Software Requirements Specification

for

Duck Tower Defense

Version 1.0 approved

Cody Chu, William Robichaux, Ezekiel Velasquez, Lucas Riccitelli, Brian Larosiliere, Emory Salaun, Gabriel Badolato

Team K

9/19/2022

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 2

1.1 Purpose 2

1.2 Document Conventions 2

1.3 Project Scope 2

1.4 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 User Classes and Characteristics 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 Assumptions and Dependencies 2

3. System Features 2

3.1 System Feature 1 2

3.2 System Feature 2 (and so on) 2

4. Data Requirements 2

4.1 Logical Data Model 2

4.2 Data Dictionary 2

4.3 Reports 2

4.4 Data Acquisition, Integrity, Retention, and Disposal 2

5. External Interface Requirements 2

5.1 User Interfaces 2

5.2 Software Interfaces 2

5.3 Hardware Interfaces 2

5.4 Communications Interfaces 2

6. Quality Attributes 2

6.1 Usability 2

6.2 Performance 2

6.3 Security 2

6.4 Safety 2

6.5 [Others as relevant] 2

7. Internationalization and Localization Requirements 2

8. Other Requirements 2

Appendix A: Glossary 2

Appendix B: Analysis Models 2

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Whole team | 9/19/22 | Basic start to our project using libGDX | 0.1.0 |
| Whole team | 10/17/22 | Updating features and Stimulus Response | 0.2.0 |
| Whole team | 10/31/22 | Updated to comply with the editor’s notes | 0.3.0 |
| William | 11/21/22 | Updated the 3.X part of the document to comply with the editor’s notes and cleaned up the formatting | 0.4.0 |
| William | 11/23/22 | Added Towers as a feature in 3.X | 0.4.1 |
| Whole Team | 12/13/22 | Completed any blank sections and finalized the document | 0.5 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This SRS is organized so that readers will be able to fully understand the scope and purpose of our project, in addition to explaining overview of the project, it will provide a surface level look into the inner workings of our game. The document begins with describing the purpose and scope of the project and introducing any outside references used in it. In section 2, more emphasis is placed on describing how our program will fit together to work. In section 3, the systems features are listed, whether they are necessary features, or quality of life features. Section 4 and 5 describe how data/users interact with the program. Section 6 describes the quality aspects of our program.

## Purpose

This guide is intended to be read by both developers and users. For the developer, it will hopefully explain the general structure, data types, conventions, and flow of the program. For the user, it will hopefully provide a brief overview of how the game works.

## Document Conventions

There is no specific document conventions in this SRS.

## Project Scope

This project is a tower-defense game in which the players (users) will defend their stockpile of bread against an endless horde of ducks. Using a variety of defenses and strategies, each playthrough and level should play slightly differently with each upcoming horde of ducks proving to be more challenging than the last. The purpose of this project is to have a fully functioning tower defense game. Features will include multiple levels, multiple towers, different types of enemies, and ways to track progress.

## References

LibGDX - https://libgdx.com/dev/

# Overall Description

This is a Tower defense game focused towards casual gamers.

This game will be produced using the Java programming language along with the libGDX framework. We will be using Microsoft teams as an instruction manual for our game. Github will be used as our hosting platform for version control and collaboration. It will be ran on a desktop (windows). It will be a tower defense game in which users (players) will have to manage money, placement of towers, and their health in order to survive against an onslaught of ducks.

## Product Perspective

Our project is an independent product focused on developing a new program for the entertainment of users. It utilizes the libgdx framework in Java. It is an entirely new product with no relation to other projects.

## User Classes and Characteristics

The Users of this program are casual gamers looking for a game that is fun and easy to play but will get harder as the game progresses. This game is mainly targeted towards children as it’s an easy game with simple controls, and it’s easy to learn.

## Operating Environment

This project will eventually become a fully functioning computer game to be played on a desktop. This website is to be hosted using GitHub. No extensive technical infrastructure is needed. The game will be programmed for use on a Windows computer, for English Speaking countries because that is all I can speak.

## Design and Implementation Constraints

This project will be written exclusively in Java, using libGDX as our graphics library. Therefore, any desktop that can run Java will be able to run our program. This project is only built to run on windows computers.

## Assumptions and Dependencies

Using libGDX is a dependency for this project. Our developer environment is the eclipse IDE. Getting the team set up and familiar with the following tools is important for our game. We must make sure that the tools we are using fit the capabilities of what we want to achieve.

# System Features

## Main Menu A menu that grants access to different screens throughout the program.

### Description

The Main Menu is the first screen the user sees upon launching the game. The user will be able to start the game at level 1, go to the settings screen and quit the game, from this screen. (High Priority)

### Stimulus/Response Sequences

#### Start Game Button Pressed ****Stimulus:** The player presses the start game button **Response:** The screen changes to the level Screen and the player is now able to play the first level of the game.**

#### Quit Button Pressed ****Stimulus:** The player presses the quit button **Response:** The game prompts the user with a popup asking if the user is sure they want to quit the game. The user can then click cancel or quit. If the user clicks quit again the game closes. If the user clicks cancel, the pop up closes.**

#### Settings Button Pressed ****Stimulus:** The player presses the settings button **Response:** The screen changes to settings screen where the player can change the settings of the game.**

### Functional Requirements

#### Change Screens to the level 1 screen on respective button press

#### Change Screens to the Settings screen on respective button press

#### Exit the program on respective button press

#### The Background image of the screen will be placed when the screen is made.

## Tile map The class that defines the level, using a 2d array of tiles.

### Description A class that contains all the information about the current level. Reads information from a plain text file and converts it to a map. Contains an array of tile objects.

### **Stimulus/Response Sequences**

#### Tower Placement Outline **Stimulus:** The player clicks on a button corresponding to the tower they want to buy. **Response:** A translucent tower hovers where the user's cursor is, turning red if the tower cannot be placed at the current position. The image should snap to a grid.

#### ****Placing tower**** ****Stimulus:**** Clicking on the level on the square the player wants the tower placed. ****Response:**** If the player has enough currency, the tower is placed on the map. If not, the player is told they do not have the required currency,

### Functional Requirements

#### convert a text file into an array of characters

#### display the map onto the screen

#### Convert an array of characters into a usable TileMap.

## Currency

Can add and subtract from the player’s amount of money.

### Description As a game player, I need a form of currency so that I can place down my towers of defense during rounds. (High priority)

### Stimulus Response Sequence

#### Starting Currency **Stimulus:** Player begins game. **Response:** Add starting currency for player and display on screen

#### **Earning Currency** **Stimulus:** the player earns Currency by destroying ducks on the map. **Response:** The Currency displayed on screen will show that it is increasing by a certain amount depending on the type of duck destroyed.

#### **Remove Currency** **Stimulus:** player places a tower down. **Response:** currency will be removed from player and display will change

### Functional Requirements

#### Must be able to add and subtract currency from user

#### Must provide user with a base starting currency

#### Must be able to detect if user is at 0 currency; currency cannot be below 0

## Round Start Button

A button that starts the next round.

### Description A button on the game screens allows the user to click it to start the round. This will start the process of summoning the enemies onto the map and start their assault on the player. This allows the player time in-between clicking round start to plan, build towers, or use the bathroom, without having to worry about losing the game. -Priority High

### Stimulus/Response Sequence

#### Button Press **Stimulus:** The player presses the round start button in-between rounds. **Response:** the game will spawn a predetermined list of enemies at the start location, with a .5 second delay between enemies, until the list is empty. The button disappears until the end of the round.

3.4.2.2 ***Increase Currency***

**Stimulus:** The player starts the round. **Response:** The player earns money for starting the round

### Functional Requirements

#### The button becomes unavailable until the end of the round.

#### Round starts when the button is pressed.

#### The player earns money

#### Ducks spawn

## Health System

Handles the health of the player

### Description As a game player, I have a health bar that needs to be sustained in order to finish the duration of the game.

### **Stimulus/Response**

#### **Damage** **Stimulus:** The enemy passes a certain boundary to reach the bread. Response: The health bar loses health due to the enemy passing through the et boundary.

#### **Healing** **Stimulus**: After a certain amount of time or random health bar drop gives the user the ability to gain health. **Response**: The player gets health and health is added to the health bar.

#### **Level** **Stimulus:** Player starts game with a specified amount of health. **Response:** The player starts the game with the specified amount of health.

### Functional Requirements

#### A health indicator is needed to keep track of player health

#### A health indicator is needed to keep track of duck health, does not have to be displayed

#### Must be able to reduce player health indicator

#### Must trigger some type of loss screen if player health is depleted.

#### Must delete duck from game if duck health is depleted.

#### Must add health if random health drop if picked up

## Ducks

The enemy of the game.

### Description

The Ducks class is used to handle data and interactions that happen with the ducks/enemies of the game. It can create a duck, damage, and remove a duck object when asked.

### Stimulus/Response

#### Create **Stimulus:** At the beginning of the game and periodically throughout the round. Game should check to see how many ducks are currently alive to not spawn too many ducks at once. **Response:** This method should be called to create ducks.

#### Take Damage **Stimulus:** When a tower shoots at a duck the projectile will collide with the duck. **Response:** The method take damage should be called with the amount of damage done.

#### Die **Stimulus:** This method will be called when the duck’s health drops to 0. **Response:** The method will set the duck’s location to null.

#### Move **Stimulus:** This method will take in the tile map and use recursion to find a path to the bread/objective. **Response:** The method will smoothly move the duck across the indicated path.

#### Different Difficulty Ducks **Stimulus:** Depending on the round number and difficulty of the current round. **Response:** The Ducks will have different colors/designs to indicate different levels of ducks.

### Functional Requirements

#### Ducks can add and subtract from their health.

#### Ducks will automatically move along the track tiles when placed on a game screen

#### Duck Objects will be deleted when they reach the player’s home tile.

#### Duck objects will be deleted when health hits zero or below.

## Sound

Plays the different sounds throughout the game.

### Description

There would be background music playing during the game, copyright free music. Additionally, there would be sounds on certain actions (towers shooting, ducks taking damage, movement, towers being placed).

### Stimulus/Response

#### Background Music

**Stimulus:** User starting the game. **Response:** Background music starts playing and continues.

#### Towers Shooting Sounds

**Stimulus:** Tower targets and shoots a duck. **Response:** Tower shooting sound is played.

#### Duck getting hit sound

**Stimulus:** Duck gets hit by the tower. **Response:** Duck getting hurt sound is played.

### Functional Requirements

#### A sound is played when a Duck reaches the home tile.

#### A sound is played on a Ducks death, and when it gets hit.

#### Different sounds are played based on which tower is attacking.

#### A background track will loop continuously when the game is started.

## Towers

### Description

The towers are the main interaction the user has with the game. The user is a allowed to buy a tower if they have enough currency, they are then allowed to place the tower on an empty tile on the map. The tower will then shoot at any ducks in range of the tower.

### Stimulus/Response

#### User places Tower with enough currency

**Stimulus:** The user presses the button to buy a tower then clicks on an empty tile on the map, and the user has enough money to purchase the tower. **Response:** The tower will be placed on the tile and the cost of the tower will be subtracted from the User’s currency.

#### User places Tower without enough currency

**Stimulus:** The user presses the button to buy a tower then clicks on an empty tile on the map, and the user does not have enough money to purchase the tower. **Response:** The tower will not be placed, and the user will get a popup stating they don’t have enough money to buy the tower.

#### Duck is within range of a Tower

**Stimulus:** Ducks move within range of Tower. **Response:** The Tower will shoot at the ducks, prioritizing the duck that is furthest along the track.

#### User tries to place a Tower on an invalid Tile

**Stimulus:** The user presses the button to buy a tower then clicks on a filled or invalid tile. **Response:** The tower will not be placed, and the user will receive a popup stating they cannot place the tower in that location.

### **Functional Requirement**

#### The Towers deal damage to ducks

#### Towers attack ducks that have moved the furthest along the track.

#### Towers can only attack ducks within range

#### Have a cost and subtract from currency when bought/placed

## Options Menu

### Description

When the options button is clicked to access the options menu, the game will display a new screen with various buttons and sliders that correspond with different settings that interact with different aspects of the game.

### Stimulus/Response

#### Player Clicks the back button

**Stimulus:** The user clicks the button labeled “back”. **Response:** The user is returned to the main menu.

### **Functional Requirement**

#### The menu has buttons that interact with volume

#### The menu has buttons that interact with resolution

#### The menu has a button that returns to the main menu

# Data Requirements

## Logical Data Model

This program does not take any data in as input and does not return any as an output either.

This program has no save states and will reset by closing and opening the program. The only data that is manipulated is data the program has initialized.

## Data Dictionary

Tile – has a Texture : texture, x-position : int, and y-position : int : used to display the map to the screen

Pos – data class, has an x-position : int, and a y-position : int

Tower- size: int, cost: int, attack: int, aSpeed: int, range: int, locationRow: int, locationCol: int, texture: Texture. It’s used to hold the location, texture and stats of the towers.

Bullet- currentXpos: int, currentYpos: int, destinationXpos: int, destinationYpos: int, speed: int, damage: int, img: Texture

## Reports

Our application does not produce any reports.

## Data Acquisition, Integrity, Retention, and Disposal

The program utilizes very little data and saves none of it upon closing. The program initilizes textures and objects when the screens are changed in the game and when the user places a tower. Upon closing a screen, the objects are disposed of.

# External Interface Requirements

## User Interfaces

The user interface will have buttons of similar style and size throughout any screens. On the main menu screen there will be 3 buttons, new game, load game, and options. In the options menu there is buttons to turn the volume up and down, and to return to the menu. And in game, there are buttons to go back to the menu, see a tutorial, or buy a tower. There will also be a GUI in which the User’s health, round, and currency is displayed.

## Software Interfaces

This software is completely independent of any other software components, and no other external services are needed.

## Hardware Interfaces

The game will be completely controlled with a mouse, and if there is enough time, keyboard shortcuts will also be used. The game will be controlled primarily through clicking buttons, each button corresponding to a different function.

## Communications Interfaces

This software will not use any communication functions.

# Quality Attributes

## Usability

The game was designed to be easy to use with every button that was implemented being useful guides to where it takes you in the game itself. For the main menu there are three buttons that will either start the game, teach you about how to play the game, or take you to the options menu that includes resolution and volume. We wanted the game to have simple return buttons that can take you to any screen hence the escape, return to game, and options button. These allowed the user to go from any screen to the other no matter the screen you are on. We had felt that it would be easiest this way so the user can never get lost throughout the game traveling from different screens.

## Performance

Our game will essentially operate with graphics like an 8bit game, meaning that it will not need many resources to operate reliably. For performance, we wanted the game to run at a smooth 30fps, as that is a number that is both achievable and also smooth to the eye.

## Security

As there is no protected information used in our program, security is not a top priority right now. If we were to create a User database with login information in the future, a secure database with proper protection would be a necessity.

## Safety

There are no concerns about possible loss, damage, or harm that could result from the use of our product.

## [Others as relevant]

N/A

# Internationalization and Localization Requirements

The game itself is built for a user that understands English and if we added the currency class, they would need to have an understanding in U.S. currency. Some requirements that we had were making the currency universal so that a user with basic knowledge of how numbers work would be able to understand. With the rest of the game although it was created in English all the buttons and text can be changed if needed to the desired language required. For the actual gameplay our requirements were to keep the game simple so any user can play the game. Even without directly reading everything one test run of the game should allow you to realize what is going on. For towers just having a button to click then clicking the screen was designed so that all that is required is a mouse click. We explored bringing in different key entries to allow the user to place things, but not all keyboards are the same when you go international so keeping the button on the screen would suffice. Other than the language barrier the game should be able to be enjoyed by any nation or culture it is available to.

# Other Requirements

N/A

Appendix A: Glossary

N/A

Appendix B: Analysis Models

N/A