basic graphics of the game scene are finished.

In SpriteKit, we do not use storyboard for game development. Instead, we create the SpriteKit Scene to build the user interfaces. After launching the game app, the program runs the GameViewController file first. We set the first scene of *Functions Fun* to be the Menu Scene in the GameViewController file. We develop each scene separately by creating the SpriteKit Scene, then call the presentScene method to switch between different game scenes.

5.2 Physics and Collisions

5.2.1 Physics World

In SpriteKit, SKPhysicsWorld is the drive of the physics engine in a scene [Apple 2019b]. We do not need to create SKPhysicsWorld directly, the system will create a physics world and add it to the scene's physicsWorld property. The physics world allows users to set important properties like gravity, respond to collision between physics bodies using contactDelegate, and perform custom collision detection or hit testing. Figure 12 shows how to set the gravity in the physics world and respond to collisions in current scene.

```
self.physicsWorld.gravity = CGVector(dx: 0, dy: -9.8)
self.physicsWorld.contactDelegate = self
```

Figure 12. The Application of Physics World in Code

5.2.2 CategoryBitMask, CollisionBitMask, and ContactTestBitMask

Before introducing these three bit masks, we need to explain the physics body of the node. The SKPhysicsBody is an object that adds physics simulation to a node. Only a node that has the physics body could be affected in the physics world. When a scene processes a new frame, the system performs physics calculations on physics bodies that attach to nodes. The calculations

include the effect of gravity, friction, forces, and impulses on a body and collisions with other bodies. The isDynamic property, a Boolean value, controls whether a physics body is affected by gravity, friction, collisions with other objects, and forces or impulses applied to it.

The bit mask is used to classify and distinguish physics objects that appear in the scene. The CategoryBitMask defines different physics bodies up to 32 categories. In other words, CategoryBitMask is the identification of a physics body. CollisionBitMask defines which categories of physics bodies can collide with this physics body. Only physics bodies in the CollisionBitMask category could collide with this physics body, otherwise physics bodies will penetrate each other. ContactTestBitMask defines which categories of physics bodies cause contact notifications with this physics body. Only physics bodies in the ContactTestBitMask category could contact with this physics body. CategoryBitMask, CollisionBitMask and ContactTestBitMask define which physics bodies collide with each other and when the game has contact interactions.

5.2.3 Detecting Collision

SpriteKit supports two interactions between physics bodies: contact and collision [Apple 2019a]. When two physics bodies touch each other is called contact. Collision is used to prevent two objects from interpenetrating each other. Contact is used when developers need to make gameplay changes while a collision occurs. The collision detection compares the body's collision mask with the other body's category mask by performing a logical "AND" operation. If the result is a nonzero value, this body is affected by the collision.

5.3 Handle Touch Events

iOS development uses the UIKit framework to determine touch events that occur in a view and call touchesBegan, touchesMoved, and touchesEnded methods to correspond to different phases of the touch event–handling process. In SpriteKit, we need to override these methods for the game scene and use them to provide a response to touch events.

5.3.1 touchesBegan

We use the touchesBegan function in-game to monitor when the touch event occurs. When a finger (or Apple Pencil) touches the screen, UIKit creates a UITouch object, sets the touch location to the appropriate point, then sets its phase property to UITouch.Phase.began. The passed-in parameters for the touchesBegan function are touches, which is a set of UITouch instances, and the event to which the touches belong. UIKit could respond to not only multiple touches but also a multiple-finger touch. For a whole touch process in a view, the UITouch set contains only one touch by default. When users touch the screen in tandem with two fingers, it will call the touchesBegan function two times with only one touch in the touches set. If users set the view's isMultipleTouchEnabled property to true and touch the screen with two fingers simultaneously, it will only call the touchesBegan function for one time with two UITouch instances in touches set.

5.3.2 touchesMoved

We use the touchesMoved function in-game to monitor the event of finger movement. After touching the iPad screen, the same finger moves around the screen and UIKit updates the touch location and changes the phase property of the touch object to UITouch.Phase.moved. In this method, we update the value of related variables according to the location of the moving touch instance.

5.3.3 touchesEnded

We use the touchesEnded function in-game to monitor the case when users lift their finger from the screen. UIKit changes the phase property to UITouch.Phase.ended and completes the whole touch sequence. UIKit will destroy the UITouch object when the touch process ends.

5.4 Game Animation

SpriteKit provides two mechanisms to animate nodes and update their properties: the Update function and using Actions. As for Actions, developers need to create an action instance for a node and call the run method to animate nodes. The Update function is called by system per-frame and allows developers to build all the game logic inside the function. Compared to Actions, the Update function is more convenient in the case that the game has more sophisticated actions to be designed. If nodes need to be updated in a wholly custom manner, the Update function is a better choice than using actions to do so.

5.4.1 Actions

The Actions run the animation on a node in the scene. The most common use for Actions is to animate changes to a node's properties. For example, developers can create actions to change the size and position of a node and make it flashing by altering the transparency. SpriteKit allows developers to create and run the action by calling the run method to run an instance of SKAction. With the duration parameter, developers could control how long the action will take to complete in seconds and SpriteKit will change the properties of the node over more than one frame rendered by the scene automatically. Developers can also chain actions together by creating an action sequence to achieve movement combinations and the animation effect overlay.

5.4.2 Update Function

The Update function is used to perform any app-specific logic to update scenes. In SpriteKit, we do not need to call the update function directly, the system calls the Update function once per frame. In the game, the system performs 60 frames per second. One frame uses only 0.0167 second. The Update function is the first function called when a game scene is presented. In the Update function, we perform the animation of nodes, calculate the physics, and check the

position of nodes to see if they collide with each other. The Update function updates the game changes 60 times per second so that users can get immediate feedback from the game.

Chapter 6: Game Functionality Description

In this chapter, we focus on describing the design of each game scene and the game functionalities for each game level based on screenshots from *Functions Fun*. We explain the interactions between users and interfaces including animations and scene switching. We also provide useful information for both developers and users on the game development and the game instruction.

6.1 Menu Scene

According to the prototype in Chapter 4.2, the first scene after the game launched is the Menu Scene as Figure 13 shows below. The Menu Scene has four buttons and a sound node button. Users can click the Play button to go to the Game Level Scene where they can choose the level they want to play. The settings button allows users to change the preference settings. Clicking the achievement button will direct to the Achievement Scene that lists all badges obtained from all levels. Figure 14 shows the About Scene after clicking the about button. It displays the description about this game app on a white, scrollable board. The back button on the top left corner is used to go back to the Menu Scene. The sound node button controls the background music to be on and off. When users click the sound button, it turns off the music and change the sound image to mute as Figure 13 shows. Users can click again to set it back. The game app will remember the current background music status when users switch between different game scenes.

6.2 Settings Scene

The settings Scene allows users to unlock all game levels and reset all the users' achievements. Figure 15 shows the UI design of the Settings Scene. We use the switch button to switch between the status of locking and unlocking game levels. As for resetting achievements, we use a reset button. When users click the reset button, it pops out a warning window to ask users if they do want to reset the achievement shown in Figure 15. Since the reset button will clear all achievement records, the warning alert helps prevent from touching the button by mistake and losing records.

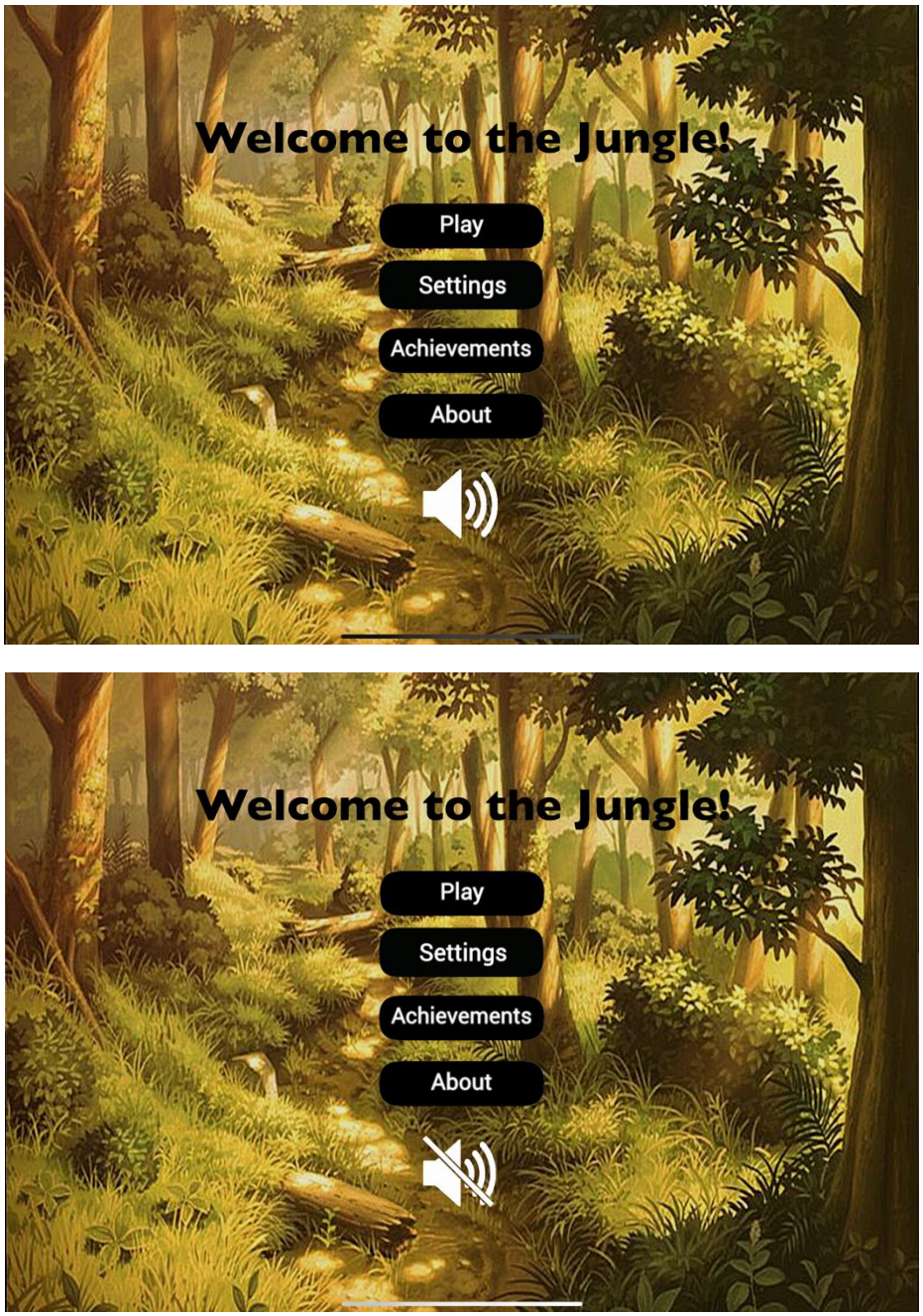

Figure 13. Menu Scene

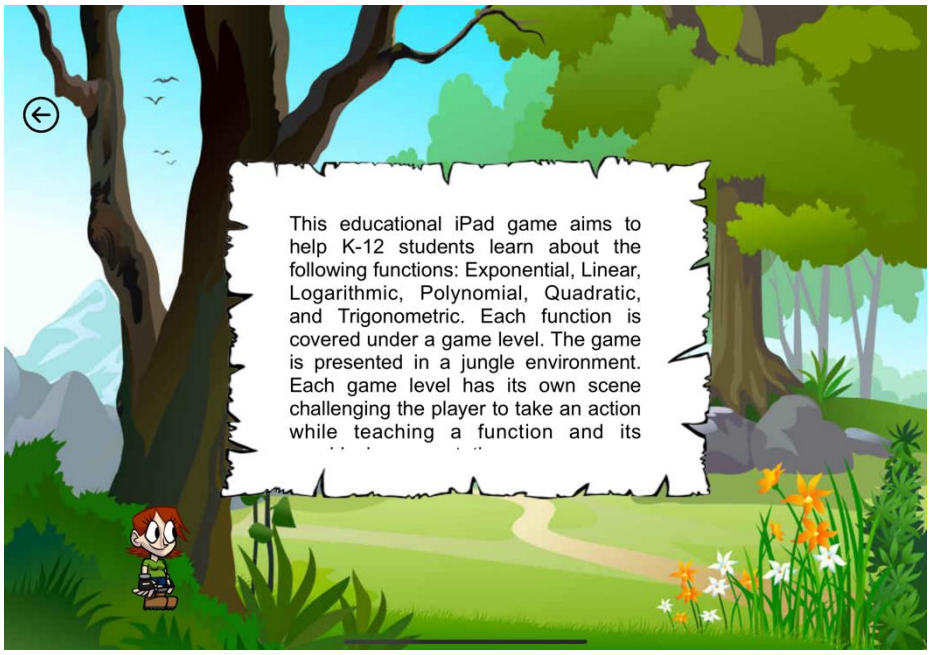

Figure 14. About Scene

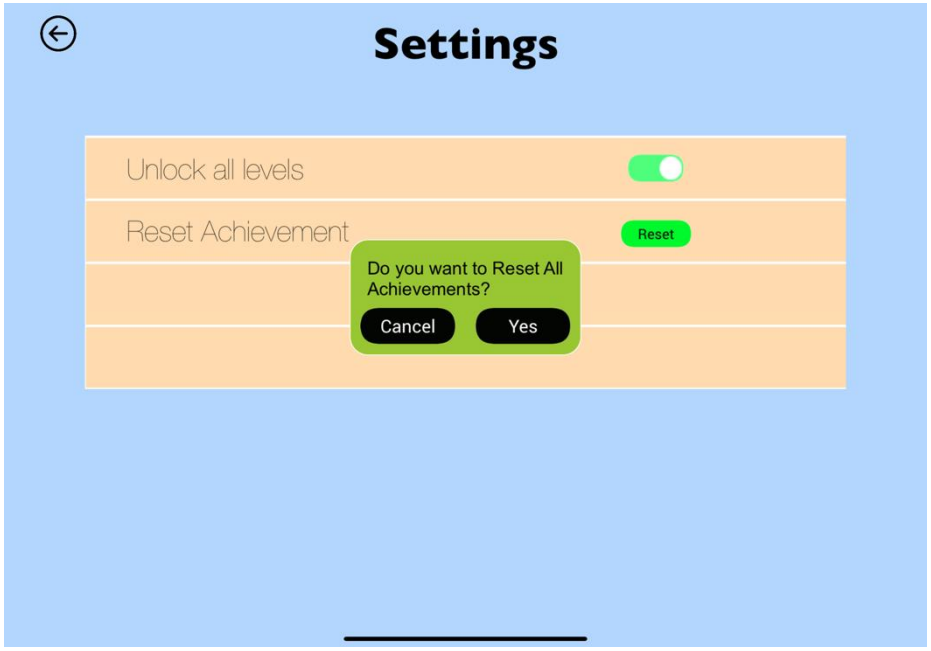

Figure 15. Clicking the Reset button in Settings Scene

6.2.1 Unlock all Levels

All game levels are locked except the first level when the game is first launched. Only when users complete the challenge in current level with three stars can the next level be unlocked. Users can only play the unlocked levels from level 1, which is the easiest, to the hardest level 6. The goal of using locked levels is to let students learn functions from the game step by step in order of difficulty. Adding the choice of unlocking all levels to the Settings Scene is very helpful for teachers who need to check each level without accomplishing all the challenges. Figure 16 shows the Game Level Scene after switching the unlock all levels condition to true. At this time, users are allowed to play any game levels.

6.2.2 Reset Achievements

The reset achievement button deletes all the achievement records in the game app and resets the Achievement Scene to its original status. With this functionality, users can restart the game again with no achievement badges. In the Achievement Scene, we set the alpha property of badge nodes to 0.1 to make them transparent as Figure 17 shows. The alpha property is the transparency value of the node's contents. The range of the alpha is 0.0-1.0 and its default value is 1.0. We increase the transparency of the nodes to give users a signal that they have not got the badges. When the alpha value equals 0.0, the node disappears in the screen. We will use this property a lot later to hide nodes from the screen.

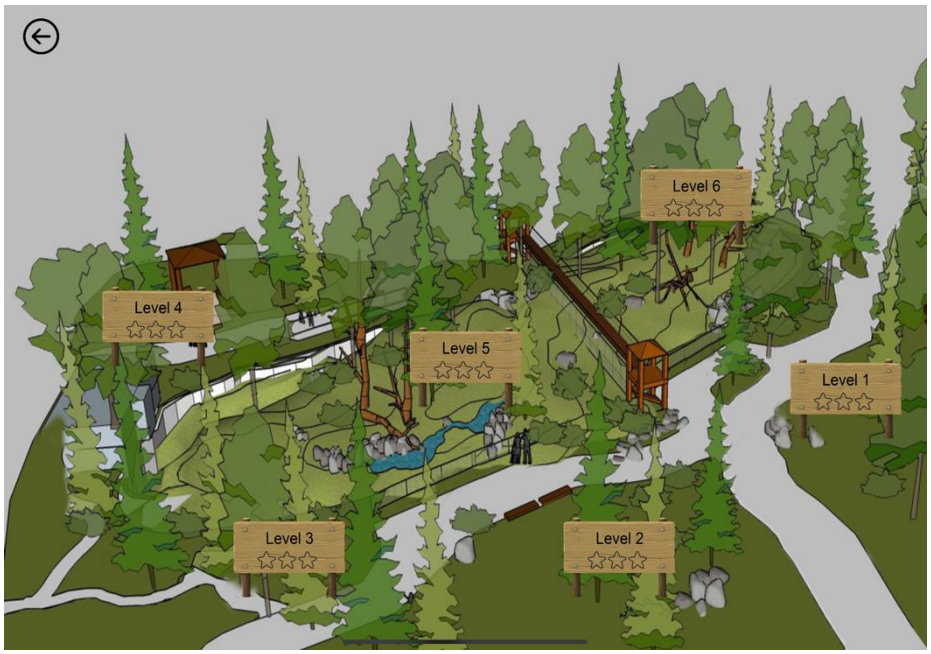

Figure 16. Game Level Scene with Unlocked Levels

Figure 17. Achievement Scene after Resetting Achievements

6.3 Achievement Scene

The Achievement Scene lists all the badges users got from six game levels. Users can click the level buttons on the left side of the screen to navigate through different levels. For the level that users choose, we increase the size and decrease the transparency of the level button. On the right side of the screen, we set the alpha property of the obtained badge nodes to 1 and the others stay the same. When users touch the badge node, a window that conveys the description of that badge pops out. The message window disappears when users lift their finger from the node. Figure 18 shows the screenshot of level 2 achievement when users press the Proficient badge node. The message says that when users get three stars in level 2, they will get the Proficient badge. The top left back button allows users to go back to the Menu Scene.

6.4 Game Level Scene

After users click the Play button in the Menu Scene, the game app switches to the Game Level Scene as Figure 19 shows. We use the wooden sign board to represent the level label node. Click the level label node will go to the game scene and start the adventure. In the screenshot, the user finishes the level 1, level 2, and level 3 with 3 stars. But the user only gets 1 star in level 4, which can not unlock the level 5. For level 5 and level 6, the level label node has a locker on the right side. While users click the locked level, an alert message shows up and prevents users from going to the Game Scene.

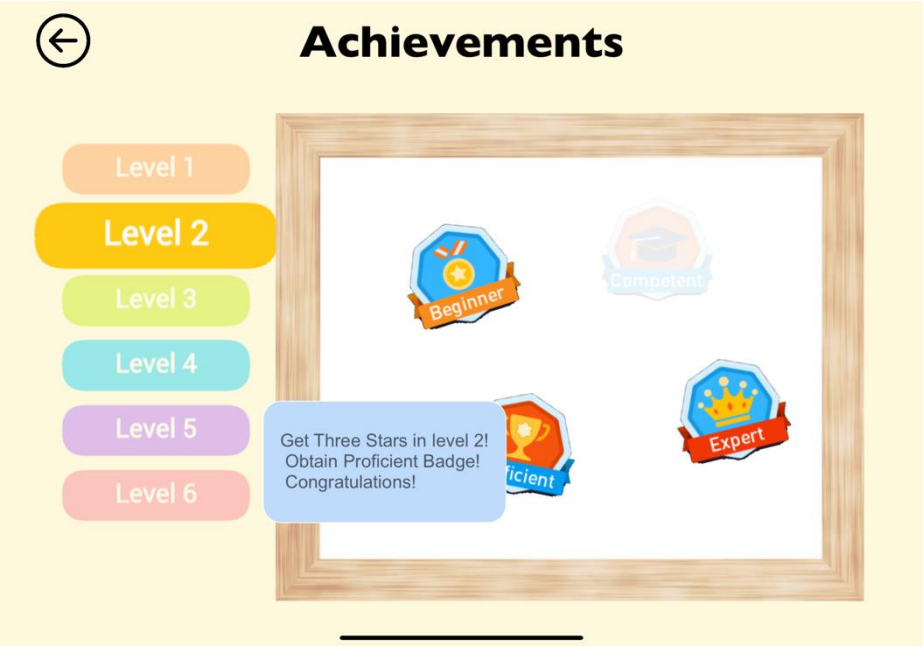

Figure 18. Achievement Scene: All Obtained Badges for Level 2

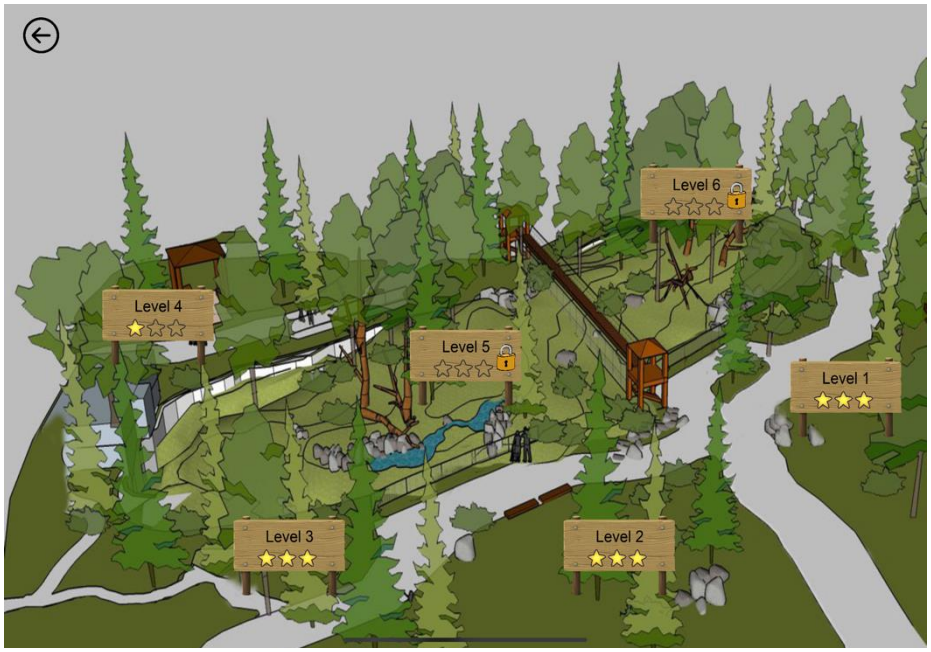

Figure 19. Game Level Scene

6.5 Game Scene

6.5.1 Pause/Resume/Replay the Game

Users are allowed to pause, resume and replay the game. For each level, the Pause button and the Replay button are on the top right corner of the screen. Clicking the Pause button will pause the game. Then the Pause button becomes the Resume button as Figure 20 shows. We add a flashing effect to the Resume button to remind users of the status of the game. In order to create the effect that the Resume node is flashing, we run a sequence of the actions on the Resume node to change the transparency of the resume node. When the game is paused, all the buttons are deactivated so that users need to resume the game first by clicking the Resume button. Once the game is resumed, the Resume button will change back to the Pause button. When users finish the game or they want to restart the game, the Replay button will allow users to replay this level. An alert window pops out and asks users to make sure whether they want to restart the game as Figure 21 shows. The game is paused at this time for users to make a decision. Clicking the Cancel button in the alert window will continue the game while the OK button restarts the game.

6.5.2 Help Information

We provide full instructions on how to play the game and how to complete the challenges. A question mark node on the top left corner of the screen offers help information to users. As Figure 22 shows, after users click the question node, the game is paused and a white board with text shows up. Users can touch the text and scroll up and down to read through instructions. The white board will disappear when users touch the background of the screen or click the question node again.

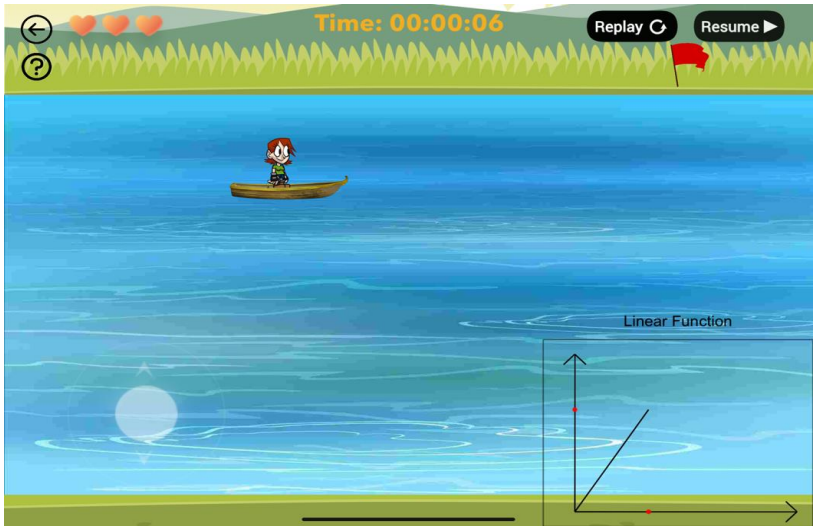

Figure 20. Clicking the Pause Button in Game Level 1

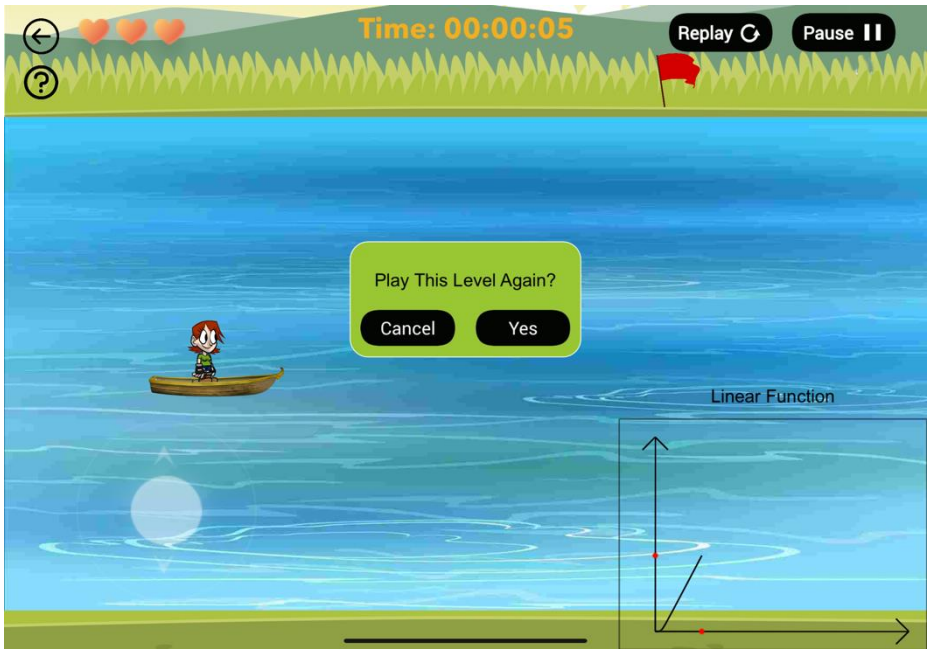

Figure 21. Clicking the Replay Button in Game Level 1

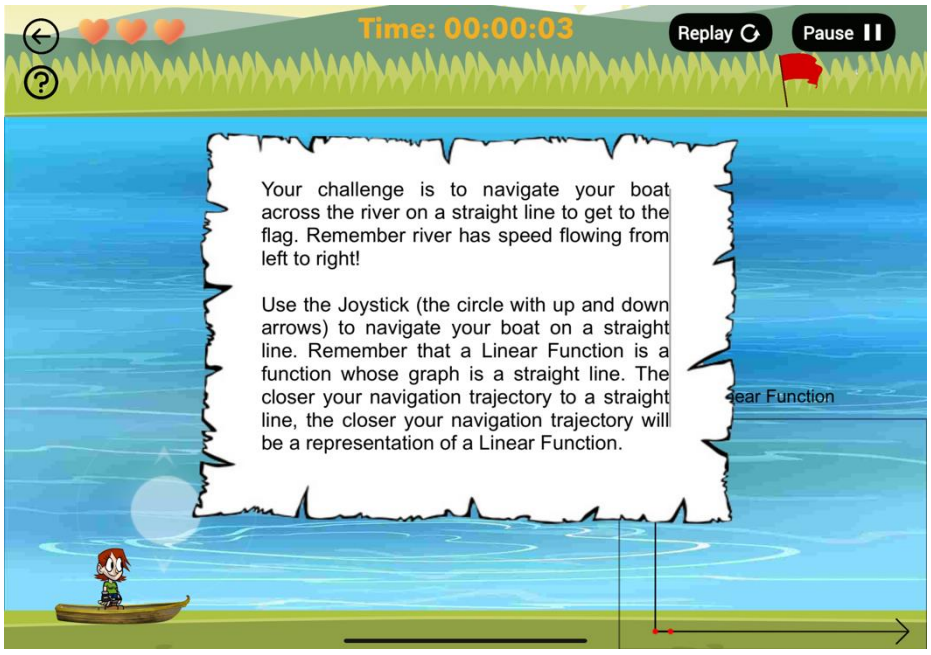

Figure 22. The Help Information in Game Level 1

6.5.3 Time and Life System

On the top of the screen, a time label records the amount of the time that users have spent on the game. The time label is in the format of "HH:mm:ss" as Figure 22 shows. When the game pauses, the time label stops counting. Figure 23 shows that we use the scheduleTimer function to achieve the time counting functionality. The scheduleTimer creates a timer and schedules it on the current run loop in the default mode. The timer calls the updateTimer function once per second. In the updateTimer function, we update the text for the time label.

```
timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector:
    #selector(updateTimer), userInfo: nil, repeats: true)

@objc func updateTimer() {
    timerCount += 1
    let timeString = String(format: "%02d:%02d:%02d", timerCount/3600,
        (timerCount/60)%60, timerCount%60)
    timeCounterLabel.text = "Time: \(timeString)"
}
```

Figure 23. Update Timer

We also build the life system for each level. The player has three chances to complete the challenge. We use the heart icons on the top left of the screen to represent the lives. Players will lose a heart when they fail to accomplish the challenge. We add the flashing effect on those hearts to remind users that they lost one life.

6.5.4 Game Level 1: Linear Functions

Figure 24 shows the game scene for game level 1. The bottom left corner is the joystick that controls the speed of boat. Users can press the joystick knob and drag it up and down to change the vertical speed of the boat. The movement of the wood on the screen indicates the river speed. The bottom right corner of the screen shows the trajectory of the boat, which is a linear function graph. The moving speed of the red dot on y-axis represents the boat speed that is controlled by the users using the joystick. The moving speed of the red dot on x-axis represents the river speed. The challenge of this game level is to use the joystick to navigate the boat with respect to the river speed and reach the flag.

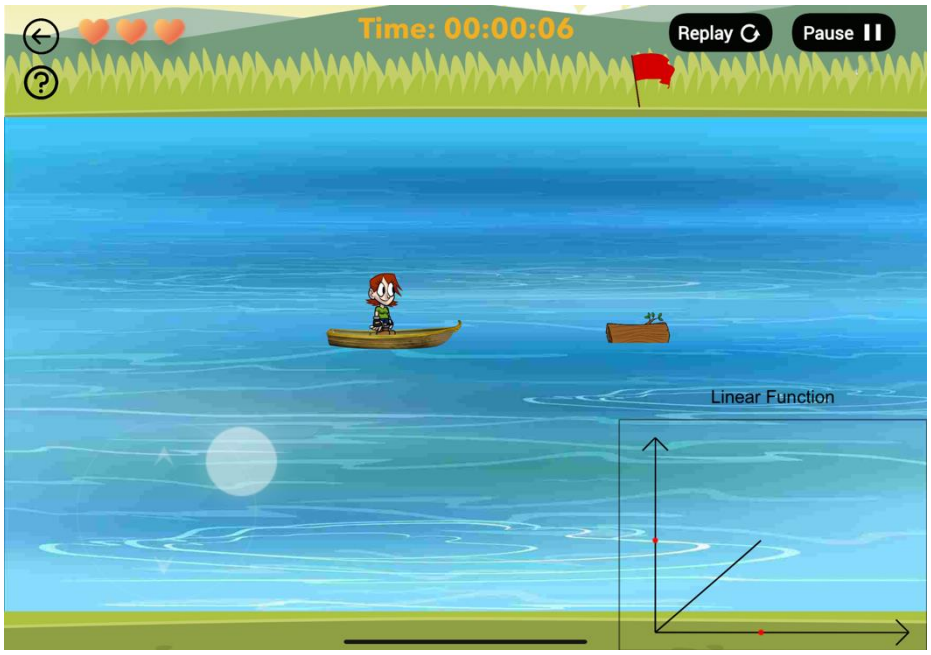

Figure 24. Game Level 1

6.5.5 Game Level 2: Quadratic Functions

Figure 25 shows the game scene for game level 2. At the beginning of the game, the character rides the bike on the left small hill and prepares to jump over the water according to the chosen direction and speed. We randomly set the slope of the small hill between 10 to 36 degrees. The left hill and the right hill are mirror symmetric. On the bottom of the screen, an arrow node rotates from facing right to up and a speed energy bar increases the energy from left to right. Users need to choose the direction and the speed of the bike to jump over the water and land in the safe area. The red line on the right side of the ground represents the safe area. Because of the initial speed and the effect of the gravity, the trajectory of the bike is presented as a quadratic function graph. We draw the trajectory on the bottom right corner coordinate. The two red dots on the x and y axis show how quantitative values of two variables change with each other. Failing to land in the safe area by jumping out of the right side of the screen or falling into the water loses one heart. Figure 25 shows the situation where the player has lost one life.

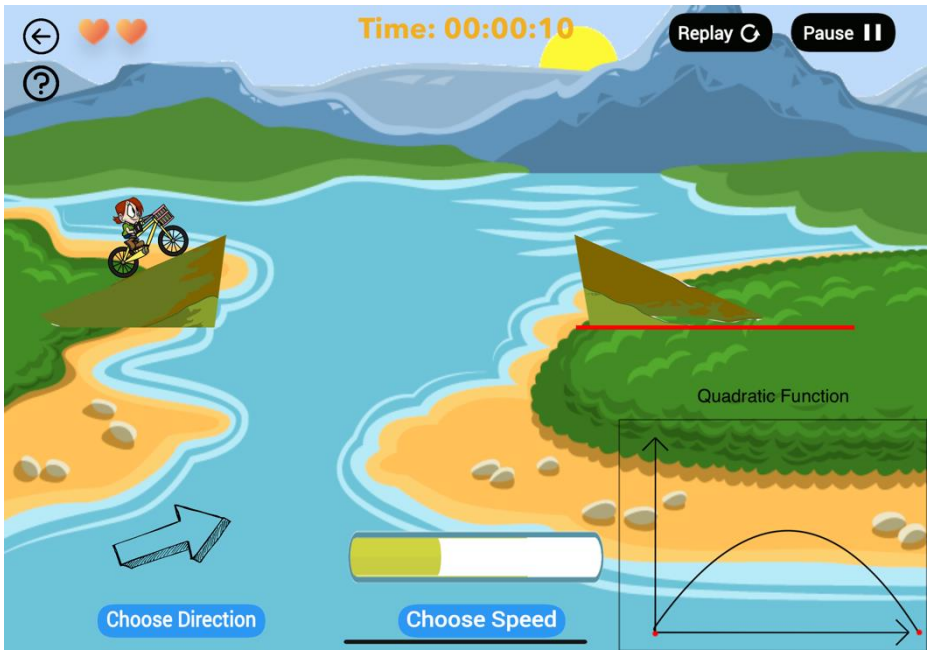

Figure 25. Game Level 2

6.5.6 Game Level 3: Exponential Functions

Figure 26 shows the game scene for game level 3. This game level is made of the 10 ground platforms. The position of the platforms is in the shape of the exponential function graph. We randomly create the exponential function with the base number between 1.6 to 2.8. Users need to move the joystick left and right to control the character to move left and right while at the same time, tap the screen to jump between platforms. The challenge is to jump from the first platform to the last one and reach the flag as Figure 26 shows. Falling down or touching the fire will lose one heart. The learning objective of this game level is to let students experience the growth of the exponential function while playing the game. The exponential function graph on the bottom right corner corresponds to the position of the platforms. In the game, the 10 dots represent the 10 platforms. In the function equation, the 10 dots represent x values from -4 to 5. When users land on a new platform, the color of the dot changes from black to red to tell users which platform the game character is standing on and how many platforms are left. The game only shows the next one platform, so users need to jump over platforms one by one. It avoids the situation that users skip platforms and jump to the last platform directly. The rapid growth of the exponential function graph causes the huge vertical difference between two platforms. When two platforms can not be drawn in the same screen and users can not see where the next platform is, they will feel lost while jumping from one platform to another. We add the guide arrows to let users know the direction of the next platform. By following the green arrows shown in Figure 26, users can easily land on the next platform.

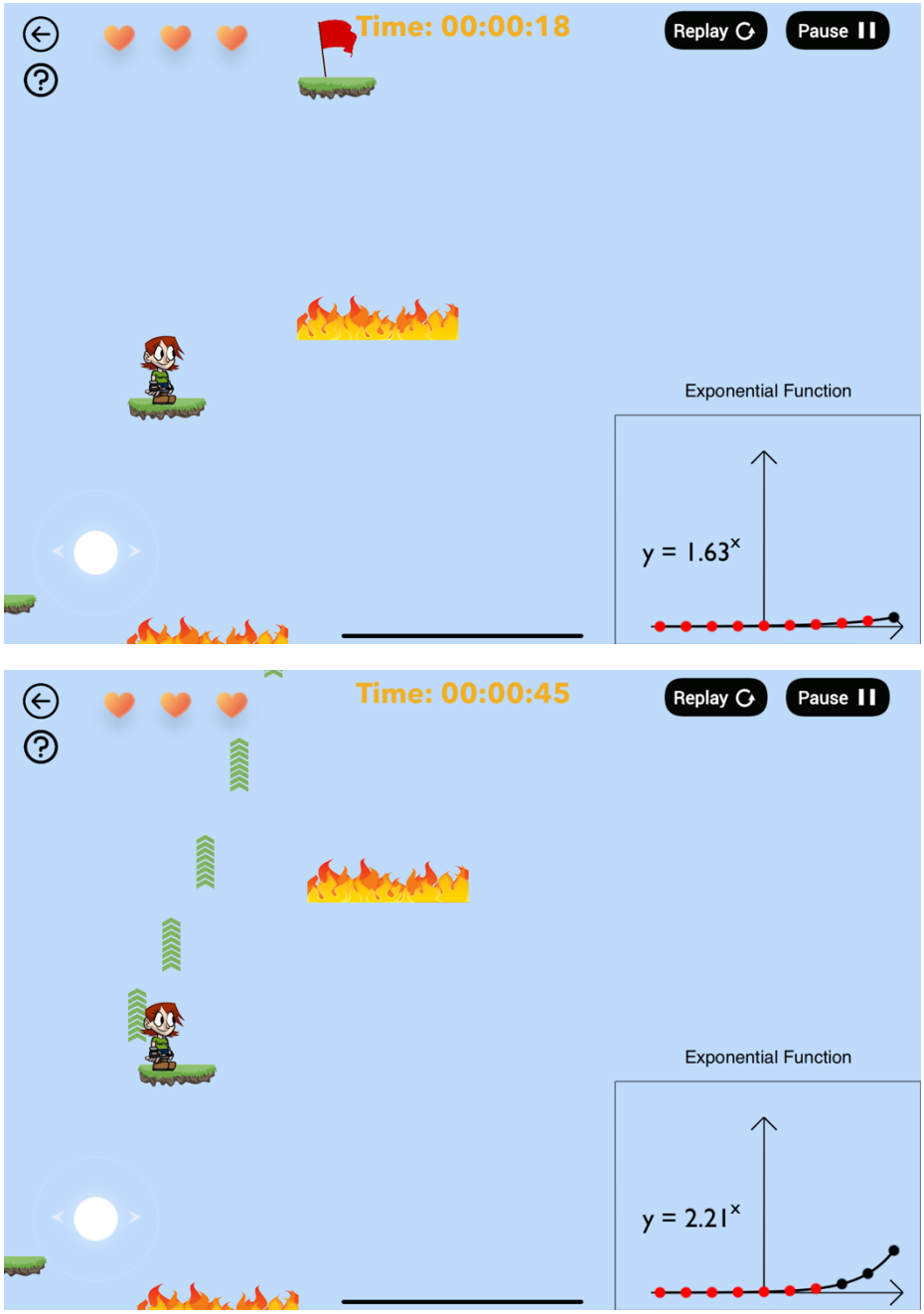

Figure 26. Game Level 3

6.5.7 Game Level 4: Logarithm Functions

Figure 27 shows the game scene for game level 4. In this level, the game character needs to use the springboard to jump over the cliff. In order to activate the springboard, users must adjust the springboard shape to match the given logarithm function equation below the time label. We

randomly create the given logarithm function. One characteristic of logarithm function graphs is that they all pass through the point (1, 0) in the coordinate plane. So we use that point as the supporting point for the springboard. We also draw grids on the coordinate plane as background. Users can use the grids to shape the springboard by substituting (x, y) values. This level combines the changing values of two variables with the function graph and stimulates students to learn function graph with quantitative thinking. Users can tap the game character node and drag it up and down to reshape the springboard to represent the graph of the given logarithm function. Users can also click the Finish button to check whether the shape of the springboard matches the given logarithm function graph. If the shape of the springboard matches the logarithm function, the game character will perform an animation to jump over the cliff. Otherwise, users lose one heart.

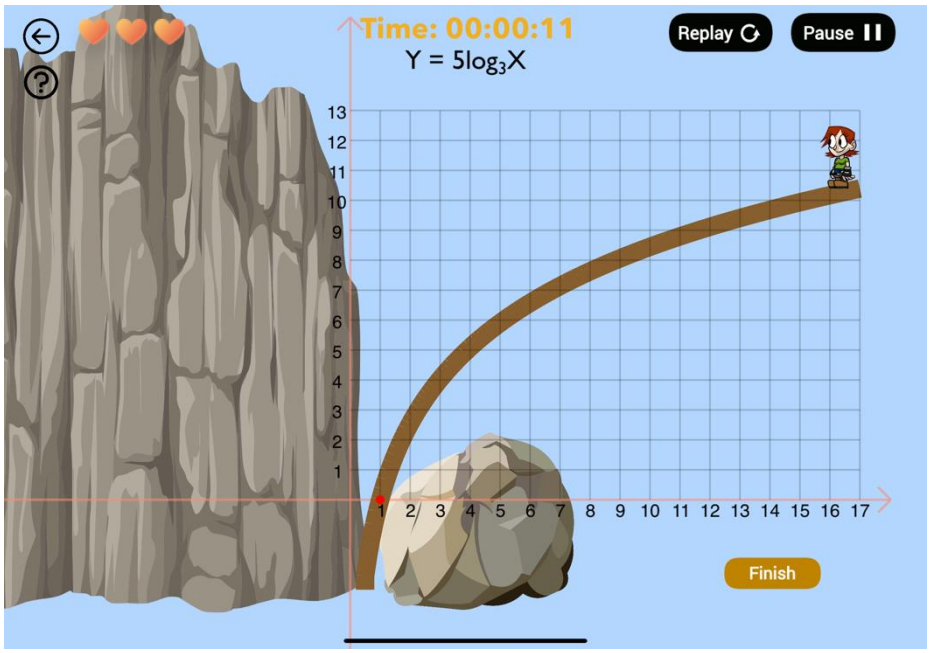

Figure 27. Game Level 4

6.5.8 Game Level 5: Trigonometric Functions

Figure 28 shows the game scene for game level 5. The formula of a sine function can be written as $y = A\sin(\omega x + \varphi)$. Parameter A represents the Amplitude of the sine function, ω influences the Period of the sine function and φ is the Phase Shift. We create the function by randomly choosing the value for these three parameters. The challenge of the game level is to reproduce the given trigonometric function graph below the time label using the water wheel. We combine these three parameters with the water wheel size, the graph period and the position where the graph begins. Users can interact with the two sliders, one arrow node, and a GO button on the bottom of the screen to create the trigonometric graph. Move the red button left or right to make the water wheel smaller or larger. Tap the arrow button to set the start position. The arrow rotates 90 degrees counterclockwise for each tap. Move the green button left or right to set the period of

the graph; the range of the slider is from π to 2π. Clicking the GO button starts drawing the graph. As Figure 28 shows, after users click the GO button, the water wheel rotates counterclockwise from the start position. In the coordinate, it generates the graph in black color and with red dots. After drawing the graph, it compares the distance between 5 black dots and 5 red dots. If the absolute distance is less than 20, users complete the challenge, the trigonometric graph flashes several times and the water wheel disappears. Otherwise, users lose a heart.

6.5.9 Game Level 6: Polynomial Functions

Figure 29 shows the game scene for game level 6. The challenge of this game level is to design a roller coaster according to the given polynomial function by identifying its roots. An intersection point on the x axis identifies a root. As Figure 29 shows, the highest power of the polynomial function $y = x^4 - 4x^3 - 3x^2 + 18$ is 4, which determines the number of function roots. Users can easily find the roots from the intersection point on x axis. But in order to determine the multiplicity of a root, users should look at the function graph carefully, substitute some x values into the expression, compute the corresponding y values and check with the graph. In the process of finding all roots for the polynomial function, students are expected to understand the quantitative relationship between x, y variables and how the graph looks when choosing roots.

The given polynomial function is under the time label and the function graph shows in the center of the screen. Eight root buttons are listed on the bottom left of the screen. Users should choose the correct roots according to the graph. By clicking the root button, it adds the root to the equation label on the bottom of the screen. If users tap the root button by mistake, the Clear button will delete all the roots after the equation and restart again. Clicking the Check button checks the roots with the function. If users select all the roots correctly, the game will perform the animation of the game character riding the roller coaster. Otherwise, users lose a heart. When running the animation, we will only show the roller coaster by removing all buttons and the equation label.

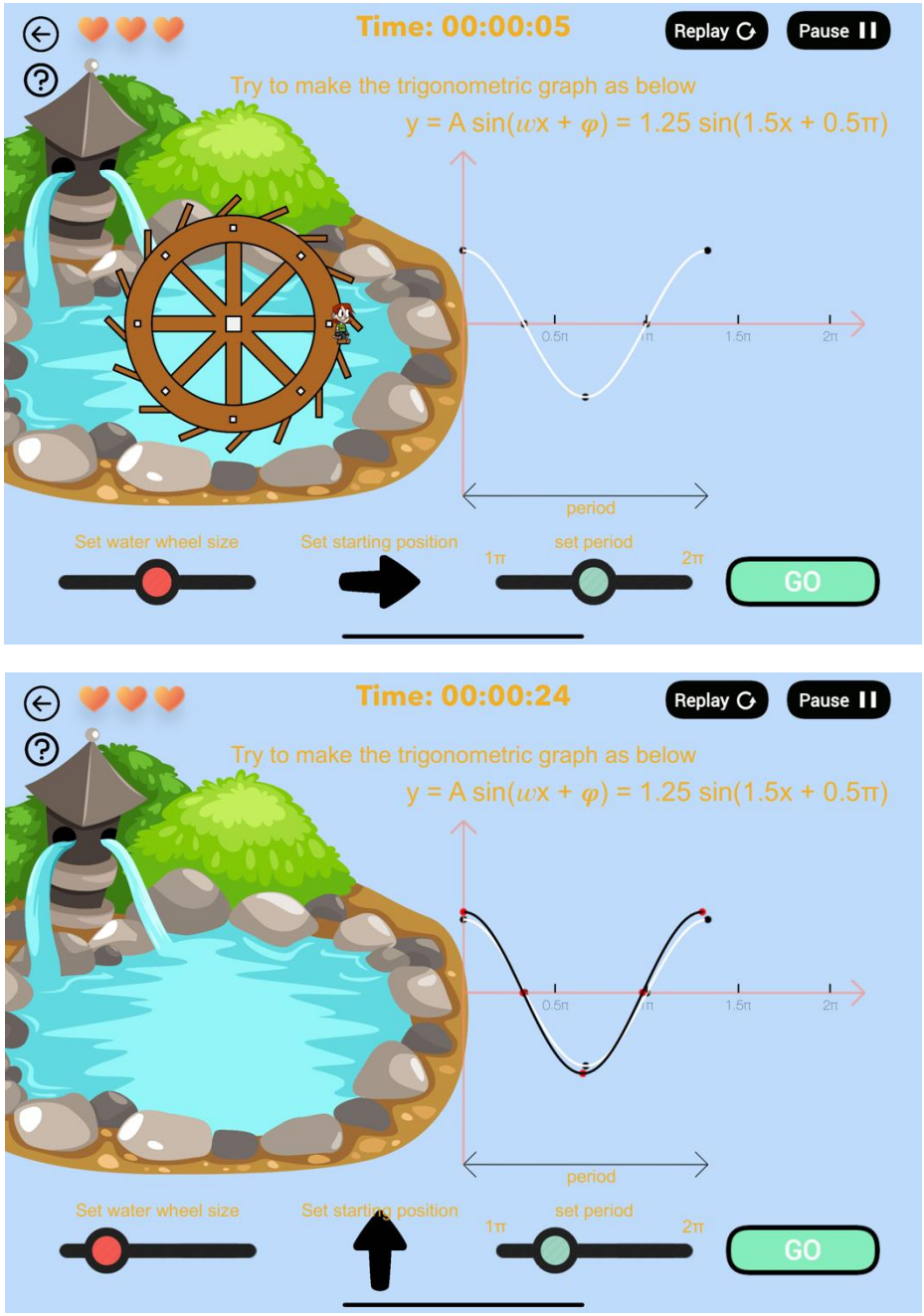

Figure 28. Game Level 5

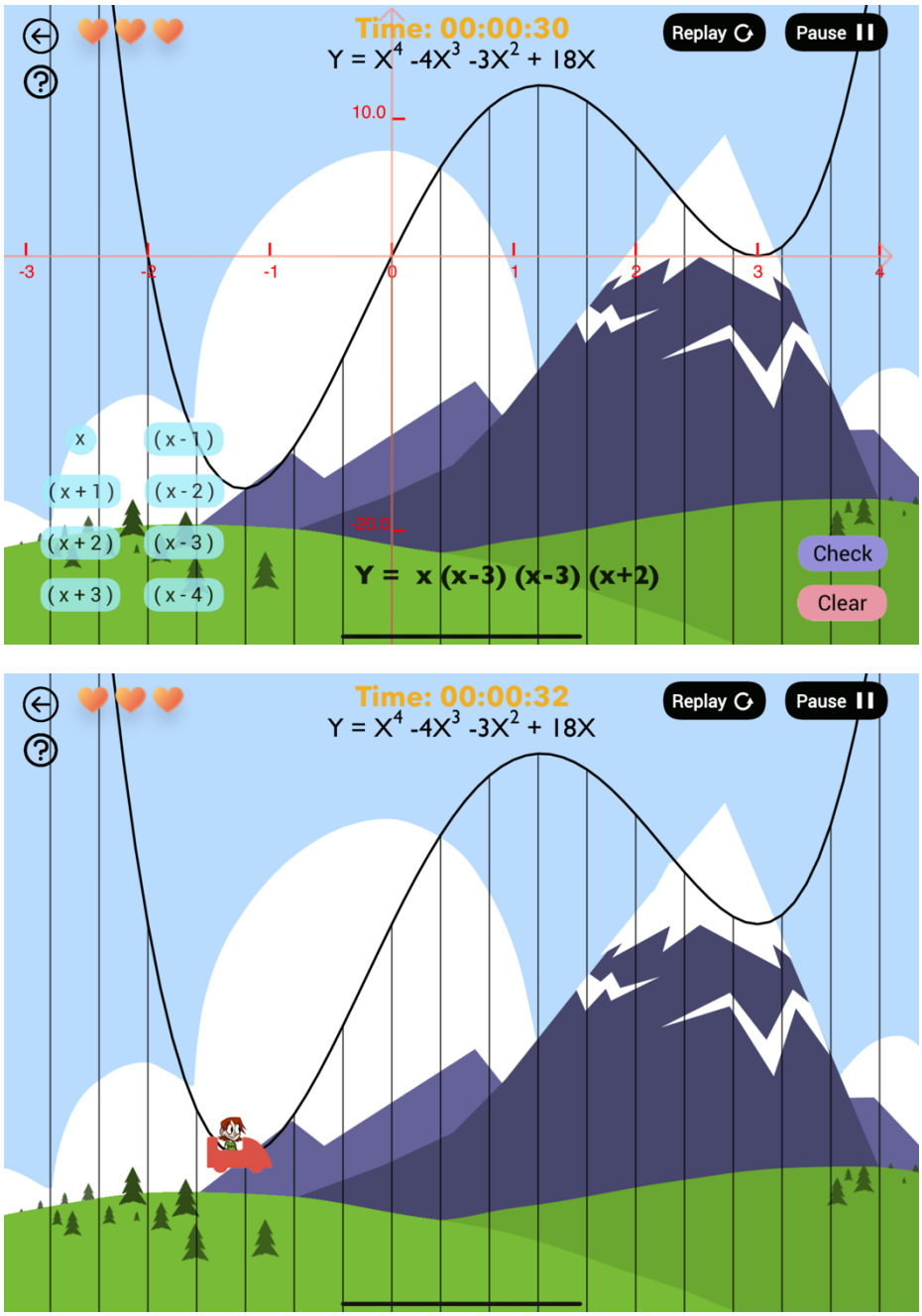

Figure 29. Game Level 6

Chapter 7: Player Performance Evaluation

In this chapter, we explain the two consequences after each gameplay: completing the game challenges or failing the game. We also introduce the reward system that is used to evaluate the performance of players.

7.1 Game Results

Function Fun provides immediate feedback to players after each attempt. According to the life system, players have three chances to play the game in each level. The condition of success is to accomplish the challenge within three attempts. If players complete the game, *Functions Fun* plays the applaud sound to celebrate and shows the congratulation sign in a pop-out window as Figure 30 shows. The congratulation window conveys the information of players' performance. The stars indicate how many attempts players use to finish the game. It also shows the badges players got from this game play. The screenshot in Figure 30 shows that the player finishes the game on the first attempt and gets two badges at this time for game level 5. Clicking the "Got it!" button closes the congratulations window. Players can then check the graph they generated in the game and decide to replay the current level or go back to the game level scene by clicking the buttons on the top of the screen.

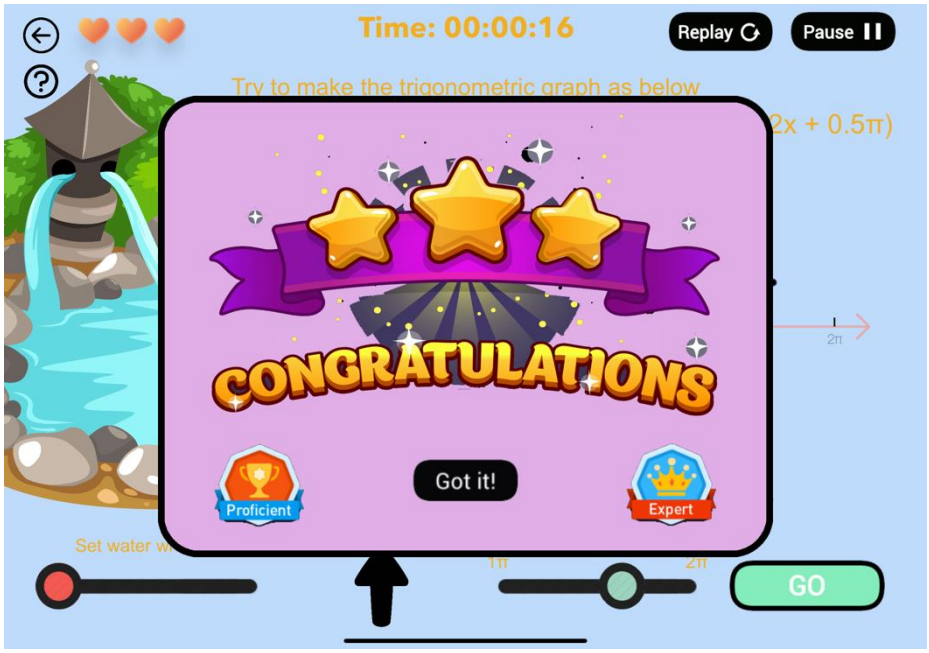

Figure 30. Congratulation Window for Game Level 5 with 3 Stars.

In game level 5, if players fail to generate the given trigonometric graph for 3 times, the game scene switches to the game over scene. The game over scene indicates the failure of the adventure for current game level. The game over scene does not have the back button to go back

to the game. Players can only choose to go back to the game level scene or replay the current game level by clicking the buttons on the game over scene.

Figure 31. Game Over Scene

7.2 Evaluation System: Obtain Achievement Badges

Functions Fun uses the time and life system to evaluate the performance of players. If players finish the game on the first try, they will get 3 stars for that game level. The remaining hearts decide how many stars players will get. Players still succeed in the game level when they only get 1 star. The next level will be unlocked only when players get three stars in the previous level. So the more stars that players get from the game, the better they perform. In *Functions Fun*, we also use 4 badges: Beginner Badge, Competent Badge, Proficient Badge and the Expert Badge to evaluate how they perform in the game. The Beginner badge can be obtained when players get one star in the game. Getting two stars obtains the Competent badge. Getting three stars obtains the Proficient badge. Only when players finish the game within a certain time, they can get the Expert badge. According to the game design of each game level, the time of getting the Expert badge for the 6 levels are: 20 seconds, 5 seconds, 30 seconds, 20 seconds, 20 seconds, and 40 seconds. The less time players spend on completing the game challenge, the more familiar players become with the function graphs.

All achievement records are saved in a property list. In Swift, the property list, called *plist*, is an XML file that contains key-value data. The top level element of the *plist* file is the root, which is a dictionary. Under the top level, we create six dictionaries for six game levels. We need to store the related game records for the player and update the record information for each game play. In

a normal game, the system usually records the highest scores for the player. *Functions Fun* records the maximum number of stars players got and the least time players used, along with the information of achievement badges for each level. Figure 32 shows the initialized records for game level 1 in the *plist* file. After each game play, we update these key-value pairs and rewrite them into *plist* file. Every time we need these information, we can read them from the *plist* file.

Key	Type	Value
▼ Root	Dictionary	(7 items)
unlock	String	false
▼ level1	Dictionary	(6 items)
highestStar	Number	0
timeCount	Number	100,000
beginnerBadge	Number	0
computentBadge	Number	0
proficientBadge	Number	0
expertBadge	Number	0
▶ level2	Dictionary	(6 items)
▶ level3	Dictionary	(6 items)
▶ level4	Dictionary	(6 items)
▶ level5	Dictionary	(6 items)
▶ level6	Dictionary	(6 items)

Figure 32. The Initialized Plist file for Storing the Players' Records

Chapter 8: Functions Fun Self-Evaluation

This chapter provides a self-evaluation of *Functions Fun*. The self-evaluation is conducted to review the quality of the game app based on the game UI design, the game functionality design, the game interaction design. As developers, our goal is to develop the quality guaranteed application for users. DEG Quality Assurance (QA) provides the planned and systematic activities that are established throughout the DEG life cycle to make sure that the DEG software application possesses certain characteristics required for a set of intended uses of the DEG [Aslan and Balci 2015]. In the design stage, we follow the DEG life cycle to develop the game. In this chapter, we will discuss the quality of *Functions Fun* based on the below 12 quality indicators provided from the DEG development methodology [Aslan and Balci 2015].

8.1 Acceptability

Acceptability is the degree to which the DEG fulfills its requirements and learning objectives [Aslan and Balci 2015]. The goal of developing *Functions Fun* is to help secondary students to form the quantitative thinking about the function graphs. According to the learning objectives, we design the game scene for each game level. During the game, players control the movement of the character and we use the trajectory of the character to represent the function graph. The interface on the bottom right corner of the screen draws the function graph in the coordinate plane while the character is moving. Combining the game and the generation of the function graph provides students an interesting and straightforward way to understand how the value change of two variables influences the graph. By playing *Functions Fun*, students may not just remember the shape of the graph using perceptual cues in the process of learning functions, they will gain quantitative understanding of function graphs through animation.

8.2 Challengeability

Challengeability is the degree to which the user finds the DEG to be exciting, stimulating, and inspiring to play [Aslan and Balci 2015]. We conduct six game levels for players to enjoy their adventures in the jungle. Each game level teaches one kind of function graph with increasing difficulties. For example, the first level is to learn linear functions and the last level is for polynomial functions. In this way, players learn functions from easy to hard and gain confidence and encouragement from playing the game. In the game, players are asked to accomplish a specific challenge for that level. Only when players get 3 stars in the game can they unlock the next level. People usually get excited and curious about the unknown things. The strategy we employed here is to stimulate players to continue playing the game until they get 3 stars and unlock the new level.

8.3 Clarity

Clarity is the degree to which the DEG is unambiguous and understandable [Aslan and Balci 2015]. To make the game app understandable and easy to use, we must tell players the rules of the game, including the goal of the game and the instruction of how to play the game. *Functions*

Fun is an adventure game under a jungle theme. The task is very simple: players need to accomplish all the challenges during the adventure trip. Before playing the game, the About scene introduces the basic background information about *Functions Fun* to players. In the game, we provide the help information for players including what is the challenge for each level and how to play the game. Since we explain the game clearly, players will not feel confused about where to start and what to do in the game.

8.4 Effectiveness

Effectiveness is the degree to which the DEG improves the effectiveness of learning the subject in a significantly better way in comparison to other pedagogies [Aslan and Balci 2015]. Compared to the traditional teaching method, the game is a special pedagogy in the education field and stimulates the students' interests in learning. *Functions Fun* teaches students function graphs in an entertaining and relaxing way by playing games. In the game, students take initiative in function graph learning instead of acquiring knowledge from teachers. On the other hand, *Functions Fun* applies 2D animations to draw function graphs with the co-varying variables x and y. It overcomes the limitation of static graph images that are used in traditional education. Although *Functions Fun* is a game app, it can become a useful and effective tool for learning function graphs.

8.5 Engageability

Engageability is the degree to which the user is captivated and addicted by DEG playing [Aslan and Balci 2015]. *Functions Fun* has six different game levels for six different functions. Each game level is under a different game scene with different background images. Players will not stay in the same scene and play the same game for 6 levels with different difficulties. Each game level is designed as a game for players, which attracts players to spend more time on the game. Players can only play the first level at the beginning; the rest of game levels are locked. Only when they got three stars, the next level can be unlocked. Driven by curiosity, players will be addicted to play the next level.

8.6 Enjoyability

Enjoyability is the degree to which the user finds the DEG playing to be fun [Aslan and Balci 2015]. We will discuss the enjoyability of *Functions Fun* in three aspects: the game story and genre, the gameplay design and the learning objectives. First, *Functions Fun* is an adventure game under the jungle theme. The adventure game makes players feel excited and curious while the jungle theme is very fresh and attractive. As for the gameplay design, *Functions Fun* has rational difficulty management and reward system, which makes it an enjoyable game. For each game level, we gradually increase the game difficulty. Players will not feel the game is too hard to play and will not get bored. Obtaining achievement badges after players accomplish the challenge encourages players to continue learning and playing. While playing games, players can learn function graphs and form quantitative thinking of graphs. Learning knowledge from games is a fun activity for secondary students compared to traditional learning methods.

8.7 Interactivity

Interactivity is the degree to which the user actively interacts with the DEG during playing [Aslan and Balci 2015]. Good interaction design effectively communicates the functionality of the application with users, defines behaviors that response to users, informs users about the state changes in game scene, and prevents game error. In the game scene, the pause button, the replay button, the help button and the back button allows players to pause the game, replay the game, receive help from the game instruction and go back to the Game Level Scene. *Functions Fun* provides immediate feedback to players. During the game, *Functions Fun* updates the position changes of the character on the scene to interact with players. An alert window with a warning message pops out to prevent users from tapping buttons by mistakes.

8.8 Localizability

Localizability is the degree to which the DEG can easily be adopted, preferably via preferences or options, (a) to satisfy the needs of languages other than English, and (b) to local standards such as decimal separator, currency symbol, time zone, calendar, etc. [Aslan and Balci 2015]. *Functions Fun* only supports English as the language for the application. In terms of local standards, in game level 3, we support a maximum of two decimal places for the base number in exponential function equation. The time system of *Functions Fun* only displays the time that players spent on the scene. *Functions Fun* does not support displaying the current time, the date and different currency symbols.

8.9 Rewardability

Rewardability is the degree to which the DEG gives rewards (e.g., points, money, trophies, certificates) to the user so that the user feels a sense of accomplishment [Aslan and Balci 2015]. *Functions Fun* has a reward system to award players for accomplishing the challenges. We use four different achievement badges, beginner, competent, proficient, expert, as rewards to encourage players to play the game and learn function graphs through games. According to the different performance, players will obtain different badges as reward. *Functions Fun* has the Achievement Scene to display all the badges players obtained from the six game levels.

8.10 Simplicity

Simplicity is the degree to which the DEG can be understood without difficulty [Aslan and Balci 2015]. UI design of *Functions Fun* is very easy to understand and to interact with. *Functions Fun* uses meaningful names for buttons and game nodes in different game scenes so that players can know the purpose of each scene without confusion. Likewise, *Functions Fun* simplifies the user interaction to its main components with easy to use and intuitive interfaces. As for the game experience, the game rules and the level challenges are simple. Players can check the instructions by clicking the question button in the game scene.

8.11 Transformativeness

Transformativeness is the degree to which the DEG transforms the subject learning in a significantly better way in comparison to other pedagogies [Aslan and Balci 2015]. In the game design stage, we build real-life models for six functions and according to these models, we design the game scene. We transform the learning of function graphs into solving real-life problems. Using real-life examples for game design increases the positive affect and the efficiency in learning process since these examples are related to players' experience. Players will be interested in playing the game and pay more attention to the animation of function graphs compared to just learning function graph images.

8.12 Usability

Usability is the degree to which the DEG can easily be employed for its intended use [Aslan and Balci 2015]. We will discuss the usability of the application from two aspects. From the user interaction aspect, the structure of *Functions Fun* allows players to easily navigate between different scenes. The user interface works properly on all iPad versions. Users can interact with interfaces fluently and get timely responses. From the functionality aspect, the 2D animation of drawing function graphs helps players to have a better understanding about the quantitative value changes of x and y variables in functions. Players can obtain the appropriate awards according to their performance.

Chapter 9: Conclusions and Future Work

9.1 Conclusions

This thesis proposes a game-based application *Functions Fun*, aimed to help secondary students and teachers engage with function graphs. According to the research studies, a large number of students use their way to learn function graphs by remembering the graph shape without reasoning about the quantitative changes about x and y variables. *Functions Fun* attempts to help students to form quantitative thinking in the process of learning about graphs. In the game, 2D animations and game effects allow students to explore how quantity changes influence the graph within the coordinate system. In this way, *Functions Fun* serves as a teaching tool in mathematics education. Compared to other teaching tools, *Functions Fun* makes learning more enjoyable to students, increases the learning efficiency, and breaks the limitations of traditional teaching media.

We modularize and structure the development of *Functions Fun* according to the DEG life cycle. The DEG life cycle is the foundation of game development and provides a framework for developers to produce game applications in a planned and systematic way. The whole development process is divided into four stages: problem generation, game idea generation, game design and implementation, and self-evaluation. We also apply the Evolutionary and Incremental development principles to our game design. With more experience and insight obtained, we developed the game in increasing levels of detail and identified the increments with different versions.

Transforming learning objectives into games is the most time-consuming challenge for game development. It requires imagination, creativity, and rich experience with different games. In the game idea generation stage, we discovered and discussed the advantages of using real-life models to generate game ideas, found real-life models to represent six kinds of function graphs and, in turn, these models inspired the design of the game challenges.

Functions Fun is designed to operate on all iPad devices. We guaranteed that the implementation of the user interface is compatible for different screen sizes by testing on different simulators. We also conducted playtesting and risk analysis for UI design and interaction design. Then we collected suggestions from different groups of people to improve the unreasonable design drawbacks and make the application more user-friendly. After using 12 quality indicators to perform the self-evaluation of the application, we believe *Functions Fun* can serve as an interesting teaching tool for learning function graphs in mathematics education.

Functions Fun is not only a useful tool for teachers and secondary students in the education field, but is also an interesting project in software engineering. Our work is an application of the DEG development methodology, which can serve as a resource on game development for other developers. Using real-life models to transform graph learning into games can also provide inspiration for developers who are interested in DEG development.

9.2 Future Work

In the development process, we focus more on how to achieve the functionality and fulfill the learning objectives. While we spend a large amount of time on game idea generation, there are several other areas that can be improved to make *Functions Fun* an attractive game. The current version of *Functions Fun* completes the basic requirements and functionality as a teaching tool. In the future, we are planning to increase the game content, beautify the user interface to improve user engagement, and make the game more fun to users. Managing the challenge difficulties properly and establishing a perfect reward system are game strategies we can use to make users feel more excited and encouraged while playing the game.

The game design of the *Functions Fun* educational app could be improved with the following:

1. Improve the artwork for the game, including the interface beautification, and designing good-looking images for background and game nodes.
2. Add more tasks within the game level to increase the game operation difficulties. For example, add barriers in game level 3 and add gold coins for game level 2 to encourage players to get more money for higher scores.
3. Improve the evaluation and reward system. We can add a score system to evaluate the players' performance and put in more badges for rewards.
4. Design other game modes. Develop the endless mode to combine all six functions together. Randomly set the challenges to increase the game experience for players and check the learning results after playing the game.

In the future, we want to build a system for *Functions Fun* to support multi-players to play the same game together in either competition or collaboration mode. We also want to build the user account system to keep track of game records for all players and to analyze the learning results from playing *Functions Fun*.

REFERENCES

Ang, K.C. (2001), "Teaching Mathematical Modelling in Singapore Schools," *The Mathematics Educator 6*, 1, 63-75.

Apple (2019a), "Apple Developer Document: About Collisions and Contact" https://developer.apple.com/documentation/spritekit/skphysicsbody/about_collisions_and_contacts

Apple (2019b), "Apple Developer Document: SKPhysicsWorld" https://developer.apple.com/documentation/spritekit/skphysicsworld

Apple (2019c), "Xcode" https://developer.apple.com/xcode/ide/

Aslan, S. and O. Balci (2015), "GAMED: Digital Educational Game Development Methodology," *Simulation: Transactions of the Society for Modeling and Simulation International 91*, 4 (Apr.), 307-319, doi: 10.1177/0037549715572673.

Boehm, B. (1986), "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Engineering Notes 11*, 4, 14-24.

Gainsburg, J. (2008), "Real-World Connections in Secondary Mathematics Teaching," *Journal of Mathematics Teacher Education 11*, 3, 199-219.

GeoGebra (2018), "Geogebra," https://www.geogebra.org/

Gravemeijer, K., and M. Doorman (1999), "Context Problems in Realistic Mathematics Education: A Calculus Course as an Example," *Educational studies in mathematics 39*, (1-3), 111-129.

MathCentre (2009), "Polynomial Functions" http://www.mathcentre.ac.uk/resources/uploaded/mc-ty-polynomial-2009-1.pdf

Mathdemos (2010), "Sinusoids: Applications and Modeling" http://www.mathdemos.org/mathdemos/sinusoidapp/sinusoidapp.html

MathisFun (2017), "Real-world Examples of Quadratic Equations" https://www.mathsisfun.com/algebra/quadratic-equation-real-world.html

MathisFun (2018), "Amplitude, Period, Phase Shift and Frequency" https://www.mathsisfun.com/algebra/amplitude-period-frequency-phase-shift.html

Moore, K. C., T. Paoletti, and S. Musgrave (2013), "Covariational reasoning and invariance among coordinate systems," *The Journal of Mathematical Behavior 32*, 3, 461-473.

Moore, K.C. and P.W. Thompson (2015), "Shape thinking and students' graphing activity," In *Proceedings of the 18th Meeting of the MAA Special Interest Group on Research in Undergraduate Mathematics Education* (RUME Pittsburgh, PA, 2015). pp. 959-965.

Pierce, R.U., and K.C. Stacey (2006), "Enhancing the image of mathematics by association with simple pleasures from real-world contexts," *ZDM 38*, 3, 214-225.

Pizza Fractions 1 (2015), “App Store Preview for Pizza Fractions 1,” https://itunes.apple.com/us/app/pizza-fractions-1/id374084320?mt=8

Pressman R. S. and B. R. Maxim (2015), *Software Engineering: A Practitioner's Approach*, 8th Edition, McGraw-Hill Education, New York, NY.

Sommerville, I. (2011), *Software Engineering*, 9th Edition, Addison-Wesley / Pearson Education, Boston, MA.

Stein, M. K., J.A. Baxter, and G. Leinhardt (1990), “Subject-matter knowledge and elementary instruction: A case from functions and graphing,” *American Educational Research Journal 27*, 4, 639-663.

Swift.org (2019), “About Swift” https://swift.org/

Thompson, P.W. (1994), “Images of Rate and Operational Understanding of the Fundamental Theorem of Calculus,” *Educational Studies in Mathematics 26*, (2-3), 229–274.

Ying, T., O. Balci, and A. Norton (2019), “CandyFactory Educational Game,” https://shark.cs.vt.edu/CandyFactory