CS 319 - Object-Oriented Software Engineering

Project Analysis Report - Iteration 2

Galaxy Invaders

Group 3J
Umer Shamaan
Osman Can Yıldız
Cemre Osan
Tanay Toksoy

Table of Contents

1. Introduction……5
2. Overview……6
    - 2.1. Gameplay……6
    - 2.2. Enemy Types……7
        - 2.2.1. Shooter……7
        - 2.2.2. Kamikaze……8
        - 2.2.3. Splitter……8
        - 2.2.4. Boss……9
    - 2.3. Bullet Types……9
        - 2.3.1. Basic Bullet……9
        - 2.3.2. Laser……10
            - 2.3.2.1. Player Laser……10
            - 2.3.2.2. Enemy Laser……10
        - 2.3.3. Missile……10
    - 2.4. Power Ups……11
        - 2.4.1. Shield Power-Up……11
        - 2.4.2. Speed Power-Up……11
        - 2.4.3. Life Power-Up……12
        - 2.4.4. Bullet Power-Up……12
3. Functional Requirements……13
    - 3.1. Player Requirements……13
    - 3.2. Game Requirements……13
    - 3.3. Play Game……14
    - 3.4. How To Play……15
    - 3.5. Settings……16
    - 3.6. Credits……16

4. Non-functional Requirements……16

4.1. Performance……16

4.2. Security…… 16

4.3 Supportability……16

4.4 Extendibility……17

4.5 Portability……17

5. System Models……18

5.1. Use case model……18

5.1.1. Gameplay……18

5.1.2. Main Menu Navigation……23

5.2. Dynamic models……27

5.2.1. Start Game……27

5.2.2. Controlling Player Ship……28

5.2.3. Firing Bullets……29

5.2.4. Checking Player Health……30

5.2.5. Hitting Enemies……31

5.2.6. Acquiring Bonus……32

5.3. Class Diagram……33

5.3.1. Model Part……34

5.3.2. View Part……35

5.3.3. Control Part……36

5.4. Activity Diagram……37

5.4.1. Main Menu Activity Diagram……37

5.4.2. Play Game Activity Diagram……38

5.5. State Diagrams……39

5.5.1. Player……39

5.5.2. Shooter……40

5.5.3. Kamikaze……41

5.5.4. Splitter……42

5.6. User interface and Screen Mock-ups ……43

5.6.1. Menu……43

5.6.2. How to Play……43

5.6.3. Settings……44

5.6.4. Credits……45

5.6.5. Game Screen……45

References……49

1. Introduction

Martians have invaded Earth and are after mankind's most prized resource, metal. Elon Musk, the don of the biggest space institution of Earth, Space-X, invents a spaceship to fight back. Galaxy Invaders is an arcade-style space shooter game inspired by Defenders. [1] The role of the player is to shoot down enemy spaceships to advance and finish the game.

We decided to work on this game as the gameplay mechanic is very simple yet there will be several interactions between the objects in the game and we will have to develop hierarchies and associations between those objects. Furthermore, we will have more freedom to make the game behave as we want. The most important reason perhaps is that as the game is extremely entertaining, we will have a fun time developing this game.

2. Overview

Galaxy Invaders is a 2D game in which the players aim is to shoot down the invader spaceships and prevent them from abducting humans.

2.1. Gameplay

(Player's Spaceship)

The player will be able to move the ship with ↑, ←, ↓, → keys and they will make the ship go up, left, down and right respectively. The player movement speed is considered as %100 movement speed while enemies' are different. Furthermore, the space key will allow the player to shoot a horizontal projectile which will be used in order to shoot down invaders. Getting shot by invaders or colliding with them will result in loss of the current ship. The player will be given 3 spaceships as life and the player's health will be 100 Health at the beginning of the game. After having lost 100 health, the player loses 1 spaceship. After having lost all lives, the game is over. Occasionally, the invaders will try to abduct humans and the player will get 100 bonus points by saving them by destroying enemies. However, shooting humans will inflict 50 points penalty.

2.2. Enemy Types

There are four types of enemies, which behave in a different manner ie. Shooter type of enemy tries to shoot down the player and abducts humans, kamikazes locks on player spaceship and tries to collide with it , Boss tries to slam into the player ship with its powerful spread fire, and Splitter splits into Kamikaze or Shooter randomly after getting shot. Defeating a certain amount of enemy, the player will culminate in a difficult boss battle.

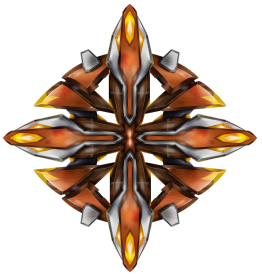
2.2.1. Shooter

Shooter enemy type shoots orange laser which deals 20 damage to player's spaceship. Shooter's collision with spaceship deals 25 damage to the spaceship. Shooters can also move up and down randomly with 120% movement speed. So the movement of this enemy a little bit unpredictable. Also Shooter has 35 health.

2.2.2. Kamikaze

Kamikaze does not shoot but they move aerobically with the purpose of colliding with the player's spaceship. The collision with spaceship deals 30 damage to the enemy spaceship. Kamikazes are really fast so their movement speed is 150% movement speed. Also Kamikaze has 25 health.

2.2.3. Splitter



Splitter enemy type shoots orange laser which deals 20 damage to player's spaceship, the same as shooter type of enemy. Splitter's collision with spaceship deals 25 damage to the spaceship. Splitters can also move up and down randomly with 120% movement speed, the same as shooter. So the movement of this enemy a little bit unpredictable. Also Splitter has 50 health. Splitter's special skill is that

when Splitter is destroyed, the Splitter splits into 2 enemies which are two Shooters and or two Kamikazes.

2.2.4. Boss

The boss come up at five-k levels. Boss moves with 50% movement speed up and down only. However, it deals 25 damage which is the same as shooter's laser but Boss shoots multiple fire so getting shot can be deadly. Collision with Boss is 100 damage which means one shot to player's spaceship. Boss has 500 health .

2.3. Bullet Types

2.3.1. Basic Bullet

This bullet type is the spaceship's original bullet without any upgrade from power-ups. This bullet is the slowest type of bullet with 150% movement speed and it deals 25 damage.

2.3.2. Laser

2.3.2.1. Player Laser

Player laser is a blue laser that faster than basic bullet with 250% movement speed and it has more damage than basic bullet so blue laser deals 35 damage. After the player got a power-up for laser, spaceship's basic bullet or missile evolves into blue laser.

2.3.2.2. Enemy laser

Enemy laser is an orange laser. This laser deals 20 damage to the player's spaceship health. It has %150 movement speed.

2.3.3. Missile

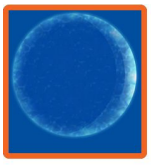
Missile is the most powerful bullet type with its high damage capacity. After the player got a power-up for missile, spaceship's basic bullet or

laser evolves into missile. Missile has the same speed as basic bullet which is %100 movement speed but it is able to destroy the bosses in seconds with 50 damage per missile.

2.4. Power-Ups

Power-ups drop from enemies with a chance when they are destroyed. Power-ups make the player's spaceship more powerful. Power-up types are shield power-up, speed power-up, health power-up and bullet power-up.

2.4.1. Shield Power-Up



If a player picks up shield power-up, the player's spaceship gains a shield which can absorb one hit from enemy laser or one collision with the enemy.

2.4.2. Speed Power-up

If a player picks up speed power-up, the player's spaceship gains speed so that player can move faster and dodge the enemy laser and kamikazes. While regular movement speed of the enemy is 100%, the

movement speed of the player spaceship becomes 120% of its regular speed with this power up. For instance, if the player collects 1 speed power-up. It's movement speed becomes 120% and collects one more time then the movement speed of the player becomes 144%.

2.4.3. Life Power-up

If a player picks up life power-up, the player gains one extra spaceship which determines the player's life.

2.4.4. Bullet Power-up

If a player picks up bullet power-up, the player's spaceship's bullet type evolves into laser or missile.

3. Functional Requirements

3.1. Player Requirements

- The user shall be able to launch the game.
- The user shall be able to click the exit button to exit the game.
- The user shall be able to use the arrow keys to move the player up, down, left, or right.
- The user shall be able to press spacebar to attack enemies.
- The player can pick up bonuses.
- The user shall be transferred to the Main Menu screen again if the player has no life left.

3.2. Game Requirements

The Player, Enemies, bonuses and the map in the game are dynamic objects because all components in this game are moving. As it is a kind of Defender game, even the map is moving as the player moves on it.

The game should include requirements below:

- The game will be implemented with Java language.
- The game will be displayed on a Java Frame and will be 1300px x 800px in size.
- The game shall have randomly generated game play elements properly.
- The game shall have been controlled by a keyboard.

- The game must have a main menu screen with buttons that allow navigation to the game screen, How to Play screen, Settings screen and Credits screen.
- The current score are displayed on the game screen.
- Music will play throughout the game.
- A sound will play when the player fire bullets.T
- The game will feature appropriate sound effects for enemies.

3.3. Play Game:

The player will start the game by clicking on the Play button. The player controls a spaceship with control buttons mentioned in How to Play scene. Enemy spaceships which are Kamikaze, Shooter and Splitter starts to randomly appear from both sides of the screen. The player is able to control the spaceship both vertically, horizontally and diagonally. Also the player can use spaceship' fire by pressing spacebar. The amount of the enemy ships increases over time. The main objective is to avoid taking damage, saving humans, destroying a specific number of enemy ships and then defeating a boss.

To see points gained from enemies, enemies' damages , bullet types and bullet damages, two tables are shown below.

SHIP TYPE	NAME	HEALTH	BULLET DAMAGE	COLLISION DAMAGE	POINT GAINED	MOVEMENT SPEED
	SHOOTER	35	20	25	10	120%
	KAMIKAZE	25	—	30	15	150%
	SPLITTER	50	25	25	5+(10 OR 15) II 15 OR 20	120%
	BOSS	500	20 (RAPID FIRE)	100 (ONE SHOT)	100	50%
	PLAYER	100	25 (BASIC BULLET)	25	—	100%

BULLET TYPE	NAME	BULLET DAMAGE	MOVEMENT SPEED
	BASIC BULLET	25	150 %
	PLAYER LASER	35	250 %
	PLAYER MISSILE	50	100 %
	ENEMY LASER	20	150%

3.4. How to Play:

The player can go to this section to learn about the game controls and rules. This section will also tell the player about how much damage will each type of projectile do and the information on the bonuses.

3.5. Settings:

Here the player can control volume and turn game music on or off. Settings scene can be opened from the main menu and the pause scene.

3.6. Credits:

This section will display the name and the main roles of the developers. Credits can be opened from the main menu only.

4. Non-functional Requirements

4.1. Performance:

The game works at least 30 frames per second so that the game will be fluent.

4.2. Security:

In the game, only player high scores are kept in a text file so that there is not any important information that is kept about the user. The key point for security is also that the game doesn't need to internet connection. Thus, there will be less threat from online interference as the game runs offline.

4.3. Supportability:

The game should be able to be run on all of the desktop computers supporting Java.

4.4. Extendibility:

It should be possible to add new objects such as new types of ships to the project without affecting the other objects. Furthermore, the attributes of the current objects such as speed should be able to be changed separately without affecting the other objects.

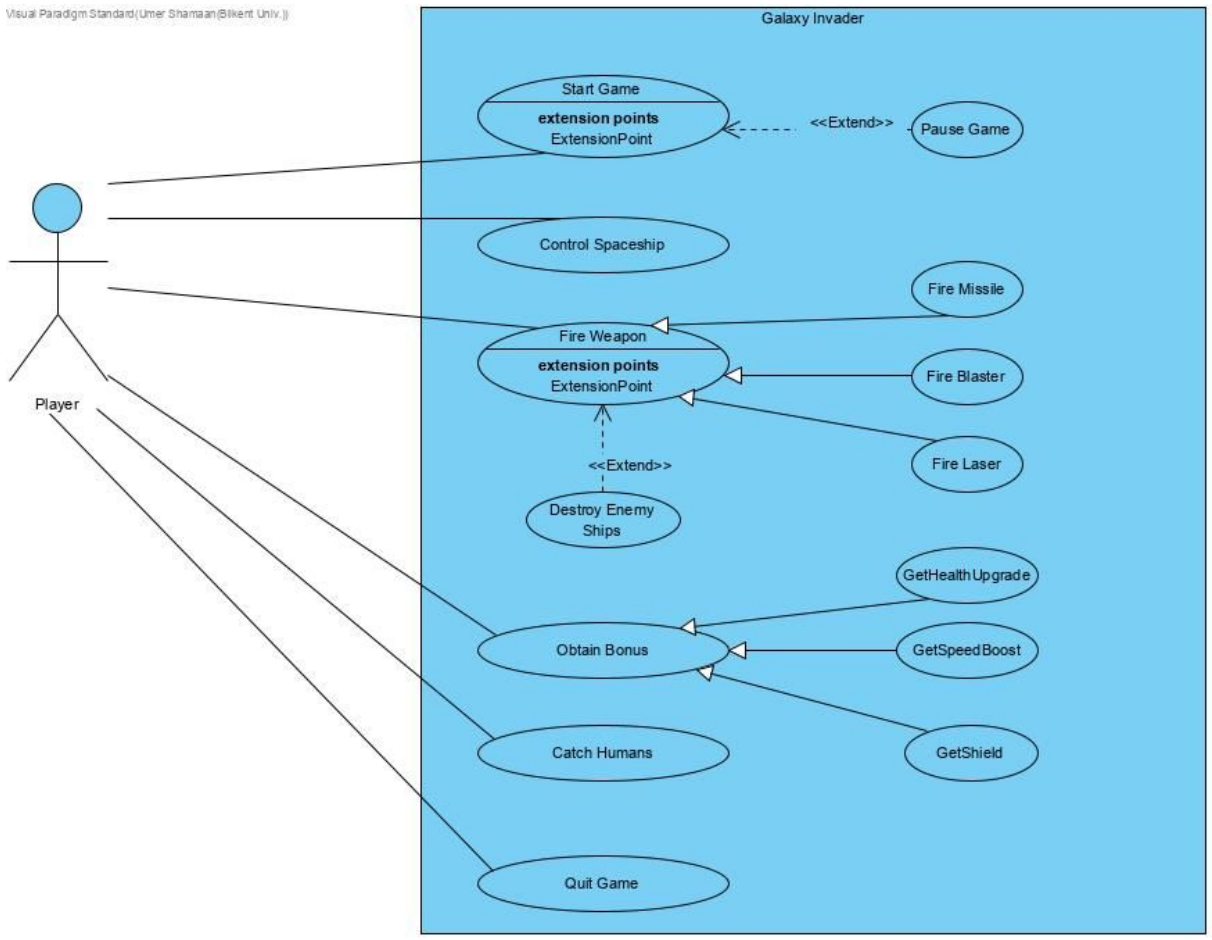
4.5. Portability:

It should be possible to move the game from one desktop computer to the other without creating any problems.

5. System Models

5.1. Use Case Models

5.1.1. Gameplay



Use Case 1:

Use Case: Start Game

Primary Actor: Player

Entry Conditions:

- The player must click on ‘Play Game’ button on the Main-Menu.

Exit-conditions:

- The player should defeat all Enemies.
- The player should click in the Esc key.

Successful Event Flow:

1. Players start the game.
2. They fire bullets by pressing the fire key.
3. The fired bullets hit the enemy ships and decrease their health.
4. All of the Enemy ships get destroyed after their hp reaches zero.

Use Case 2:

Use Case: Control Ship

Primary Actor: Player

Entry Conditions: Player has to start the game.

Exit-conditions:

- Players have to destroy all Enemy Ships
- Player has to press escape

Successful Event Flow:

1. Player starts the game.
2. Player presses the arrow key.

Use Case 3:

Use Case: Fire Weapon

Primary Actor: Player

Entry Conditions: Player has to start the game

Successful Event Flow:

1. Player starts the game.
2. Player presses the space-bar key.

Use Case 4:

Use Case: Obtain Bonus

Primary Actor: Player

Entry Conditions:

- Player has to start the game.
- Player has to destroy a certain amount of enemy ships.

Exit Conditions:

- Player collides with the bonus

Successful Event Flow:

1. Player starts the game.
2. Player defeats a certain amount of enemies.
3. The system spawns a bonus.
4. The player Collides with the Bonus object.

Use Case 5:

Use Case: Catch Humans.

Primary Actor: Player

Entry Conditions:

- The Player has to start the game.
- The Enemy ship should spawn the human with it.
- The player has to destroy that ship.

Exit-conditions:

- The player collides with the human.

Successful Event Flow:

1. The player starts the game.
2. The player defeats an enemy ship carrying the human.
3. The system drops the human towards the ground.
4. The player catches the human.

Use Case 6:

Use Case: Exit Game

Primary Actor: Player

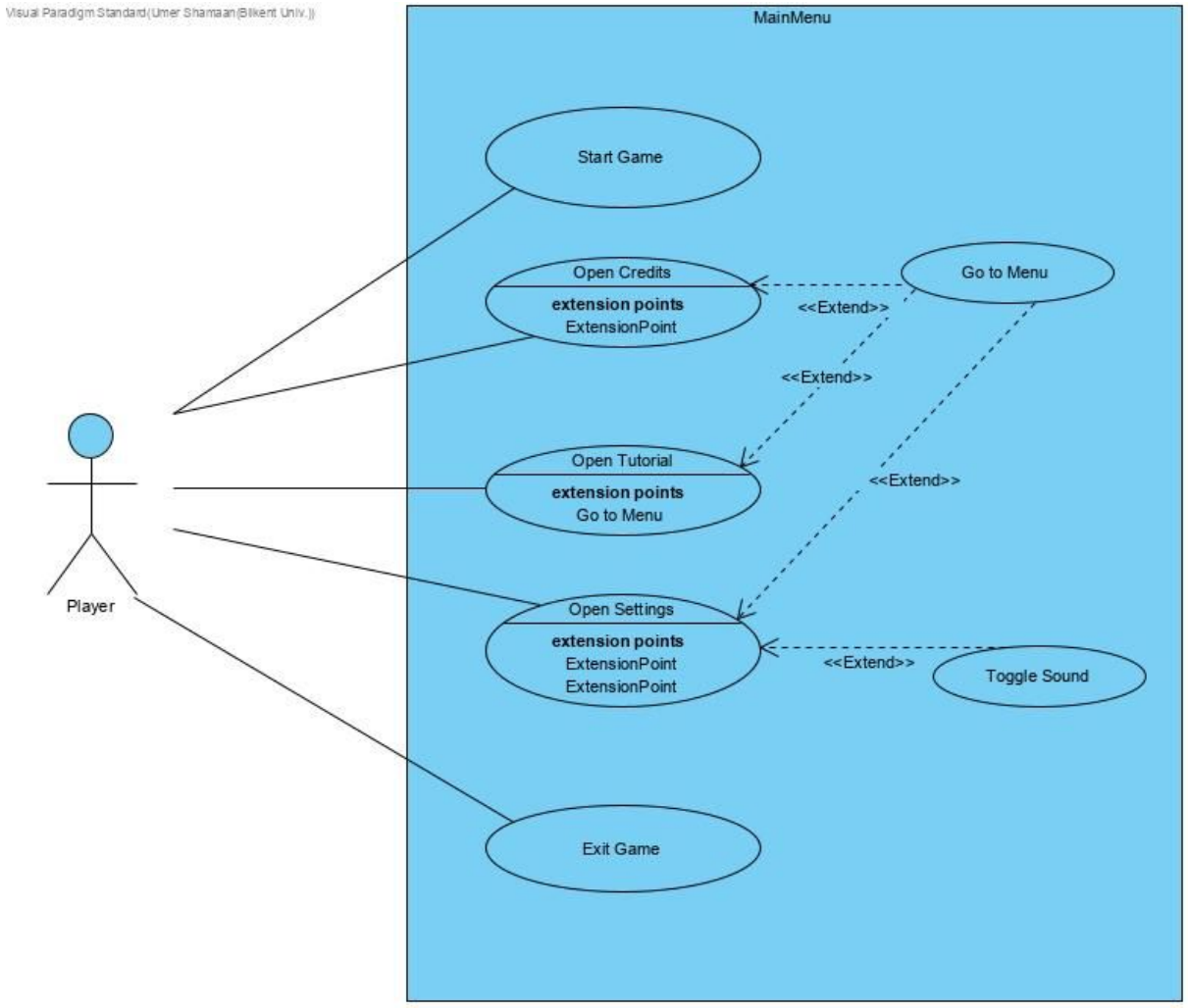
Entry Conditions:

- The player starts the game

Successful Event Flow:

1. The player starts the game.
2. The player presses on the escape key.

5.1.2. Main Menu Navigation



Use Case 1:

Use Case: Start Game

Primary Actor: Player

Entry Conditions: The player presses ‘Start Game’ on the main menu.

Exit-conditions:

- The exits the game window by clicking on the close button on the window

- The player presses on the escape key

Use Case 2:

Use Case: Open Credits

Primary Actor: Player

Entry Conditions:

- Player has to be on the Main Menu Screen

Exit-conditions:

- Player clicks on ‘Go Back’ button.
- Player presses the escape key.

Successful Event Flow:

1. Player clicks on the Credits Button on the main menu.
2. Player exits the Credits Panel by pressing escape or the back button.

Use Case 3:

Use Case: Open Tutorial

Primary Actor: Player

Entry Conditions:

- Player has to be on the Main Menu Screen

Exit-conditions:

- Player clicks on ‘Go Back’ button.
- Player presses the escape key.

Successful Event Flow:

1. Player clicks on the ‘How to Play’ Button on the main menu.
2. Player exits the Tutorial panel by pressing escape or the back button.

Use Case 4:

Use Case: Open Settings

Primary Actor: Player

Entry Conditions:

- Player has to be on the Main Menu Screen

Exit-conditions:

- Player clicks on ‘Go Back’ button.
- Player presses the escape key.

Successful Event Flow:

1. Player clicks on the ‘Settings’ Button on the main menu.
2. Player changes the sound Settings
3. Player exits the Settings Panel by pressing escape or the back button.

Use Case 5:

Use Case: Exit Game

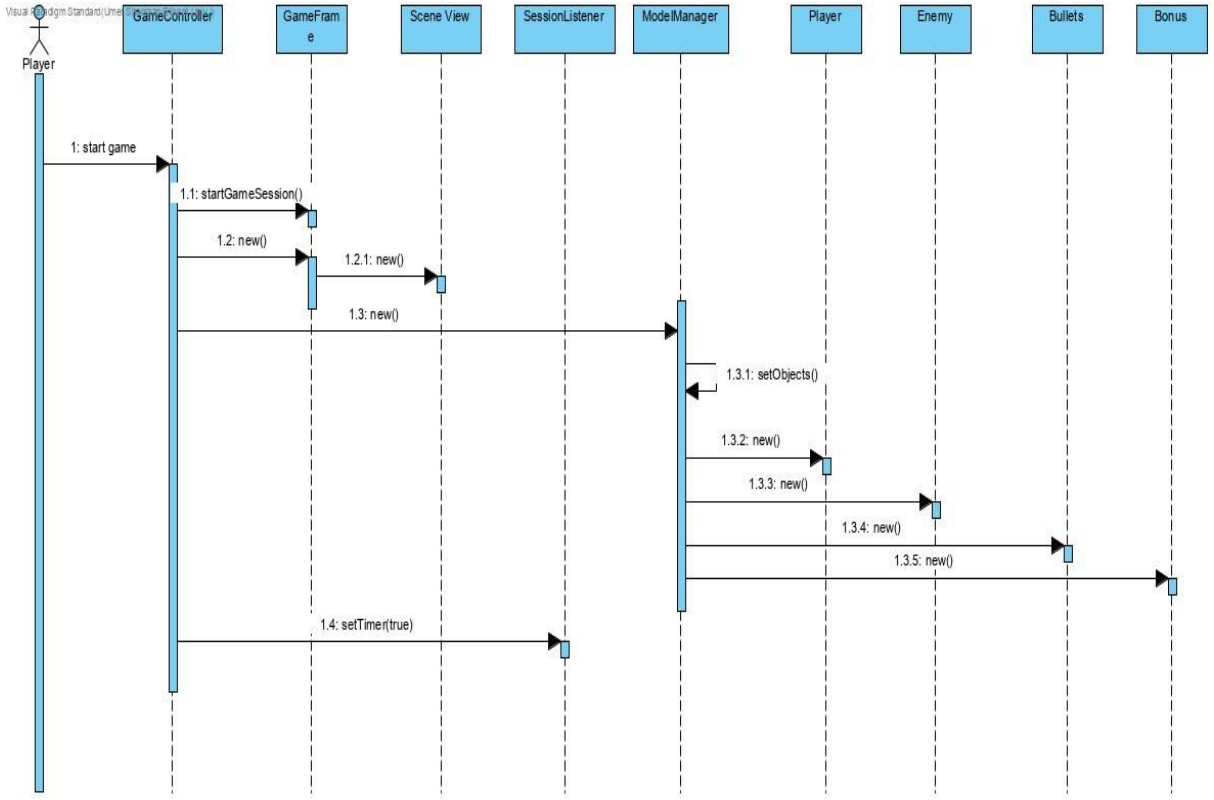
Primary Actor: Player

Successful Event Flow:

1. Player clicks on the ‘Quit Game’ Button on the main menu.
2. Player exits the Credits Panel by pressing escape or the back button.

5.2. Sequence Diagrams

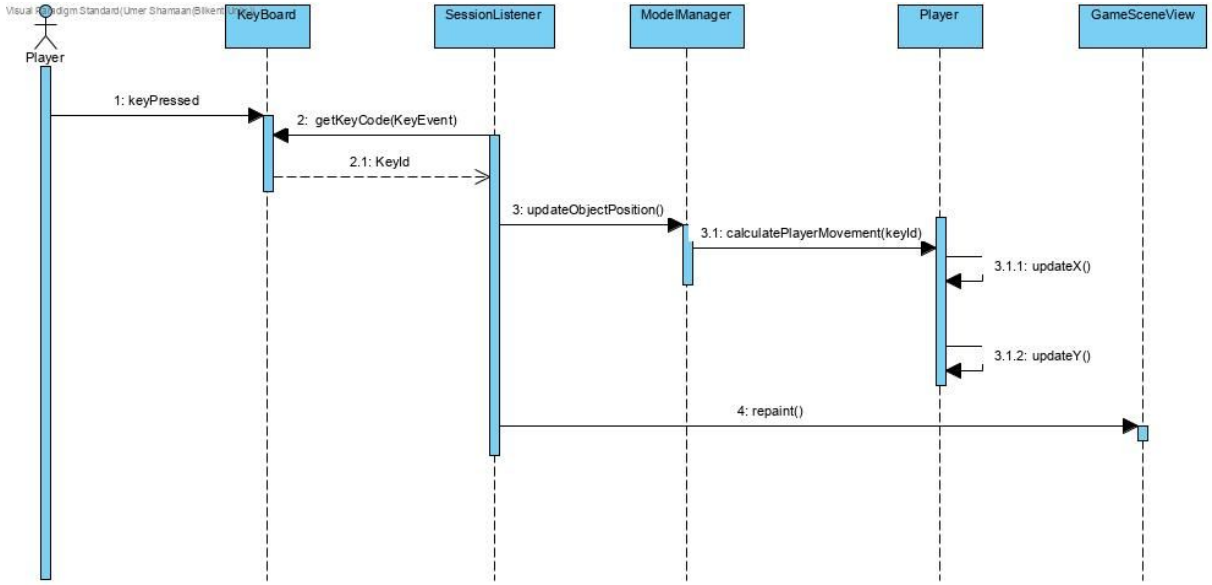
5.2.1. Starting a Game



Scenario: The player starts a new game.

When the player starts a new game from the menu, the GameFrame initializes the Scene View on which the actual gameplay will take place. The Game Controller also intializes the Model Manager which in turn creates the instances of Model Objects which will be observed on the new Scene. After that the Timer of the Session Class starts to tick.

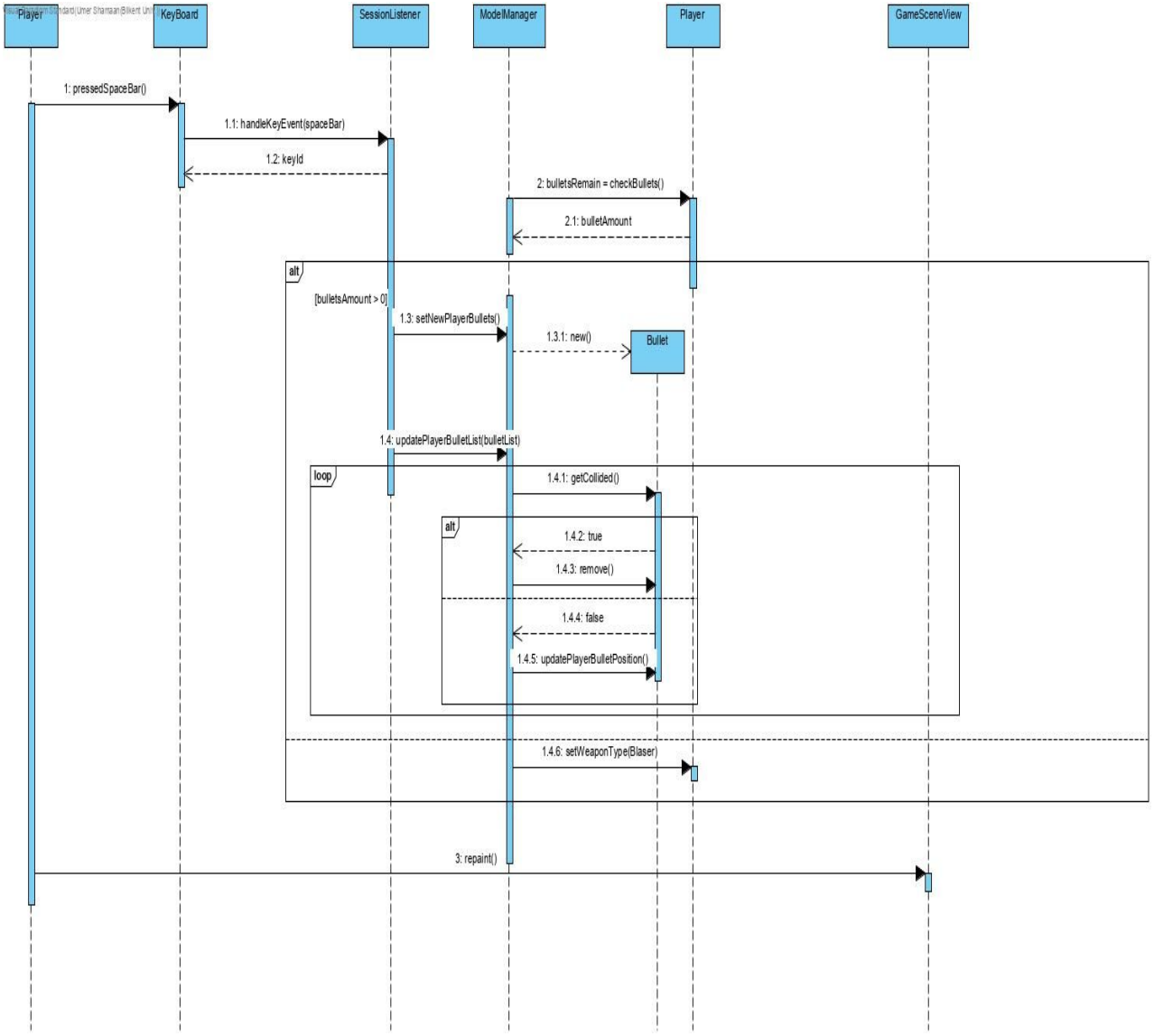
5.2.2. Controlling Player Ship



Scenario: The player presses on an arrow-key to control the ship.

The keylistener determines the code of the key that has been pressed. The session listener then calls on to the ModelManger to update the x-coordinate or the y-coordinate of the player ship depending on the key pressed.

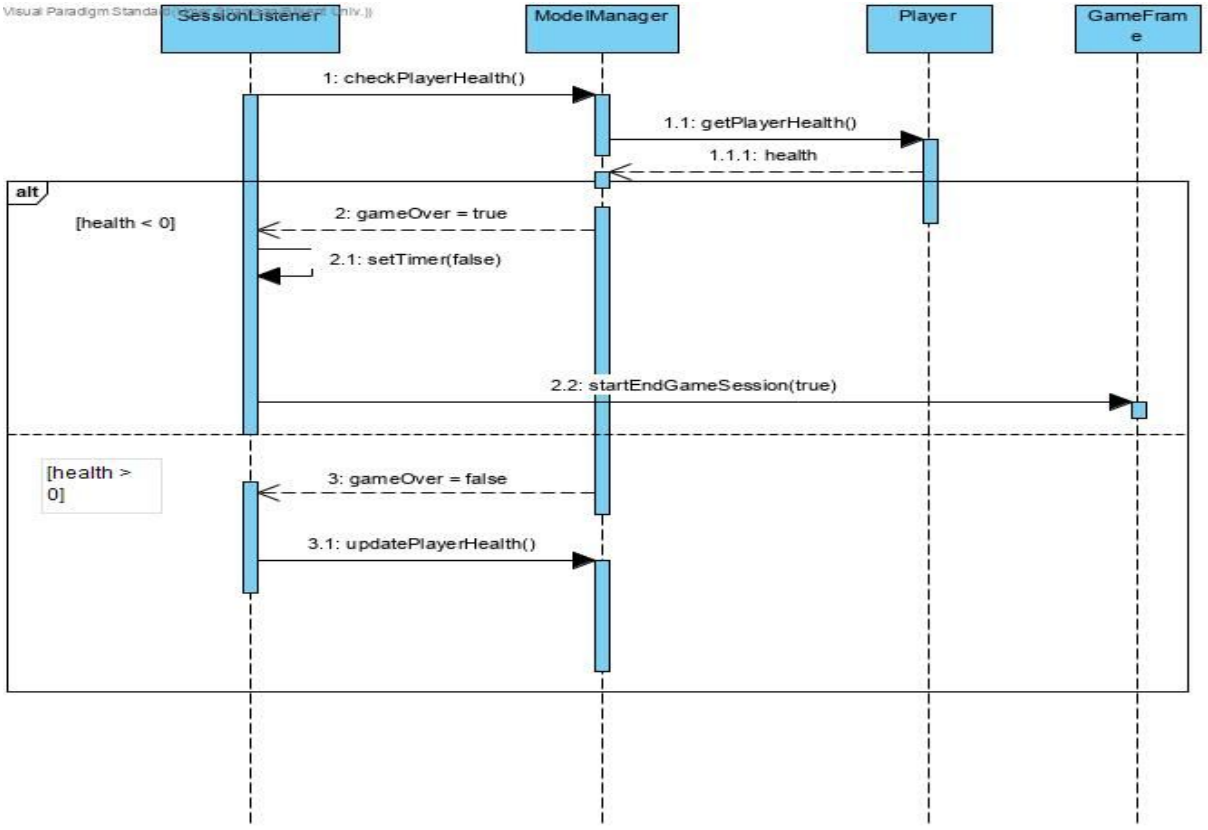
5.2.3. Firing Bullets



Scenario: Firing Bullets.

The keylistener determines the code of the key that has been pressed. The Model Manager checks if the player ship has any remaining bullets (other than blaster) and then creates a new instance of the bullet and adds it to the list of the player bullets. It then continuously checks if the bullet has collided with an object. If so the bullet is removed from the list. Otherwise, its position is updated.

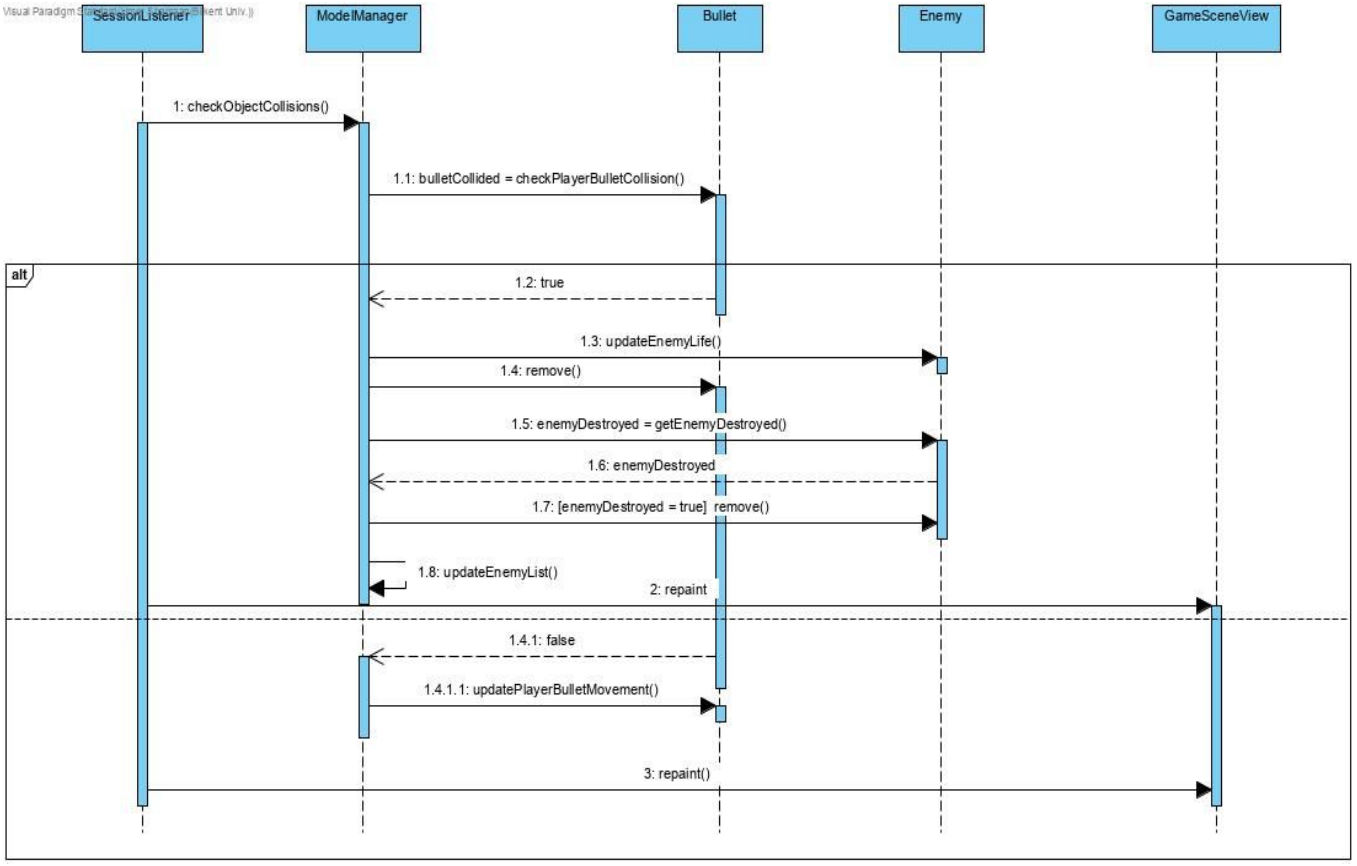
5.2.4. Checking Player Health



Scenario: Checking players' health to determine if game is over after they are hit

The action listener checks for the player health through the model class after every tick. If the health reaches zero, the timer is stopped and the game is over. Otherwise, the player health is just updated and the game continues to run.

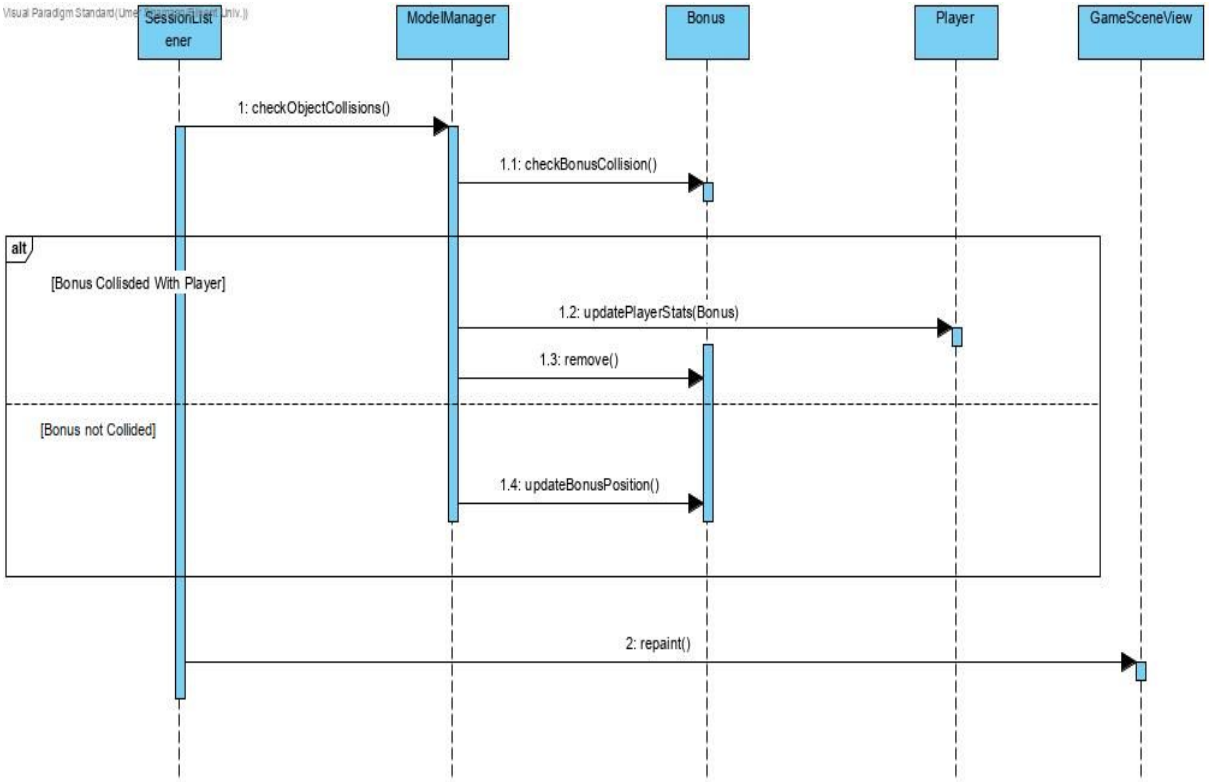
5.2.5. Hitting Enemies



Scenario: The player bullet hits an enemy.

The action listener checks through the Model Manager if the player bullet has hit one of the enemy ships in the list. It then checks if the status of the Enemy has been set to “destroyed”. The ship is removed from the list if the status is true. If the Enemy has not collided with the bullet, it’s position is simply updated after the tick.

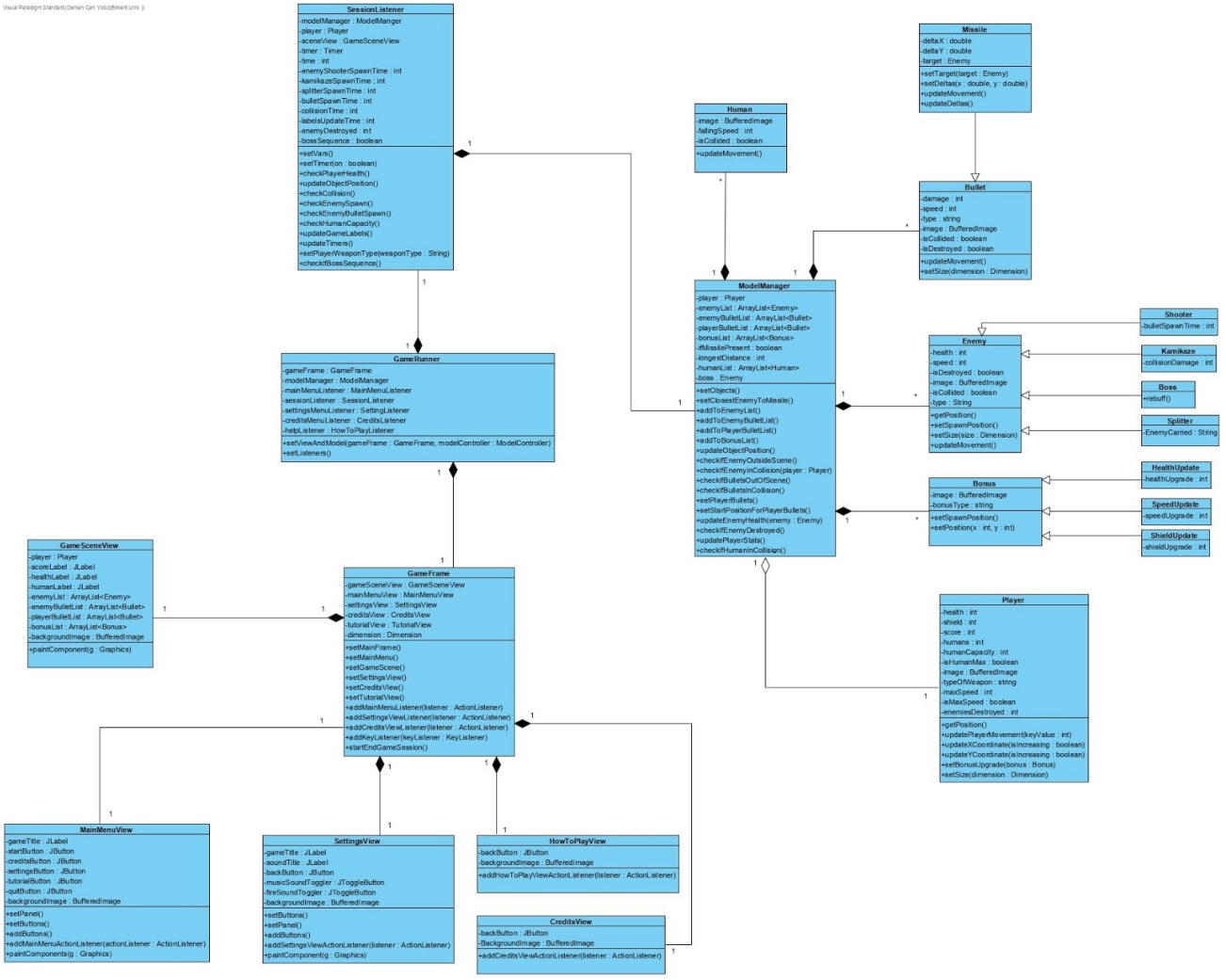
5.2.6. Acquiring Bonus



Scenario: A bonus appears in to the scene and the player hits or misses it.

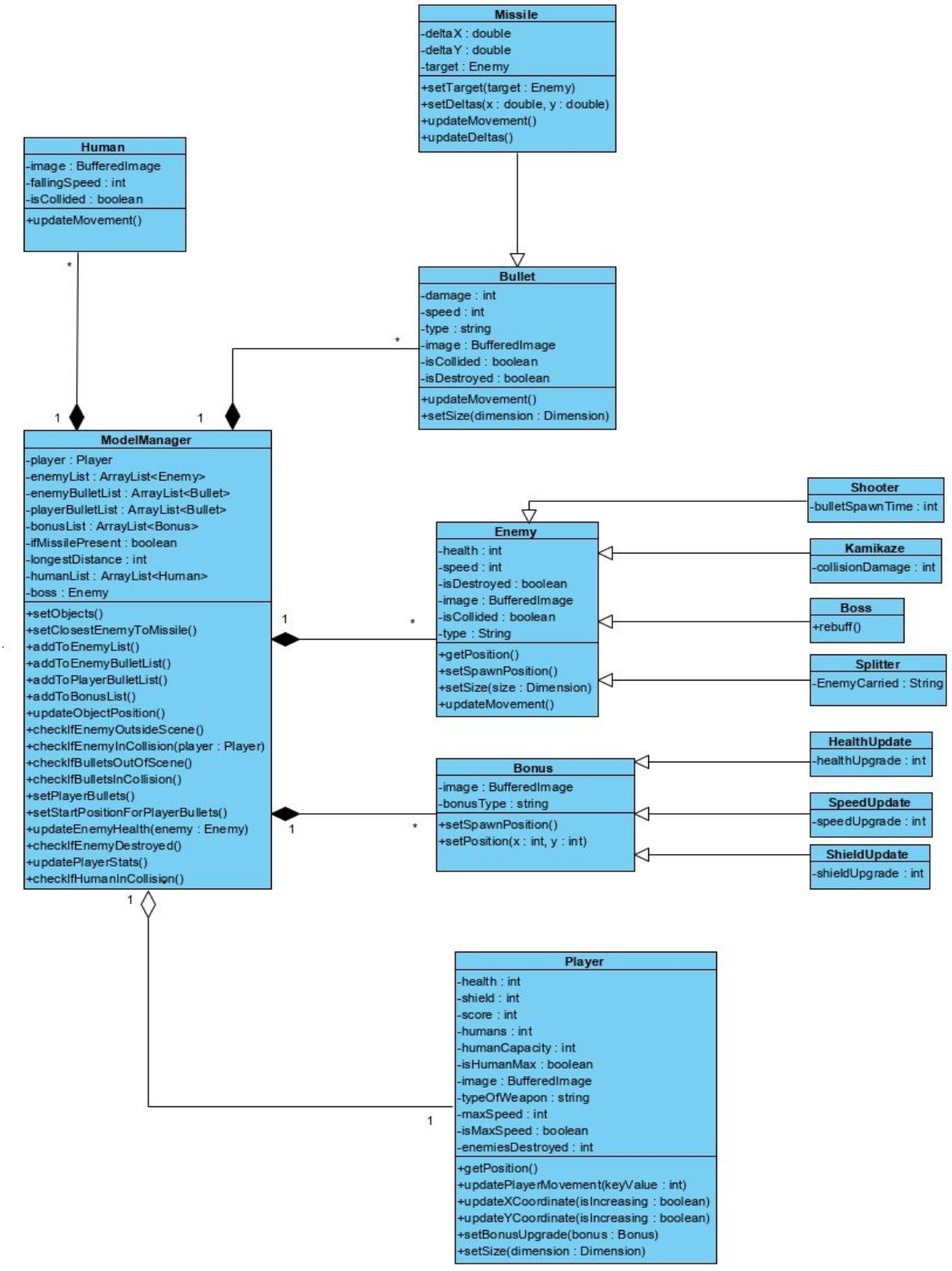
The action listener checks through the Model Manager if the player collided with one of the bonuses in the list. In the case of the collision, the player stats are appropriately updated according the type of bonus it has collided with. Otherwise, the position of the Bonus object is updated.

5.3. Class Diagrams

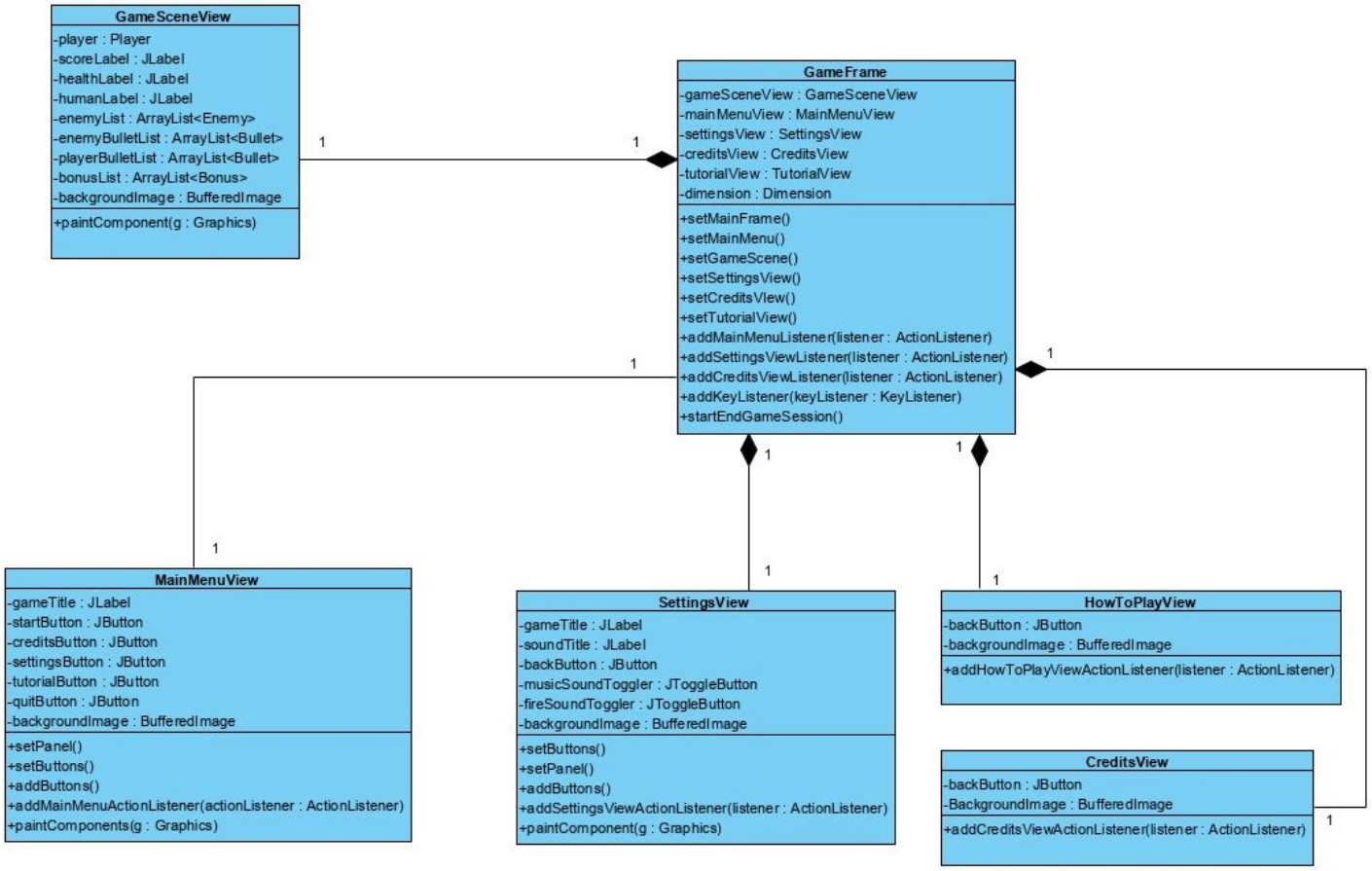


To see the class diagram we divided into 3 pieces.

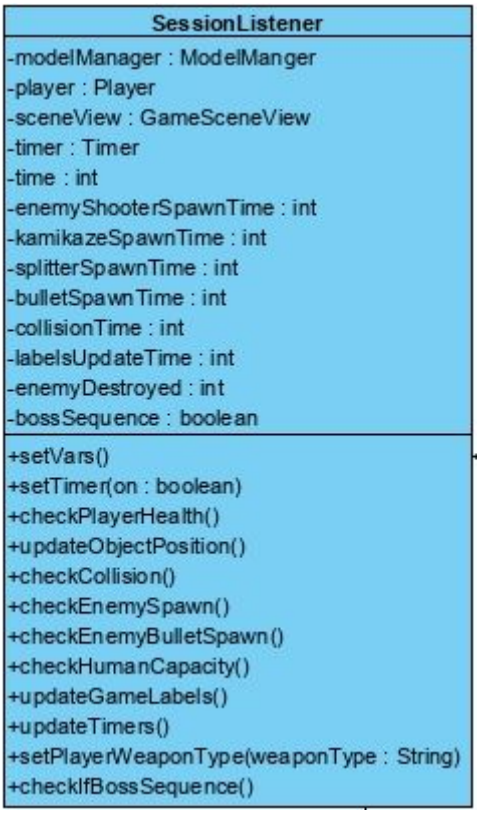
5.3.1. Model Part



5.3.2. View Part

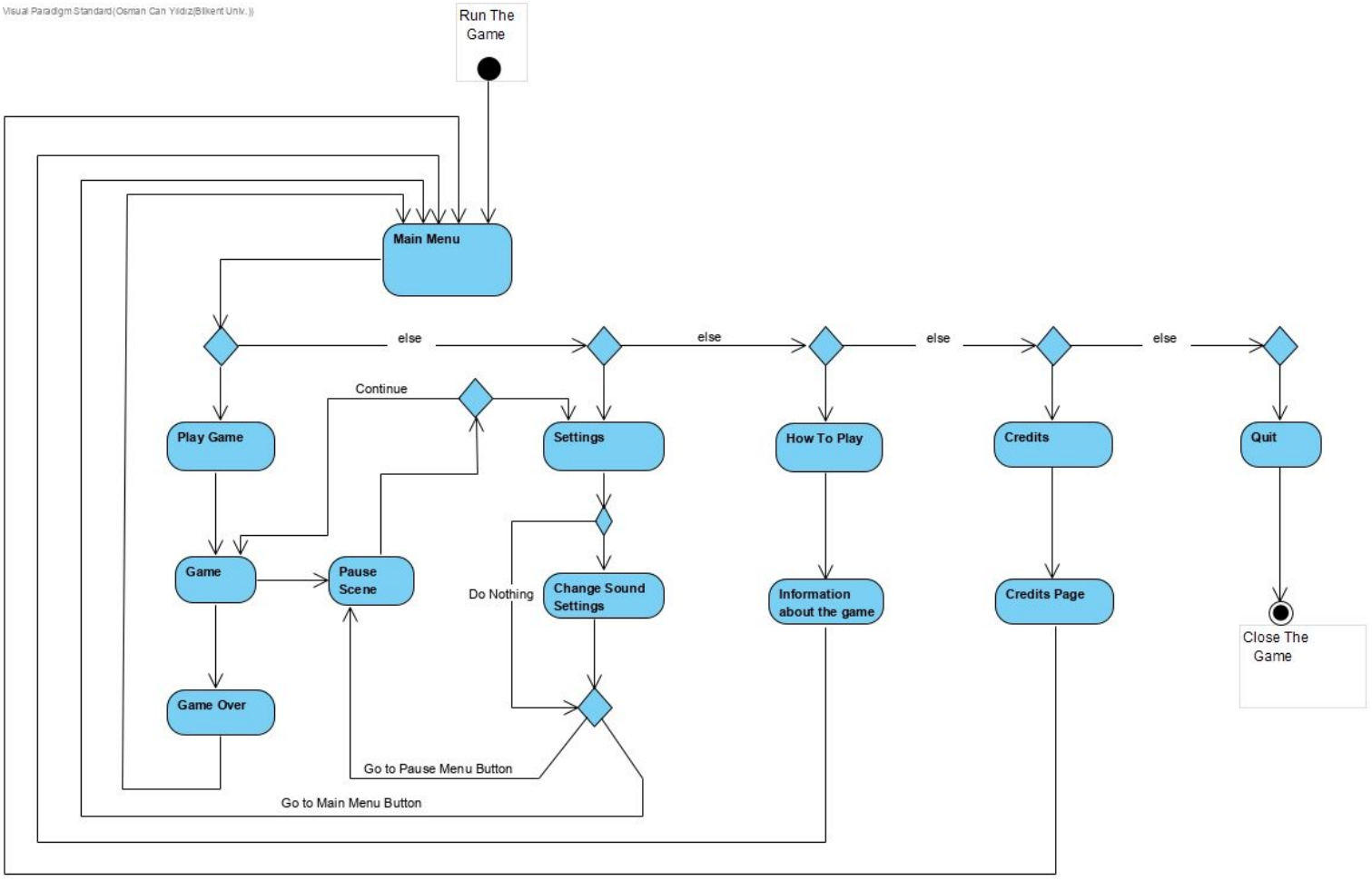


5.3.3. Controller Part

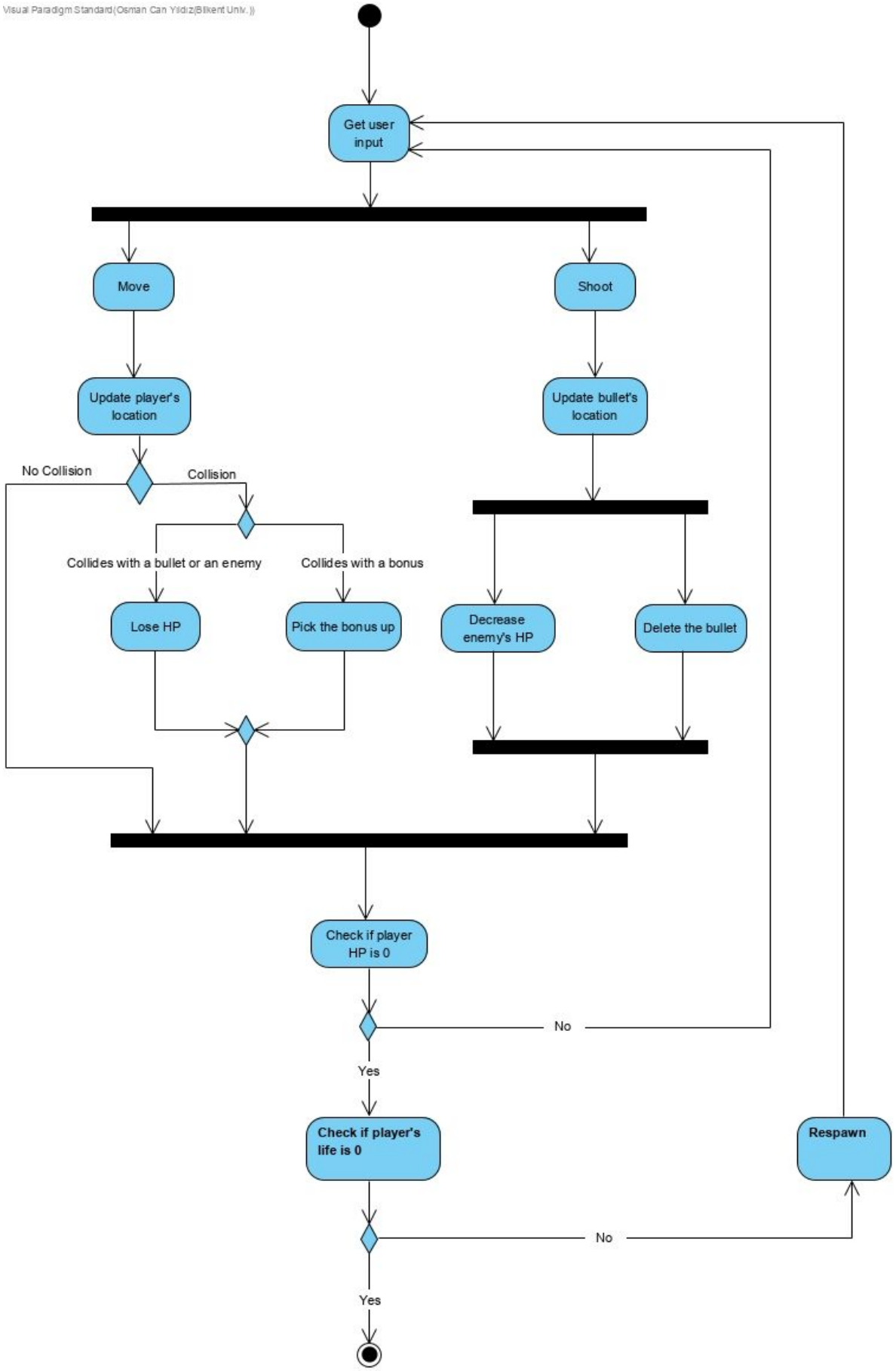


5.4. Activity Diagram

5.4.1. Main Menu Activity Diagram

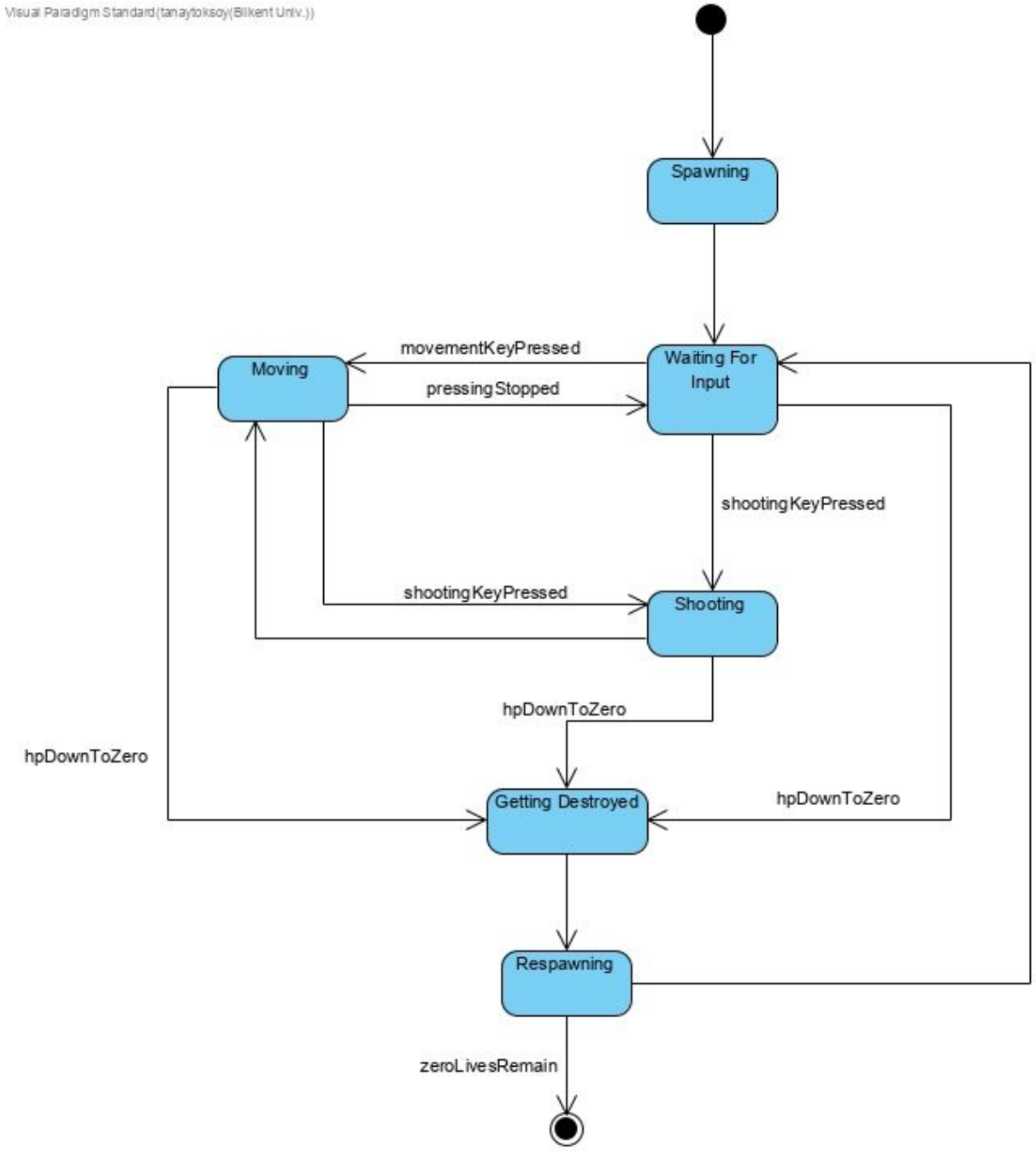


5.4.2. Play Game Activity Diagram

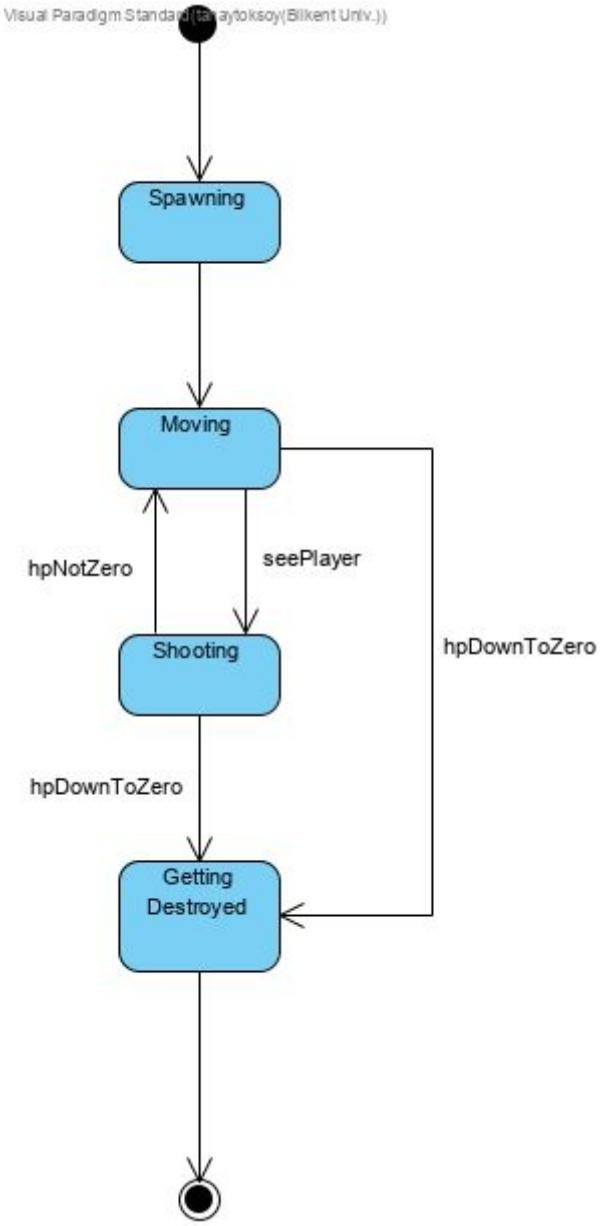


5.5. State Diagrams

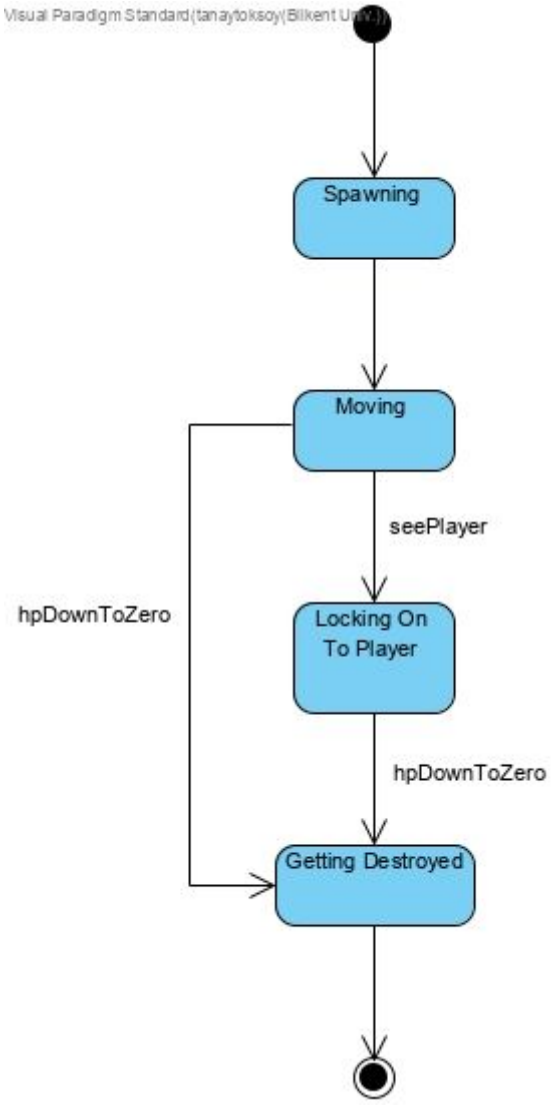
5.5.1. Player



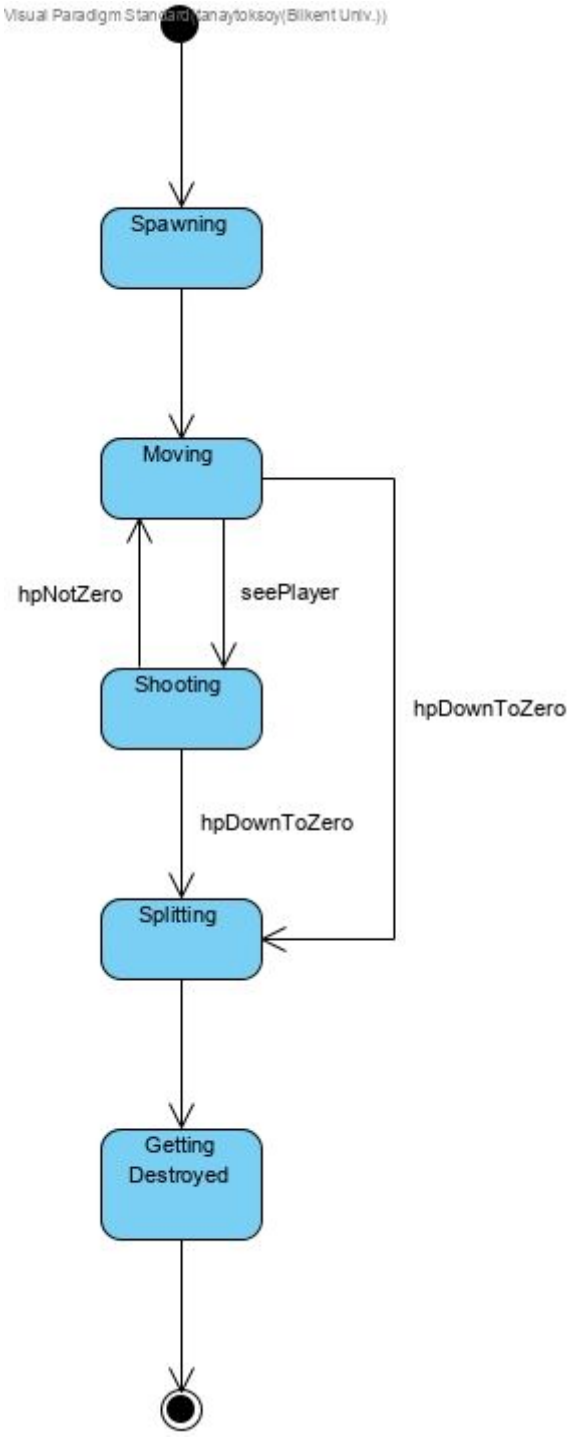
5.5.2. Shooter


Visual Paradigm Standard (tarayoksoy(Bilkent Univ.))
Spawning
Moving
hpNotZero
seePlayer
Shooting
hpDownToZero
hpDownToZero
Getting
Destroyed

5.5.3. Kamikaze



5.5.4. Splitter



5.6. User interface and Screen Mock-ups

5.6.1. Menu

In menu screen, four buttons are provided that they are “start new game”, “how to play”, “settings” and “quit”.

5.6.2. How to Play

In how to play screen, there is a short definition about the purpose of the game. Action buttons which are moving left,right,up and down and firing button are shown below the definition of the game.

5.6.3. Settings

In settings screen, player can turn on or off the music and sounds of the game.

5.6.4. Credits

5.6.5. Game Screen

In game screen, the game will be running like the samples. Enemies come from left and right and also they shoot to spaceship. Rescued humans that are attached to enemies can be seen on the bottom left corner of the game after destroying enemy. While score is on the top left corner, health which shows how much a player can die is on the bottom right corner.

Bullet power-up is dropped from enemies. In this frame, Bullet Power-Up will upgrade basic bullet to laser as seen below.

The bullet type of the player's spaceship is changed from basic bullet to laser.

Kamikazes are dangerous because they are hard to see because of their camouflage and they move unpredictably.

Boss is the most powerful enemy and its special rapid fire is seen below so boss's fire forces player to move carefully.

Player’s spaceship is protected by shield power-up. After one hit or collision, the barrier will be destroyed.

References

[1] "Defender (1981 Video Game)." *Wikipedia*, Wikimedia Foundation, 9 Nov. 2019, https://en.wikipedia.org/wiki/Defender_(1981_video_game).