  
  
  
  
CS 319 - Object-Oriented Software Engineering  
Analysis Report  
  
  
SpaceGuard  
Group 9 Section -2

Ege Karaarslan

Ferit Özcan

Mehmet Ali Deligöz

Halil Bülent Orhon

Contents

[1. Introduction](#_Toc433736954) 3

[2. Requirement Analysis 4](#_Toc433736955)

[2.1 Overview 4](#_Toc433736956)

[2.2 Functional Requirements 6](#_Toc433736957)

[2.3 Non-Functional Requirements 8](#_Toc433736958)

[2.4 Constraints 8](#_Toc433736959)

[2.5 Scenarios 9](#_Toc433736960)

[2.6 Use Case Models 10](#_Toc433736961)

[2.7 User Interface 15](#_Toc433736962)

[3. Analysis 21](#_Toc433736963)

[3.1 Object Model 21](#_Toc433736964)

[3.1.1 Domain Lexicon 21](#_Toc433736965)

[3.1.2 Class Diagram 22](#_Toc433736966)

[3.2 Dynamic Model 24](#_Toc433736967)

[3.2.1 Activity Diagram 24](#_Toc433736968)

[3.2.2 Sequence Diagram 26](#_Toc433736969)

[4. Conclusion 30](#_Toc433736970)

[5. References 31](#_Toc433736971)

**1. Introduction**

SpaceGuard is a 2D java implemented shooting game influenced by Space Invaders. Games like Space Invaders aim to shoot and destroy aliens with a laser cannon and earn as many points as possible. The game that influenced us is following.

<http://www.oyunskor.com/oyna600.htm>

In our game, we plan to have different features. In SpaceGuard, the game will have three modes. Easy mode will aim to destroy aliens and reach to a certain score limit. Medium and hard modes will aim to destroy aliens and reach to a certain score limit within a certain time limit where the aliens are harder to be destroyed.

Medium and hard mode will allow its user to get a high score as medium and hard modes will end only when the player has no life left.

The game will be a desktop application and will be controlled by a keyboard.

-

# **2. Requirement Analysis**

## **2.1 Overview**

SpaceGuard is a shooting/arcade video game that mainly inspired by Space Invaders. It differs from the original game by extra features and modified gameplay. Like any other game embracing infinite gameplay, the main purpose of this game is destroying aliens with a laser gun. Each hit from the SpaceGuard weakens or destroys aliens depending on strength of the aliens. The game will continue forever unless the player has no lives or doesn’t exit the game intentionally.

### **2.1.1 Game Play**

The player has the ability to move spaceship horizontally by using left/right arrow keys. Since the game is a 2D fixed shooter game, spaceship is only allowed to move between left and right borders at the bottom of screen. In order to survive longer, player must avoid bombs and destructive powerups dropping from aliens.

To make the game more competitive and keep it exciting, we decided to change the main design from goal focused (levels, saving the princess etc.) to infinite gameplay. Therefore, the player will always be motivated to play again.

### **2.1.2 Play Modes**

There are 3 different play modes in the game.

* *Easy Mode*: This mode is for beginner level and like a tutorial. The only goal is surviving until achieving the target score by avoiding destructive powerups/bombs. When an alien is hit, only the addition operation occurs. There is no time limit.
* *Medium Mode*: When the player selects medium mode, gameplay gets more complicated. Firstly, the frequency of destructive powerups and bombs increases. There is a time limit. Therefore, the player is supposed to collect time bonuses which drop from aliens.
* *Hard Mode*: When the player selects hard mode, in addition to medium mode, in order to increase risk, different operations such as divison, multiplication, substraction are added.

### **2.1.3 Spaceship**

The spaceship is the only thing that user can control manually. User can shift the spaceship left and right to aim and hit the aliens. By using keyboard, user can fire to target. Spaceship also has some powerups. This powerups can be gained from the aliens. Powerups varies in different types. The effects of the powerups will be negative or positive on the spaceship. For example, when spaceship catches speed powerup, it will increase spaceships speed so that it will easier to aim to aliens.

### **2.1.4 Aliens**

There will be three type of aliens in the game and the aliens is the funny part of the game. They can be destroyed by one hit, two hit or three hits from the spaceship. User gains points by destroying the aliens. Aliens has shift left and right and they throw down bombs. If aliens hit spaceship, spaceship will lose one of its lives.

## **2.2 Functional Requirements**

## **2.2.1 Play Game**

SpaceGuard is a 2D shooting game. The aim of the game is to destroy the aliens and score higher by using the laser of the spaceship. In the beginning player has 3 lives and if a bomb thrown by the alien hits the spaceship player loses one of his/her lives. The game is over when the player loses all of 3 lives and wins if the score limit is reached. The game ends in easy mode if the player reaches the target score. However, as one of the objectives of this game is to reach highest score the game continues until the player loses all of his/her lives or reaches the time limit in medium and hard mode.

Player will face bombs thrown by the aliens and power downs which show up from the destroyed aliens. These features makes the game more challenging. Additionally, power ups that are added makes the game more entertaining.

While playing this game, the player is supposed to move the spaceship and shoot at the same time. This game requires the user to keep track of the spaceship and the aliens, shoot according to their movement directions and keep track of the operation to be done in the hard mode where the operation will be either +, -, \* or /. As a result of playing this game, this will enable the player to develop hand-eye coordination along with the ability to manage multiple objectives.

### **2.2.2 Change Settings**

The program provides easy mode as a default to its players. Player can change the game mode from easy to medium or hard and background picture. If the player wants to change play modes or background he selects the preferred background and play mode before he begins to play the game. If a player wants to go back to the default settings a player can chose default system settings. Additionally, whenever the game is closed and reopened the default system settings apply.

### **2.2.3 Pause Game**

During the gameplay, player can pause the game. When game is paused, from the pause menu a player can click “continue” to continue playing the game. However, when a player closes the application while the game is paused, player loses any game progress.

### **2.2.4 View Help**

User will find instructions and explanations about the game in the help menu.

This help menu includes:

-Controls

-Main goal of the game

This help menu will help user to learn the main concept of the game and teach user how to control the spaceship. Also user will be allowed to learn about the power ups and power downs.

### **2.2.5 View Credits**

Player will be able to get information about the game developers.

## **2.3 Non-Functional Requirements**

### **2.3.1 Comfortable Interface**

The interface of the game will be designed in a way that user can easily find anything about the game and also thanks to this interface, user can play this game effortlessly. Interface is one of the most important non-functional property of a game. As a group, we will try to make our interface as user-friendly as possible to make users feel comfortable while playing our game.

### **2.3.1 Graphic Quality**

Graphics is one of the most important property of the video games. So, we want to make our graphics attractive by selecting interesting images for spaceship and aliens. Also graphical smoothness is very important for us. Because we want to create a game with smooth graphics to not disturb user's eyes.

### **2.3.2 Low Response Time**

To increase the playability of the game we will try to make our game with lowest response time possible. It means that, the game will respond quickly to the user's spaceship movement so that there will be no delay in the game and this will increase the game experience.

## **2.4 Constraints**

The game will be implemented in Java.

## **2.5 Scenarios**

Scenario 1: Player requests to start game by pressing play game button from Main Menu. After that System initializes the game manager and creates graphics objects. Then the game loop starts and the system checks if there is any hit. Game manager, updates itself and an alien drops down a bomb. Then the system checks if the game is over and the loop continues until the game is over.

Scenario 2: Player desires to change game settings. He presses change settings button in the main menu. System displays settings. Player changes settings according to his desire. After that he presses save, System updates new settings by Game Manager.

Scenario 3: Assuming that the Player has changed settings, as in the Scenario 1, Player desires to use default settings provided by the system at the startup. The player presses change settings from Main Menu. System displays settings. Player presses save and the system applies the default settings.

## **2.6 Use Case Models**

2.6.1 Main Model

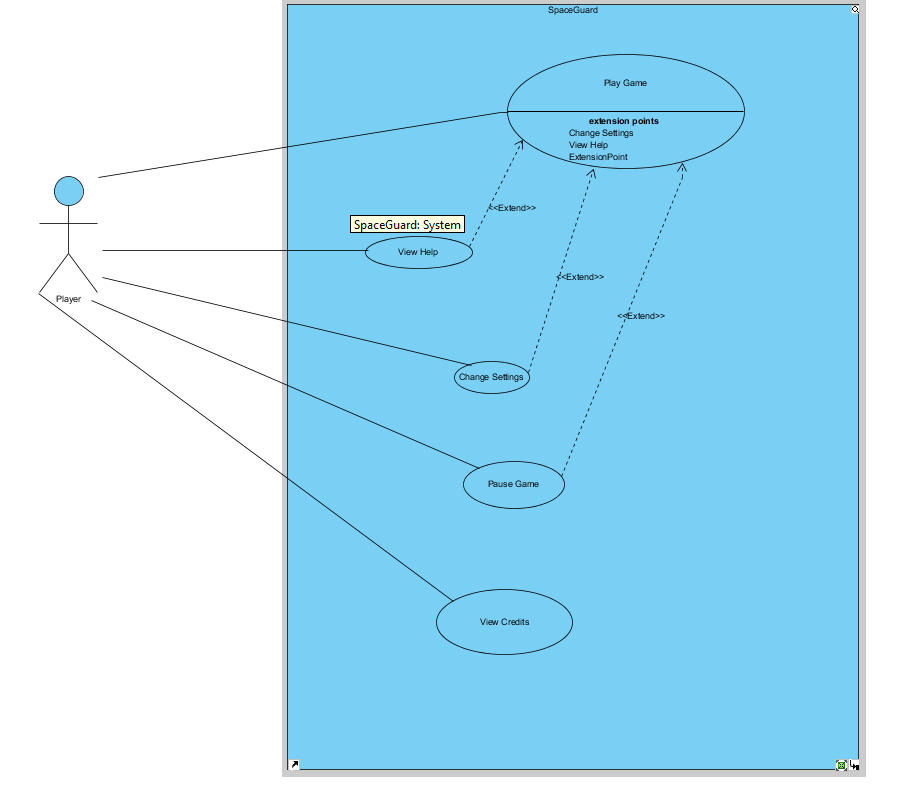
This section gives information about the main use case model of SpaceGuard game. Detailed explanations of use cases are in explained below.

Figure 1: Main use case model of the system

**2.6.2 Play Game**

**Use Case Name:** Play Game

**Actor:** Player

**Stakeholders and Interests:**

* Player aims to survive as long as possible and make the highest score.
* System records each player’s score

**Pre-condition:**

* Unless the player changes game settings, easy mode is selected as default.

**Post-condition:** None

**Entry Condition:** Player selects “Play Game” button from main menu

**Exit Condition:** Player returns to main menu by interrupting the gameflow.

**Success Scenario Event Flow for Easy Mode:**

1. Game is started by System.
2. Player starts to play without time limit.
3. Player plays until he/she pass the target score.
4. System asks player to change game mode or continue with the same mode.

*If the player chooses to continue with same mod, Step-3 is repated until player loses all lives.*

1. System updates highscore list
2. System returns to Main Menu

**Alternative Flows for Easy Mode:**

**3.1)** Player tries to survive until achieving the target score:

**3.1.1)** Player fires the laser gun by using keyboard (Default: Space Bar).

**3.1.2)** Laser hits alien.

**3.1.3)** System removes aliens that get hit, if they are type of OneHitAlien.

**3.1.4)** System updates the score with respect to value of the shot alien.

*Steps 3.1.1 - 3.1.4 are repeated until achieving target score*

**3.1.5)** Player finishes the easy mode, after achieving target score.

**3.2)** Player gathers the powerUps during game:

**3.2.1)** Laser hits alien.

**3.2.2)** System removes aliens that get hit, if they are type of OneHitAlien.

**3.2.3)** Power up pops up, if the shot alien has one.

**3.2.4)** Player gathers the power up by using spaceship.

**3.2.5)** System applies the power up, if power up hits the spaceship.

**3.3)** Player gets hit by falling bombs:

**3.3.1)** Bombs falls down from the top of screen, randomly.

**3.3.2)** Player gets hit by falling bombs.

**3.3.3)** System removes one life from spaceship, if the player has any.

**3.3.4)** System records the score and returns to main menu, if the player has no lives.

**Event flow for medium/hard mode:**

1. Game is started by system.
2. Player starts to play with a time limit.
3. Player plays the game until time is up or the player has no lives.
4. System updates highscore list.
5. System returns to main menu.

**Alternative flows for medium/hard mode:**

**3.1)** Player gets hit by falling bombs:

**3.1.1)** Bombs falls down from the top of screen, randomly.

**3.1.2)** Player gets hit by falling bombs.

**3.1.3)** System removes one life from spaceship, if the player has any.

**3.1.4)** System records the score and returns to main menu, if the player has no lives.

**3.2)** Player gathers the powerUps during game:

**3.2.1)** Laser hits alien.

**3.2.2)** System removes aliens that get hit, if they are type of OneHitAlien.

**3.2.3)** Power up pops up, if the shot alien has one.

**3.2.4)** Player gathers the power up by using spaceship.

**3.2.5)** System applies the power up, if power up hits the spaceship.

**3.3)** Time is up.

**3.3.1)** Player cannot collect enough time bonus.

**3.3.2)** System records the score and returns to main menu

**3.4)** Player collects points

**3.4.1)** Player fires the laser gun by using keyboard (Default: Space Bar).

**3.4.2)** Laser hits alien.

**3.4.3)** System removes aliens that get hit, if they are type of OneHitAlien.

**3.4.4)** System updates the score with respect to value of the shot alien and the current operation.

*Steps 3.4.1 - 3.4.4 are repeated until achieving target score*

## **2.7 User Interface**

**Main Menu**

When player starts the application, main menu shows up after the splash screen.  
Main menu has five options: Play Game, Settings, Help, Credits, Exit Game. Application is terminated when user selects “Exit Game” option.



Figure 2: Main menu mock up

**Play Game**

If the player chooses Play Game, game starts with default settings. Here is a screenshot from easy mode gameplay.

Figure 3: Easy mode screenshot

**Pause Game**

The game can be stopping by pressing escape button during the game. Users can resume to the game or quit existing game by clicking on the buttons.

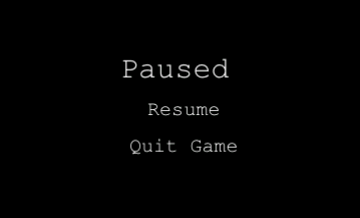
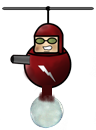
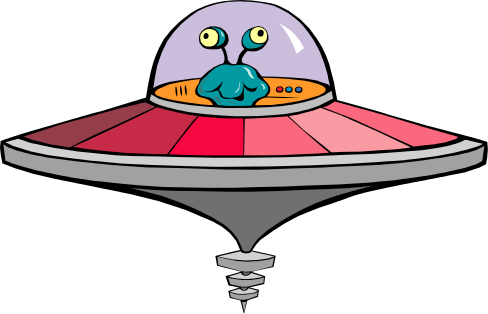


Figure 4: Pause Menu of the Game

**Alliens**

In the game, we have three types of aliens. One hit aliens, two hit aliens and three hit aliens. One hit and two hit aliens are moving right and left and bombing constantly. Three aliens are being produced to avoid player to keep pressing left and right buttons which causes to a fix game play. Spawn rate of three hit aliens is %20. They calculate if it is going to hit the player if they release the bomb in that moment or not by physical calculations.



One hit alien Two Hit Alien Three Hit Alien

**PowerUps**

There are %10 possibility that an alien can drop a power up instead of bomb.There are three kind of power up.

Health power up refills the health bar.

C:\Users\Ege\Desktop\ferit3119.rtfd\bomb.png The big bomb kills any enemy alive at that moment.

C:\Users\Ege\Desktop\ferit3119.rtfd\slowdown.png The snail slows the game for ten seconds but not the spaceship.

C:\Users\Ege\Desktop\ferit3119.rtfd\anim2.pngThe spaceship is the object that is controlled by the user and throws laser.

CHANGE SETTINGS

When player presses on the change settings button, a list of options are displayed. These are change difficulty, change background image and apply default settings. If player does not make any change, the default settings are used by the system. Player can select only one difficulty among easy, medium or hard and only one background image for gameplay. Background image is selected using the paddles in the figure. If the player wants to use default settings he should only click on the “Default settings” checkbox.



Figure 5: Change settings mock up

**Help:**

If player enters help menu, there will be a text shown in the screen including playing instructions of the game which means the aim of the game, information about the buttons of the game and information about the power ups. Player can go back to the main menu by clicking on the Back button.



Figure 6: Help menu mock up

**Credits:**

When Player enters the Credits Menu, player will see the contact information of the game developers and their name. Player can go back to the main menu by clicking on the Back button.



Figure 7: Credits menu mock up

# **3. Analysis**

## **3.1 Object Model**

### **3.1.1 Domain Lexicon**

One hit alien: Simple robots that die with one hit.

Two hit alien: Alien that dies with two hits and more challenging bomb throws.

Three hit alien: Also called sniper. It dies with two hits and bomb throws are as challenging as the two hit alien.

Snail: It is the slow down power up.

Projectile: It stands for laser.

### 3.1.2 Class Diagram

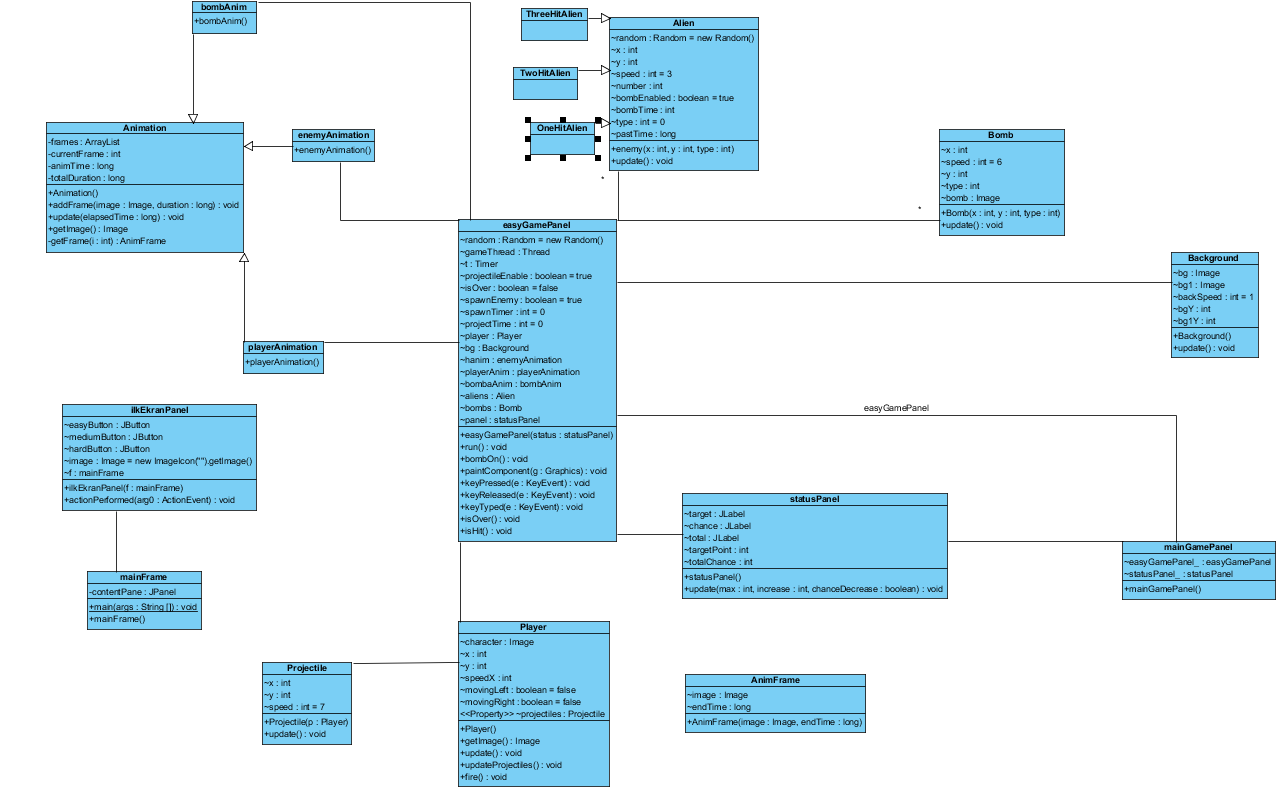
The class diagram of SpaceGuard is illustrated below.

Figure 8: Class diagram of the system including the relationships between classes

In our design, **MainMenuPanel** is the first class that will be constructed by the main method. **MainMenuPanel** will start the **GameManager** which is the control object of the system.

**GameManager** is the class where we manage and organize the game dynamics.

**Background**, **bomb**, **projectile**, **player** and **enemy** classes keep x, y coordinates and images to draw. They are the game graphics classes.

**MainFrame** keeps all of the panels inside it.

**StatusPane**l which is updated by GameManager. keeps track of health and other game  
information during the game.

**Animation** classes refresh image per miliseconds and return them so we can  
create animations

**3.2 Dynamic Model**

### **3.2.1 Activity Diagram**

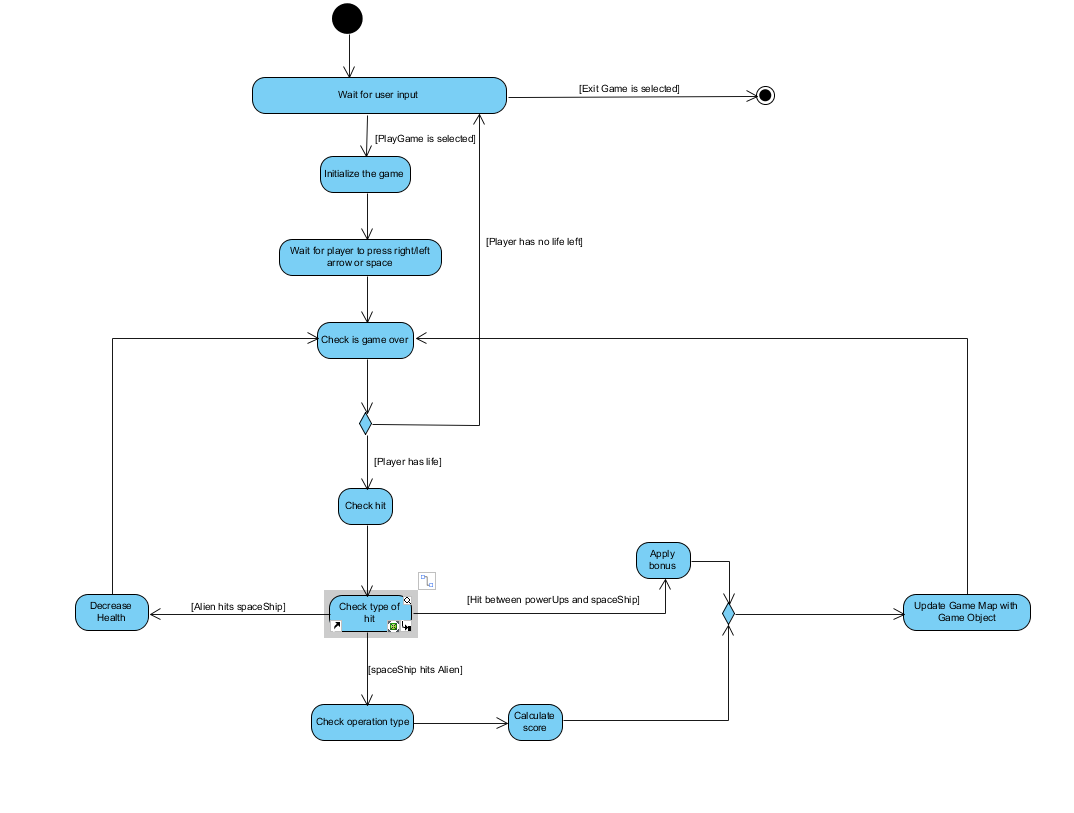


Figure 9: Activity diagram of the play game use case

When user selects Play Game, system initializes the game by creating game manager and game objects. When this is done, system waits for the user to press right/left arrow or space from the keyboard to start the game. When spaceship begins to move, the game starts and the system checks if the game is over. If player has no life left the game is over, otherwise the game continues. If player has life, system checks if there is a hit.

If the hit is between a powerUp and spaceship then the corresponding bonus feature is applied to the spaceship. If the hit is between spaceship’s laser and Alien then, the alien is destroyed. After this two cases, system updates game manager with remaining game objects. If the hit is between aliens bomb and spaceship then player loses one of his lives and the system checks if the game is over. The game is over only when player has no lives left and the system checks whether the game is over after every hit.

### **3.2.2 Sequence Diagram**

**a) START GAME**

Game Loop shown below is the Sequence Diagram of scenario 1.

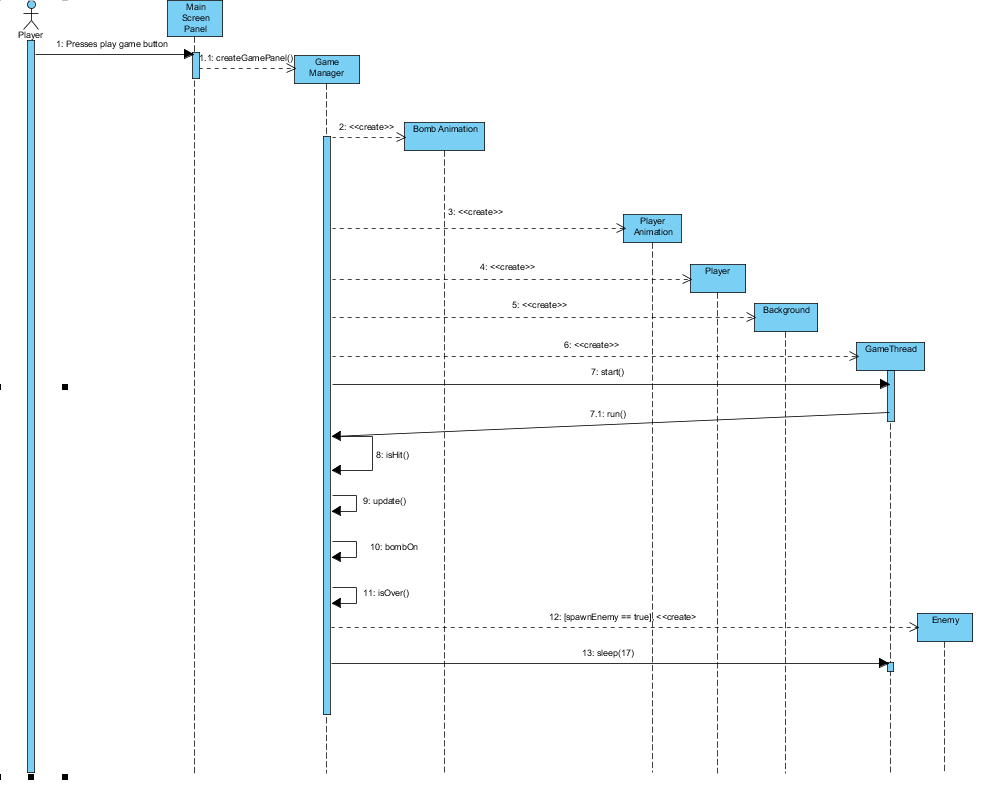


Figure 10: Sequence diagram of the start game (scenario 1)

The game loop starts with the run method comes back from the thread. It calls update() method of GamePanel, which updates coordinates of each visible image on screen. Then it calls isHit() method that checks if an enemy is hit or not. After that it calls bombOn() and isOver() methods. bombOn() make sure to fire if enemies bomb stock is enabled. Finally isOver() method checks if the game is end or not before calling thread's sleep(17) method to wait 17 milliseconds before repeating this actions

CHANGE SETTINGS

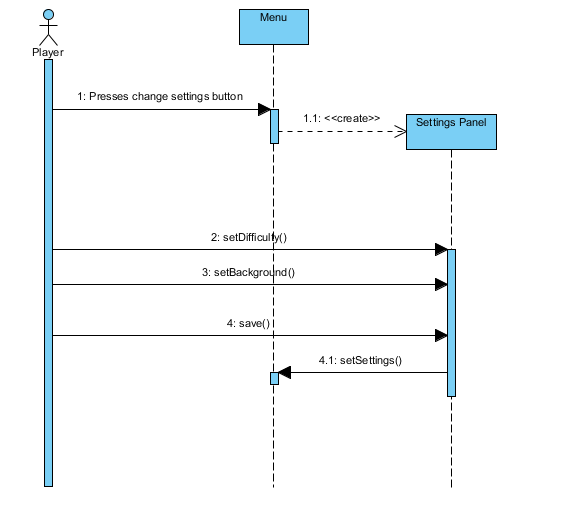


Figure 11: Sequence diagram for Scenario 2

This sequence diagram refers to the Scenario 2. In this diagram, player presses change settings button and menu creates a settings panel. Player sets difficulty, background and presses save and the changed settings are sent back to the main menu.

Scenario 3:

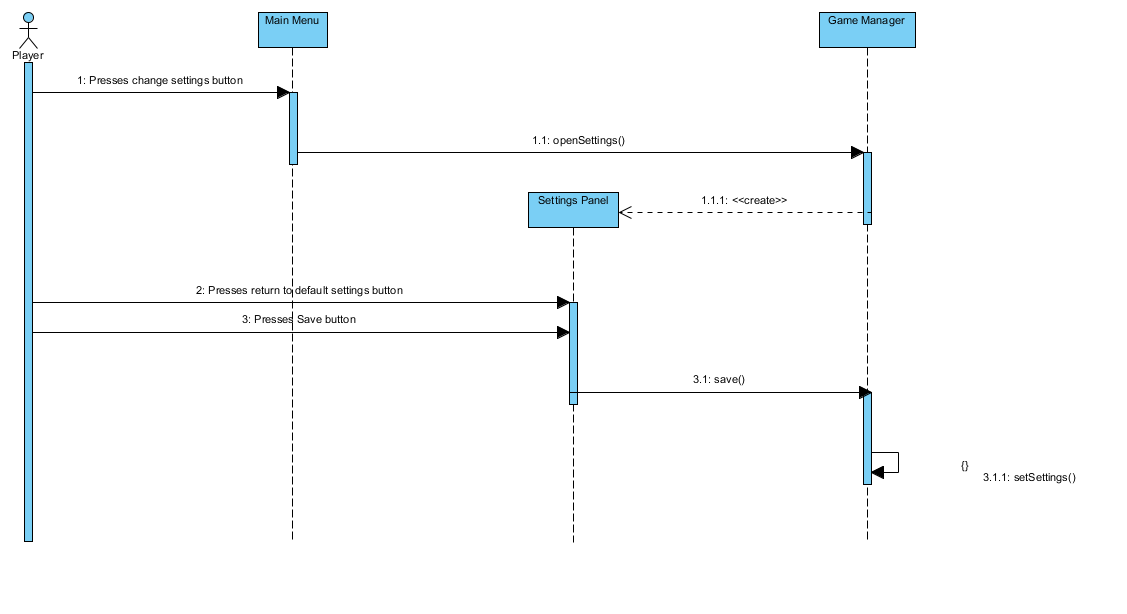


Figure 12: Sequence diagram for scenario 3

This diagram illustrates the alternative change settings scenario. In this diagram, the player has already changes settings and now player wants to set default settings back.

# **4. Conclusion**

The aim of this analysis report was to define and analyze the requirements necessary for the SpaceGuard program. Our report consists of two main sections called Requirement Analysis and Analysis.

For the Requirement Analysis section, we specified all possible requirements that a player can perform in our game, SpaceGuard. Under this section we explained functional and nonfunctional requirements, constraints, scenarios, user interface and use case model in our program. The subsections are the requirements in this game decided from our problem statement.

Scenarios describe activity of a user in our SpaceGuard game.

The Analysis section, consists of Dynamic model and object model. This part was done after the requirements are specified. In the dynamic model subsection, we explained the interactions between the user and the system. To do that, we have used a sequence diagram and an activity diagram that indicates the game play. In the object model, we have identified the necessary classes and their relationships.

To sum up, in this report we specified the required classes, relations and interactions need for building this system.

# 5. References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.SS