

# Bilkent University

## Department of Computer Engineering

---



### CS-319 Object Oriented Software Engineering Project

#### Sword & Shield: A Space Adventure

## Analysis Report

#### Group Members:

Akın Berkay Bal  
Eren Aslantürk  
Mehmet Enes Keleş  
Sadık Said Kasap

**Instructor:** Uğur Doğrusöz

**TA:** Hasan Balcı

# Table of Contents

- 1. Introduction ..... 3
- 2. Game Overview ..... 4
  - 2.1 Defense Buildings ..... 5
    - 2.1.1 Turrets ..... 5
      - 2.1.1.1 Miner ..... 5
      - 2.1.1.2 Lazerion ..... 6
      - 2.1.1.3 Good Ol’ Gun ..... 6
      - 2.1.1.4 Fryinator ..... 6
    - 2.2 Attacker Buildings ..... 7
      - 2.2.1 Reactor ..... 7
      - 2.2.2 Factories ..... 7
        - 2.2.2.1 Muscular Spaceship Co. .... 8
        - 2.2.2.2 Athletic Spaceship Co. .... 8
        - 2.2.2.3 Swedish Spaceship Co. .... 9
        - 2.2.2.4 Tough Tough Co. .... 9
- 3. Requirements ..... 10
  - 3.1 Functional Requirements..... 10
  - 3.2 Non-Functional Requirements..... 10
  - 3.3 Pseudo Functional Requirements..... 10

4.	Models .....	11
	4.1 Use Case Models .....	11
	4.2 Dynamic Models.....	16
	4.3 Object and Class Model.....	19
	4.4 GUI.....	21
5 .	References.....	24

# Analysis Report

## 1. Introduction

We decided to design and implement a game called Sword and Shield. It is a two dimensional top-down strategy/tower defense game inspired from Bloons: Tower Defense. The game will be playable by two players: the attacker and the defender. The map will consist of a base for attack units to spawn, a lane for them to walk and a lot of space for defense units to be located. There will be offensive and defensive units to be used during two different stages of the game. At the first stage, defensive player will buy and put his desired items on the map. During the second stage, attacker player will try to breach through the end of the map by spawning attack units in a limited time.

This report includes the overview of the game, describes how game will be played, it will give a brief look on the gameplay and rules of the game. It also contains functional, non-functional and pseudo requirements, use-case models with scenarios, use-case diagrams, dynamic models, object and class model and a mockup of Graphical User Interface.

## 2. Game Overview

Sword and Shield will contain three turns for every game. Turns will have different maps to reduce repetitiveness and hinder the ability of one player to get too comfortable. Turns will have a time limit of 3 minutes. There will be an attacker and a defender in the game. They will start the game with a predetermined amount of coins and they will try to set up their base with the given budget and they will earn some coins throughout the game. Defender earns coins by destroying attacker's ships. Attacker will earn coins by passing the defense line and destroying some health of the defender and through reactors which earns money to player. Coins won't get resetted every turn. Coins can be used to upgrade/buy turrets, factories or reactors.. Also, coins can be used to buy additional tiles to build more of turrets or factories.

Players can pause the game anytime they desire. If we can we want to include a save and a load function but it is not in priority list for the time.

On the top part of the GUI, there will be a timer, scores and money for each player so that they can see who is winning and how much money they have everytime they check. On the right side, defender will be able to see the turrets that can be bought and these turrets can be upgraded by clicking on them.

The first Defender will get their defense ready by using the mouse and after that, attacker will be available to build factories/reactors for attack. Timer will start after attacker builds the first factory/reactor. All turrets, factories and reactors have a maximum level of three.

## 2.1 Defense Buildings

Buildings that will be used only by the defender.

### 2.1.1 Turrets

There will be four kinds of turrets with different properties to let player decide the tactic.

Turrets' properties are the following:

- Cost
- Dimensions
- Cost of Leveling Up
- Attack Speed
- Range
- Armor Penetration

#### 2.1.1.1 Miner

Miner will produce mines which will be placed in front of the miner. It can produce at most 3 mines until at least one of them get destroyed. They will be placed at first exactly in front, then one to the left tile of the first, then the last one will be on the right tile of the first one. Mines will have area damage effect. To balance it, it will be produced in a low rate. It does not have armor penetration. Moreover it gives less damage to armored units.



**Miner**



**Mine**

### 2.1.1.2 Lazerion

Lazerion will be the fast shooter turret. Therefore, it will have the least range but second powerful projectiles for balancing. It does not have armor penetration.



**Lazerion Projectile**

### 2.1.1.3 Good Ol' Gun

Good Ol' Gun will be a machine gun. As technology is kinda old with this one, it will have the least damage, but an average range and a good rate of firing. No armor penetration with this gun.



**Good Ol' Gun**



**Good Ol' Gun Projectile**

### 2.1.1.4 Fryinator

Fryinator is the most powerful weapon for defender. It will have the most damage and most range. But as expected its rate of fire is low, it takes up two tiles and it is the most expensive one of the turrets. It has armor penetration.



**Fryinator**



**Fryinator Projectile**

## 2.2 Attack Buildings

Buildings that will be used solely by the attacker.

### 2.2.1 Reactor

Reactor can be bought by the attacker to earn money throughout the stage. It is in the shape of a nuclear reactor for the game's concept. It will earn the attacker a certain amount of coins throughout the game at a predetermined rate that can be upgraded.

### 2.2.2 Factories

There will be four kind of factories which produces attack units with different properties to let player decide the tactic.

Factories' properties are the following:

- Cost
- Production Rate
- Dimensions
- Cost of Leveling Up

Units produced by this factories have these properties:

- Speed
- Armor
- Health



### 2.2.2.1 Muscular Spaceship Co.

As the name suggests this factory will produce heavy and powerful units. Therefore, it has a low production rate.

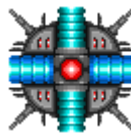
Units will be moderate in speed but will have heavy armor and high health.



**Muscular Spaceship**

### 2.2.2.2 Athletic Spaceship Co.

This factory will produce units which will be fast, have light armor and average health at average production rate.



**Athletic Spaceship Co.**



**The Athletic Spaceship**

### 2.2.2.3 Swedish Spaceship Co.

If you want very fast things which will divide in three when first one is destroyed Swedish is the way to go. This units won't have any armor and they will have low health, but as expected they will be very fast. When the first one is destroyed it will spawn three more. Production rate is low.



**Swedish Spaceship**



**Swedish Spaceships After the Division**

### 2.2.2.4 Tough Tough Co.

This is the most expensive factory, it will take 2 tiles and will have a low production rate. It will produce units with heavy armor and heavy health which can go at average speed.



**Tough Tough Spaceship**

## 3. Requirements

### 3.1 Functional Requirements

- User will be able to place buildings with mouse.
- User will be able turn on and off the game music.
- User will be able to access the help menu. Help menu contains information about game mechanics and game elements such as building and attack units.
- User will be able to pause the game.
- User will be able to reach contact developers of the game.

### 3.2 Non-Functional Requirements

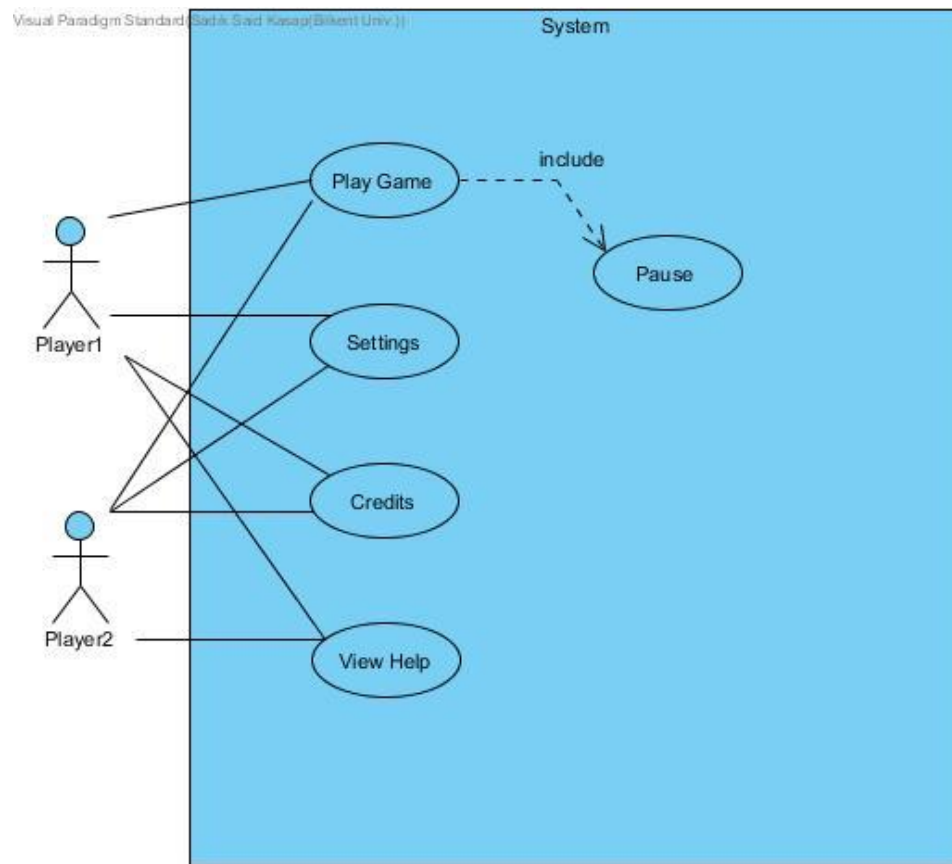
- Game graphics and user interface will be designed in a user-friendly manner. The colors will be designed so as not to disturb the user.
- Control mechanisms of the game will have low response time, which enables users to play with minimal delay.

### 3.3 Pseudo Functional Requirements

- Game will be implemented using Java Swing libraries.

## 4. Models

### 4.1 Use Case Models



*use case diagram*

## Use Case#1

**Use Case Name:** Play Game

**Participating Actors:** Player1, Player2

**Entry Condition:** Player opened the game and click the play game in main menu

**Exit Condition:**

- Players have chosen to close the game, OR
- Players have chosen to return to main menu via in-game menu, OR
- Player can exit the game.

**Main Flow of Events:**

1. Players start the game.
2. System constructs the map.
3. System started the game in the defense stage.
4. Player2 locates the defense turrets.
5. Player2 clicks to Turn button.
6. System start the attack stage.
7. Player1 places the attack units.
8. Player1 destroys the defense units and wins the turn.
9. Players return the main menu.

**Alternative Flow of Events:**

- At any time player can exit the game.(go to step 9)
- Player1 did not destroyed the defense units and Player2 won the turn.(go to step 9)
- Players continue the game after first turn. (go to step 3)

## Use Case#2

**Use Case Name:** Pause Game

**Participating Actors:** Player

**Entry Condition:** Player is already playing the game

**Exit Condition:**

- Player return main menu, OR
- Player continues the to play the game, OR
- Player can exit the game.

**Main Flow of Events:**

1. Player presses the pause button during the game.
2. The system displays the in-game menu.
3. Player continues to play the game.

**Alternative Flow of Events:**

- Player returns to main menu.

### Use Case #3

**Use Case Name:** View Help

**Participating Actors:** Player

**Entry Condition:** Player opened the game

**Exit Condition:**

- Player return main menu, OR
- Player can exit the game.

**Main Flow of Events:**

1. Player click to help button.
2. The system displays the help page
3. Player returns to the main menu.

### Use Case #4

**Use Case Name:** Change Settings

**Participating Actors:** Player

**Entry Condition:** Player opened the game

**Exit Condition:**

- Player returns to the main menu, OR
- Player can exit the game.

**Main Flow of Events:**

1. Player clicks to the settings button.
2. The system displays settings page.
3. Player chooses to turn off the sound
4. Player returns to the main menu.

**Alternative Flow of Events:**

- Player chooses to not to change the settings(go step 4)
- Player chooses to turn on the sound(go step4)

## Use Case #5

**Use Case Name:** View Credits

**Participating Actors:** Player

**Entry Condition:** Player opened the game

**Exit Condition:**

- Player returns to the main menu, OR
- Player can exit the game.

**Main Flow of Events:**

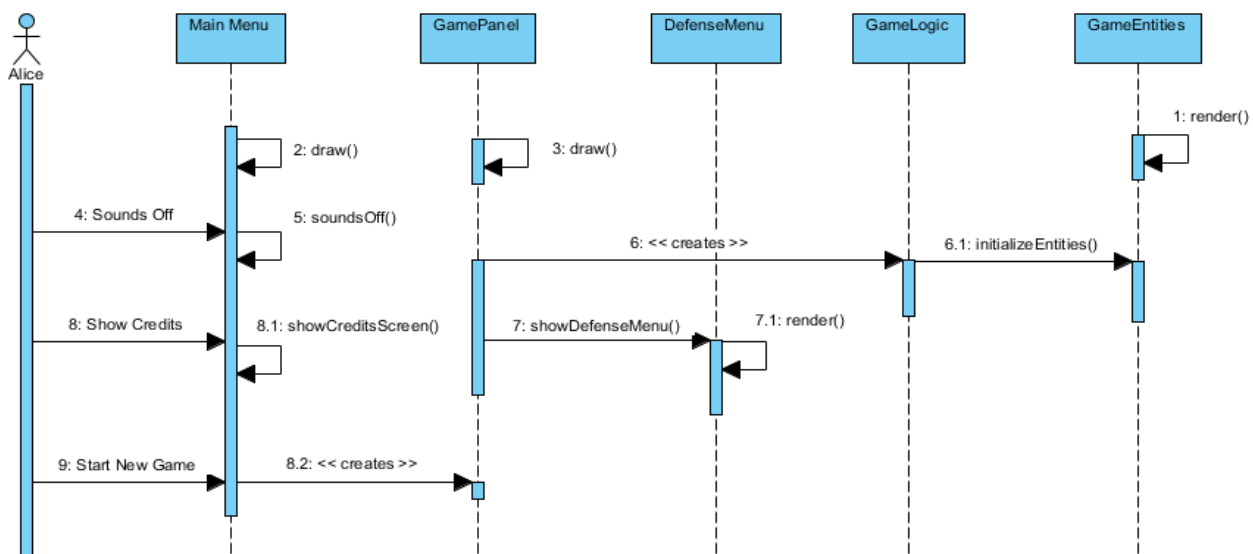
1. Player clicks to the credits button.
2. The system displays the help page
3. Player returns to the main menu.



## 4.2 Dynamic Models

### **Scenario I:** Start the Game

**Plot:** Alice doesn't know anything about the game. She happens to click to the game icon on the desktop. The menu appears on the screen. Alice gets tired of the music, clicks to "Sounds Off" button. After that, she clicks to "Credits" button to see the producers of the game. At the end she starts a new game by clicking to "Play" button to see what the actual game looks like.

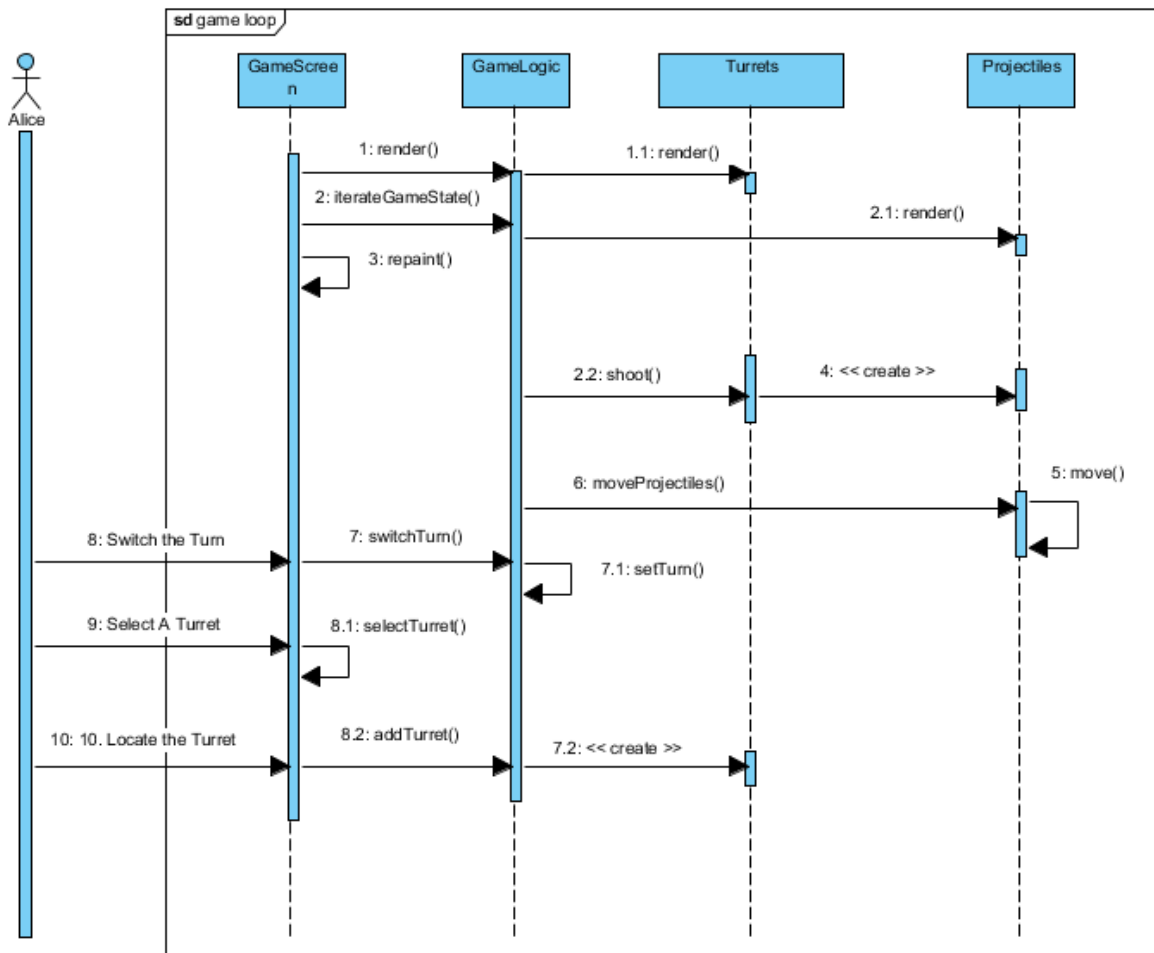


**Description:** Main menu shows up at the beginning. It handles credits screen and sound functionalities.

When the actual game starts, it is replaced by the GamePanel which also initializes a GameLogic and a Defense menu object. DefenseMenu object displays the items that the defensive user can buy. The GameLogic object initializes GameEntities which contains only empty tiles for the start. All these empty tiles have their own render function. Therefore a smooth image with all of the game components will be shown.

## Scenario II: The Defender Plays Her Turn

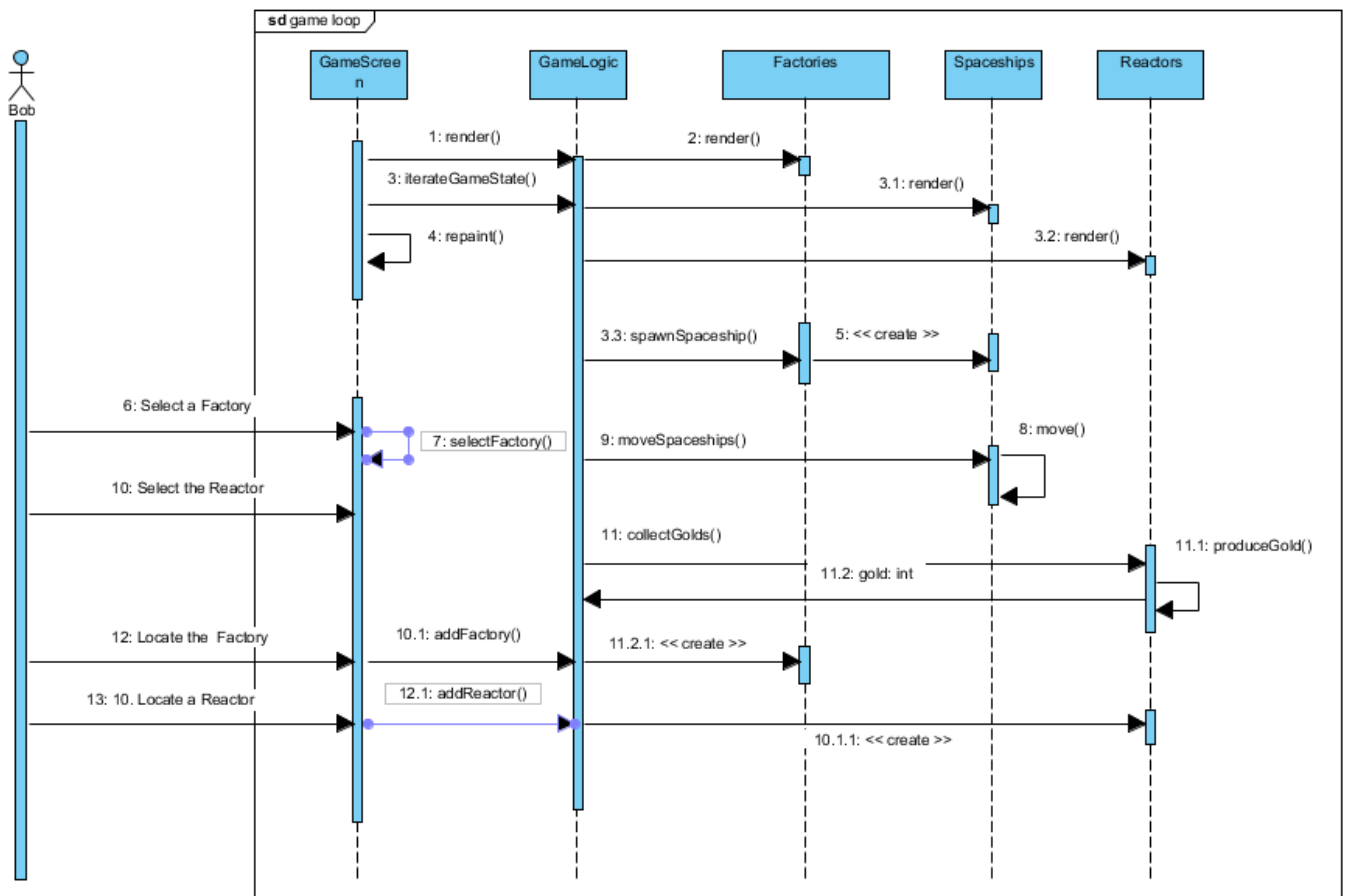
**Plot:** Alice starts a new game. She sees that the game is intended for two players. She invites her friend Bob to play with. At the beginning she has some gold to buy and locate the turrets from the “Defense Units” menu. So she does. Afterwards she clicks to “Turn” button in order to end the defense stage of the game.



**Description:** Alice puts her desired turrets on the map. First they are selected and added to the GameLogic. The turret appears on the map and it is ready to shoot. During the next stage, turrets will shoot depending on their attack speeds and projectiles will show up. But so far turrets are only rendered. Alice clicks to “Turn” button, Bob’s turn starts.

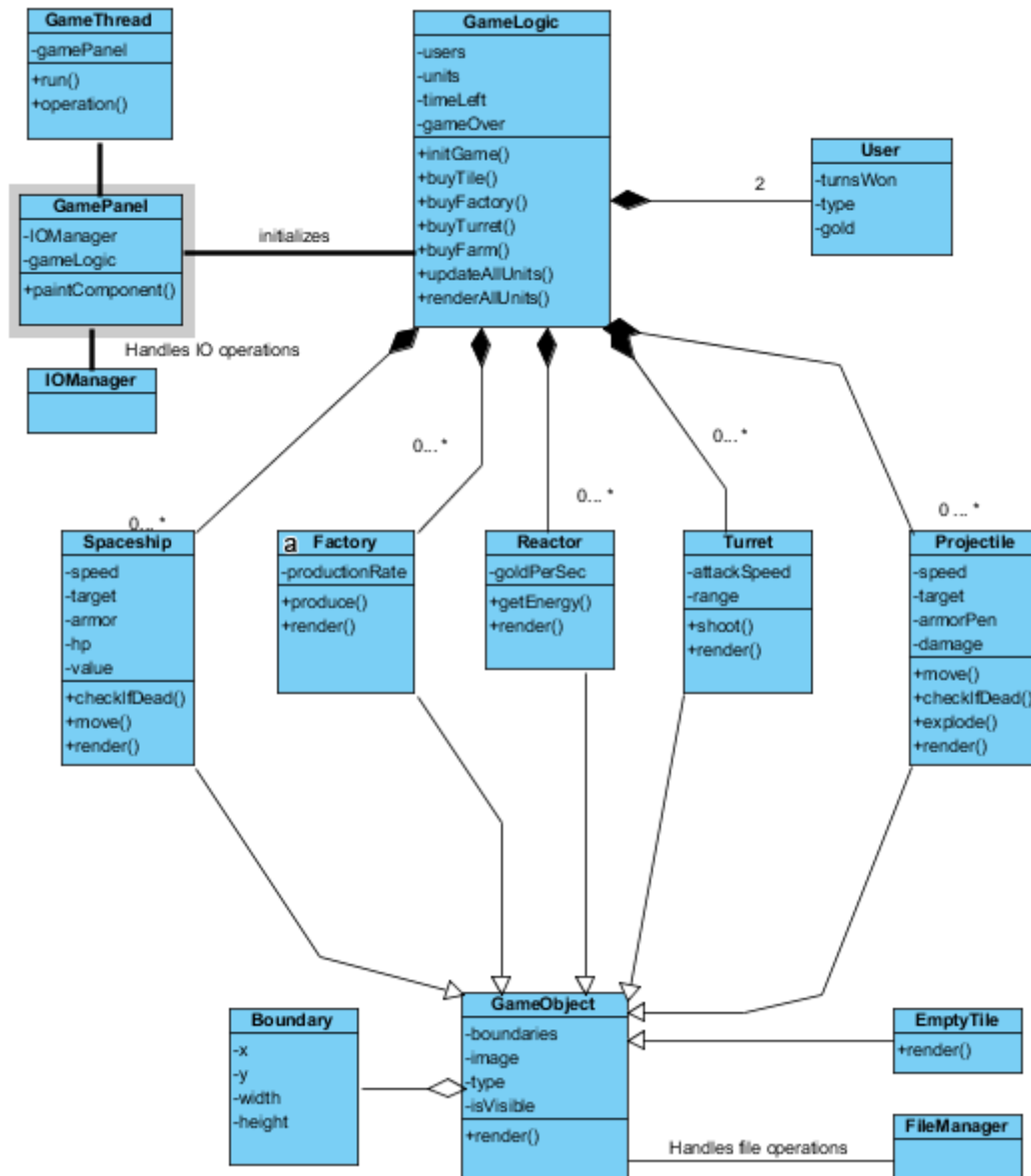
### Scenario III: The Attacker Plays His Turn

**Plot:** Bob needs to start putting reactors and spaceship factories immediately in order to pass through the defense Alice created in the previous turn. However his gold and time is limited. He selects these units from the “Attack Units” menu and puts them on the map. He sees that the reactors produce gold and the spaceship factories spawn spaceships at some rate. Also his spaceships are exposed to projectiles sent by Alice’s turrets.



**Description:** This is now Bob’s part of the game. He selects and puts reactors and factories on the map. The action start. In each iteration, factories spawn spaceships depending on their spawn rates. Reactors will produce gold and spaceships will move towards their target which is the end of the map. As it is explained in the previous diagram, turrets will start shooting projectiles towards the spaceships to prevent them from getting out of the map. Each object has its own render function therefore everything will smoothly be drawn on the screen.

## 4.3 Object and Class Model



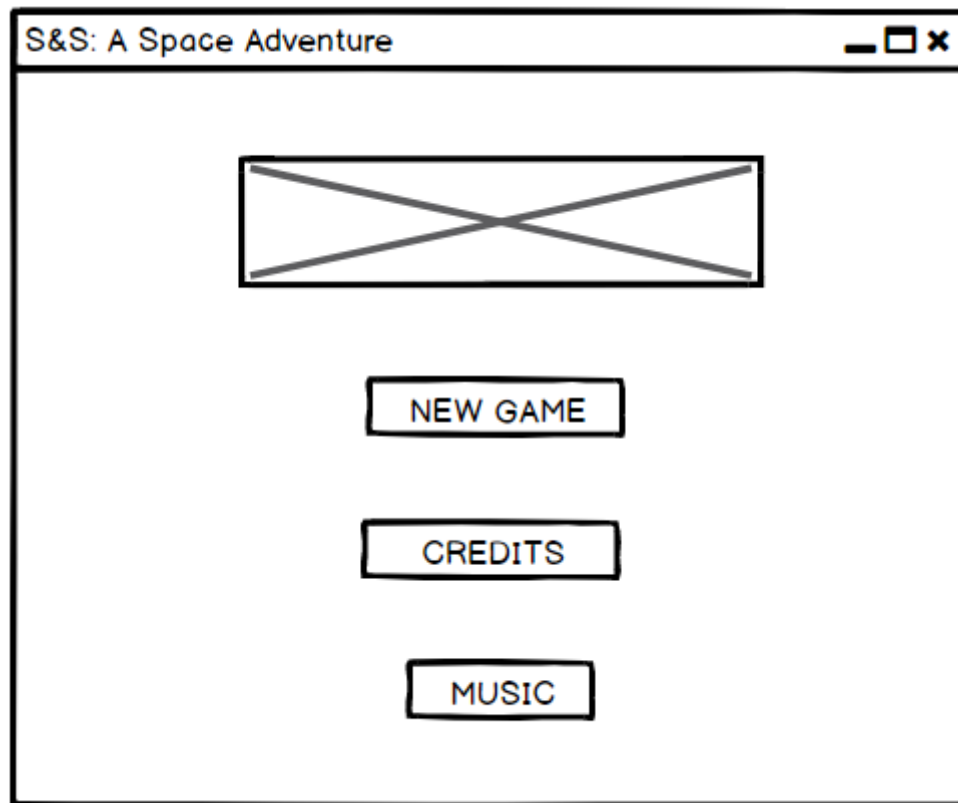
This is the basic outline of the classes that will be implemented. GamePanel is the main screen of the game. It creates a GameLogic instance. This GameLogic instance holds the current state of the game which contains the objects on the map, users, scores etc. At each tick of its timer GamePanel will render the background, call updateAllUnits() and renderAllUnits() methods of the GameLogic instance. updateAllUnits() method basically keeps the game up and running by calling GameObject methods. To elaborate, this method runs reactors, makes factories produce, spaceships move, turrets shoot and projectiles explode. Each GameObject instance has its own boundary which is a rectangle that encloses the object. renderAllUnits() method calls render() methods of all the existing GameObject instances. As the result, image of the corresponding object is drawn to its boundary.

The player can put his/her desired units on the map. This kind of moves change the current state of the game. However, GameLogic is not synchronous to GamePanel. Therefore the state will be displayed on the screen only after 1/60 seconds (60 FPS).

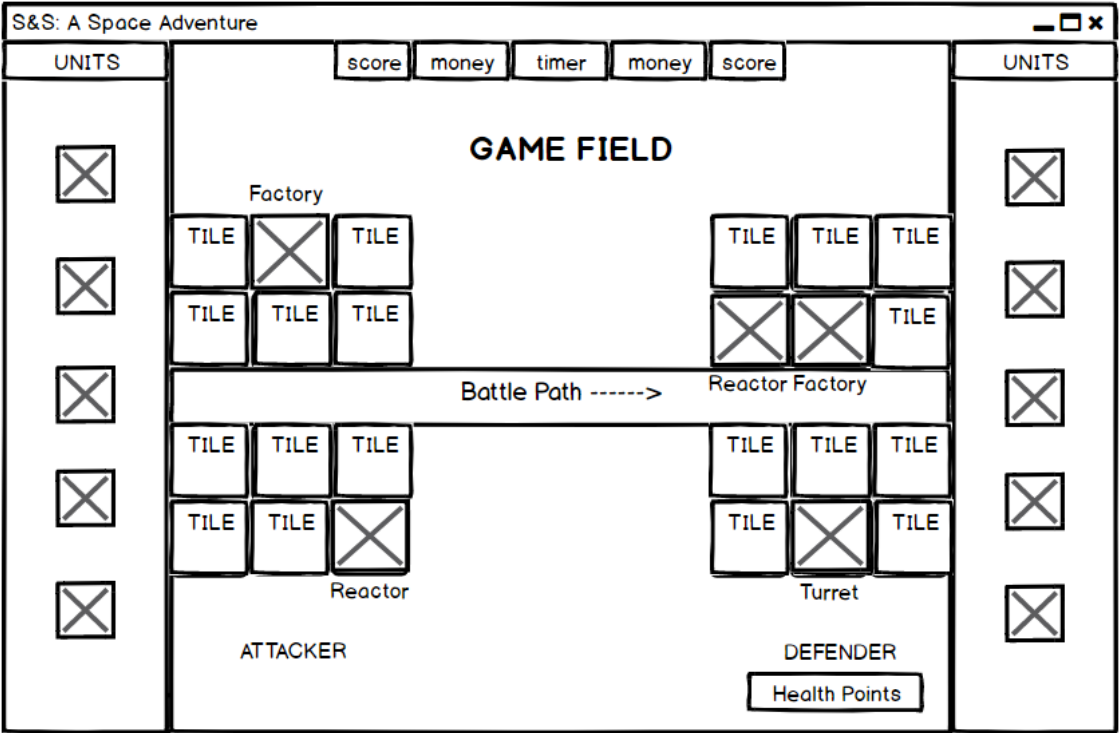
## 4.4 GUI

In our project, the interface covers a few sections: Main Menu, Game Interface, and Credits. Here, we aim that the game should be visually attractive for the user, contents should be easily accessible for the best game experience and user-friendly to enjoy the game with a delightful in game progress.

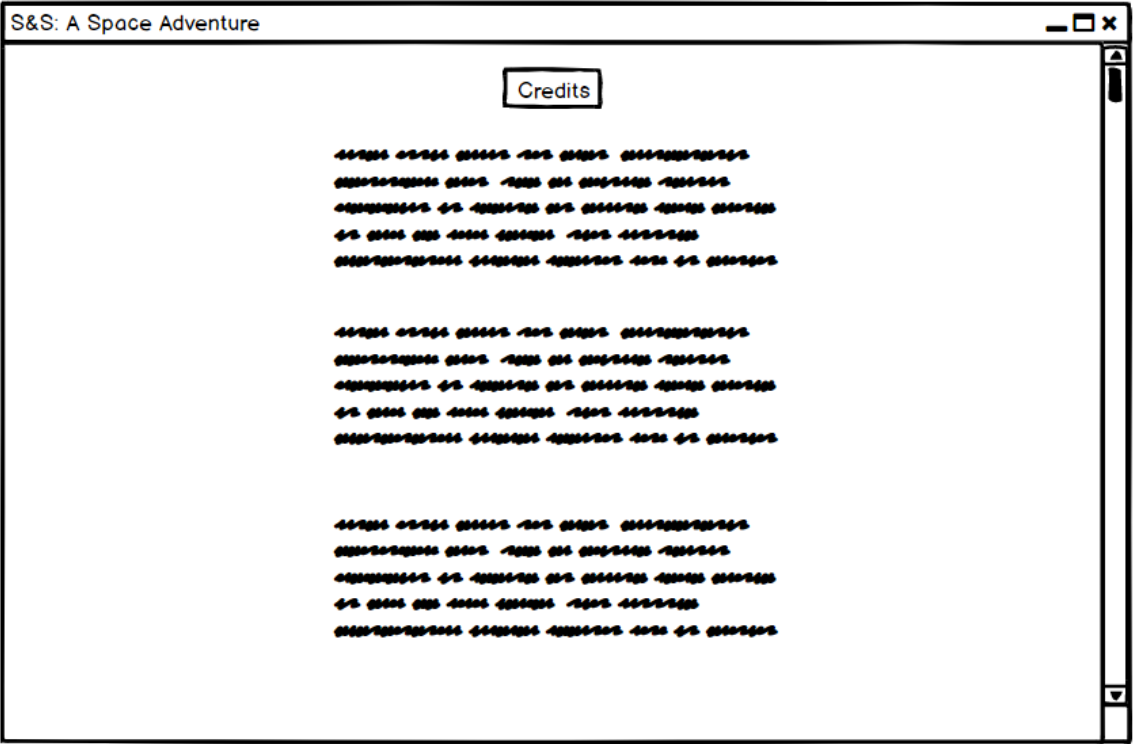
Firstly, in Main Menu interface, we have several buttons to interact: New Game, Sound On/Off and Credits. New Game's main purpose is to lead the user to Game Interface. From here, the user can advance through the screen to play the game. Sound On/Off button is an available feature for the user to enhance the game excitement with supportive background sound effects and musics. Credits button is to inform the user about the game details and contributors to the project. The design mock-up of Main Menu interface is shown below.



Secondly, Game Interface is the major frame where the game occurs. When the user comes here, as we will give information about game rules, he/she will use the play/pause button that we are planning to include and the game will progress. The panels on left and right sides of the screen will be for in-game use for the both of player1 and player2. There will also be additional informative indicators on the screen such as timer and player currency. The design mock-up of the game is as below.



Thirdly, the credits screen is as mentioned above, it aims to inform the user for the game content and contributors. This section is changeable depending on the contents of the project application. Also the mock-up is available below.





## 5. References

inspired from:

<https://ninjakiwi.com/Games/Tower-Defense/Play/Bloons-Tower-Defense-5.html#.WoiAgXYUpEY>

assets are taken from:

<https://opengameart.org/content/pack-spaceships-and-building>

<https://opengameart.org/content/research-building-for-mobile-strategy-games>

<https://opengameart.org/content/starbase>

<https://opengameart.org/content/towers-for-tower-defense>

<https://opengameart.org/content/sci-fi-shoot-em-up-object-images>

<https://opengameart.org/content/space-ship-mech-construction-kit-2>

<https://opengameart.org/content/space-ship-construction-kit>

<https://opengameart.org/content/space-ship-building-bits-volume-1>

<http://kenney.nl/assets/tower-defense-top-down>

<http://hirefreelanceartist.com/free-tower-defense-graphics.html>

<https://t-allen-studios.itch.io/turret-pack-01>

<http://ask4asset.com/construct-2-capx-google-play-high-score-and-achievements/>