Tuffy++

Group ID:
1

Group Members:
David Tu - david.tu2@csu.fullerton.edu
Douglas Galm - douglasgalm@csu.fullerton.edu
Johnson Lien - johnsonlien95@csu.fullerton.edu
Patrick Hovsepian (Scrum Master) - phovsepian@csu.fullerton.edu
Susmitha Chittem - susmitha.chittem@csu.fullerton.edu
Thao Nguyen - tnguyen29@csu.fullerton.edu

Class:
CPSC 362-01
TuTh 6:00PM - 7:50PM

Professor:
Sara Ghadami

Submission date:
12/08/2016

Revision History

Name	Date	Reason For Changes	Version
Johnson Lien	11/17/16	Second Iteration. Updating information	
Johnson Lien	12/4/16	Third Iteration. Additional information.	

Table of Contents

Vision and Scope

Business Requirements

Our requirements, as of now, are to implement education into Tuffy++. More specifically, the customer wants this game to be more on the educational side rather than just pure entertainment. To do this, we will first be using C++, a popular and imperative programming language, as a topic of interest to base our educational questions on.

Background

Until now, there have not been many games dedicated to learning C++. This program will give the users a fun and interactive platform to learn the ins and outs of C++ and object-oriented programming in general. We hope to be able to popularize the concepts of programming to children and young adults alike while having the option of being challenging enough for veterans to brush up on their skills.

Business Opportunity

This could be a different method to teach new programmers some neat tips to get started in learning C++. The main objective of our game is to make the players enjoy a simple game, and at the same time, learn a programming language. This game is designed for simplicity to attract a younger audience to get them started at learning a new language.

Some of the obstacles we are facing have to do with how exactly will the programming information be implemented. For example, how difficult should we make the questions? Should there be any actual code involved? When and where will the programming information be given? These questions can only be answered as the project progresses and we can see where the information will be appropriate. Some ideas for the solution to those questions, for now, would be to have the user answering multiple choice questions to advance further into the game. As the user progresses, the harder the question is, the more intermediate the information will be.

Business Objectives and Success Criteria

Success will be measured by how many downloads this game will receive. As of now, the game will be completely free. That, in itself, is a selling point. Also, we aim to release Tuffy++ for free on the Google Play Store, making it available to millions of Android users.

Customer or Market Needs

There are two main Customer Market needs this product will address:

1. **Create an educational application which teaches basic computer science and programming, that is easy to comprehend and easy to use:** The current mobile market does not offer simple and fun educational applications which teaches basic computer science and programming. We believe that users have the drive to learn but also want to have fun in the learning process as well. Our application will be simple, yet engaging to help users get the most out of their educational aspirations.

2. **Create an educational application which teaches basic computer science and programming, that is highly available to the consumer market:** The current mobile market does not have many educational applications which teaches computer science and programming. Procurement of this application in the mobile market should address this market need—all that the customers need to do is to download the free mobile application in the Google Play store.

Business Risks

Given the main intent of our product we are essentially competing with any educational facet available to the public and is free of charge: books, internet, articles, educational videos, etc.

Strategic Risks

- This is the first time our company has decided to build a mobile application.
- Our workforce consists of software engineers who are familiar with older programming languages.
- We are asking a larger and more experienced consulting company to oversee the certain developmental aspects of this mobile application.

Vision of the Solution

Our main vision is to have everyone enjoy learning C++ anywhere they want, in order to garner interest in the field and inspire others to learn more about the discipline.

Vision Statement

Tuffy++ will be an interactive learning game for children and young adults, which will teach the basics of the C++ programming language. With this game, users of all different knowledge bases will be able to learn the syntax, tools, functions, and paradigms associated with C++ and object-oriented programming.

Tuffy++ will also have an achievement system to help motivate users to continue learning and will help them visualize the knowledge they have gained through the game. Through progress, the users will be able to unlock different lessons and levels to further their understanding and knowledge.

Major Features

This section represents a list of features that are ultimately anticipated for Tuffy++. The major features are included here:

1. Tuffy++ includes an exhaustive library of interactive multiple choice C++ programming questions of a variety of topics such as data types, classes, objects, loops, etc. The library should be updated frequently as the programming language gets developed and updated in the real world.
2. Users get to choose the the difficulty of the questions to adjust to their liking. Otherwise, the difficulty will default to Easy.
3. The game is displayed with endearing and colorful graphics which appeals to people of all ages, especially targeting children and young adults.
4. Amusing soundtracks and captivating sound effects will be included.
5. The game includes certain setting options that allow users to adjust the volume of the music and sound effects as they seem fit.
6. The game is offered in offline mode. Users can play the game (after downloading and installing it on their device) without Internet access required.
7. Users, on the first launch of the game, will enter a name to use to keep track of scores. New scores and name would be needed when downloading on a new phone or reinstalling on a phone that previously had the application.
8. Tuffy++ also features a scoring system to keep track of user's progress when playing the game.
9. The game includes California State University, Fullerton's school mascot, Tuffy, that soars across the sky to dodge obstacles.

10. If user achieves a new high score, the game system will displays “High Score” at the end to let user know they have a new achievement. This feature aims to encourage and engage users in playing Tuffy++ again to beat their own score, thus learning more C++ along the way.

Assumptions and Dependencies

Our product will be a mobile game application geared towards Computer Science education. It will be developed within the Android platform and will be available in the Google Play store. It features California State University, Fullerton’s elephant mascot named Tuffy overcoming obstacles as he tries to graduate on time by answering C++ programming questions. As he gets closer towards graduation, i.e. the user reach higher levels, the questions will become more difficult and complex.

The game will also provide power-ups in order to help Tuffy with his endeavor and a scoring system which will help players keep track of his progress. We will also be using Google Play gamer accounts, leaderboards, and achievements and progress storing to help synchronize and save the game state online in case the user decide to switch or play on multiple Android devices.

Scope and Limitations

Our product is being created to promote awareness, aggregate valuable metrics, and provide insight into the future gamification of education. Making this capital investment in the uncharted area of education might encourage and promote the growth of our company into a valuable market.

Scope of Initial Release

In our initial release, we want to release the mobile game application in the Android market. By making it available in the Google Play store, all customers who own an Android device should be able to access it.

Players will experience a game unlike any other, as it takes them into the wondrous world of computer science as they help Tuffy, a cute elephant mascot, try to graduate on time! This game should help create exposure within the general population of this discipline and hopefully inspire them to pursue it.

Scope of Subsequent Releases

In the future, we are thinking expanding our potential customer base by also developing it for other platforms, such as the market for the Apple iPhone line of products.

For our subsequent releases, we are thinking of:

- Adding additional power ups in order to provide variety for the players.
- Additional avatar skins (new characters) will be added for extra in-game rewards/incentives.
- Linking the game to Google Play user account to save the game progress and allow multiple-device gameplay.

Limitations and Exclusions

Computer science is an incredibly vast field with numerous languages, topics, and practices. This game is only made to appeal to individuals in the exploratory/beginner stage of their computer science career, particularly learning C++ and object-oriented programming. In the future, either an expansion pack/version with more difficult questions should be released or an option to learn another high-level programming language.

Also, there is going to be a library consisting of, a minimum, of 15 and up to, a maximum of, 100 questions, which regularly maintainable and updatable. Unfortunately, when a user uninstalls and reinstalls the app, there is no way to keep track of which questions they have or haven’t previously answered.

Business Context

Operating Environment

The environment in which this game will be played is on Android OS. After this game is complete, we might focus on expanding the game into Apple. However, as of right now, we are focusing on completing all features of the game before moving on to the next operating system.

Tuffy++ will be available at all times as long as the users have it downloaded onto their phone. The only time Internet access will be required would be the initial downloading. Afterwards, it can be played at anytime with offline mode, even without Internet connection, unless the user wants to link the game to their Google Play account to save the game progress even after uninstalling the game.

Pre-Game Planning

1st Iteration:

Explain how did your team work on Pre-game Planning (Write 2-3 paragraphs explaining what you did as well as attaching below items).

Before we started playing our Planning Poker Cards game, we gathered every group member as soon as possible after receiving the deadline. In order to ensure everyone was properly up-to-speed on the current goals of the project, we reviewed our vision and scope. After everyone's blessing, we decided to move forward with our 5 or 6, most functional user stories. These user-stories underwent careful scrutiny of every team member to validate their priority. During the same session, the pre-game planning was initiated to speed up development time.

During the "poker" game, the project manager facilitated every team member's estimation for the given user-story. It took an average of 2 full rotations for all the team members to reach a consensus on the time for developing all the user stories. This was true for all the user-stories except our user-story TPP 03. This particular card dealt with the actual gameplay of the application. The team was primarily torn into two halves: one half believed the gameplay mechanics could be developed in a week, while the other half wanted time upwards to 3 to 4 weeks for development. After four rotations, a reasonable development time for this card was reached - 2 weeks.

Immediately after solving this dilemma, we began constructing our Trello-Kanban board so the entire team could have transparency regarding the development of this project. No individual team member is assigned a particular task, as long as progress is made to reach the necessary deadlines of the first iteration.

Explain how your team developed your system and documented activities by producing work products related to this phase. Write 2-3 paragraphs explaining what you did as well as attaching your code for the 5 functional user stories.

In order to ensure that the development of our system moved forward accordingly, we spend 10 minutes at the beginning of every meeting and 10 minutes at the end of every meeting quickly discussing every member's accomplishments, comments, or concerns. By doing so, we document every step of this first iteration on the KanBan Trello board. This board can be accessed by anyone, anywhere, anytime so if anyone has a very important thought in the middle of the night they can upload their comments to the board for team viewing.

We solely rely on the KanBan Trello board for our main objectives and tasks, but we also use other formal team tools to log our conversations, documents, source code, and other work

products. For conversations, we use Slack. For documents, and source code, we use Google Drive. For diagrams and flowcharts, we use draw.io.

2nd Iteration:

Explain how did your team work on Pre-game Planning for the 2nd iteration. Write one paragraph about the plan.

Thankfully, completing our pre-game planning for the second iteration went a lot smoother than our first iteration. This is partly due to the fact that every team member was able to get a better understanding of the product, tools, methods, and processes we were working with and provided valuable input during this essential planning process. Gauging the amount of work needed to be completed was by far the biggest team improvement—there was little-to-no friction among the team members when it came to presenting their time estimation for user stories. With that being said, it took an average of one rotation for all the team members to reach a consensus on the time required for developing all the user stories. The facilitator of the meeting was very pleased with the outcome for the portion of the iteration. Clearly, the synergy and the collaboration of the team had evolved over the duration of the time.

3rd Iteration:

Explain how did your team work on Pre-Game Planning for the 3rd sprint. Write one paragraph about the plan and include:

For the third iteration, the team's ability to work together improved compared to the work done in the previous iterations. Our team opted to take the "divide and conquer" approach to complete our work more efficiently. In this iteration, we divided the documentation and testing process to different group members. This iteration became much simpler and efficient. Our work progressed much faster since all of the members began to have a clearer idea of the project's whole concept.

Immediately after meeting with our client, our group began consolidating the feedback and worked on the final functional and nonfunctional user stories and use cases. We then divided our tasks over the one-week break period we had in order to get a head start in completing this project. Therefore, thanks to delegation and teamwork, the documentation process took less time and effort to complete.

Use Cases

Functional Use Cases

Functional Use Case 1

Use Case:

- Start

ID:

- UC 01

Primary Actors:

- User

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed.
- The game must be opened

Post-Conditions:

- Success End Conditions:
    - User must be able to see “Start” Screen
    - User must be able to start the game by tapping on “PLAY” button
    - User is directed to “Ready” Screen
- Failure End Conditions:
    - User is unable to start the game or move on to the “Ready” Screen
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User sees “Start” Screen
2. User taps on “PLAY” button
3. User reaches “Ready” Screen

Functional Use Case 2

Use Case:

- Ready

ID:

- UC 02

Primary Actors:

- User

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- The user must have tapped “PLAY” button

Post-Conditions:

- Success End Conditions:
    - User must be able to see “Ready” Screen
    - User must be able to initialize the game by tapping on anywhere on the screen
    - User is directed to “Game” Screen
- Failure End Conditions:
    - User is unable to initialize the game or move on to the “Game” Screen
- Minimal Guarantee:
    - The game does not crash
    - The game does not proceed further than the “Ready” Screen

Main Success Scenario:

1. User sees “Ready” Screen
2. User taps anywhere on screen
3. User reaches “Game” Screen

Functional Use Case 3

Use Case:

- Game

ID:

- UC 03

Primary Actors:

- User

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must have pressed PLAY button
- User musts have tapped on “Ready” screen to initialize game

Post-Conditions:

- Success End Conditions:
    - User must be able to see game screen
    - User must be able to increase elevation by tapping screen
    - User must be able to decrease elevation by not tapping on the screen
- Failure End Conditions:
    - User is unable to change elevation
    - User is unable to reach “Game Over” Screen if they hit obstruction
- Minimal Guarantee:
    - The game does not crash
    - The game does not proceed further than the “Game” Screen

Main Success Scenario:

1. User taps anywhere on “Ready” Screen to reach “Game” Screen
2. User can increase and decrease elevation of the object on the “Game” Screen to avoid obstruction
3. User reaches “Game Over” Screen after hitting obstruction

Functional Use Case 4

Use Case:

- Game Over

ID:

- UC 04

Primary Actors:

- User

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must already have started the gameplay and hit an obstacle
- User must have tapped the “Ready” Screen to initialize
- User must have hit obstruction in the “Game” Screen

Post-Conditions:

- Success End Conditions:
    - User can tap anywhere
    - User can retry the game
- Failure End Conditions:
    - User is unable to tap anywhere
    - User is unable to retry the game
- Minimal Guarantee:
    - The game does not crash and the object stops moving forward

Main Success Scenario:

1. User sees “Game Over” Screen and taps anywhere
2. User gets to retry the game

Functional Use Case 5

Use Case:

- Restart

ID:

- UC 05

Primary Actors:

- User

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must have already started the gameplay and hit an obstacle
- User must have tapped the “Ready” Screen to initialize
- User must have hit obstruction in the “Game” Screen
- User must have seen the “Game Over” Screen
- User must see the “RETRY?” text

Post-Conditions:

- Success End Conditions:
    - User is navigated back to the “Ready” Screen after tapping anywhere on the “Game Over” Screen
    - User is navigated back to the “Ready” Screen after tapping “RETRY?”
- Failure End Conditions:
    - User is not navigated anywhere after tapping the “Game Over” Screen
    - User is not navigated anywhere after tapping the “RETRY?”
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User taps anywhere on the “Game Over” Screen
2. System navigates back to the “Ready” Screen

Functional Use Case 6

Use Case:

- **“**RETRY?” Button

ID:

- UC 06

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- User must already have started the game and hit an obstacle
- User must have tapped the “Ready” Screen to initialize
- User must have hit obstruction in the “Game” Screen
- User must have seen the “Game Over” Screen

Post-Conditions:

- Success End Conditions:
    - User is navigated back to the “Ready” screen after tapping on the “RETRY?” button
- Failure End Conditions:
    - User is not navigated anywhere after tapping the “Game Over” Screen
    - User is not navigated anywhere after tapping the “RETRY?” button
- Minimal Guarantee
    - The game does not crash

Main Success Scenario:

1. User taps on “RETRY?” button
2. User is navigated to the “Ready” Screen

Functional Use Case 7

Use Case:

- **“**BACK” Button

ID:

- UC 07

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened

Post-Conditions:

- Success End Conditions:
    - User sees a small dialog box when tapping Android “BACK” button
    - The box contains the message “Press back again to close this app”
    - The game quits when user taps Android “BACK” button one more time
- Failure End Conditions:
    - The game quits when user taps on “BACK” button the first time
    - User does not see a dialog box when tapping Android “BACK” button
    - The dialog box does not contain any message
    - The game does not quit when user taps Android “BACK” button again
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User taps on the Android “BACK” button
2. Game system displays a small dialog box
3. The game system displays the message “Press back again to close this app” on the dialog box
4. Tapping on Android “BACK” button again quits the game
5. Not tapping on Android “BACK” button again does nothing to the game

Functional Use Case 8

Use Case:

- Top Score

ID:

- UC 08

Primary Actors:

- User and Game System

Secondary Actors:

- Development Team and Client

Pre-Conditions:

- Game has started and the user pressed “PLAY” button
- User has played the game and successfully died

Post-Conditions:

- Success End Conditions:
    - The user’s score is saved
    - The saved scores is sorted from highest to smallest score, displaying only the highest one
    - If user’s current score is higher than the current top score, the game will display “High Score” text instead of “Game Over” text in the “Game Over” Screen
- Failure End Conditions:
    - The user’s score is not saved
    - The highest score is not displayed
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User would play the game and eventually die when Tuffy hits obstacles
2. Whatever the user’s score was when the time of death, the score is then saved
3. If the score is higher than the current top score, the game updates the current top score and displays “High Score” text instead of “Game Over” text in “Game Over” Screen
4. Otherwise, game system displays “Game Over” Screen as previously mentioned in Use Case #UC 04

Functional Use Case 9

Use Case:

- “Question” Screen

ID:

- UC 09

Primary Actors:

- User and Game System

Secondary Actors:

- Developer Team and Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must have pressed “PLAY” on the “Start” Screen
- User must have tapped to start playing on the “Ready” Screen
- User must have hit an obstruction on the “Game” Screen

Post-Conditions:

- Success End Conditions:
    - User sees a C++ question immediately after an obstruction is hit
    - User sees a question with four answer options
- Failure End Conditions:
    - User did not see a “Question” Screen
    - User did not see a C++ questions and four answer options on the “Question” Screen
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User sees a “Question” Screen when an obstruction is hit
2. The game system displays a C++ question and four answer options

Functional Use Case 10
Use Case:
Answer

ID:
- UC 10

Primary Actors:
- User & Game System

Secondary Actors:
- Developer Team & Client

Pre-Conditions:
- The game application must be installed
- The game must be opened
- User must have pressed “PLAY” on the “Start” Screen
- User must have tapped to started playing on the “Ready” Screen
- User must have hit an obstruction on the “Game” Screen
- User must have seen a C++ question with the four multiple choice answers

Post-Conditions:
- Success End Conditions:
    - User is able to select 1 of the 4 options
    - User is navigated to the “Game Over” Screen when the correct answer is selected
- Failure End Conditions:
    - User is not able to select the options
    - User is not navigated to the “Game Over” Screen when the incorrect answer is selected
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:
1. User sees a “Question” Screen with a C++ question and four answer options
2. If user selects the wrong answer, the answer will be highlighted red and a hint appears at the bottom of the screen
3. If user selects the correct answer, the answer will be highlighted green and a feedback “You are Correct!” appears at the bottom of the screen
4. Tapping the screen again navigates to the “Game Over” Screen

Functional Use Case 11

Use Case:

- Mute

ID:

- UC 11

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened

Post-Conditions:

- Success End Conditions:
    - User can successfully turn on and turn off the sound in the game
- Failure End Conditions:
    - User cannot turn on or turn off the sound in the game
    - The sound can be turned on and off in the game independently of the sound system in the device
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User sees the sound button on the “Start” Screen
2. If user turns off the sound using the Sound button, game system plays the game with no sound without affecting other sound system in the device
3. If user turns on the sound using the Sound button, game system plays the game with music and sound effects

Functional Use Case 12

Use Case:

- Name

ID:

- UC 12

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened and started for the first time

Post-Conditions:

- Success End Conditions:
    - User is prompted to type in their name at the “Start” Screen
    - Their name is then saved and shown on top of their score in the “Game Over” Screen
- Failure End Conditions:
    - User is not prompted for their name
    - Their name is not saved
    - Their name is not shown on top of their score
    - System prompts for user’s name after the first launch
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User installed and launched the game for the first time
2. User is prompted to enter their name
3. Game system saved user’s name
4. User started the game and played until they died
5. User answered the C++ Question correctly is navigated to the “Game Over” Screen
6. User’s name is shown on top of their score in the “Game Over” Screen

Functional Use Case 13

Use Case:

- Library

ID:

- UC 13

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must have played and hit an obstruction

Post-Conditions:

- Success End Conditions:
    - Game system randomly generates a new C++ question within the selected difficulty setting
- Failure End Conditions:
    - Game system generates the same C++ question every time
    - Game system generates a C++ question that is of a different difficulty setting
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario

1. User installs, launches the game, selects a difficulty setting, and starts playing
2. User hits an obstruction and dies
3. Game system randomly selects a question from the library that matches the selected difficulty setting and displays to “Question” Screen with the 4 multiple choice options

Functional Use Case 14

Use Case:

- Highscore Question

ID:

- UC 14

Primary Actors:

- User & Game System

Secondary Actors:

- Development Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened

Post-Conditions:

- Success End Conditions:
    - User is able to see the “Question” Screen and a answer C++ question after they get a high score
- Failure End Conditions:
    - Game system navigates from “Game” Screen to “Game Over” Screen when user gets a high score
    - User is unable to answer the C++ question after they got a high score
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User installs and launches the game
2. User selects difficulty and volume setting, starts playing, gets a high score, and hits an obstruction
3. Game system navigates to “Question” Screen
4. Ser is able to answer the displayed C++ question

Functional Use Case 15

Use Case:

- Redirection

ID:

- UC 15

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must have played and hit an obstruction
- User must have answered a question correctly
- User must have seen the “Game Over” Screen and tapped the “RETRY?” button

Post-Conditions:

- Success End Conditions:
    - User is navigated back to “Start” Screen
- Failure End Conditions:
    - User is navigated back to “Ready” Screen
    - User is navigated to any screen other than “Start” SCreen
    - User is not navigated anywhere
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario

1. User installs and launches the game
2. User plays and hits an obstruction
3. User answers a question correctly, sees the “Game Over” Screen, and taps on “RETRY?” button
4. Game system navigates back to “Start” Screen

Functional Use Case 16

Use Case:

- How to Play

ID:

- UC 16

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be launched

Post-Conditions:

- Success End Conditions:
    - User sees “HOW TO PLAY” button at the bottom of “Start” Screen
    - Tapping on “HOW TO PLAY” button directs user to “How to Play” Screen with simple instructions
    - Tapping anywhere on “How to Play” Screen directs user back to “Start” Screen
- Failure End Conditions:
    - Game system fails to load “HOW TO PLAY” button
    - Tapping on “HOW TO PLAY” button does not direct to “How to Play” Screen
    - “How to Play” Screen does not include any instruction
    - Tapping on “How to Play” Screen does not navigate anywhere
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario

1. User installs and launches the game
2. Game system launches “Start” Screen with “How to Play" button at the bottom
3. User taps on How to Play button and game system loads “How to Play” Screen
4. Instruction on “How to Play” Screen are: “1. Tap to ascend. 2. Let go to Descend. 3. Avoid the Obstacles. Good Luck!”
5. User taps anywhere on “How to Play” Screen and game system directs back to “Start” Screen

Non-Functional Use Cases

Non-Functional Use Case 1

Use Case:

- Graphics

ID:

- UC 17

Primary Actors:

- User
- Game System

Secondary Actors:

- Developer Team
- Client

Pre-Conditions:

- The game is installed on the device
- The game is open and running

Post-Conditions:

- Success End Conditions:
    - The game has clear, crisp graphics
- Failure End Conditions:
    - Graphics do not load
    - Graphics are unclear and ugly
- Minimal Guarantee:
    - The game does not crash

Main Scenario:

1. User opens application
2. User reaches “Start” Screen
3. Graphics is clear and aesthetically pleasing

Non-Functional Use Case 2

Use Case:

- Music

ID:

- UC 18

Primary Actors:

- Users
- Game Systems

Secondary Actors:

- Development Team
- Client

Pre-Conditions:

- The game is installed and running properly

Post-Conditions:

- Success End Conditions:
    - Game system plays a soundtrack on repeat in the background when the game is running
    - Game system stops playing music when the game is terminated
- Failure End Conditions:
    - Background music stops after a certain period of time
    - Background music never plays
- Minimal Guarantee:
    - The game does not crash

Main Scenario:

1. User launches the game
2. Game system plays a soundtrack on repeat in the background when the game is running
3. User taps “BACK” button twice to terminate the game, and the music stops playing

Non-Functional Use Case 3

Use Case:

- Sound

ID:

- UC 19

Primary Actors:

- Users
- Game System

Secondary Actors:

- Development Team
- Client

Pre-Conditions:

- The game is installed and running properly

Post-Conditions:

- Success End Conditions:
    - Game system plays a sound effect when Tuffy flaps its ears
    - Game system plays a sound effect when Tuffy hits an obstacle
    - Game system plays a sound effect when Tuffy falls down and dies
    - Game system plays a sound effect every time the screen is tapped
- Failure End Conditions:
    - No sound effect is played
- Minimal Guarantee:
    - The game does not crash

Main Scenario:

1. User runs the game
2. Game system plays a sound effect when the elephant character flaps its ears
3. Game system plays a sound effect when the elephant character hits an obstacle
4. Game system plays a sound effect when the elephant character falls down and dies
5. Game system plays a sound effect every time the screen is tapped

Non-Functional Use Case 4

Use Case:

- Text Font

ID:

- UC 20

Primary Actors:

- Users
- Game System

Secondary Actors:

- Development Team
- Client

Pre-Conditions:

- The game is installed and running properly

Post-Conditions:

- Success End Conditions:
    - Game System displays the new font for all the text and numbers
- Failure End Conditions:
    - Font is unclear and ugly
    - Font and text does not load properly
- Minimal Guarantee:
    - The game does not crash

Main Scenario:

1. User runs the game
2. Game system displays new text font for the game for all game pages screens

Non-Functional Use Case 5

Use Case:

- Compatibility Across all Android Platforms

ID:

- UC 21

Primary Actors:

- Users
- Game Systems

Secondary Actors:

- Development Team
- Client

Pre-Conditions:

- User has Android 7.0 platform on their device

Post-Conditions:

- Success End Conditions:
    - The game can be installed on Android 4.0 and above
    - Properly runs on all of the newer versions of Android
- Failure End Conditions:
    - Game is unplayable on newer versions of Android
- Minimal Guarantee:
    - The game does not crash

Main Scenario:

1. User has Android 7.0 on their device
2. User installs and runs the game on an Android device
3. The game runs successfully and functions the same way on different Android platforms between 4.0 and 7.0

Non-Functional Use Case 6

Use Case:

- Validate

ID:

- UC 22

Primary Actors

- User & Game System

Secondary Actors

- Developer Team & Client

Pre-Conditions

- The game application must be installed
- The game must be opened
- User must have played the game and hit an obstruction

Post-Conditions

- Success End Conditions:
    - User can successfully see if the chosen answer is correct
- Failure End Conditions:
    - Game system shows an incorrect answer as correct
    - Game system marks an answer correct if it's supposed to be incorrect
    - Game system allows user to press "RETRY?" even though answer is incorrect
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario

1. User installed and launched the game for the first time
2. User is prompted to enter their name
3. User can select their difficulty at home screen
4. User can select their topic at home screen
5. User started the game and played until they died
6. User can clearly see when they answer questions correctly

Non-Functional Use Case 7

Name:

- “Ready” Screen Instruction

ID:

- UC 23

Primary Actors

- User & Game System

Secondary Actors

- Developer Team & Client

Pre-Conditions

- The game application must be installed
- The game must be opened
- User must have selected difficulty and tap “PLAY” button on “Start” Screen
- User must have been navigated to “Ready” Screen

Post-Conditions

- Success End Conditions:
    - User is able to see the “Tap to Start” instruction on the “Ready” Screen to start the game
- Failure End Conditions:
    - Game does not display “Tap to Start” instruction on “Ready” Screen
- Minimal Guarantee
    - The game does not crash

Main Success Scenario

1. User installs and launches the game
2. User selects a difficulty level and taps “PLAY” button on “Start” Screen
3. User is navigated to “Ready” Screen
4. Game system displays “Tap to start” instruction on “Ready” Screen

Non-Functional Use Case 8

Name:

- Difficulty Highlight

ID

- UC 24

Primary Actors

- User & Game System

Secondary Actors

- Developer Team & Client

Pre-Conditions

- The game application must be installed
- The game must be opened
- User must be able to see “Start” Screen and 3 difficulty buttons: “EASY”, “MEDIUM”, and “TUFF”

Post-Conditions

- Success End Conditions:
    - User is able to select a difficulty button and the button becomes highlighted when selected
- Failure End Conditions:
    - The button display is not highlight when selected
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario

1. User installs and launches the game
2. Game system displays “Start” screen with 3 difficulty buttons: “EASY”, “MEDIUM”, and “TUFF”
3. Game system highlights the difficulty button when selected

Non-Functional Use Case 9
Name:

- "R.I.P." Message

ID:

- UC 25

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must hit the obstruction and see the message
- User must see the message "You died. Answer to move on." immediately after an obstruction is hit

Post-Conditions:

- Success End Conditions:
    - User can get the above message on the "Question" Screen
    - User can successfully get a C++ question with four answer options below the message and can easily answer the question
- Failure End Conditions:
    - User cannot get the message on the "Question" Screen
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User can easily understand the next step of the game through the message displayed on the screen
2. As simplicity being one of the main non functional requirements of the project, game is made to be easily understood without any set of instructions or help.

Non-Functional Use Case 10

Use Case:

- Volume Highlight

ID:

- UC 26

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- The game application must be installed
- The game must be opened
- User must be able to see the volume button on “Start” Screen

Post-Conditions:

- Success End Conditions;
    - Game system successfully highlights the volume setting when pressed
- Failure End Conditions:
    - Game system does not highlight the volume setting
- Minimal Guarantee:
    - The game does not crash

Main Success Scenario:

1. User installs and launches the game
2. User sees the Volume button on the “Start” Screen and taps on it to toggle between ON and OFF
3. Game system highlights the volume setting, ON or OFF, selected

Non-Functional Use Case 11

Use Case:

- Maintainability/Adaptability

ID:

- UC 27

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- Pre-code planning

Post-Conditions:

- Success End Conditions:
    - Future coding will be quick and easy
    - New code won't require rewriting from scratch
- Failure End Conditions:
    - Constant changes must be made to keep old code up to date with the new code
    - Spend more time debugging the code than necessary
- Minimal Guarantee:
    - Code will function properly

Main Success Scenario:

1. Code is eligible and intuitive
2. New code does not require old blocks to be overridden
3. Easy to bring in new functions to the game

Non-Functional Use Case 12

Use Case:

- Game Simplicity

ID:

- UC 28

Primary Actors:

- User & Game System

Secondary Actors:

- Developer Team & Client

Pre-Conditions:

- Game is playable

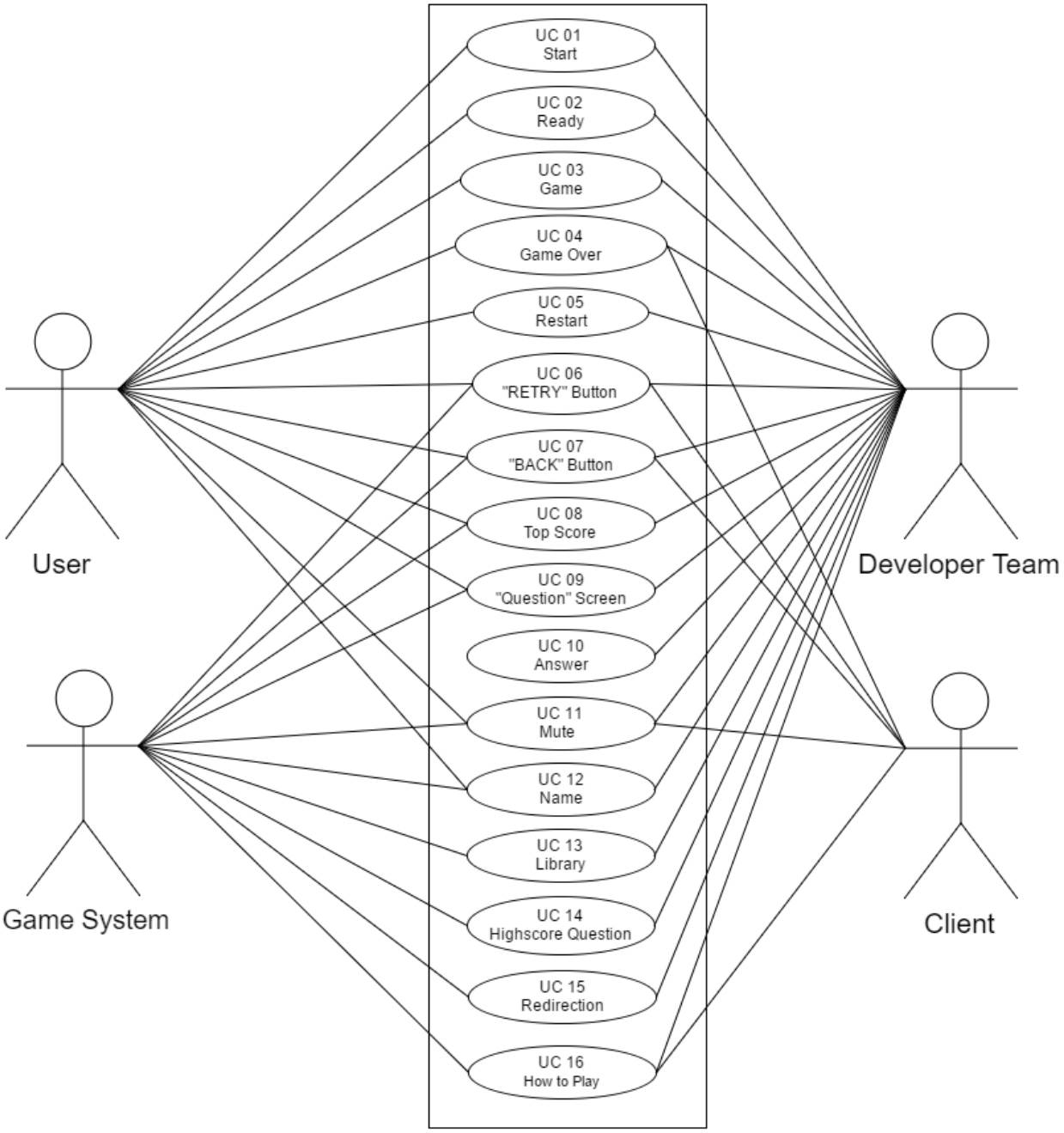
Post-Conditions:

- Success End Conditions:
    - Little-to-no instructions necessary to learn the game
- Failure End Conditions:
    - Users get confused when playing the game
    - Require more rules and instructions to play the game
- Minimal Guarantee:
    - There will not be much instructions and rules

Main Success Scenario:

1. User starts playing the game and does not require any sort of manual
2. As user progress, he understands the mechanics of the game
3. Questions will be short and understandable

Use Case Diagram



User Stories

Functional User Stories

1. **As a user, I want to be able to use the "Start" screen of the game so that I can begin to play the game.**
    a. User should be able to click "PLAY" button
    b. When user clicks "PLAY" button, page will navigate to the game screen
    c. User should see the title of the game

2. **As a user, I want to have a "Ready" screen so that I can choose when to start the game.**
    a. User should be able to see the prompt "READY?"
    b. User should be able to tap the screen and initiate the game sequence
    c. User can wait at this screen for an indefinite amount of time

3. **As a user, I want to be able to use the game screen so that I can play the game.**
    a. User should be able to tap to raise the object above obstructions
    b. User should be able to not tap and decrease the object's elevation
    c. User will fail the current run if they hit an obstruction

4. **As a user, I want to have a "Game Over" screen so that I can now when I have ended the game.**
    a. User will have the option to tap anywhere to continue
    b. User will see their final score
    c. User will see their best score
    d. User will have the option to retry

5. **As a user, I want to be able to restart the game so that I can play again.**
    a. User will be able to tap the screen's "RETRY?" button
    b. User will be navigated to "Ready" Screen

6. **As a user, I want to have a "RETRY?" button when the game is over so that I can try playing again.**
    a. User should be able to see a "RETRY?" button on the "Game Over" Screen
    b. User should be able to tap on the "RETRY?" button
    c. Tapping on the "RETRY?" button will navigate the user back to the "Ready" Screen

7. **As a user, I want to have a "Quit?" dialog box so that I can choose to quit the game and go back to the phone's home screen.**
    a. User should be able to see a dialog box popped up when tapping the Android "BACK" button
    b. The "Quit?" dialog should display this message: "Press Back again to quit the app"
    c. Tapping on "BACK" button again will quit the game
    d. Not tapping on "BACK" button again will resume the game

8. **As a user, I want to be able to have a record of my top score so that I can compare my previous results.**
    a. Game system should keep track of the highest score on the user's device
    b. Game system should display the top score at the "Game Over" Screen

9. **As a user, I want to have a C++ question when the game is over so that I can learn C++ and answer the question to restart the game.**
    a. User should be able to see the "Question" Screen after hitting an obstruction
    b. The question should be a C++ question together with four answer choices inside the "Question" Screen

10. **As a user, I want to be able to select an answer to the C++ question so that I can play the game again.**
    a. User should be able to select one of the four answer options displayed
    b. User should be navigated to the "Game Over" Screen when the correct answer is selected

11. **As a user, I want to have a button that controls sound so that I can choose to play the game muted or not.**
    a. User should be allowed to control the sound (ON or OFF) from the "Start" Screen

12. **As a user, I want to have my name recorded so that I can see my name shown on top of my score.**
    a. User should be prompted to type in their name the first time the game is launched
    b. After typing in their name, game system will display user's name on top of their score on the "Game Over" Screen

13. **As a user, I want to be asked a different C++ question every time my character dies so that I can learn new C++ knowledge**
    a. A C++ question is randomly generated from a library of C++ questions based on the selected difficulty setting

14. **As a user, I want to answer a C++ Question even when I get a new high score, so that I can learn C++ at all times**
    a. Game system should display a C++ Question even when user achieves a new high score

15. **As a user, I want to be directed to the "Start" screen instead of "Ready" screen after I tap the "RETRY?" button so that I can select a different difficulty level if I want to.**
    a. Game system navigates back to "Start" Screen instead of "Ready" Screen after ser taps "RETRY?" button on the "Game Over" Screen

16. **As a user, I want to have an instruction screen on how to play the game so that I know how to play.**
    a. Game system includes a How to Play button on "Start" Screen
    b. Tapping on the button opens up a "How to Play" Screen with simple instruction on how to play the game
    c. Tapping anywhere on "How to Play" Screen navigates back to "Start" Screen

Non-Functional User Stories

1. **As a user, I want to be able to clearly see the graphics so that I can enjoy the game more.**
    a. Users should be able to see all the graphical details clearly
    b. User should enjoy the visuals presented by the game
    c. When the user launches the application, the instructions written on the screen should be clear and concise

2. **As a user, I want to be able to hear music so that I have more fun playing the game.**
    a. Game system should include a background soundtrack.
    b. The background soundtrack will be played on repeat as long as the game is opened

3. **As a user, I want to be able to hear sound effects for each interaction so that the game is more appealing to me.**
    a. Game system should play sound effects for when the elephant character flaps its ears, i.e. when user taps the screen
    b. Game system should play sound effects for when the elephant character hits an obstruction
    c. Game system should play sound effects for when the elephant character dies/falls down

4. **As a user, I want the game to have a new text font so that I can see the text more clearly.**
    a. Game system should display new text font for all the text: game name, buttons, scores, C++ questions and answers, and other text

5. **As a user, I want to be able to play Tuffy++ on all versions of Android platform from 4.0 (Kit Kat) to the new 7.0 (Nougat) so I am not restricted to using only one version of Android.**
    a. User can download, install, and play Tuffy++ on all version of Android from 4.0 to 7.0
    b. There should be no discrepancies in the game system, display, and user experience when the game is run on different Android versions

6. **As a user, I want to know what the controls are so that I can play the game.**
    a. User should be instructed on how to play the game
    b. The instruction should be displayed in simple text on the “Start” screen

7. **As a user I want to have “Tap to start” instruction on the “Ready” screen, so that I know how to start the game.**
    a. User should be able to see the “Tap to start” instruction to start the game

8. **As a user I want to know which difficulty I selected so I can learn C++ accordingly.**
    a. A user should be able to clearly see the difficulty setting

9. **As a user I want to see “You died. Answer to move on.” message displayed, so that I can now the next step to continue the game.**
    a. User should be able to clearly see the message after an obstruction is hit

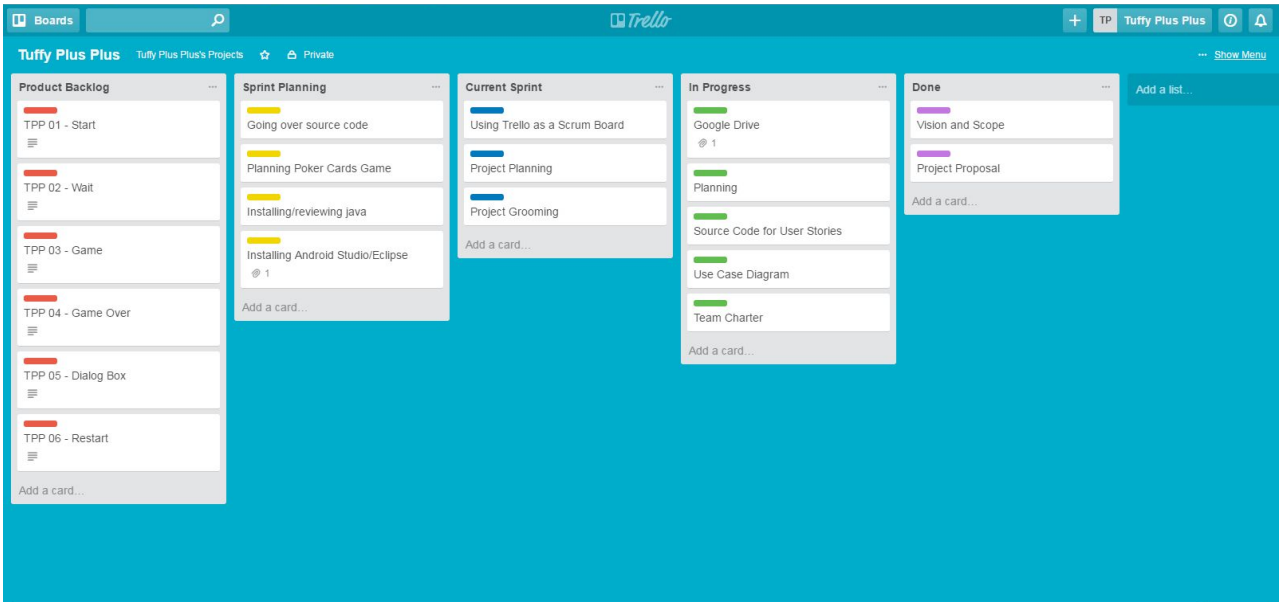
10. **As a user I want to have the volume button highlighted , so that I can know which volume setting is selected**
    a. User should be able to see the volume button highlighted when pressed

11. **As a developer, I want the code for the game to be maintainable so that we can maintain and continue developing it for future iterations and releases.**
    a. The code for the game should be well-organized in appropriate folders
    b. The code for the game should be easy to read and easy to modify
    c. Encapsulation should be used for particular methods or object creation to ensure readability.

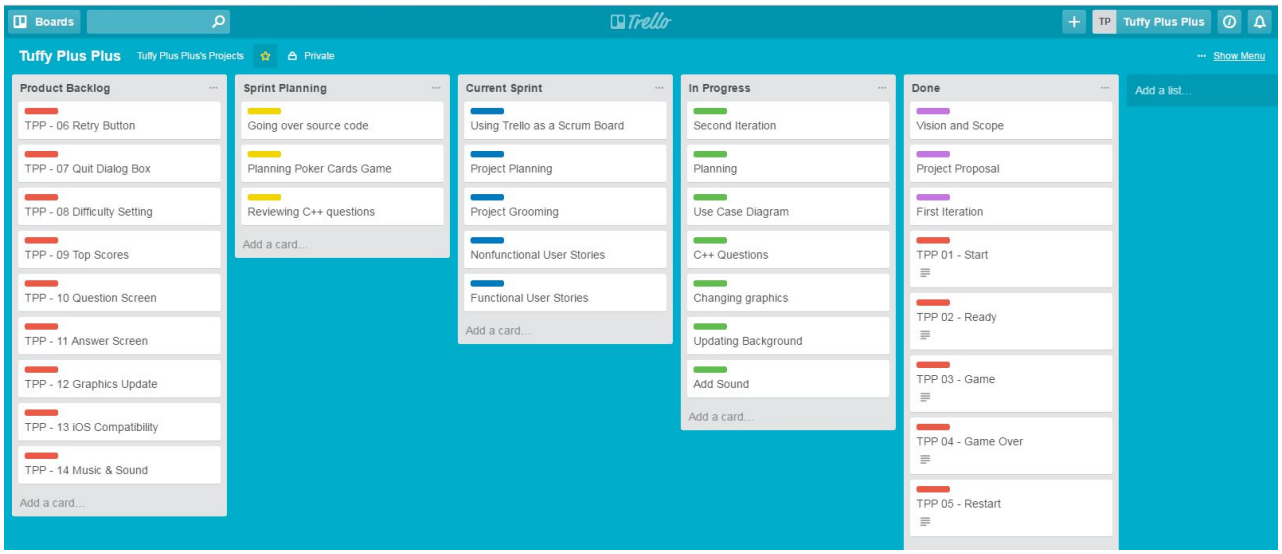
12. **As a user, I want the game to be simple so that I can play with minimum instructions**
    a. The gameplay controls should be intuitive
    b. The navigation methods must be clear and obvious
    c. The instructions should be short, concise, and easy to understand

Trello Kanban Board

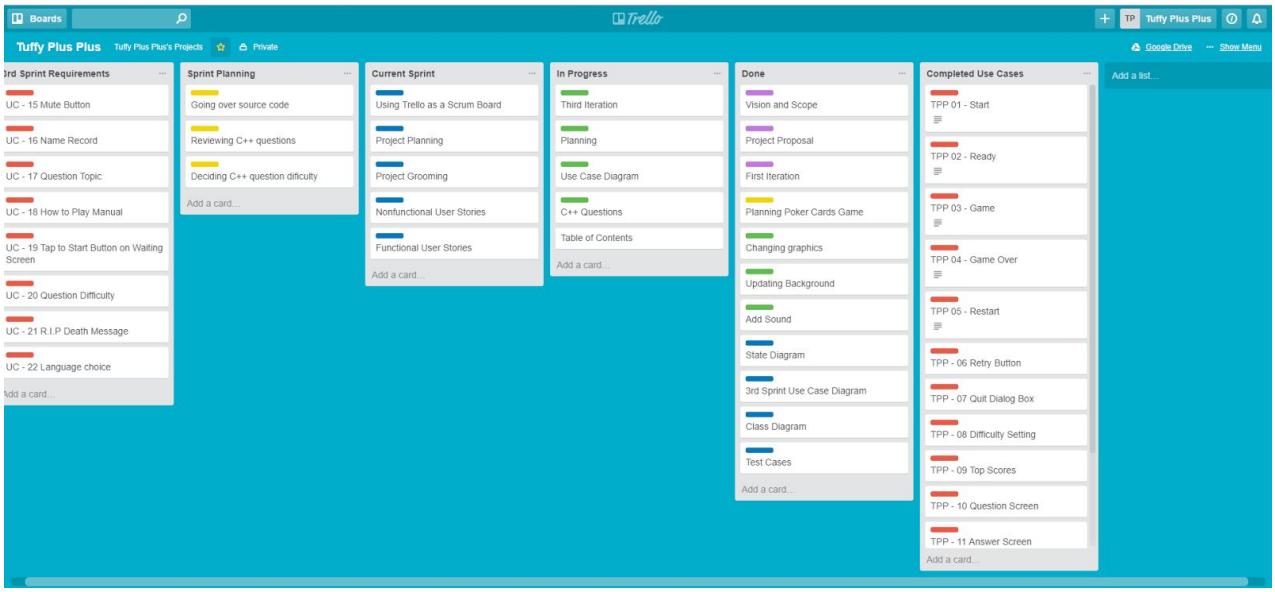
1st Iteration:



2nd Iteration:



3rd Iteration:



Staging and Grooming

1st Iteration:

Explain how your team worked on staging/grooming: identifying requirements and how did you prioritize them for the First Iteration.

After reviewing all of the features listed on the Vision and Scope document, we decided on some key factors to include into the first iteration and excluded the secondary features. This is primarily why we felt like the staging/grooming portion of this project was integral to its success. Our team consists of very passionate and creative individuals, hence some of the features that we plan on integrating into our application were almost "out of this world." With that being said, in order to preserve the collaboration and representation of every team member, we had a very simple and democratic method to choose our critical and non-critical user-stories - voting.

The project manager told every team member (3 graduate students and 3 undergraduate students) to write 5 key features that would really like to see implemented in the first iteration. The project manager already knew that there would be no possible way to actually complete 30 game critical user-stories. Given a time frame of roughly two weeks to produce a work product, we decided to cut down a lot of the features we initially wanted to implement. Every member was easily able to weed at least 3 features right off-the-bat. For some of the more sensitive features, we did an anonymous vote to see if the rest of the team wanted to keep the feature as a priority. Our main focus at this point is to just get the game running. Our revised goal then became to just get the game screen working, the gameplay mechanics to be implemented, and the "End Game" screen to be functional with our educational questions awaiting the user. This goal, though not as grand as our original plan, is a more practical, feasible, and executable step that can then lead to a fully perfect game.

2nd Iteration:

Explain your second iteration development, testing and documentation process. Write one paragraph explaining what you did as well as attaching your code for this sprint's five functional user stories (if possible, if not make a zip file and attach it to your submission).

In the first iteration, the development of the testing was slow because everyone was fairly new to working with each other and working with agile methodology. The entire process began to synergize more as time went on during the first iteration and completing working took less effort to collaborate. During the second iteration, we took an entirely different approach for the development, testing and documentation process. For the sake of efficiency, we decided to implement the "divide and conquer" method - half of the team would work on developing and

testing the game while the other half focused on completing the documentation required. This strategy expedited the completion of the second iteration.

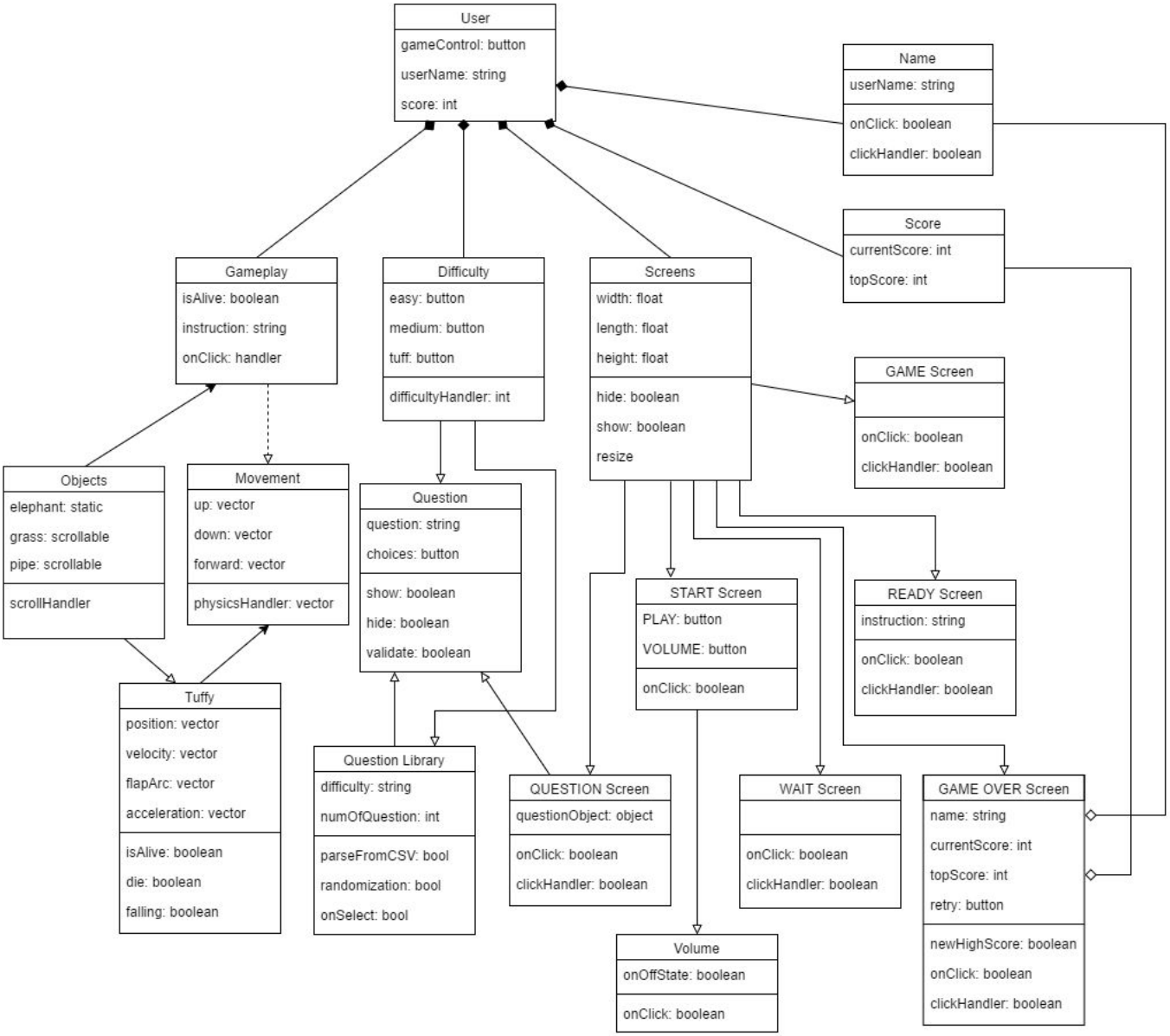
3rd Iteration:

Explain how your team worked on staging or grooming by identifying more requirements and prioritizing them for the 2nd iteration. Write one paragraph and include below items: User Stories, Sprint Backlog Screenshots.

For our third iteration, we had most of our game already completed at this point. We just needed to do some finishing touches that make the game how we originally imagined. We needed to implement our final phase of this project which was to create a library of C++ questions and importing that library into the game. These questions are what help our users learn C++ in a fun and addicting manner. Also, after having meeting with our client for a final review, we took in the feedback and decided to include a couple of add-ons into the game. We will be including an option to input a name upon the first game launch and will display it on top of their scores. Finally, users will be able to see some short instruction texts on certain game screens that clarify what to expect and how to play the game which enhances the user's gameplay experience. Other simple graphical fixes include a better indication when pressing button, a volume setting and clearer animations.

Similar to the second sprint, our team opted the "divide and conquer" method to complete our work in an efficient way. In this third sprint, the documentation work and testing process has been divided among our group members. The documentation process was made simple in this sprint as everybody in the group are used to the structure and got a clear idea of the whole process. Immediately after the meeting with our client, our group began consolidating the feedback and working on the final functional and non-functional user stories and use cases. We then divided out the tasks that needed to be completed over our one-week break period to lighten up the tasks in the long run. Therefore, thanks to delegation and teamwork, the documentation process took up less time and effort to complete compared to the previous iterations.

Class Diagram



CRC Cards

User	
- knows game controls	Gameplay
- knows difficulty	Difficulty
- knows user's name	Screens
- knows user's scores	Scores

Scores	
- knows user's scores	User
- knows top scores	
- knows current scores	

Difficulty	
- knows the difficulty	User
	Questions

Questions	
- knows the question	Question Screen
- handles what happen clicked	Difficulty
- knows the options	
- knows the answer	

Gameplay	
knows if Object (Elephant) is alive	User
	Objects
	Movement

Objects	
- knows elephant	Gameplay
- knows grass	Elephant
- knows pipe	
- handles scrolling	

Elephant	
- knows if it's alive	Objects
- knows how to fly	Movements
- knows if it's falling	
- knows its position	

Movement	
- knows up	Gameplay
- knows down	Elephant
- knows forward	
- handles physics	

Screens	
- handles when clicked	QUESTION Screen
	WAIT Screen
- knows its size	PLAY Screen
	START Screen
	READY Screen

PLAY Screen	
- handles what happens when clicked	Screens

START Screen	
- knows when the user "start"	Screens
- knows when to tell the continue to next screen	

QUESTION Screen	
- knows Question	Screens
- knows answers	Question
- handles what happens answer is chosen	

WAIT Screen	
- handles what happens clicked	Screens

Name	
- knows the name of the user	User

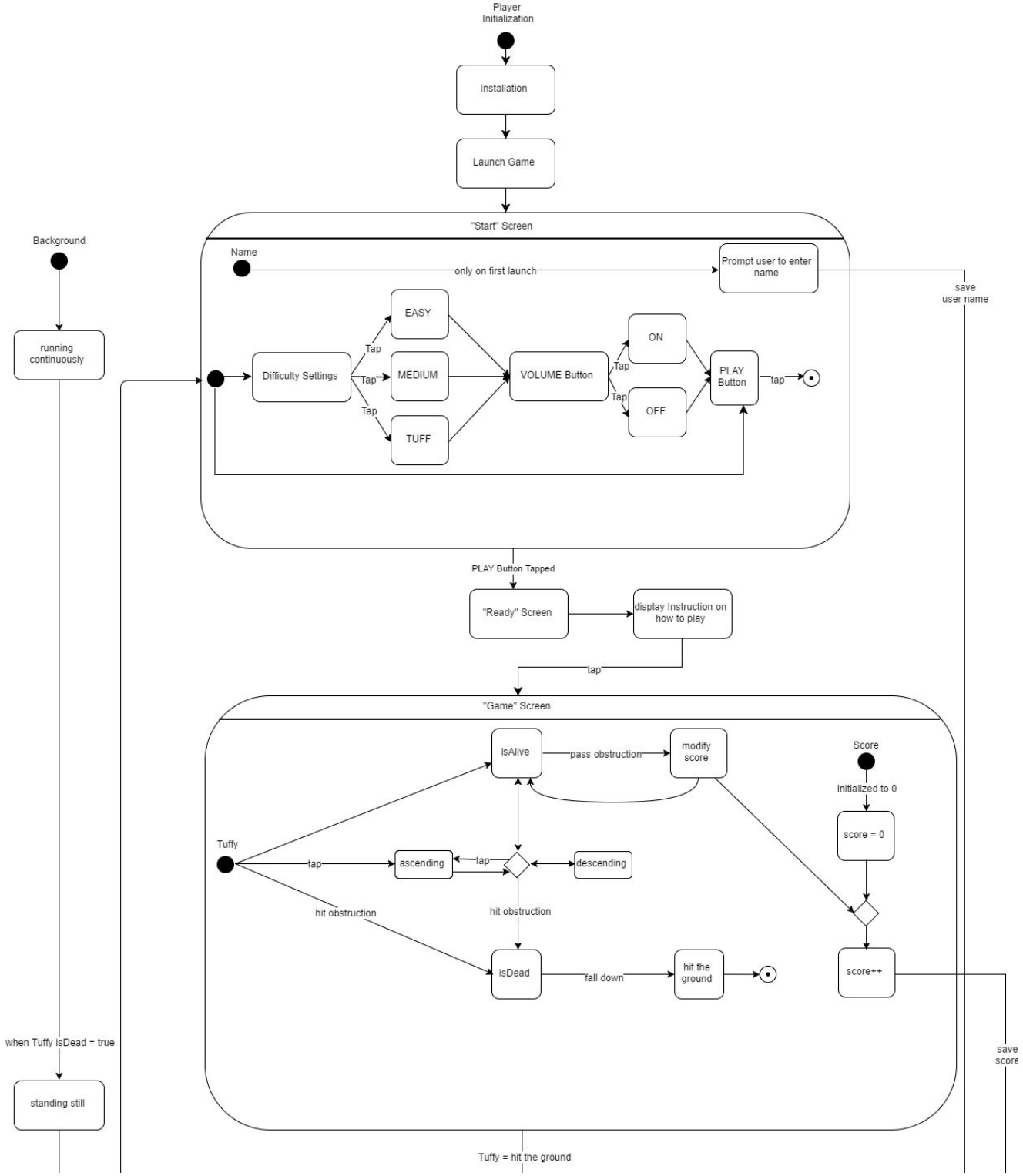
GAME OVER Screen	
- knows user name	Screens
- knows current score	Name
- knows top score	Score

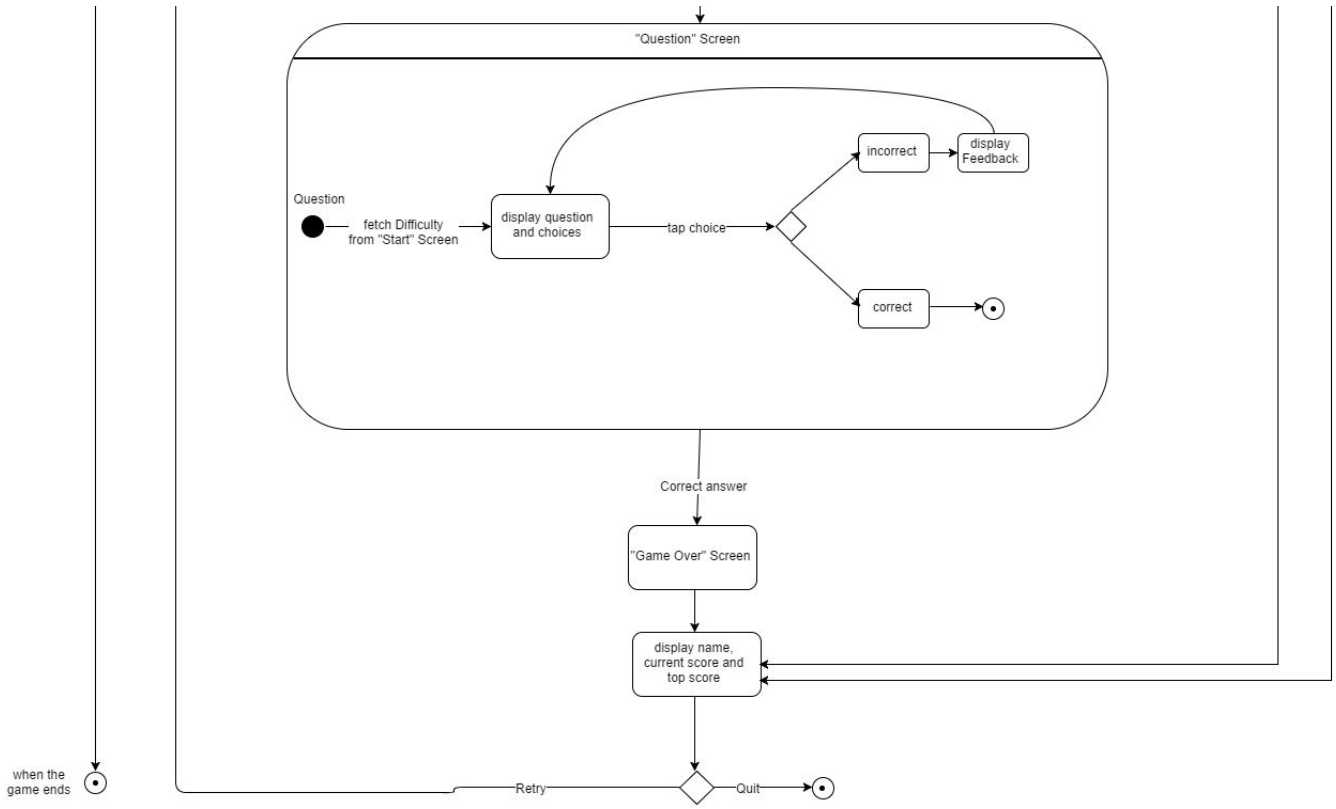
READY screen	
- knows to start the game	Screens

Volume Button	
- knows if its on or off	Start Screen

Question Library	
- knows the number of	- Question
- knows the questions accroding to the difficulty level	- Difficulty

State Diagram




"Question" Screen
Question
fetch Difficulty
from "Start" Screen
display question
and choices
tap choice
incorrect
display
Feedback
correct
Correct answer
"Game Over" Screen
display name,
current score and
top score
when the
game ends
Retry
Quit

Test Cases

Test Case 01

Test Case #: TC_01 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see if the game launches successfully	**Test Case Name:** Application launch **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- The device must have the application downloaded

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Open the application	The game navigates to the Tuff Studios Screen	Pass	
2	Application launches successfully	Game system navigates from Tuff Studios Screen to Start Screen	Pass	

Post-Conditions:
- The game navigates to the Start Screen

Test Case 02

Test Case #: TC_02	**Test Case Name:** Start Screen
System: Android Devices	**Subsystem:** Android Versions 4.0+
Designed by: Tuff Studio	**Design Date:** 12/01/2016
Executed by: Tuff Studio	**Execution Date:** 12/03/2016
Short Description: Test to see if "Start" Screen is fully functional	

Pre-Conditions:
- The device must have the application downloaded
- The application is launched

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Launch application	Game system launches the game, then navigates from Tuff Studios Screen to Start Screen	Pass	
2	Check the graphic display	Graphic includes repeating background and hovering Tuffy the elephant	Pass	
3	Check the Difficulty	Difficulty includes 3 Difficulty buttons: Easy, Med, and Tuff	Pass	
4	Check the Sound	Sound includes a toggle Sound button	Pass	
5	Check the How to Play	How to Play includes the How to Play button	Pass	
6	Check the Play button	Tapping on PLAY button leads to Ready Screen	Pass	

Post-Conditions:
- Start Screen displays graphic, a functional menu of options, and a "PLAY" Button
- Tapping on "PLAY" button navigates to the "Ready" Screen

Test Case 03

Test Case #: TC_03 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see if "Ready" Screen is fully functional	**Test Case Name:** Ready Screen **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- Game application has been started
- The Start Screen loads successfully

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Press "PLAY" on the "Start" Screen	The game navigates to the "Ready" Screen	Pass	
2	Instruction should be shown on the screen	"Tap to play" is shown on the screen	Pass	
3	Tap the screen button to indicate user is ready	Game proceeds to "Game" Screen	Pass	

Post-Conditions:
- User can now start tapping the screen to play the game

Test Case 04

Test Case #: TC_04 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test the game mechanics in "Game" Screen	**Test Case Name:** Test Game Screen **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- Game application is installed onto the mobile device
- User has started the application
- Start Screen loaded successfully and the user pressed "PLAY"
- User tapped on the screen to play the game

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Tap the screen to make Tuffy fly	Tuffy should elevate vertically	Pass	
2	Avoid continuous obstacles	Each obstacle successfully passed increments the user's score	Pass	
3	Hitting an obstruction	Tuffy falls down and game system navigates to "Question" Screen	Pass	

Post-Conditions:
- User continues to play the game until an obstacle is hit
- Game can theoretically last infinitely long as long as user does not die

Test Case 05

Test Case #: TC_05 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see what happens after obstacle is hit	**Test Case Name:** Question Screen **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- User must have completed the above steps
- User must have hit an obstruction, causing Question Screen to appear

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Check the "Question" Screen instruction	Instruction says "You died. Answer to move on."	Pass	
2	Check the Question	Game system displays a question that matches the selected difficulty level	Pass	
3	Check the Answer Options	Game system displays 4 Answer Options that match the question	Pass	

Post-Conditions:
- Question Screen appears with an instruction text, a C++ question, and 4 answer options

Test Case 06

Test Case #: TC_06 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to make sure question and answers work properly	**Test Case Name:** Answering a Question **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- User must have completed the above steps and the previous conditions are met
- User is at Question Screen

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	User taps one of the 4 answers	The chosen answer will either be correct or incorrect	Pass	
2	User gets the answer incorrect	The selected answer option is highlighted red and a hint is displayed at the bottom of the screen	Pass	
3	User gets the answer correct	The selected answer option is highlighted green and a feedback text “You are correct” is displayed at the bottom of the screen	Pass	
4	After the correct answer is selected	Game system navigates to Game Over Screen	Pass	

Post-Conditions:
- The screen will continue to loop until answer is correct
- After the answer is correct, the user is navigated to the Game Over Screen

Test Case 07

Test Case #: TC_07 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see if "Game Over" Screen is fully functional	**Test Case Name:** Game Over Screen **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- User must have completed the above steps and the previous conditions are met
- User is at Game Over Screen

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Check User name	User name is displayed correctly	Pass	
2	Check Score	Score and Top Score are displayed correctly	Pass	
3	Tap anywhere on screen except the "RETRY?" button	Game system does not navigate anywhere	Pass	
4	Tap on "RETRY?" button	Game system navigates back to "Start" Screen	Pass	

Post-Conditions:
- Tapping on "RETRY?" Button causes game system to navigate back to "Start" Screen

Test Case 08

Test Case #: TC_08 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see if the game's volume turns on or off	**Test Case Name:** Sound Button **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- User must be in the Start Screen

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Tap on the Sound button	Sound either turns on or off depending on current state	Pass	

Post-Conditions:
- The game volume turns "off" if it was "on" when user pressed the button
- The game volume turns "on" if it was "off" when user pressed the button

Test Case 09

Test Case #: TC_09 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test to see what happens if user gets a new high score	**Test Case Name:** High Score **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- Game must be installed and launched
- User must be playing and have achieved a new high score
- User must have hit an obstruction and answered a C++ Question correctly
- User must be in “Game Over” Screen

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Check to see if High Score appears	Game system displays High Score where Game Over text should be	Pass	
2	Check actual Score	Score and Top Score are the same, being the new High Score user just achieved	Pass	

Post-Conditions:
- High Score is displayed correctly on “Game Over” Screen
- User can tap “RETRY?” Button to play again

Test Case 10

Test Case #: TC_10 **System:** Android Devices **Designed by:** Tuff Studio **Executed by:** Tuff Studio **Short Description:** Test the music and sound system of the game	**Test Case Name:** Music and Sound **Subsystem:** Android Versions 4.0+ **Design Date:** 12/01/2016 **Execution Date:** 12/03/2016

Pre-Conditions:
- Game must be installed and launched
- Sound must not be turned off on Android device and/or by Sound button in "Start" Screen

Steps	Action	Expected System Response	Pass/ Fail	Comment
1	Launch the game	Background music plays starting from "Start" Screen until application closes	Pass	
2	Avoid an obstruction	Sound effect plays every time Tuffy pass an obstruction	Pass	
3	Hit an obstruction	Sound effect plays when Tuffy hits an obstruction	Pass	
4	Turn off screen display	Background music and sound effect stops playing	Pass	
5	Turn screen display back on	Background music and sound effect continues playing	Pass	
6	Tap "BACK" button twice to quit the game	Background music and sound effect stops playing	Pass	

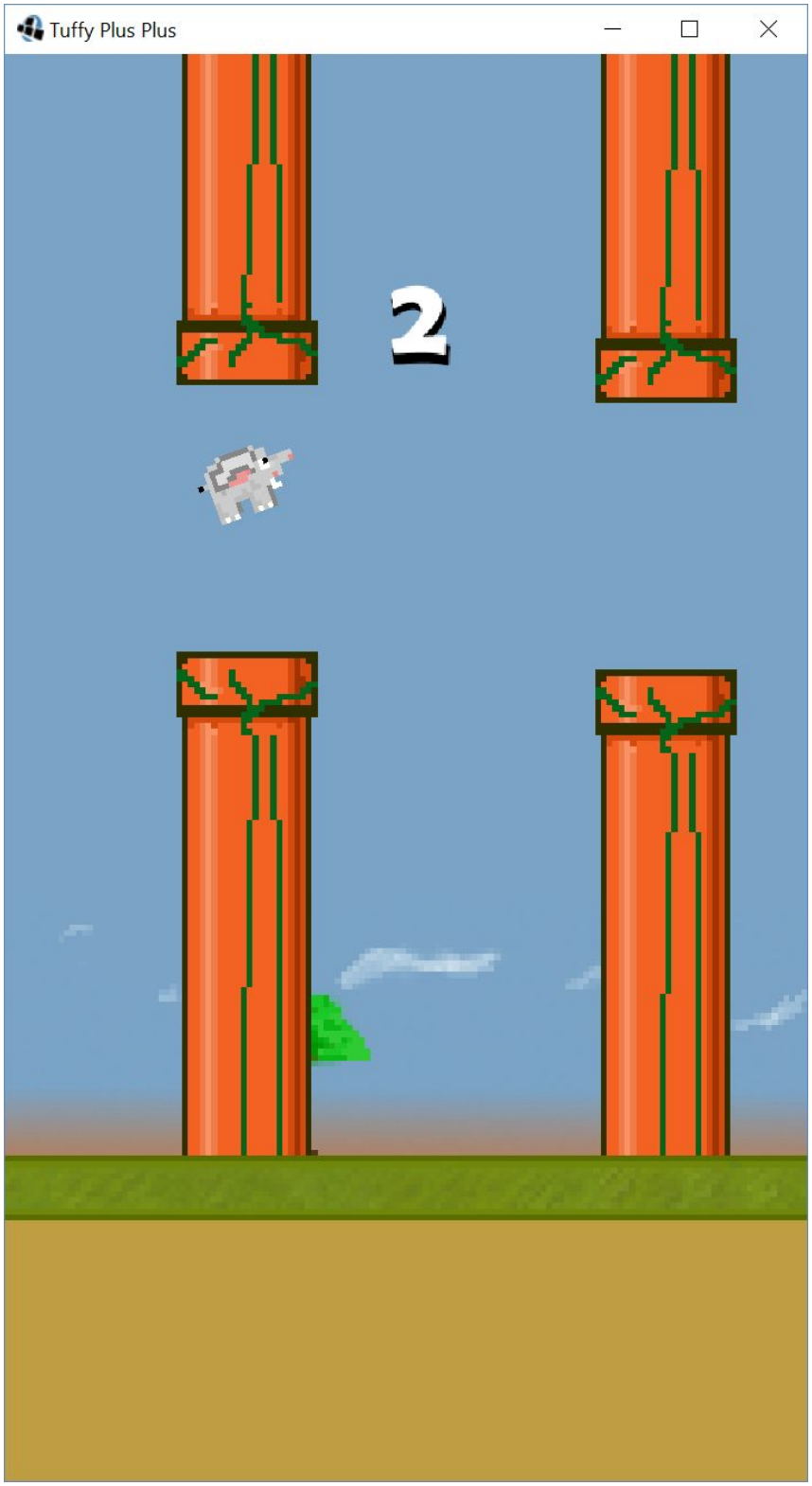
Post-Conditions:
- Background music and sound system works properly in the game

User Manual

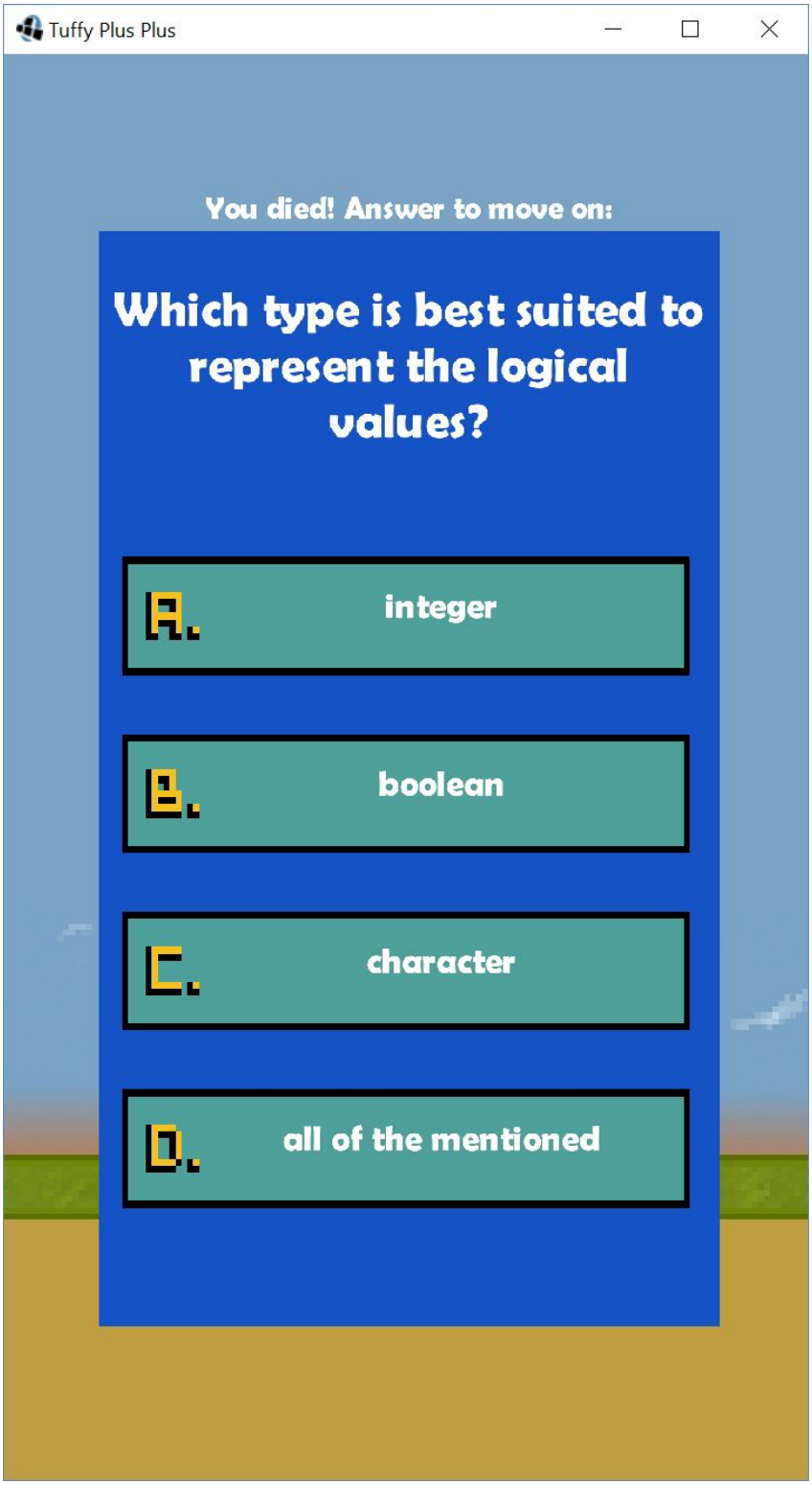
(All screenshots, except for the last one, are taken from the desktop version of the game)

"Start" Screen with 3 Difficulty Options: Easy, Medium, and Tuff.
If you are a new user, type in your name and click OK. Select a Difficulty and tap "PLAY" button to start the game.

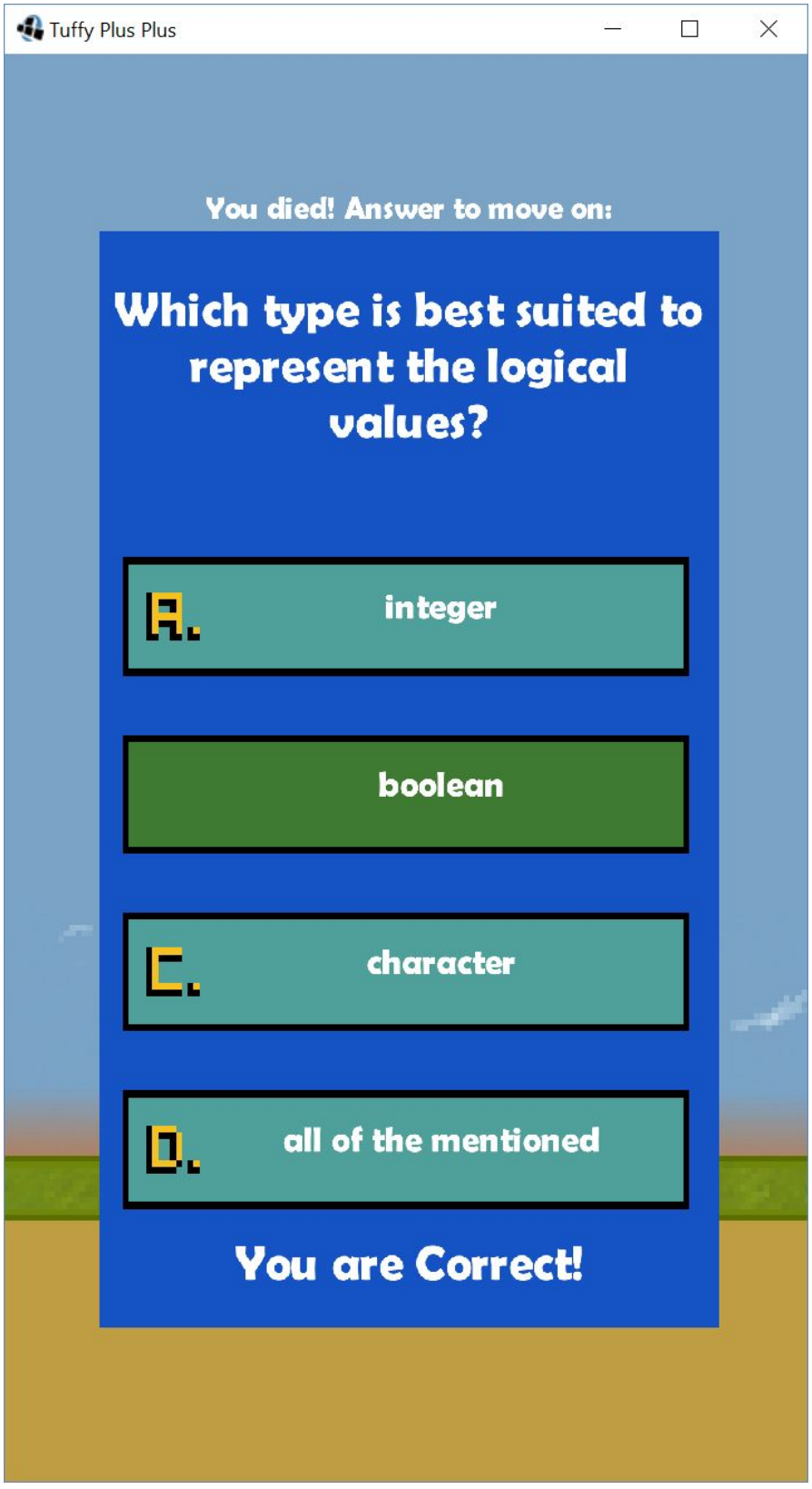
“Ready?” Screen. Tap on the Screen to start playing.



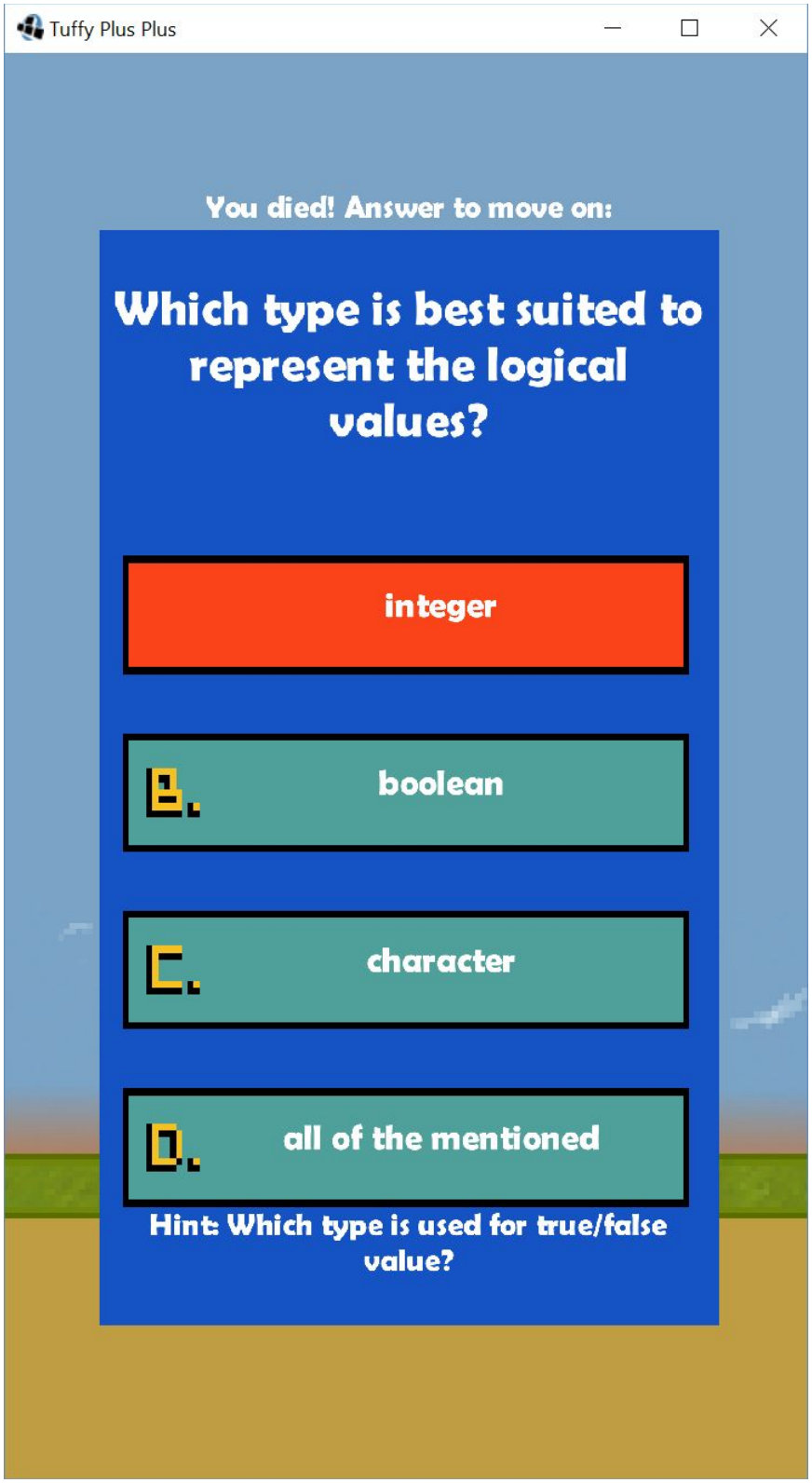
“Game” Screen. Tap on screen to elevate Tuffy to dodge obstructions.



“Question” Screen. Displays the question and the possible answers for the user to choose from



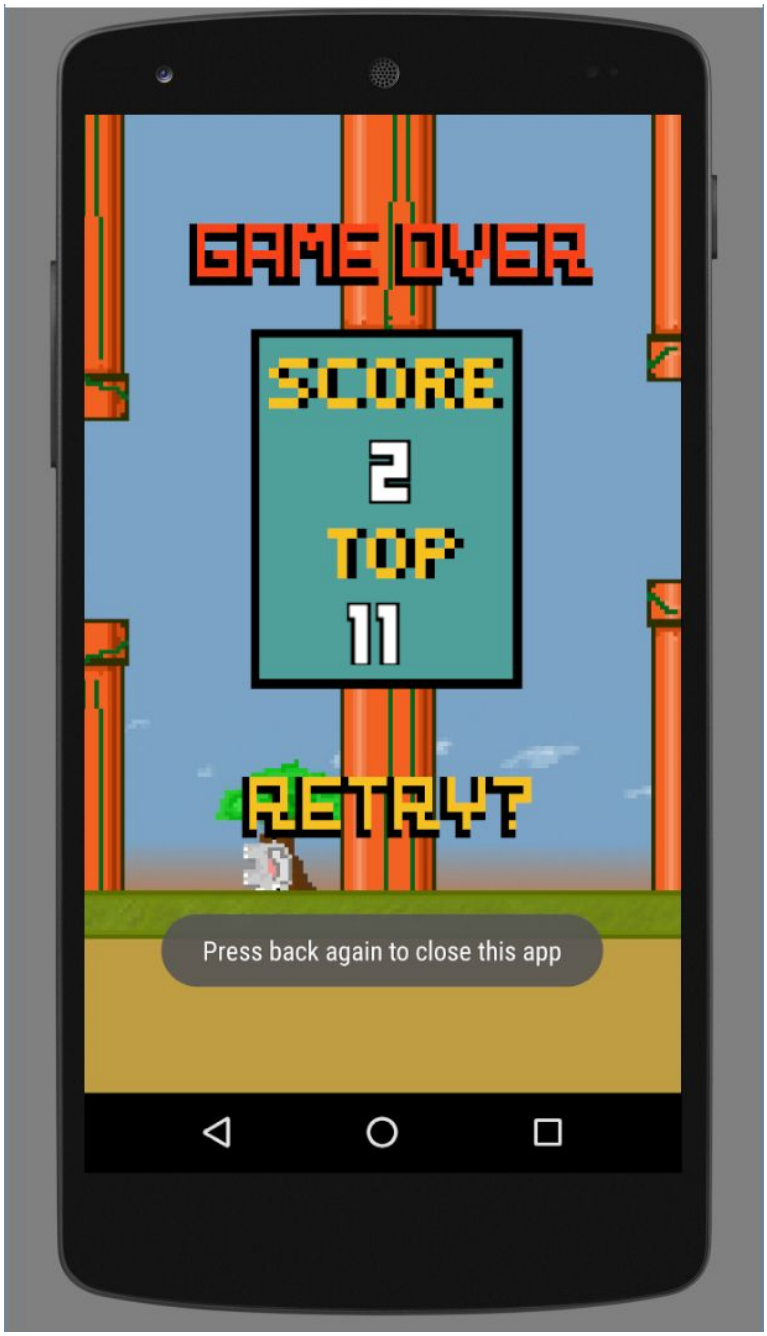
If you answer the question correctly, the correct answer will turn green.



However if you pick the wrong answer, the answer will turn red. Use the hint to help you answer the question!

After answering the question correctly, “Game Over” screen displays a Retry button and the current & top scores

When user's score is higher than current top score, "Game Over" Screen replaces "Game Over" text with "High Score".



If you are playing the Android version, the user can press the “BACK” button twice to exit the game.

References

1. “Day 1-12: Setting up LibGDX.” *Kilobolt*. N.p., 2015, Web. 06 Oct. 2016.

2. Dewalt, Matt, “Elephant Ballerina Stock Photos & Illustrations.” *Pinterest*. N.p., n.d. Web. 06 Oct. 2016

Team Charter

Course Title	CPSC 362	All team members participated in the creation of this charter and agree with its content. **Date** 10/06/2016
Instructor	Sara Ghadami	
Course Dates	8/22/2016-12/16/16	

Team Members (Contact Information)

Name	Address (city, state)	Phone	Cell	Email
Patrick Hovsepian	Fullerton, CA	555-555-555	555-555-555	phovsepian@csu.fullerton.edu
Johnson Lien	Fullerton, CA	555-555-555	555-555-555	johnsonlien95@gmail.com
David Tu	Fullerton, CA	555-555-555	555-555-555	david.tu2@csu.fullerton.edu
Douglas Galm	Fullerton, CA	555-555-555	555-555-555	douglasgalm@csu.fullerton.edu
Susmitha Chittem	Fullerton, CA	555-555-555	555-555-555	susmitha.chittem@csu.fullerton.edu
Thao Nguyen	Fullerton, CA	555-555-555	555-555-555	tnguyen29@csu.fullerton.edu

Team Member Skill Inventory (Areas individual members can contribute)

Patrick Hovsepian	▪ Adobe Photoshop, Illustrator, Premiere ▪ MS Word, Excel, PowerPoint ▪ C++, Java, Python ▪ Project Management ▪ NI LabVIEW, Revit, Visual Studio, AutoCAD Architecture
Douglas Galm	▪ C++, Java, Android Studio, C#, PHP, SQL ▪ Eclipse, Visual Studio, PyCharm
Johnson Lien	▪ Google Docs ▪ C++, Java
David Tu	▪ SQL, C++, Java, HTML ▪ Visual Studio, Eclipse, Android Studio, Microsoft Project, Visio, Word, Excel, Powerpoint, Access, Outlook ▪ SDLC, QA & Release Management, Dashboarding
Susmitha Chittem	▪ MS Word,Excel,PowerPoint ▪ C++,Java ▪ NI Labview,Matlab,Multisim
Thao Nguyen	▪ C++, Java, Android, Python, and some Shell Scripting ▪ MS Word, Excel, PowerPoint, and OneNote ▪ Adobe Acrobat Reader, Photoshop, and Illustrator ▪ Windows 7, 8, and 10; Linux (Ubuntu); and Android ▪ English and Vietnamese

Team Goals (Project goals, team process goals, quality goals, etc.)

- Develop a game that will actually teach C++ in a fun and entertaining way
- Learn more about game development
- Learn how to communicate effectively using a tool other than texting for a group project
- Learn how to document efficiently because industry standards require an expert level of documentation
- Practice communication with stakeholders and developing business skills
- Feel comfortable with a new a group of individuals that we never met before
- Manage a perfect work/life balance while completing this project
- Learn about the System Development Lifecycle and apply software engineering knowledge learned in class

Team Roles (Define roles of each member to achieve goals)

Patrick Hovsepian (Scrum Master)	▪ Helping with the literature that get's outputted to all the group documents ▪ Managing meetings and communications with all stakeholders ▪ Keeping the team organized and responsible for their respective parts and pieces ▪ Managing daily standup meetings ▪ Setting up team meetings outside of class and setting up the agenda
Douglas Galm (Project Manager)	▪ Assisting with document editing and addition ▪ Keeping track of the source code ▪ Managing the entire team's android development environment ▪ Implementing user-stories in every sprint
Johnson Lien (Developer)	▪ Managing final document ▪ Testing working product and looked for inconsistencies ▪ Working on user manuals and technical documentation ▪ Scheduling meetings and locations
David Tu (Developer)	▪ Reviewing current system ▪ Presenting ideas for product enhancements ▪ Working closely with analysts, designers and the rest of the team members with code design ▪ Product testing in controlled environments before going live ▪ Maintaining the system once they are up and running
Susmitha Chittem (Developer)	▪ Helping in reviewing the documents and participating in regular meetings ▪ Presenting ideas in developing the game scenario and setup ▪ Working with all the group members in the implementation of the software and ideas
Thao Nguyen (Graphic Designer)	▪ Creating integrated graphics and visual effects systems that are used in the design and development of the project ▪ Testing codes and systems for animation, audio, memory and streaming features, among other areas ▪ Researching and learning to use new and free graphic and designing tools ▪ Researching the market and communicating with team members for the graphic requirements before designing user interface and user experience

Ground Rules (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

- All team members are highly encouraged to attend meetings, but not required to attend.
- All communication must be done on Slack so logs of conversations may be seen by everybody, unless of emergencies, Texting/Calling/Email is permitted.
- All documents are saved on a cloud-based system so that everyone has an updated version of progress
- To make sure everyone is learning the fundamentals of Software Engineering, our group also encourages class attendance if possible.
- If anyone has any extra skills related to computer science, they are encouraged to share them with the group even if it requires having them to complete more work.
- If any one member has trouble, ask for assistance (can be anything). Communication is key.
- Any objections should be made clearly and efficiently so group can adjust accordingly.
- Conduct any research and report new information to group so everyone can open up options on coding projects.
- If someone cannot attend a meeting, let other members know to prepare another meeting suitable for everyone.
- All team members focus on learning the material to contribute further knowledge for other projects.

Time Commitments/Availability (Pacific Time)

Patrick Hovsepian	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)
Douglas Galm	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)
Johnson Lien	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)
David tu	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)
Susmitha Chittem	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)
Thao Nguyen	▪ Tu, Th 4PM - 6PM ▪ Tu, Th 7PM - 8PM ▪ Tu, Th 6PM - 7PM (Occasionally)

Conflict Management (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

- Schedule time conflictions due to overlapping classes
- Some group members work and go to school so they have very limited availability

Risk Management (What are potential barriers to the achievement of these goals?)

- Environment setup and learning Android Studio IDE.
- Biggest risk: Google's cloud services shutting down and we lose our main data source.
- Back-up: Saving source files and documents on multiple flash drives and computers.

Team Evaluation Criteria (List evaluation criteria that will be used to evaluate team members objectively.)

- Attendance to group meetings
- Willingness to communicate
- Contribution in group meetings
- Work done outside of meetings
- Willingness to share ideas and thoughts