Terminal

Galaxy

Project Report

**California State University, Northridge**

COMP 380/L Introduction to Software Engineering

Summer 2017

**Prepared By Group 1:**Tuan Bui  
Ahmed Al Balushi  
Arthur Zargaryan  
Damian Linares  
Dillon Hour  
Moises Zurita  
Phu Nguyen

July 24, 2017

TABLE OF CONTENTS

**Cover Page**1

**Table of Contents**

**Revision History**

**1. Project Description**

1.1 Project Overview

1.2 The Purpose of the Project

1.3 Project Scope

1.4 Stake Holders

**2. Requirements and Specifications**

2.1 Use Case Diagram and Descriptions

2.1.1 Start Game

2.1.2 Set Controls

2.1.3 Main Menu

2.1.4 Level Select

2.1.5 Level

2.2 Functional Requirements

2.2.1 User Interface

2.2.2 Hardware Interface

2.2.3 Software Interface

2.2.4 Communication Interfaces

2.3 Nonfunctional Requirements

2.3.1 Performance Requirements

2.3.2 Safety Requirements

2.3.3 Security Requirements

2.3.4 Software Quality Attributes

**3. Design3-6**

3.1 System Modeling, System Architecture and Patterns **4-5**

3.2 4 +1 Views**5**

3.3 Design Principle and Design Patterns**6**

**4. Testing6-8**

4.1 Execution Testing**7**

4.2 Non-execution Testing**7**

**5. Project/Process8-10**

5.1 Scheduling**8**

5.2 Cost Estimation**8**

5.3 Product Review**8**

5.4 Product Retrospective**8-10**

**6. Conclusion and Future Work10**

**Appendix A: Glossary10**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date Changed** | **Reason For Change** | **Version** |
| Group 1 | 08/24/2017 | Final Draft | 1.0 |
|  |  |  |  |

**1. Project Description**

**1.1 The Project Overview**

Terminal Galaxy is an independent and self-contained product for the course COMP 380 for the computer science majors. This product is currently in development by a team of 7 members and with a goal and mindset to complete within 6 weeks. Developing the game Terminal Galaxy will offer a great opportunity for each member to enhance or develop new skills that can be applied to the real-world. Tools that will be used to enrich our skill-sets but not limited to, are C#, GitHub, and Unity.

Terminal Galaxy is a fast action packed 2D side-scrolling space shooter game. The user will have the option to command his or her spacecraft with a virtual joystick, or have access to the device’s accelerometers allowing one to use the tilt feature. The user will be challenged with 6 unique levels and distinctive enemies in which a score will be displayed regardless of the outcome. The user will enter a level with goal to eliminate enemies with either lasers or bombs, and with a total of no more than two hits by an enemy or object (asteroid). Completion of a level will unlock a new level and granting the user to continue his or her quest. Terminal Galaxy will be available on multiple platforms PC, Android OS, IOS, and a total new feature on apple called the AR version.

Terminal Galaxy will run on Windows, Mac OS, Android, and IOS platforms, as well as within all major web browsers Chrome, Firefox, and Internet Explorer.  Android platforms will be limited to Operating systems 4.0 (Nougat) or above. User hardware in terms of mobile devices will also be a limiting factor to how well the game runs. Desktop users should have no issues running Terminal Galaxy.

Terminal Galaxy is being produced in a span of six weeks. This will potentially limit optional features we would like to implement. Porting to IOS may prove challenging. User hardware may be a severe limiting factor when it comes to our minimum performance standards for Terminal Galaxy and may limit features.

**1.2 The Purpose of the Project**

The purpose of this project is to create and deliver a fully functional Shoot ‘Em Up style game in the span of six weeks. Basic features such as a system to keep the players scores, a level selector, a variety of enemies and boss battles will be included in this game. The product is intended to run on all major operating system platforms and will be playable on mobile devices as well. Therefore, this document will outline the tools and information needed to start this development.

**1.3 Project Scope**

Terminal Galaxy is a side scroller/top down shooter game. The purpose of the game is for the on-the-go gamer, who does not have a lot of time to play time consuming games. Each player will have a two minute experience on each level with six different levels to beat. The players objective is eliminate as much enemies as possible into order to obtain a high score. Our goals are to release a fully functional product with 6 levels, 6 varieties of enemies, 2 different power ups and a score keeper system in which will be playable along the web browsers and mobile devices

**1.4 Stake Holders**

The primary stake holders of Terminal Galaxy will be Group One Games themselves. Dillon Hour, Arthur Zargaryan, Damian Linares, Tuan Bui, Ahmed Al Balushi, Moises Zurita, and Phu Nguyen are all owners of Terminal Galaxy and they have put in their own assets and work hours to produces this application. We do not have any sponsors or advertisements because this was a small project that was done within a short time period.

**2. Requirements and Specifications**

**2.1 Use Case Diagram and Descriptions**

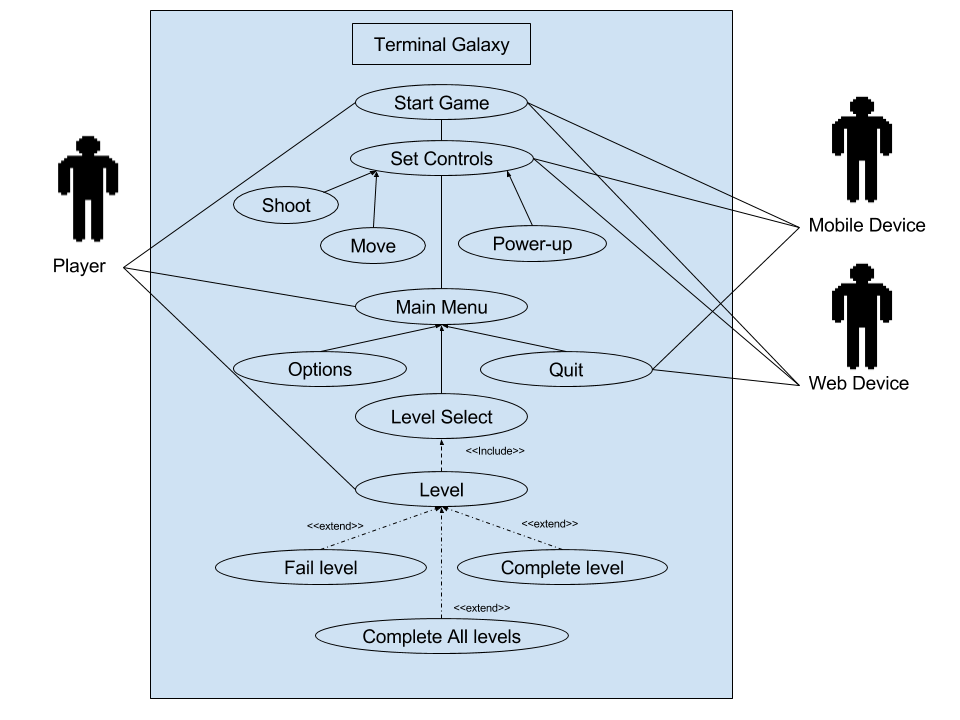
**Use Case Terminal Galaxy**

**Terminal Galaxy**

Description - Starting up Terminal Galaxy on a supported device or browser will load the game in the correct configuration. The player will start on the main menu where they can access the Mission List to load individual levels until the game is completed.

|  |  |  |
| --- | --- | --- |
| **Scenario** | The standard operation and use of Terminal Galaxy from the player and or system’s perspective | |
| **Triggering Event** | When the user loads the game. | |
| **Actors** | Player - Users playing the game  Mobile Device - The device they are playing on  Web Browser - The medium they play the game on | |
| **Related Use Case** | N/A | |
| **Stakeholders** | N/A | |
| **Pre-condition** | The user must be on a browser capable of running Javascript and WebGL  Or the user must have an Android phone of API 25  Or the user must have an iPhone of API TBD | |
| **Post-Condition** | Load Keyboard controls if on a browser  Touch controls if on phone | |
| **Flow of Events** | **Actor** | **System** |
| 1. Start the game  3. Select “Mission List”  5. Select a Level to play  7. Intro Scene to Level  9. Play the level  11. Complete the level  13. Plays next level  15. Completes the game  17. Exits the game | 2. Load the main menu and set correct controls  4. Load a list of levels unlocked or level 1  6. Load Selected Level  8. Start Gameplay  10. Wait for completion of level  12. Unlock next level and saves  14. Loads nexts level until all have be unlocked  16. Returns to main menu |
| **Exception** | |  |  |  | | --- | --- | --- | | **Step** | **Condition** | **Action Description** | | 2a. | If step 2 fails | The game will unload and return to OS | | 4a | If step 4 fails | Unlock all levels | | 6a | If step 6 fails | The game will unload and return to OS | | 8a | If step 8 fails | The game will unload and return to OS | | 10a | If player fails | Load GameOver UI, return to 8 or 2 | | 10b | If step 10 fails | The game will unload and return to OS | | 12a | If step 12 fails | Unlock all levels | | 14a | If step 14 fails | The game will unload and return to OS | | 16a | If step 16 fails | The game will unload and return to OS | | |

Terminal Galaxy is a side scrolling shooter designed for the web and mobile environment.  The game will be loaded on the local environment and once Start Game is triggered it will trigger Set Controls event.  Our Move(), Shoot(), PowerUp() will be using the Cross Platform Input manager class to allow the code to be versatile enough to be used for both web and mobile applications.  Once the controls are set for the environment the game will load the Main Menu. The Main Menu will be central hub of the game where the user can select Level Select, Options, and Quit.  Level select is the list of levels the player has unlocked in their play through or just the first level if it is a new game.  Once a level is completed the next level will unlock but if the player fails then the player must repeat the level to progress.  This will progress until the player unlocks all the levels and beats the game in its entirety.

****

## 2.1.1 Start Game

**Description and Priority:**

High priority. Initializes application.

**Stimulus/Response Sequences:**

Called when user initializes the application through a web browser or app on mobile device.

## 2.1.2 Set Controls

**Description and Priority**

High priority. Checks for presence of a mobile or WebGL version of the game. Depending on the version, sets Move(), Shoot(), and PowerUp() accordingly.

**Stimulus/Response Sequences**

Set Controls is called when the application is first initialized.

**2.1.3 Main Menu**

**Description and Priority**

Main interface that player will use to navigate. Will contain Options, Level Select, Quit, Continue, and New Game buttons.

**Stimulus/Response Sequences**

Main menu will be initialized after Start Game and Set Controls are called

**2.1.4 Level Select**

**Description and Priority**

Level Select interface is where the player will choose which level to play.  This will also hold the player’s progression since more levels will be unlocked as the player completes more levels until finished.  The Priority of Level Select is medium, while it is important it is not the core aspect of the program as whole.

**Stimulus/Response Sequences**

The Level Select interface will access the save data the user has stored on the device.  If there is no data saved then the only level that the player can select will be Level 1.  When the player completes a level they will unlock the next level and this progression will be saved.  If save data exists then the player will have multiple levels to select.

**2.1.5 Level**

**Description and Priority**

The Level is the core of game play for Terminal Galaxy.  The level be designed for a 2 minute experience with 10 seconds dedicated to the introduction scene and the exit scene.  This is the highest priority for the project since it is the core of game play.

**Stimulus/Response Sequences**

The Level will be very interactive, the Move(), Shoot(), and PowerUp() will become key in if the player completes the level or fails the level.  If the player’s health reaches zero or less the Level will fail requiring the player to retry the level.  If the player completes the level then the next level will unlock but if he has all levels unlock he will return to the main menu.

**2.2 Functional Requirements**

WEB REQ-1: Touch Screen or Keyboard

WEB REQ-2a: Firefox Browser

WEB REQ-2b: Chrome Browser

WEB REQ-3: Javascript

WEB REQ-4: WebGL

WEB REQ-5: Available RAM to load game

iOS REQ-1: iPhone iOS

OS REQ-2: Touch Screen

iOS REQ-3: Available memory to install game

ANDROID REQ-1: Android 7.1.1 API 25 Rev 1 or greater

ANDROID REQ-2: Touch Screen

ANDROID REQ-3: Available memory to install game

## 2.2.1 User Interfaces

The user will have access to several menus. The main menu will contain a level select button, new game button, high scores button, options button,exit button, and a continue game button. Main menu will load upon launching the game. When the user pauses the game a menu with restart level and exit level buttons will be displayed. Upon player death a menu with exit and restart buttons will be displayed.

## 2.2.2 Hardware Interfaces

* Android Device - With about 80% of all mobile devices being Andriod means this is a necessary requirement to meet a large audience.  Android is a platform made by Google and Version 7.1.1 Rev 1 devices meets the needs to render and play the game at an acceptable performance.
* iPhone - Apple’s iOS makes up 18% of all mobile devices with the iPhone being the most popular iOS device.  Support for this device is TBD.
* Touch Screen - The touch screen is the most common control interface for mobile devices today.  Terminal Galaxy will take advantage of this commonality by using simple and streamlined controls to reduce complexity and cluster for a clean interface.
* Keyboard - The most standard interface for web browsing on a personal computer.  This will be our default controller for all gameplay using a web browser.

## 2.2.3 Software Interfaces

* Chrome - Of all browsers Google’s Chrome web browser makes for over 50% of all usage.   Chrome supports large variety of plugins and a minimalistic design. Chrome also supports HTML5 which is required to use WebGL.
* Firefox - The Mozilla Corporation created a free and open-source browser called Firefox which accounts for 10% of all usage.  Terminal Galaxy makes heavy use of WebGL which is also made by Mozilla, which makes Firefox the most compatible browser for Terminal Galaxy.
* WebGL - WebGL is a Web Graphics Library built on the JavaScript API used to render interactive objects that include both 3D and 2D graphics on compatible web browsers such as Mozilla Firefox or Google Chrome.  This makes it so the user does not require any plug ins for the browser because it makes use of the HTML5 elements and OpenGL ES 2.0.  By using an existing and proven library for the basis of the game we can ensure that time is spent more on gameplay development and that Terminal Galaxy has the reliability that gamers today demand.
* Javascript - Terminal Galaxy will require Javascript to load WebGL and will be the initial check to see if the browser is compatible with our software.
* iOS - All Apple mobile devices such as iPad and iPhone run iOS.  We will be running our application on iOS so we can deliver a product to Apple users as well.  Support for this device is TBD.
* Android Operating System - The Android Operating System must be of Android 7.1.1 Nougat API level 25 revision 1 or greater to run the installed software.  Released on August 22 2016 increased the versatility of the development platform.  The platform switched to an OpenJDK environment and also has support for Vulkan graphics API. This operating system minimum requirement also ensures that we have a device capable of running Terminal Galaxy.

## 2.2.4 Communications Interfaces

There will be no communications interfaces.

**2.3 Nonfunctional Requirements**

## 2.3.1 Performance Requirements

The game will run in the 30-60hz range (30-60 frames per second).

## 2.3.2 Safety Requirements

There will be no safety requirements.

## 2.3.3 Security Requirements

Security features will not be implemented and there will be no user authentication.

## 2.3.4 Software Quality Attributes

**Nonfunctional Reliability -**

In order to ensure the user has a level of consistency-play in the game Terminal Galaxy, a high level of reliability will help accomplish minimal interruptions between, and during gameplay. Three areas that are focal points: Probability of failure on demand (POFOD), Rate of occurrence of failures (ROCOF), and Availability (AVAIL).

POFOD: The reliability metric Probability of Failure on Demand (POFOD) is a metric that defines the probability that a demand for service from a system will result in system failure. The game will demand services from: tilt axis controls, processor, memory, storage, user interface (touch-screen), battery, operating system. The projected goal is to have a greater than 90% reliability rate. To measure the reliability the Mean Time between Failure formulas will be used:

|  |  |  |  |
| --- | --- | --- | --- |
| Mean Time Between Failure | MTBF | (Hours / Failure Count) | Average length of time an application runs before failing. |

ROCOF: The reliability metric Rate of Occurrence of Failures (ROCOF) sets out the probable number of system failures that are likely to be observed relative to a certain time period.

AVAIL: The reliability metric Availability is the probability that a system will be operational when a demand is made for service. It is an industry norm for software to continuously have future software updates for improvement; therefore servers will need to be accessible. These include but not limited: updated operating system, newly discovered/reported software bugs, update game (features: adding new levels, multiplayer, global high scores, stability performance), re-download game due to data corruption. To calculate this metric the following formulas are used to measure availability:

|  |  |  |  |
| --- | --- | --- | --- |
| Mean Time Between Failure | MTBF | (Hours / Failure Count) | Average length of time the application runs before failing. |
| Mean Time to Recovery | MTTR | (Repair Hours / Failure Count) | Average length of time needed to repair and restore device after a failure. |
| Availability | AVAIL | (MTBF / (MTBF + MTTR)) \* 100 |  |

**Nonfunctional** **Robustness –**

It is a great quality attribute that has a great chance of previewing Terminal Galaxy’s behavior in certain situations. 3 factors have been taken into consideration in determining robustness and they are as following, the reset time, chances of failure and data loss rate. in these situations, the percentage rate is an estimation and does not hold final value. As estimated, in a case of application crash, the time it will take for it to reset is not going to exceed 30 seconds. Meanwhile, this estimates the application of having a 25% chance of failing. In case of application crash, data loss has a rate of 10%.

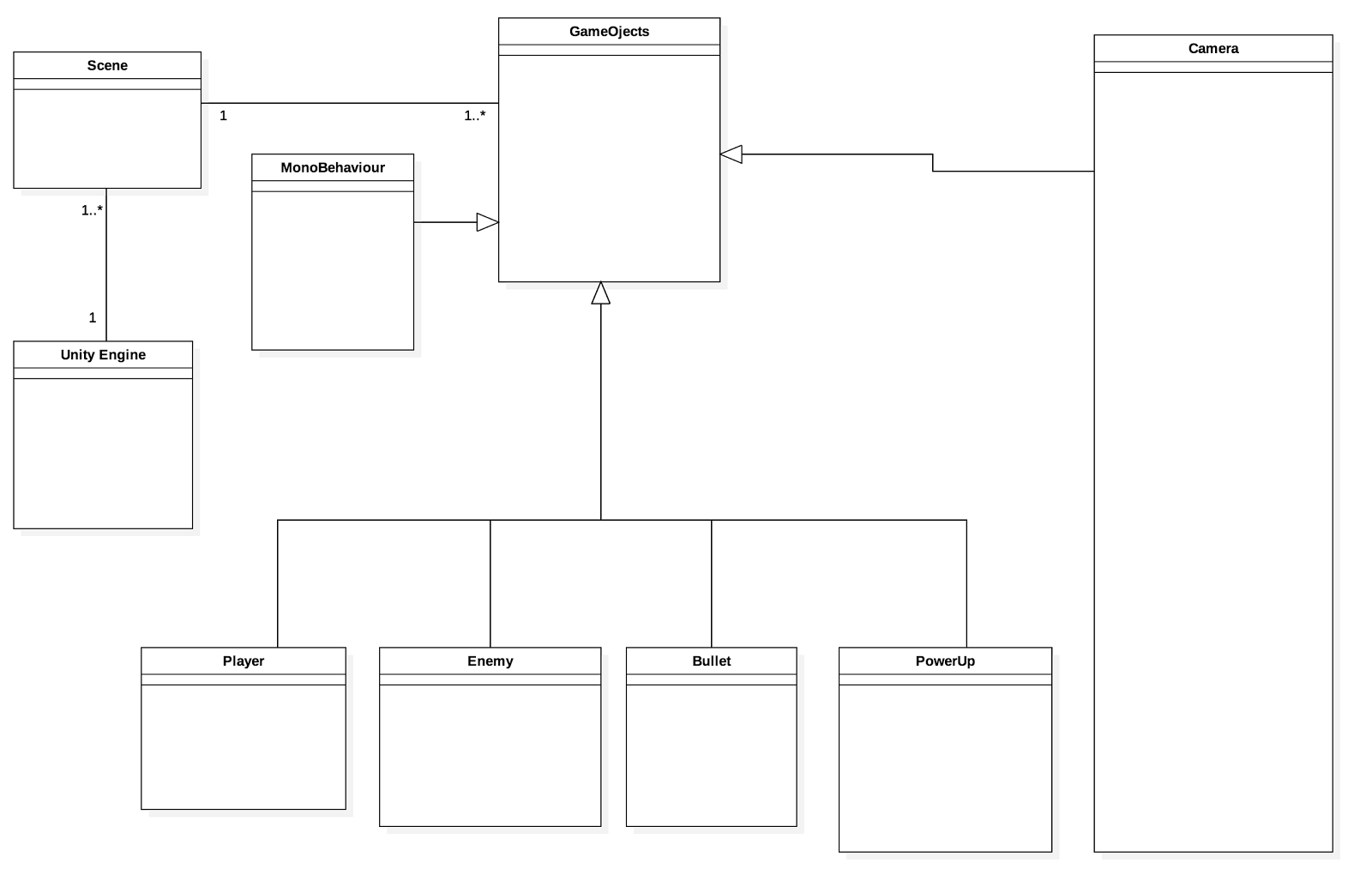
In the first factor of reset time process, the chances of the application activity being interrupted by any other sort of application is rather apparent, which can cause the application to crash. Whether it is a phone call on the phone or a web crash, in this scenario, when the user decides to re-launch the application, this whole process time will not exceed the indicated time of 30 seconds. This factor is very much related to the next factor of chances of failure. Even though, some activities might not cause the application to crash or stop functioning. The failure rate however, is estimated to be around 25%. Major factors have been taken into consideration that can cause the application to fail, including activity interference, others like the system processing strength or the operating system of the phone running the application, yet there are many other factors have been neglected, the rate of failure can decrease at the end with accurate measures.

The third factor is the percentage of data loss. Eventually, when the application stops working, data of the application might get lost. To be specific, a player might be playing on Terminal Galaxy, when some activity might cause the application to stop working, whatever level that user was on or achievements earned may not be saved and the user would have to repeat the level when the application is restarted. There is however a 10% chance the data might get lost in the case the application encounters any failures or when it crashes.

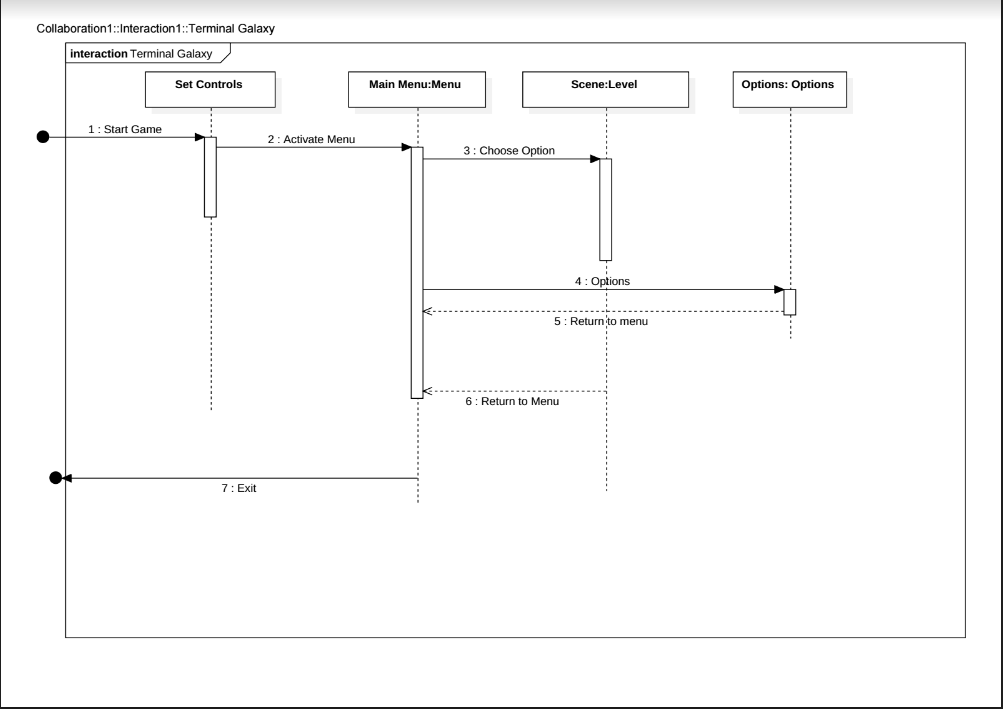
**3. Design**

**3.1 System Modeling, System Architecture, and Patterns**

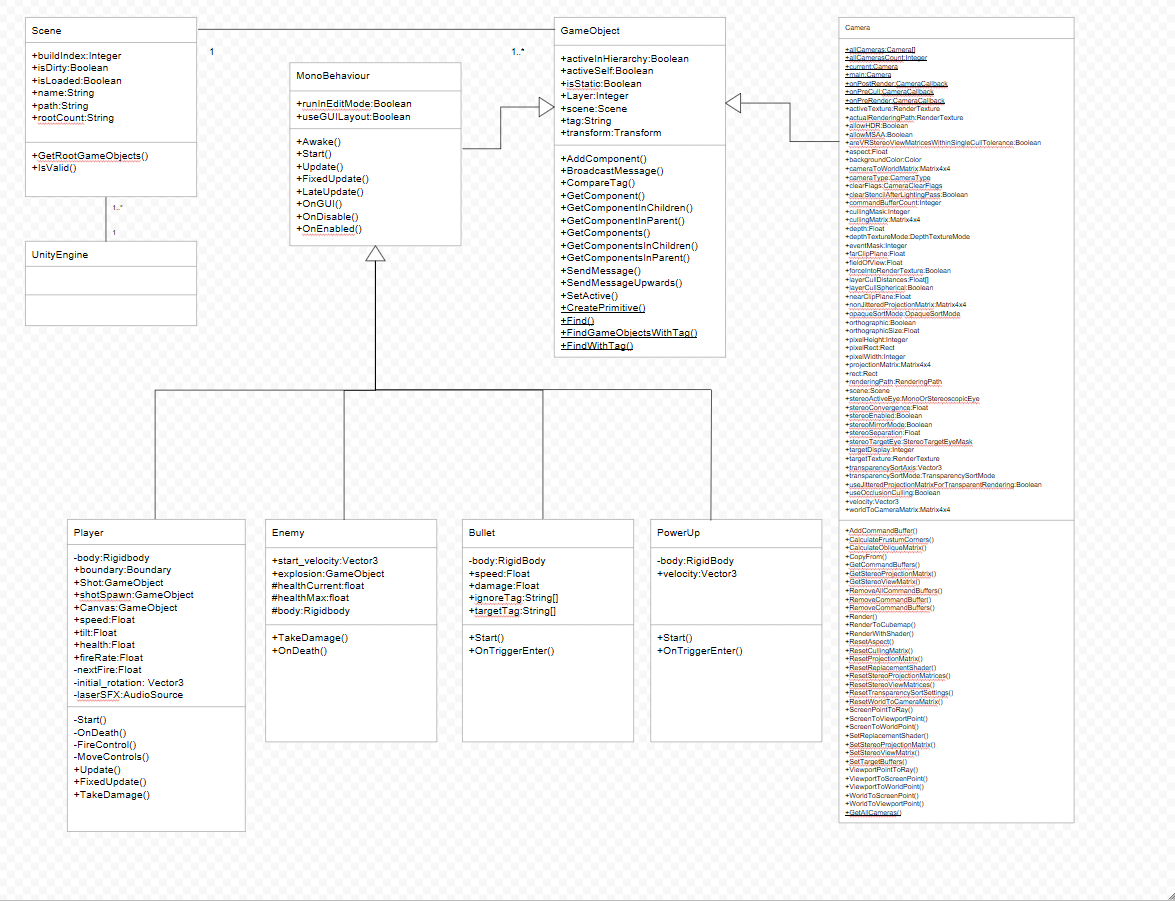
**Class Diagram with Relationships:**

**System-level Sequence Diagram**

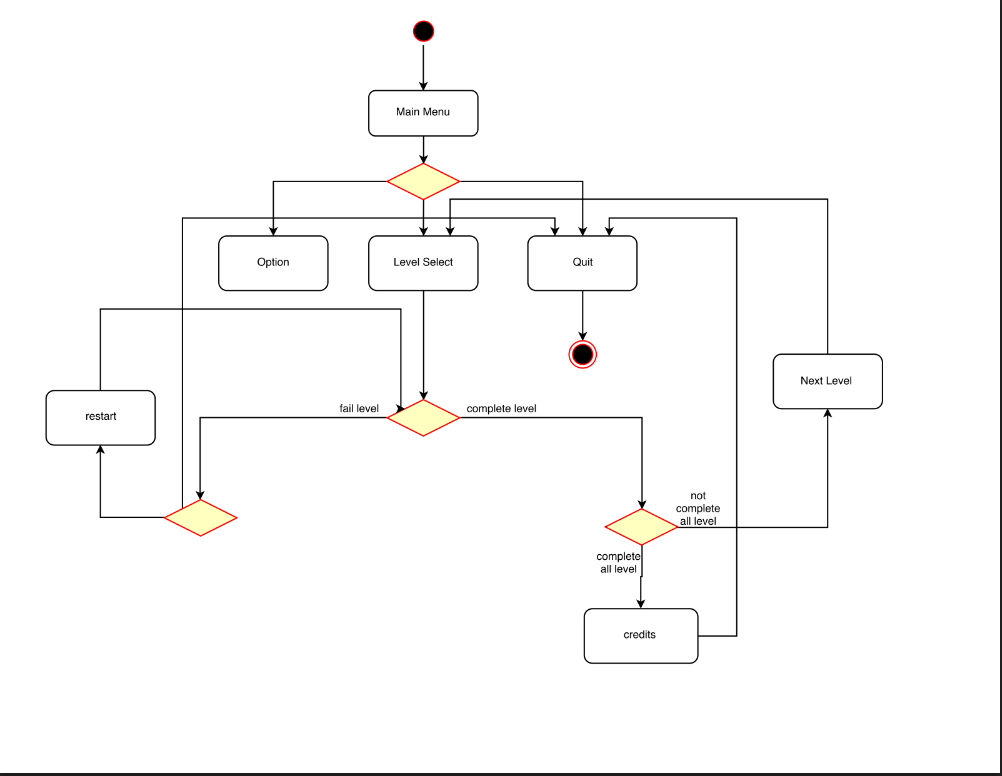
**System-level Sequence Diagram**



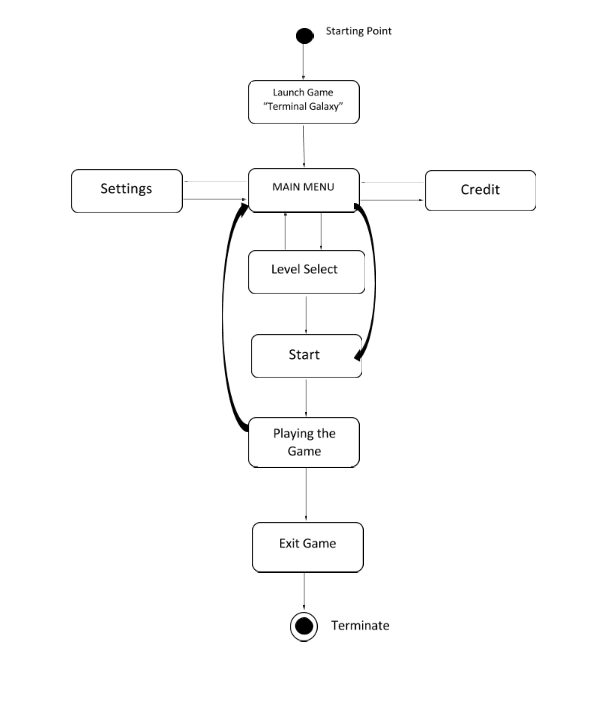
**Detailed Class Diagram**



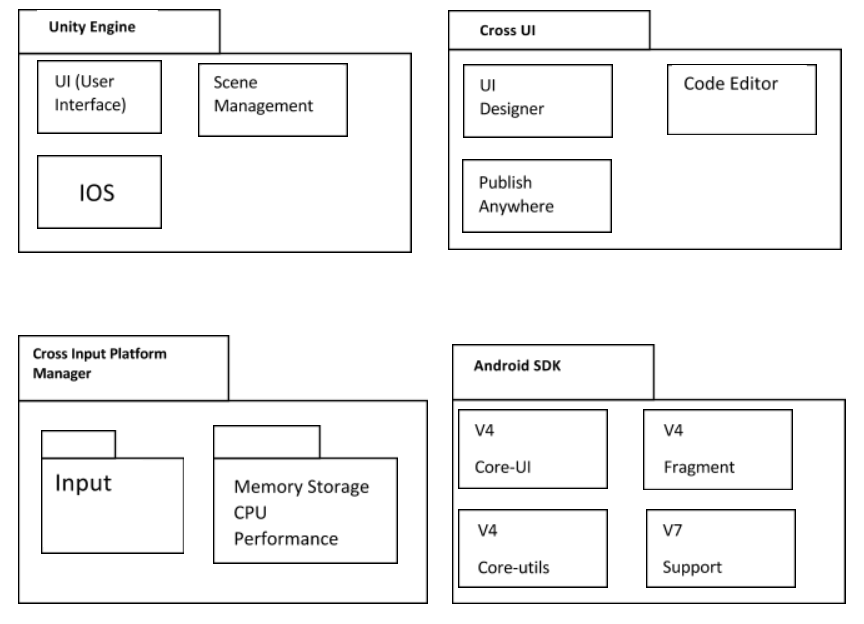
**System-level Activity Diagram**



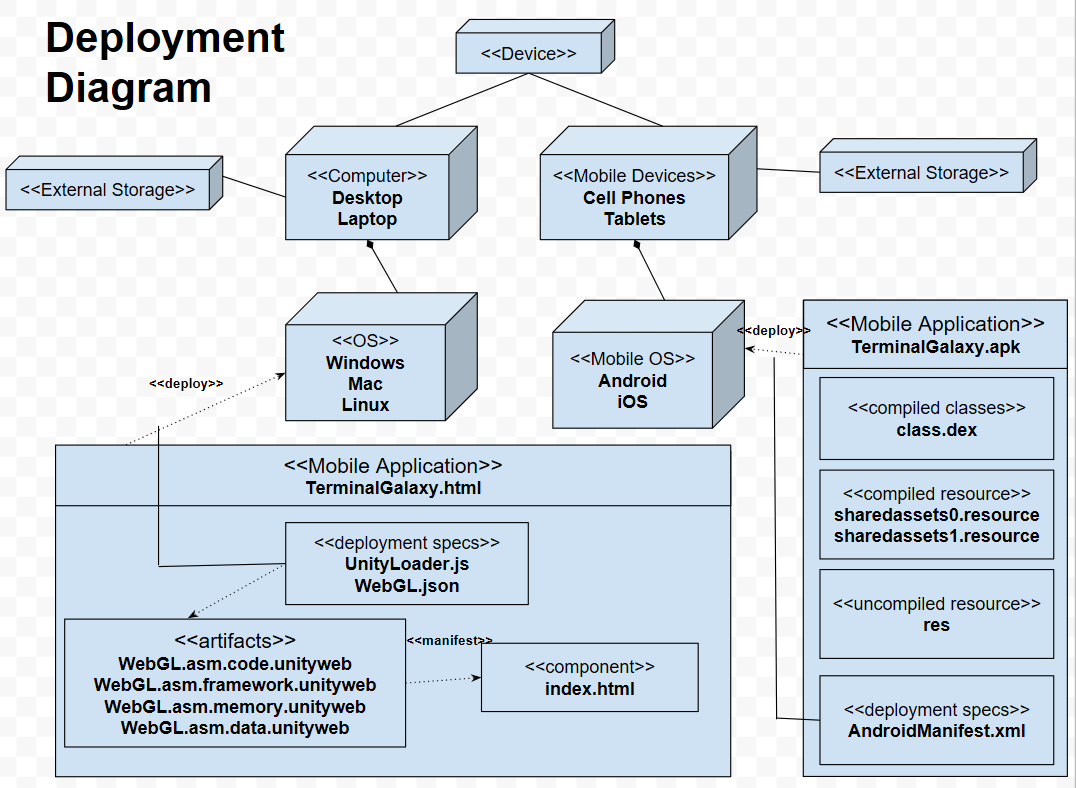
**System-level State Chart Diagram**



**Package Diagram:-**



**Deployment Diagram**



**3.2 4 +1 View Model**

Deployment Diagram

Package Diagram

Component Diagram

Class Diagram

Objects Diagram

Sequence Diagram

LOGICAL VIEW

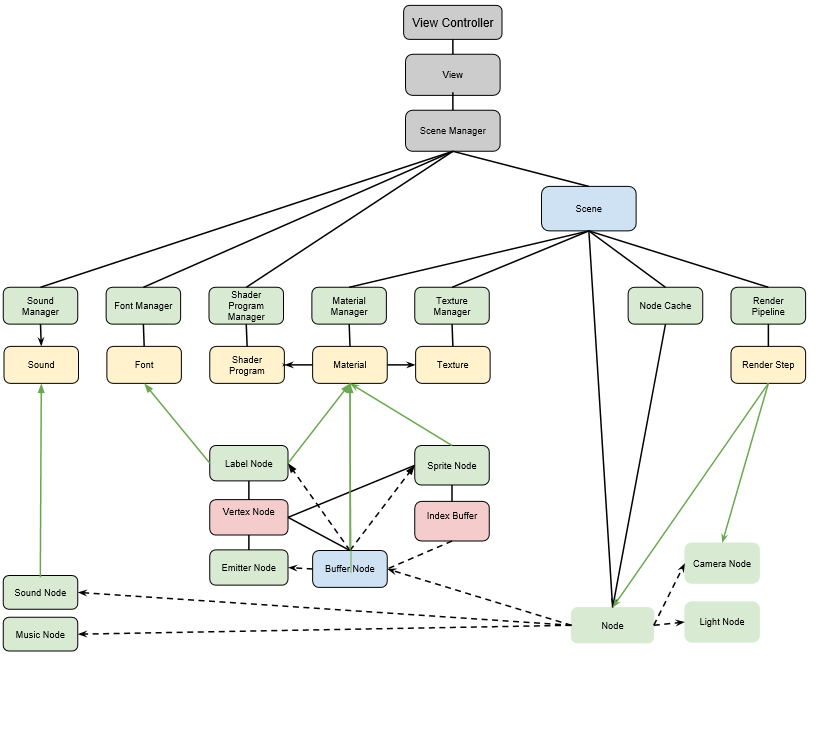
PROCESS VIEW

IMPLEMENTATION VIEW

PHYSICAL VIEW

SENARIO VIEW

**3.3 Design Principles and Design Patterns**



**4. Testing**

**4.1 Execution Testing**

**Test Plan Identifier**

Group One Games has composed a master plan in which goals were both set and completed. The application has been tested and modified several times using GitHub. GitHub is a program that the developers/programmers use to communicate and share codes.

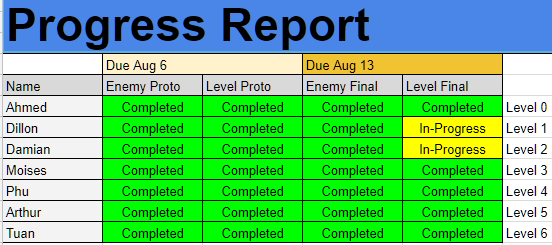
Throughout this process, each programmer were each assigned to create a level and to complete a certain game task assigned to them; such as the main menu, pause control, gamer over control, ect..

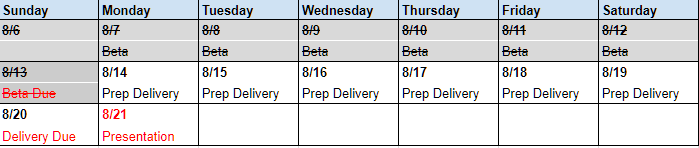
The programmers would submit changes and add new additions for the application and post it onto GitHub, so that other programmers could download it and test the changes themselves. The revision numbers are based on the number of merges and commits made by the team. Therefore, the team has about:

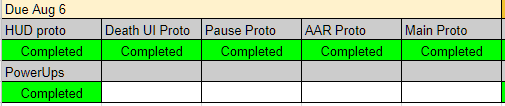
Merges: #262

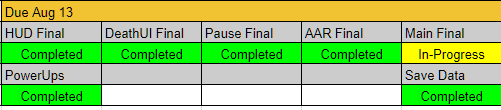
Commits: #794

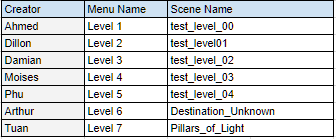
**Test Items**

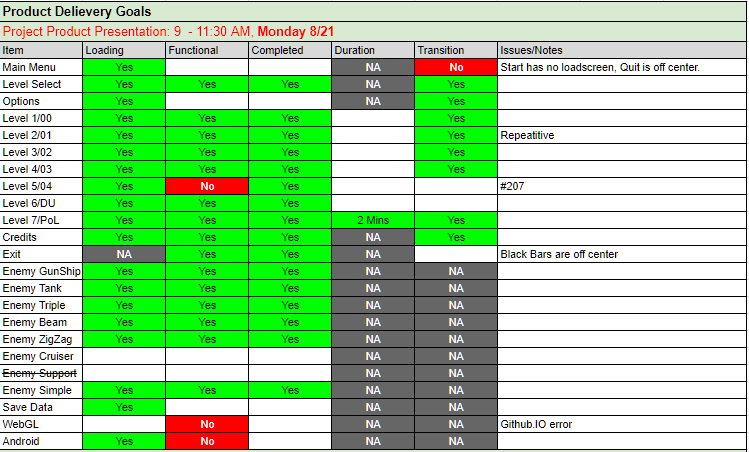
****

****

****

****

****

****

**Software Risk Issues**

The risk that we will be encountering is data management which deals with the load, save, and erase data. This would be an issue for the user if the game does not load, or data becoming corrupted.

There will be no third party products involved, which will eliminate the need for any third party support for any software issues.

**Features To Be Tested**

The most important thing that needs to be tested is the level because the player will have a two minute experience and we want to make sure that there are no bugs inside the level.

Next we want to make sure that the game play is fully functional and fun to play. We need to make sure all the loading screens are working properly and if the sounds are working properly. Lastly, we need to make sure that the user interface is fully functionally and easy to use and understand.

**Approach**

Our main approach is to make sure the game is playable and usable throughout the whole game play experience. We have used GitHub, Discord, and Google Doc/Sheets/Hangouts to communicate and keep track of each other progressions.

**Item Pass/Fail Criteria**

Below will be a list of question we ask each other to make sure that the game has the minimal amount of bugs and errors possible. These questions include:

* Does it load?
* Do all the functions function?
* Is it complete?  (Requirements and Specifications)
* Continue/New game: Does the game function from beginning to end?
* Level Select: Does game progression work?
* Does fade in work?
* Assert checks
* Do all UI functions work?
* Functionality (Player movement, death, etc.)
* Does it meet specifications?
* Can you complete the level?
* Does it transition ok? (Main menu-->Next level-->Quit etc.)
* Does every level load properly? Is it functional? Does it meet requirements?
* Does it save the data?
* Does it run properly on mobile?
* Does it run properly on WebGL?

**Suspension Criteria and Resumption**

**Requirements**

Script warning will be acceptable, but errors will not be acceptable and will be fixed right away once discovered.

**Test Deliverables**

Things that will be used for deliverables include:

* Google Docs/Sheets (will be used to compose documents and deliver specific instructions)
* GitHub (Allows programmers to share and add codes)
* Google (For research)
* Unity (Game designing and scripting)
* Google Play Store Documentations
* Errors logs will be attach to asserts

**Environmental Needs**

There will be some requirements that the user must and some these include:

* An up to date web browser that run WebGL
* Mobile Device
* Internet Accessibility
* Android 6.0 or higher
* Dual Core Processor

**Staffing and Training Needs**

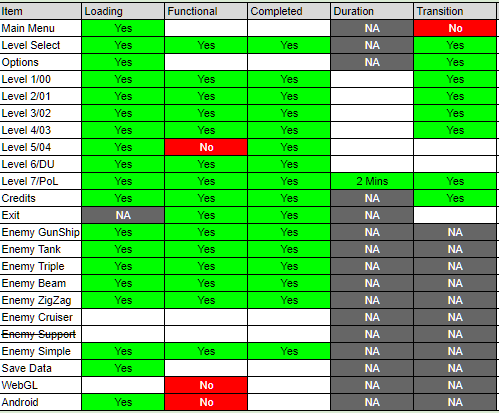
Each member will be trained by the project lead; Tuan Bui. He will be training the team in:

* Unity
* C#
* GitHub
* Google Docs/Sheets/Hangouts
* Discord

**Responsibilities**

Each member in the group will have been assigned an individual task and will also be assigned a universal task. The universal task includes creating a unique enemy and level.

The individual task varies on the programmer. Below will be a list of task that was completed throughout this developmental process.



**Schedule**

Communication is very important since time was limited. We met four times a week and had online meetings on Discord or Google Hangouts. Tasks were assigned on GitHub.

Schedule:

Face to Face Meetings:

Monday-Thursday: 9AM-1PM

Online Meetings:

Friday-Sunday: All day

**Planning Risks and Contingencies**

We ran into several issues throughout this process these included:

* GitHub merging conflicts (possibility of wiping another's work progress)
* Time and implementations (Time was limited)
* Difficult task might take longer
* 2D game
* Risk of saving builds that are not fully functional
* Reverting to beta phases
* Submitting beta phasing

**4.2 Features Not To Be Tested**

Debug levels will not be test. However, they will be used to fix the actual levels. Additional script warning will not be test as well, instead will be using null references or debug logs to get rid of these warnings.

**5. Project/Process**

**5.1 Scheduling**

Terminal Galaxy was designed for a 6-week production cycle from planning to release. The scope of the project was limited so that the product would not only be finished but polished with in those 6 weeks. Considering that may members were new to the tool required for the production of Terminal Galaxy, there was a whole week set aside for just learning tools. The first week was simple task to test the skills learned, the player prototype. The next was originally planned to be for the 6 unique enemy prototype but was changed to be two weeks and include the level prototype as well. Week 2 and 3 was dedicated in the creation of the enemy and level prototype which is the first fully playable version of both. Week 4 was dedicated to finish up all content and get the game playable from beginning to end, the Beta Phase. Week 5, was used to finalize the whole game, as well as testing and bug fixing. Week 6 is release on the Google Play Store.

* **Week 1: Player Prototype**
* **Week 2 + 3: Enemy/Level Prototype**
* **Week 4: Beta Phase**
* **Week 5: Finalize and Testing**
* **Week 6: Submission**

**5.2 Cost Estimation**

Terminal Galaxy used only free assets and copyright free music to reduce cost as much as possible. The submission to the Google Play store required a fee of 25 dollars. The total work hours are estimated at 746 hours but the work effort costed nothing since this was a class assignment. The calculated KLOG per person-month and the cost function points per person-month is estimated at 3.227 KLOG.

Total Costs: $25

**5.3 Product Review**

Terminal Galaxy exceed original planned expectations. In 6 weeks Group One Games was able to produce 8 levels (2 more than planned), 11 enemies (5 more than planned), and 3 boss battles.

**5.4 Product Retrospective**

Terminal Galaxy is a limited game and needs more context to stay competitive in the mobile gaming environment. This can only be solved with more development time. The game does have a good foundation to expand on and can easily add more content when the development continues.

**6. Conclusion and Future Work**

Terminal Galaxy was a great entry for our first game. It helped the whole team get experience in working with other programmers in a large project. The design methodology learned well help in any future works. The next game made will have a much stronger foundation and speed up production by a large scale. A few of us plan on producing an Augment Reality version of Terminal Galaxy for iOS. While others plan on a using what we learned to develop the Senior Project. While growing pains were expected as a group we were all able to overcome this and contribute to the product as a hole.

**Appendix A: Glossary**

* **Accessibility:** Hardware and software technologies that help visually or physically impaired people to use the computer
* **UI (User Interface):** the means by which the user and a computer system interact, in particular the use of input devices and software.
* **Merges:** Merging takes the changes from one branch (in the same repository or from a fork), and applies them into another. This often happens as a pull request (which can be thought of as a request to merge), or via the command line. A merge can be done automatically via a pull request via the GitHub web interface if there are no conflicting changes, or can always be done via the command line
* **Commit:** A commit, or "revision", is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. Commits usually contain a commit message which is a brief description of what changes were made.
* **Comprise**: is defined as any computing resource whose confidentiality, integrity or availability has been adversely impacted, either intentionally or unintentionally, by an untrusted source. A compromise can occur either through manual interaction by the untrusted source or through automation.
* **Data Corruption**: Data corruption is a when data becomes unusable, unreadable or in some other way inaccessible to a user or application. Data corruption occurs when a data element or instance loses its base integrity and transforms into a form that is not meaningful for the user or the application accessing it.
* **Interface:** a device or program enabling a user to communicate with a computer.
* **Bugs**: A software bug is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.
* **Debug**: Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.
* F**unctionality** Asserts: an assertion is a statement that a predicate (Boolean-valued function, a true–false expression) is expected to always be true at that point in the code. If an assertion evaluates to false at run time, an assertion failure results, which typically causes the program to crash, or to throw an assertion exception.
* **Specifications**: A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform.
* **API** - Application Programming Interface
* **AVAIL** - Availability
* **HTML5** - HyperText Markup Language 5
* **MTBF** - Mean Time Between Failure
* **MTTR**- Mean Time to Recovery
* **OpenJDK** - Open Java Development Kit
* **TBD** - To be determine
* **RAM** - Random Access Memory
* **WebGL** - Web Graphics Library