CS 319 - Object-Oriented Software Engineering

Analysis Report

Armageddon

<u>Group 4</u>

Özge Karaaslan	*21301835*
Onatkut Dağtekin	*21300801*
Teyfik Rumelli	*21102161*
Saner Turhaner	*21100475*

Deadline: 28/10/2015

Course Instructor: Özgür Tan

Contents

List of Figures:

1. Introduction

As a group we intended to design and implement a game called *Armageddon* which is a multi-directional shooter space game based on destroying enemy spaceships, asteroids and surviving. There are five levels in the game. Transitions between sequential levels are smooth such that game continues when level goes up. If player passes these five levels successfully the game ends. If player is destroyed by enemy spaceships or asteroids at three times the game is over. A level is cleared when the player destroys all enemy ships which are coming in certain number, otherwise level continues until player destroys ships moving on the screen or player's ship is destroyed with the result that game is over.

Aim of this project is to make users feel the joy of ability games with minimal and smooth graphics. Armageddon is a kind of arcade game; playing Armageddon needs coordination of mentality and sleight of hand together. So users should think faster and decide faster while playing the game. As a group our principle while creating the idea of this game is keep it as simple as possible while making it as much as entertaining and irreplaceable.

In this report we are going to explain the overview of the game, we specify the gameplay contents and we are going to indicate general rules of Armageddon. This report includes the architectural patterns which will be used in our game. As an analysis report it also gives detailed information about functional requirements, non-functional requirements and constraints of the project. In the "System Models" part; inclusive scenarios, use-case models, object models, dynamic models and interfaces of the game will be analyzed. At the end of the report, there are "Glossary" and "References".

2. Requirements Analysis

2.1. Overview

At the beginning of our game, player controls a spaceship which moves from left to right. First, there are just asteroids in the space which are coming towards our spaceship. Player can skip out from these objects; spaceship can fly any direction on the screen by pressing direction buttons in keyboard. Our essential mission is keeping spaceship alive. Then enemy spaceships come within few minutes. At the very first level of the game enemies are not very harmful. These harmless spaceships coming first are generally fling from right to left and do not shoot, but if we hit any of these spaceships it will decrease our shield strength. In progress, enemies are going to be more harmful starting with first level.

The player, controlling a spaceship, must destroy twenty enemy spaceships in each level in order to level goes up. These twenty spaceships are certain ships which do not go away until player destroys them. They come from right side of the screen like others but when they come they start moving at y-axis on the right side and shooting. Apart from that, fuel of the spaceship that player controls is limited and it decreases constantly. Enemy spaceships enter the playfield from the right side of the screen. The aim of the enemy spaceship is simply to destroy the player's ship. The level is completed when all the enemy spaceships are destroyed. During the game, there will be space station in each level that supply fuel, repair the shield and give specific ammunition which are controlled by one of “a” or “s” buttons of keyboard. Extra ammunition will be shown on the left down of the screen with its number and specified button.

While the game is on, the game music is playing constantly and there will be fire and explosion sounds. The player can stop the game at any time and simple settings panel appears when the game is paused. Player can adjust music and sound levels using this panel and continue afterwards. Moreover, after every level is passed a short password reflects on the screen for the game can be loaded later from that level. It allows the player to resume the game afterwards via

“Enter a Password” option of main menu. User collects variety of points when it destroys space objects. These points are recorded after the game is over if user’ points are enough to get into the first ten highest scores. And also there is “High Scores” option in the main menu for players to see top ten highest scores.

Armageddon consists of 5 levels. At every level the enemy spaceship types are varied. Their appearance will also differ too. In addition to this our spaceship ammunition will be enhanced level to level. For instance, default type gun destroys GoblinY enemy spaceships with 3 times shoot. But Laser destroys them by 2 shoots. Besides, rocket destroys a GoblinY enemy at first shoot. Shoot appearances will also differ in the game.

2.2. Functional Requirements

Play Game:

Armageddon’s main purpose is to destroy enemies and objects that it may encounter. At the start, the player has three lives and a full shield strength. The game is over when the shield is depleted and the ship gets hit by one of the enemy ships and the player has no more lives. The game will consist of 5 levels. To get to the next level, the player must survive the waves of enemy ships and asteriods.

The player will be able to strengthen his/her ship by going into the stations to get the ships’s shield recovered, to fill its fuel deposit and to receive a better weapon by luck. When he/she gets his weapon upgraded more difficult enemies will appear making the game more challenging and enjoyable.

Change Settings:

Other than the default settings implemented by the system, the user can change it as well. These settings include:

- Sound On/Off
- Music On/Off

Basically changing these settings would make the game more quiet and the user can change them while playing the game and pressing they escape key. Both of the options will be on by default.

High Scores:

Players can see the top 10 scores in a list view with the person’s name and his/her score. This list can be seen as a motivation to the players since a player might want to get into that list.

About the Game (Credits):

Players will get general information about the game such as:

- Weapon Types
- Enemy Types
- Controls

By reading the player will be instructed on how to play the game and enjoy it in a better way. It will also have a part about the creators of the game.

Pause Game:

While playing the game the user can pause it to exit the game or change the settings. If he chooses to exit all progress will be lost and player’s score won’t be recorded for the high scores list.

Enter Password:

The player can enter a password to get to a specific level. If the player enters a password to get to a specific level his score will start from zero. Passwords will be given at the end of a level. As a reminder to the player, the password will be printed on the screen again if the ship gets destroyed.

2.3. Non-Functional Requirements

Performance:

To increase performance we will use limited hard drive space and store passwords and high scores on text files. By reading those information from the text files it will be easier to create a table and use it whenever necessary by doing so the application will not waste memory. Since the game is for single player, it will not use a server which means there is no need to wait for response from other sources.

User Friendliness:

The game will not have complex structure so the user can understand it easily. Simple controls mean that the user does not need to have lightning reflexes to play this game. The game can be opened whenever user wants to play it and since it is a single player game user has only himself/herself to compete against.

2.4. Constraints

- The game's language will be English.
- The game will be developed and implemented in Java.
- If the user wants to play the the game he/she must have Java Runtime Enviorment installed.
- The space this game will occupy will not go beyond 30 MBs.

2.5. Scenarios

Scenario #1: Starting Game

Player Saner requests to start game by clicking “New Game” button from Main Menu. After that system initializes GamePanel into the frame and GamePanel sends message to LevelManager to create objects. LevelManager initializes objects and gets images of objects; background, spaceship that player uses, enemy spaceships, space station and asteroids from the file directory of the game. Then GamePanel locates all objects to appropriate locations indicated in the LevelManager according to LevelNo. While loading the level, GamePanel reads last settings from the text file. Lastly, GamePanel starts the game loop and repaints all the objects related to Level and updates the game panel.

Scenario #2: Shooting War

Player Ozge requests to start the game by clicking “New Game” button from Main Menu. Assuming that steps of the previous sequence diagram are performed, in the Game Panel whole game dynamics are arranged; current location of the spaceship that Ozge uses is taken, space objects and bullets of the enemy spaceships that are on the screen are manipulating, spaceship that Ozge uses is moving and shooting. GamePanel also checks collisions; assuming that player Ozge’s spaceship has been collided with the bullet of the enemy spaceship and is damaged, Ozge directs the spaceship using direction keys from the keyboard in order to avoid the enemy bullets. After that Ozge shoots and her bullet is collided with the enemy then the enemy spaceship is destroyed.

Scenario #3: Changing Settings

Player Ozge request to start a game by pressing the “New Game” from Main Menu. Assuming that the steps of the Start Game sequence diagram are performed, Ozge wants to change the game setting as she desires. In order to do that Ozge pauses the game by pressing Esc key from the keyboard. Small SettingPanel shows up on the game screen and game is stoped. Ozge changes sound and music volumes by sliding bars on this panel. New volume settings are arranged by the SettingPanel. After that Ozge clicks to “Return to Game” button and game continues.

Scenario #4: Entering Station

Player Saner request to start a game by pressing the “New Game” from Main Menu. Assuming that the steps of the Start Game sequence diagram are performed, Saner's spaceship runs out of fuel and his shield is damaged during the game. Saner presses down button from the keyboard and spaceship enters the station in order to repair his shield and load fuel. Game is paused when spaceship is in the space station. Then it gets an extra bullet from space station which is specific to the current level. After a certain period spaceship leaves the station and game continues.

2.6. Use Case Model

2.6.1. Use Case Diagram

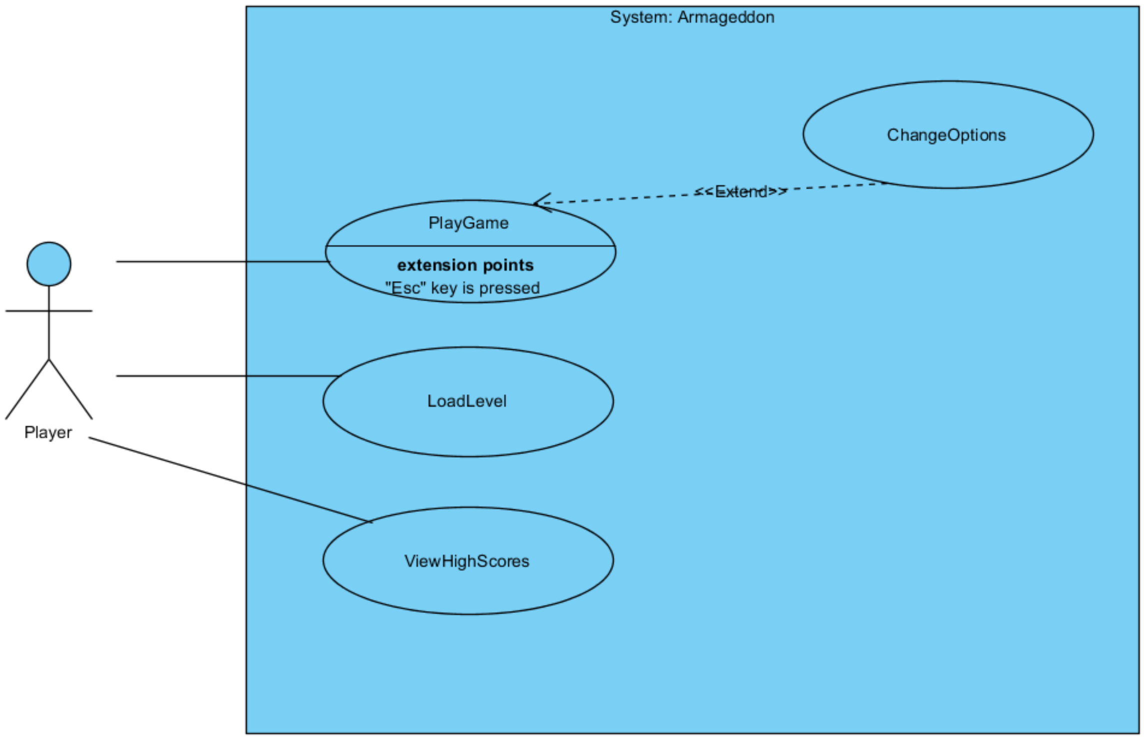


Figure 1: Use Case Diagram for the System

2.6.2. Use Case Descriptions

Use Case - 1

Name: PlayGame

Actors: Player

Entry Condition: The game is started and on the main menu player should choose "Play Game" operation.

Exit Condition:

- All the five levels are completed successfully.
- Player has lost all of his/her lives.
- Player stops the game by pressing "Esc" key from keyboard and decides to return main menu by mouse left click.

Main Flow of Events:

1. Level 1 is constructed by default and player starts to play game.
2. Player uses direction keys from keyboard for direction of the spaceship and "a", "s", space buttons from keyboard for shooting.
3. Spaceship shoots and destroys enemy spaceships and it is damaged by fires of the enemy spaceships and asteroids that hit it.
4. Fuel of the spaceship is decreasing constantly, it enters to space station and station gives extra ammo and fuel to spaceship.
5. Player finishes all the levels successfully. Greeting message shows up at the end.
6. Score of the player is displayed. If the score is higher than tenth high score, player will be asked to enter a nickname.

7. If so player enters a nickname (If score is not higher enough then the game returns main menu automatically).
8. The score is recorded and high score table is updated accordingly.
9. Player returns to the main menu by clicking related button on the high score menu.

Alternative Flow of Events:

- Player has lost all of his/her lives. Game over message is displayed on the screen.Then high score menu shows up automatically. Player returns to the main menu by clicking related button on the high score menu.
- Player stops the game. Small options panel is displayed on the game screen. Player returns to Main Menu by clicking “Return Main Menu” button on this panel.

Use Case - 2

Name: ChangeOptions

Actors: Player

Entry Condition: The game is started and player stops the game by clicking “Esc” key from keyboard.

Exit Condition:

- Player returns to the Main Menu by clicking “Return to Main Menu” button.
- Player returns to the game by clicking “Return to Game” button.

Main Flow of Events:

1. Small options panel is displayed on the center of the game screen. On this panel there are three buttons: “New Game”, “Return to Main Menu” and “Return to Game” and also two sliding bars that are related to sound and music voices.

2. Player navigates through provided menus.
3. Player changes music and sound voices by sliding these bars.
4. New settings will be saved by the system.
5. Player returns to the main menu by clicking “Return to Game” button.

Alternative Flow of Events:

- Player does not change the settings and returns to game by clicking “Return to Game” button.
- Player clicks the “New Game” button and level 1 is constructed by the system.
- Player returns Main Menu by clicking “Return to Main Menu” button.

Use Case - 3

Name: ViewHighScores

Actors: Player

Entry Condition: The game is opened and player is on the main menu to choose an operation.

Exit Condition: Player returns to the Main Menu by clicking “Return to Main Menu” button.

Main Flow of Events:

1. Player selects the “View High Scores” option.
2. System displays top 10 high scores with their user names.

Alternative Flow of Events:

- Player wants to return back to the Main Menu and presses “Return to Main Menu” button.

Use Case - 4

Name: LoadLevel

Actors: Player

Entry Condition:

- The game is opened and player is on the main menu to select an operation.
- Player is already playing a game.
- Player already passed at least one level.
- The player has the appropriate password to start at the desired level.

Exit Condition:

- Player enters the password correctly and load level.

Main Flow of Events:

1. Player selects the “Enter a Password” option to continue to play.
2. System asks for a password.
3. Player enters the password.
4. Player continues to play the game from the point where s/he passed.

Alternative Flow of Events:

- Player cannot find a level to load since there is none and returns to the main menu.
- Player cannot enter the correct password, an error message is displayed.

2.7. User Interfaces

In this part, panels and visual of objects will be given.

2.7.1. Main Menu

When user runs the game menu screen will be shown first. Menu screen contains New Game, Password, High Scores and Credits options. By clicking one of these options, user can move desired panel.

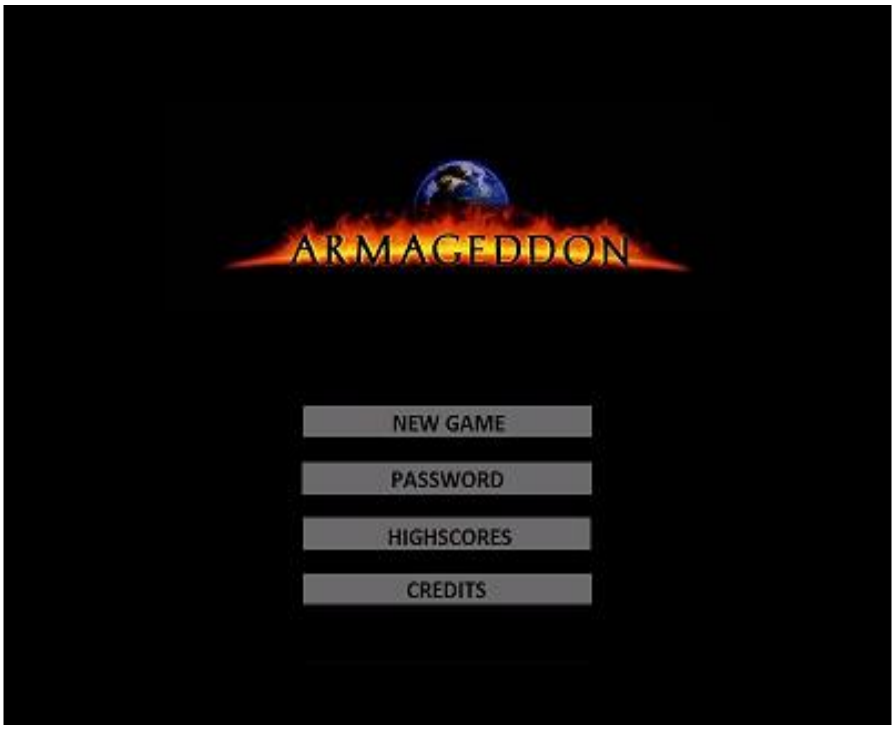
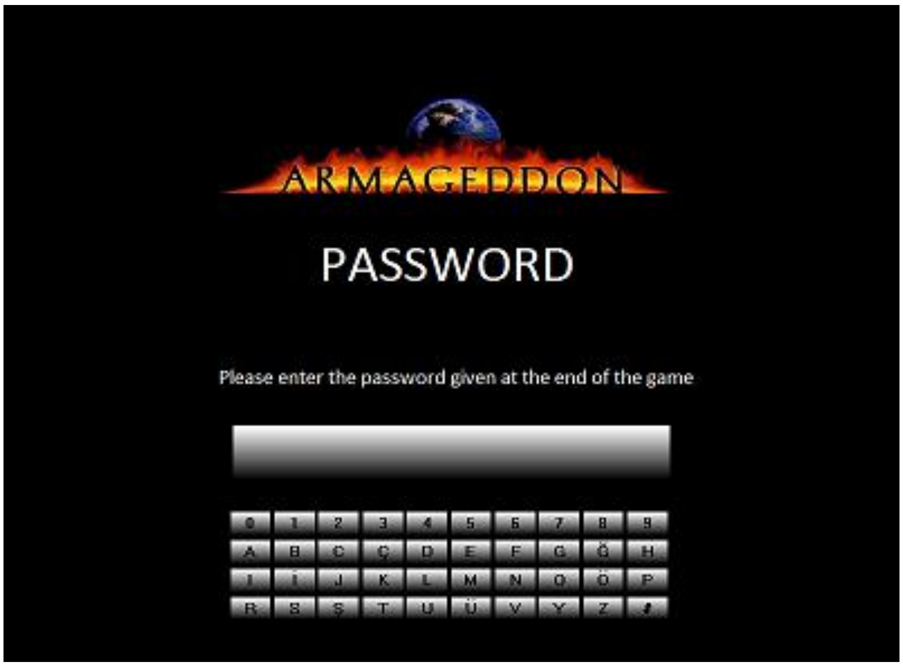


Figure 2: Main Menu

2.7.2. Password

If user selects “PASSWORD” option, this screen will be shown. In this screen, there are an input box, which takes password from user, and a keyboard. By the help of this keyboard, user can enter the password given at the end of the game. After entering the password, if the entered password is correct, according to entered password user can start game from specific level.

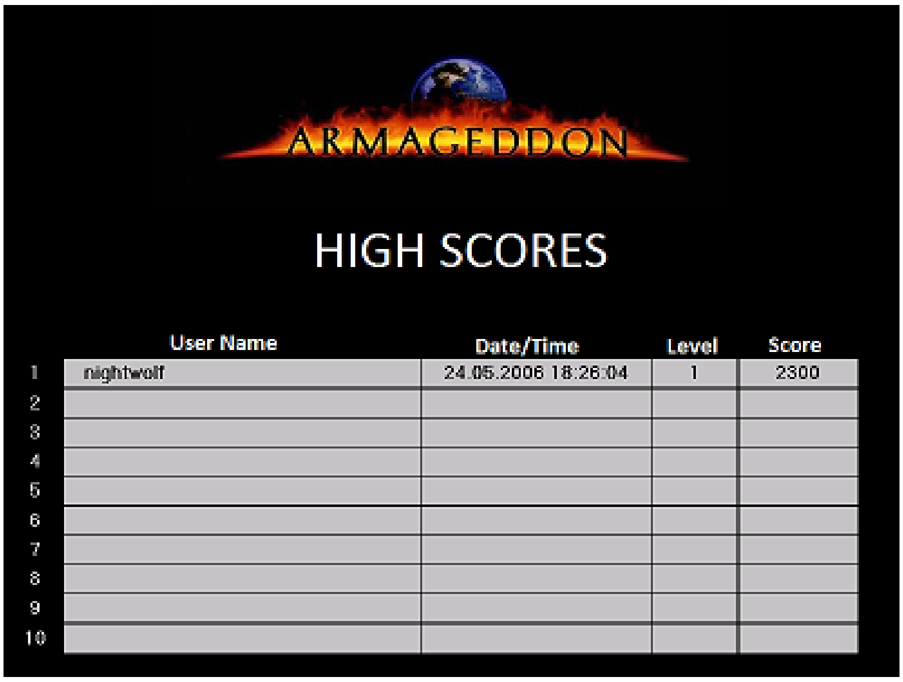


Please enter the password given at the end of the game

Figure 3: Password Menu

2.7.3. High Scores

If user selects “HIGH SCORES” option, this screen will be shown. In this screen, user can see table of the best ten score ever reached before. These scores are on display with the “User Name”-name of user-, “Date/Time” -date of this score was achieved- , “Level” -last reached level- and “Score”-total points user gained while playing-.



	User Name	Date/Time	Level	Score
1	nightwolf	24.05.2006 18:26:04	1	2300
2				
3				
4				
5				
6				
7				
8				
9				
10				

Figure 4: High Scores Table

2.7.4. Credits

When user selects “CREDITS” option, this screen will be shown. In this screen, there will be a textbox which contains some brief information about the game and development team.

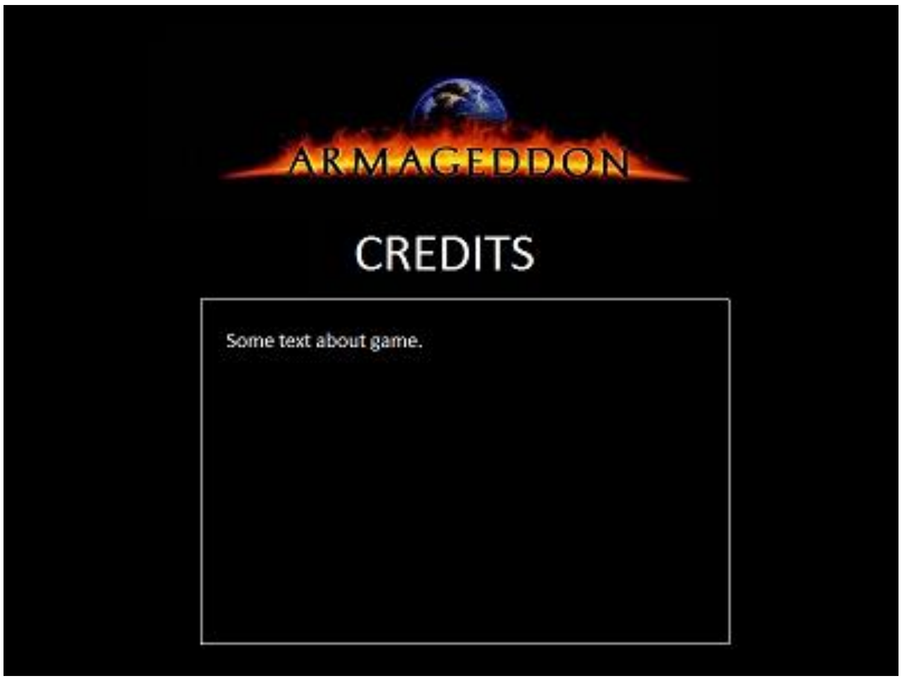


Figure 5: Credits

2.7.5. Game

When user selects “NEW GAME” option, this screen will be screened. This screen is where user plays the game so it contains various objects which will be shown one by one below. Left top of the page, level of game and total score that user collects will be shown while playing.

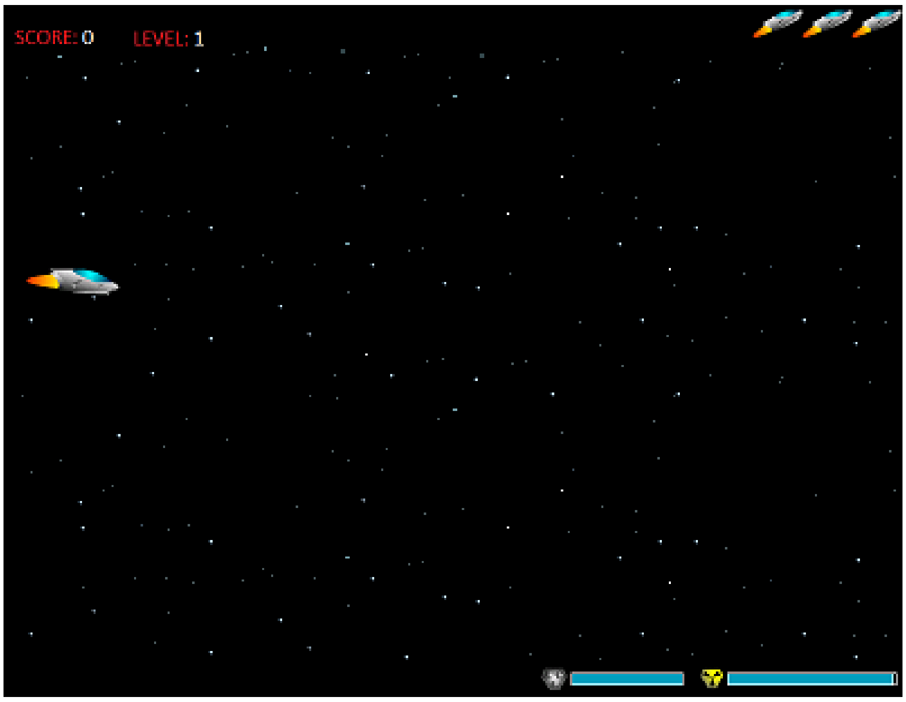
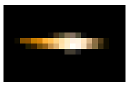


Figure 6: Gameplay View

2.7.5.1. Ammunition Types

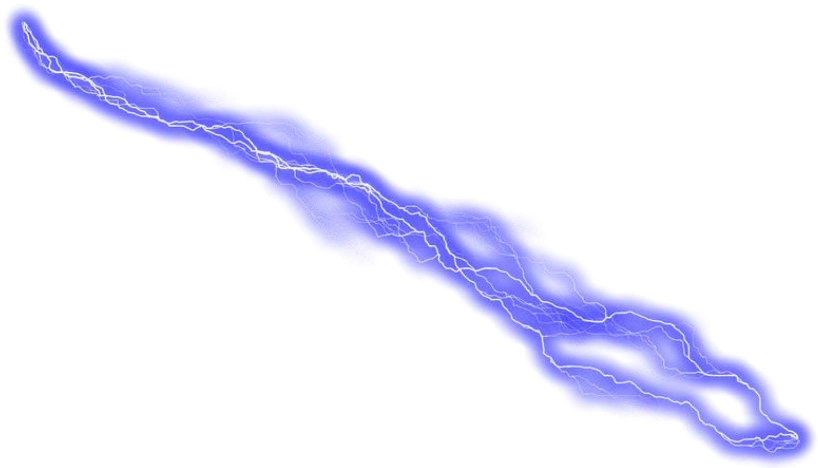
- Default: Default type of ammunition of the player's spaceship. It is weakest ammunition of the game. Its power of fire is 1 unit.



- Laser: This weapon will be given at second level, in the space station. Its power of fire is 2 units, so it is two times more powerful than default ammunition. When it is given it replaces with default ammunition. Therefore it is not bonus weapon.

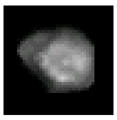
- Rocket: It is given at third level, in the space station. Station gives one hundred rockets as bonus ammunition in this level. Its power of fire is 3 units.

- Electric: It is given at fourth level, in the space station. It is not numerable weapon. When the player takes this weapon there will be charging bar on the underside of the screen. When this bar is fully charged player can use this weapon and when it is used lightning flashes and it destroys all of enemies on the screen at that time.

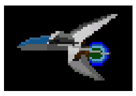


2.7.5.2. Enemy Types

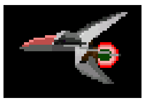
- Asteroid: Space objects which are coming towards player's spaceship. When player hits one of them it decreases vast amount of shield strength.



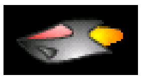
- GoblinX: It is most harmless enemy spaceship in the game. It flies through right to left without shooting. But when it hits shield will be damaged tragically.



- GoblinY: Enemy spaceship that comes from right side of the screen but never goes away until player destroys it. These spaceships come in certain number for certain levels which is at most twenty. When players destroy all of them level goes up.



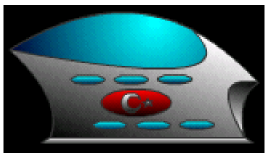
- GoblinZ: Just like GoblinY spaceship, but it goes from right to left on up and down sides of the screen. It shoots crosswise but appearance is same with GoblinY.
- BomberX: This is a powerful enemy spaceship which launches rocket. This weapon is same with our ammunition which is called rocket. Just like GoblinX these spaceships goes through right to left on the screen but unlike GoblinX, they shoot on their routes.



- BomberY: Just like GoblinY these are coming from right side of the screen and then moving towards y-axis until they die. They are coming in limited numbers for each level. Appearance of them is same as BomberX.
- BomberZ: Its way and shooting route is same as GoblinZ and its weapon is same with BomberX and BomberY. Appearance is slightly different than other Bomber's.

2.7.5.3. Special Types

- Station: Station is where user can load fuel, repair space ship's shield and get some sophisticated bullets.



2.7.6. Settings Menu

User can reach this screen from game menu by pressing "ESC" button. With the help of this screen user can modify music settings and sound settings. Also new game can be started. After the completing desired settings, user can back to game by clicking "BACK TO GAME" or back to main menu by clicking "EXIT".

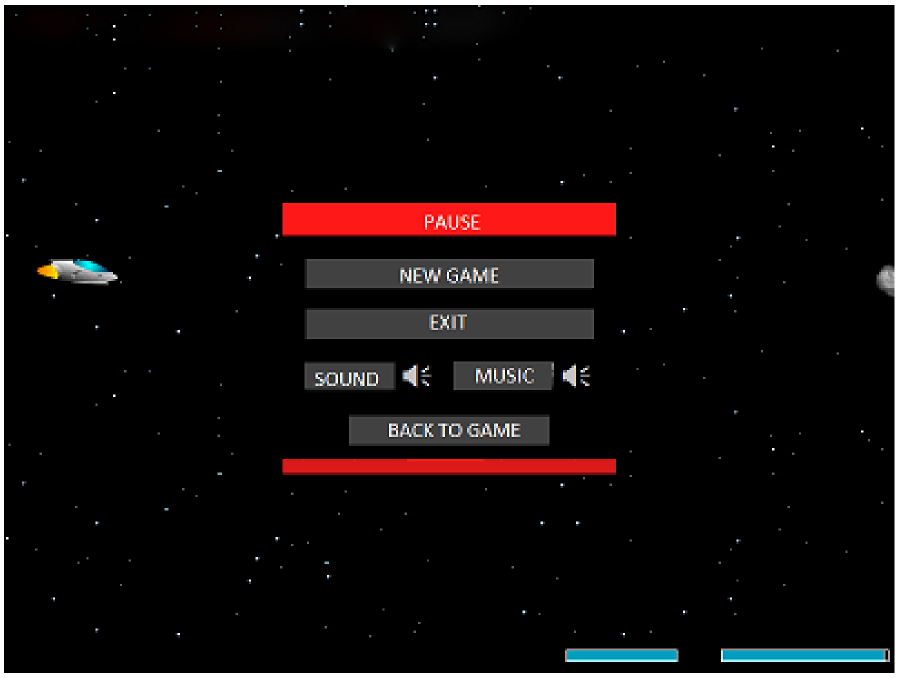


Figure 7: Settings Panel

3. Analysis

3.1. Object Model

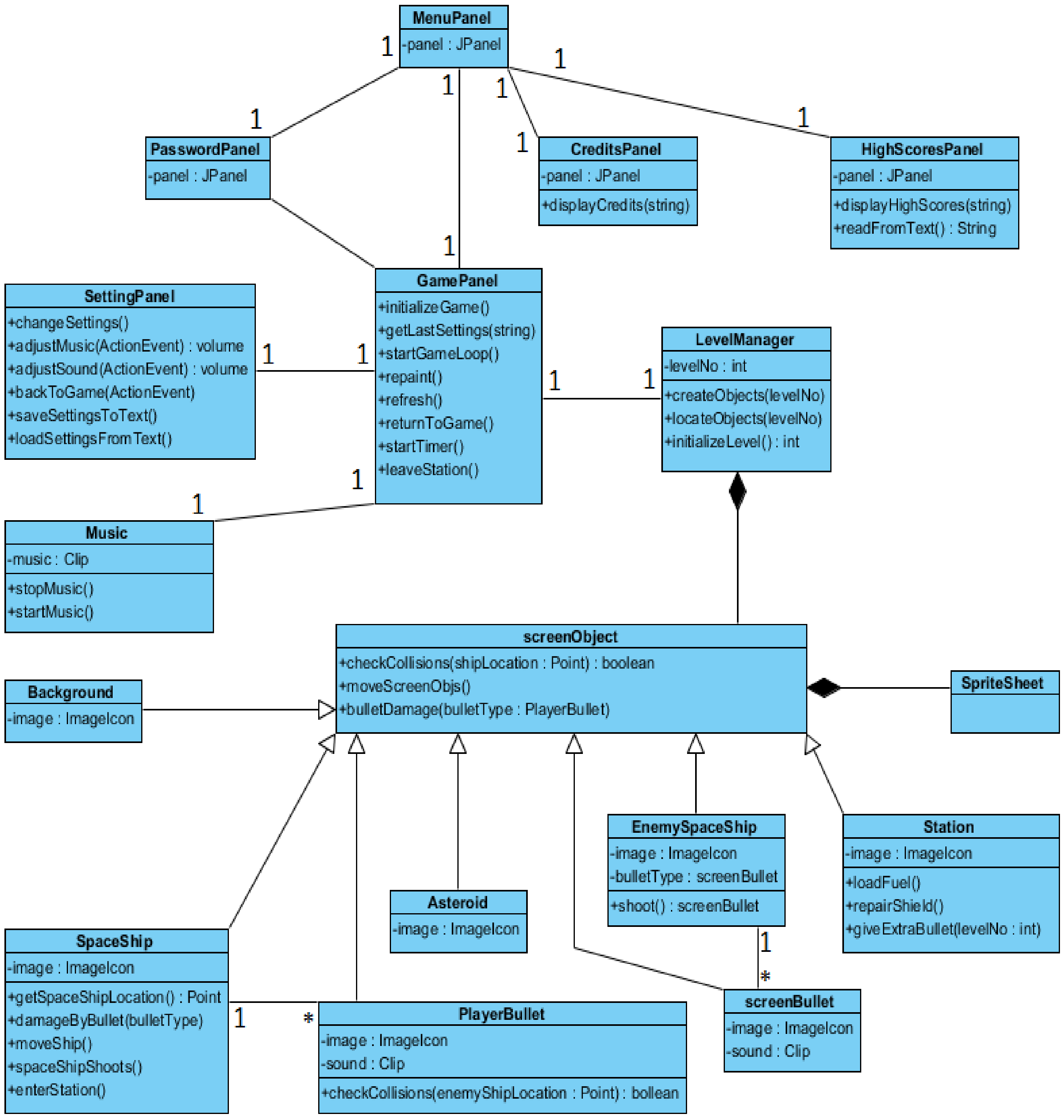


Figure 8: Class Diagram for the System

The object model of ARMAGEDDON game is illustrated above. This diagram contains 17 classes.

- **GamePanel:** This is the most significant class. This class is where game is organized and interacted with user at most. When user plays the game images are shown by this class panel. GamePanel controls inputs and object relations.
- **LevelManager:** LevelManager class manages game difficulty according to level number. It decides type of enemies, their numbers and holds them.
- **screenObject:** screenObject is the most important class for visualization because this class is where collisions are checked according to location of objects, and also where objects can move. Below this class there are seven entity classes. These classes have different image icons, speed, direction and some operations.
    - <u>SpaceShip:</u> SpaceShip class hold image of spaceship that user controls. With the help of this class user can move and shoot to destroy enemies.
    - <u>PlayerBullet:</u> When the user shoots, this class is called and move until a collision occurs.
    - <u>Asteroid:</u> Asteroid is a type of obstacle.
    - <u>screenBullet:</u> screenBullet is bullets that enemies fire.
    - <u>EnemySpaceShip:</u> This class consist enemy spaceships with different ship types, bullet types and line of action.
    - <u>Station:</u> Station is where user's spaceship enters and gets bonuses.
    - <u>Background:</u> Background is movable object for getting movable background view.
- **Music:** Music class contains game music and sounds of some objects.

Other panels consist of extended JPanel. They interact with user and do some operations that are explained interface part.

3.2 Dynamic Models

3.2.1 State Chart

State Chart Diagram for the Ship:

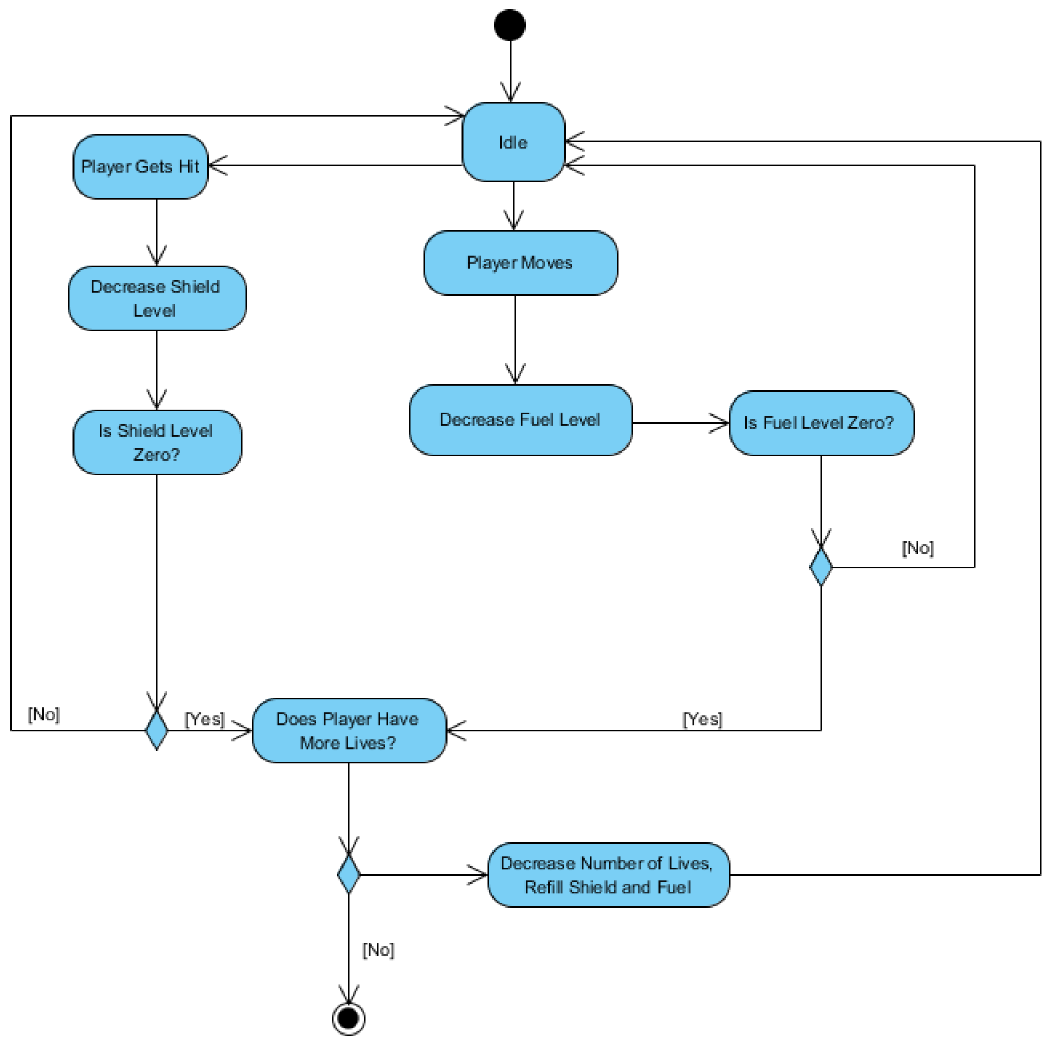


Figure 9: State Chart Diagram for the Ship

This state chart diagram explains the behavior of the ship the player uses. If the user uses direction keys to move the ship the fuel level will decrease. The ship's fuel level will also decrease automatically since the ship moves forward continuously. When the ship collides with one of the enemy ships its shield will decrease. If either of shield or fuel is fully depleted, one life will be removed from the player and if the player has no more lives, the ship is destroyed and the game ends.

3.2.2 Activity Diagram

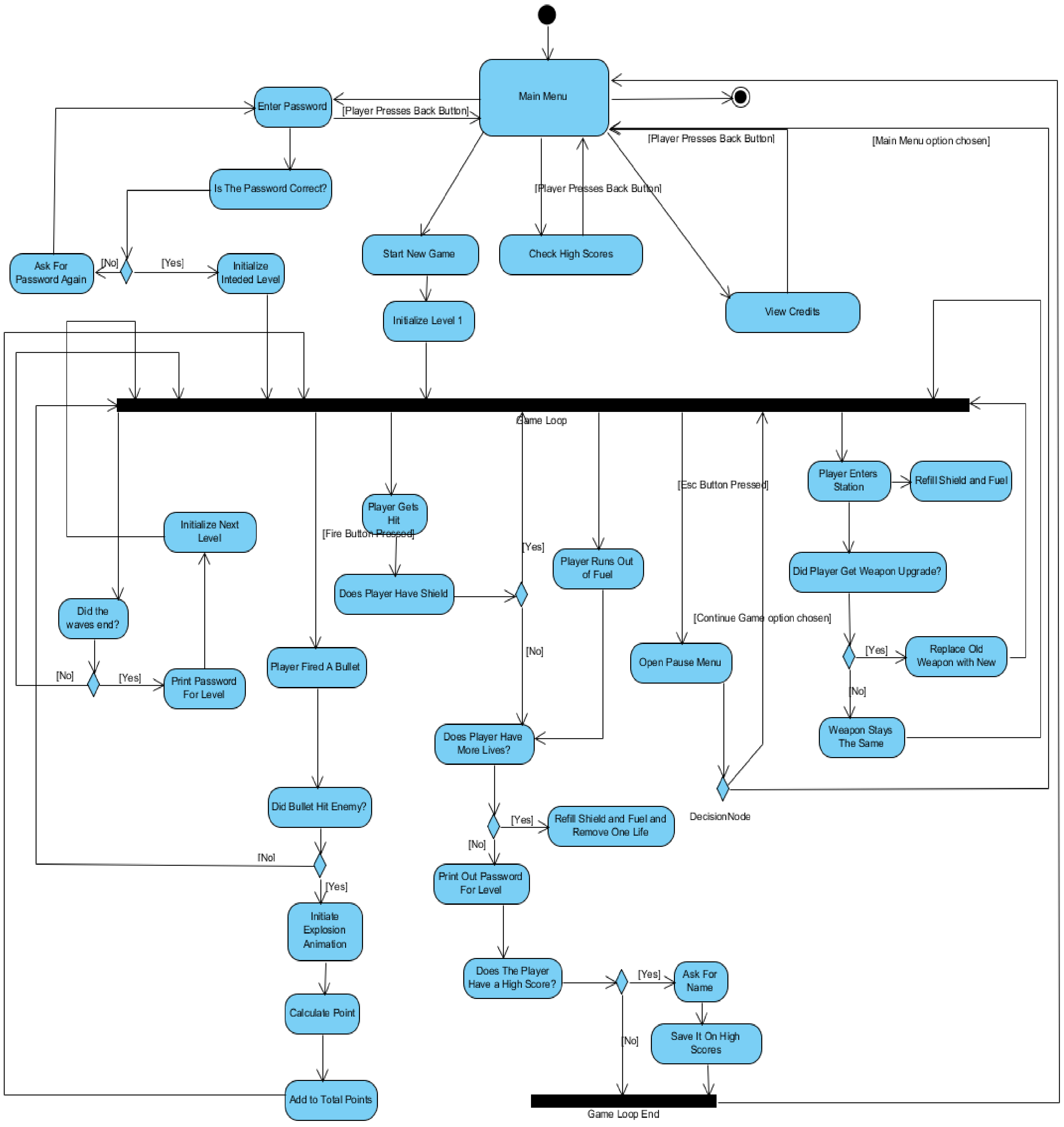


Figure 10: Activity Diagram for the Game

This state chart shows the main flow of the game. The game is initialized from the main menu and the user has options to enter a password, start a new game, view high scores, view credits and exit game. The user can go back to the main menu from all of these options except start a new game option and exit game option. The system checks if the user has entered the correct password. If it is correct, the system initializes the desired level and starts the game loop, else asks for the password again. If the user selects the new game option the system initializes the first level and starts the game loop. If the user has fired a bullet, the system checks for collisions. If a bullet hits an enemy, the system initiates an explosion animation and adds the appropriate point to the score. If the player gets hit, the system checks the current shield level. If the current shield is zero, the system checks the number of lives. If the number of remaining lives is zero the system prints out a password for the level and asks for a name if the player has done a high score and goes back to the main menu. If the player has lives left, the ship's fuel and shield level is refilled. If the player presses the escape the pause menu shows up and the player can change the settings of the game, continue the game or return to the main menu.Else the game continues normally. When a player enters a station the fuel and shield level is refilled and the ship might get a weapon upgrade if it does get an upgrade, the new weapon takes the place of the old one. If a level ends (i.e., the waves have ended) the system checks if the ship has survived, if it does the system initializes the next level. If the user decides to exit the game from the main menu, the game closes.

3.2.3 Sequence Diagram

Scenario #1: Starting Game

Player Saner requests to start game by clicking "New Game" button from Main Menu. After that system initializes GamePanel into the frame and GamePanel sends message to LevelManager to create objects. LevelManager initializes objects andgets images of objects; background, spaceship that player uses, enemy spaceships, space station and asteroids from the file directory of the game.Then GamePanellocates all objects to appropriate locations indicated in the LevelManager according to LevelNo. While loading the level, GamePanel reads last settings from the text file. Lastly, GamePanel starts the game loop and repaints all the objects related to Level and updates the game panel.

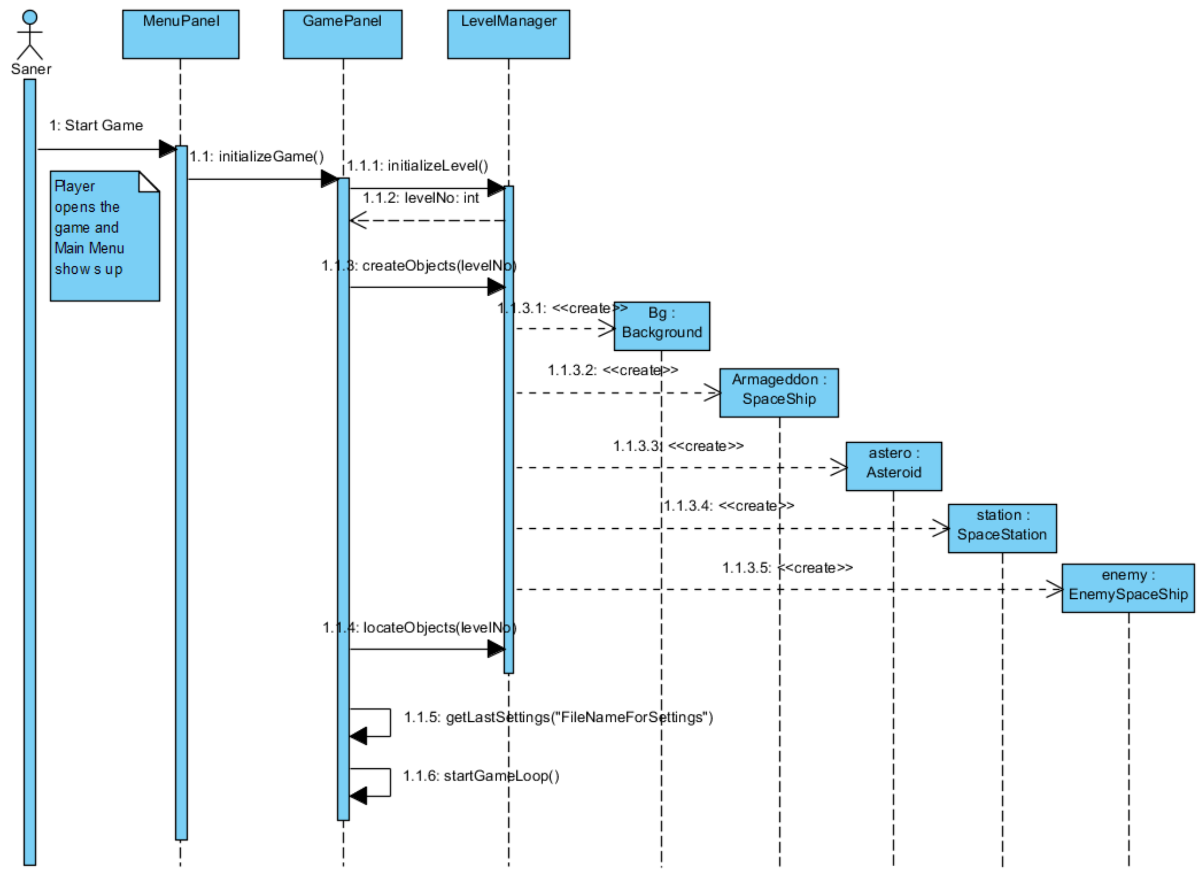


Figure 11: Sequence Diagram for Starting Game Scenario

Scenario #2: Shooting War

Player Ozge requests to start the game by clicking “New Game” button from Main Menu. Assuming that steps of the previous sequence diagram are performed, in the Game Panel whole game dynamics are arranged; current location of the spaceship that Ozge uses is taken, space objects and bullets of the enemy spaceships that are on the screen are manipulating, spaceship that Ozge uses is moving and shooting. GamePanel also checks collisions; assuming that player Ozge’s spaceship has been collided with the bullet of the enemy spaceship and is damaged, Ozge directs the spaceship using direction keys from the keyboard in order to avoid the enemy bullets. After that Ozge shoots and her bullet is collided with the enemy then the enemy spaceship is destroyed.

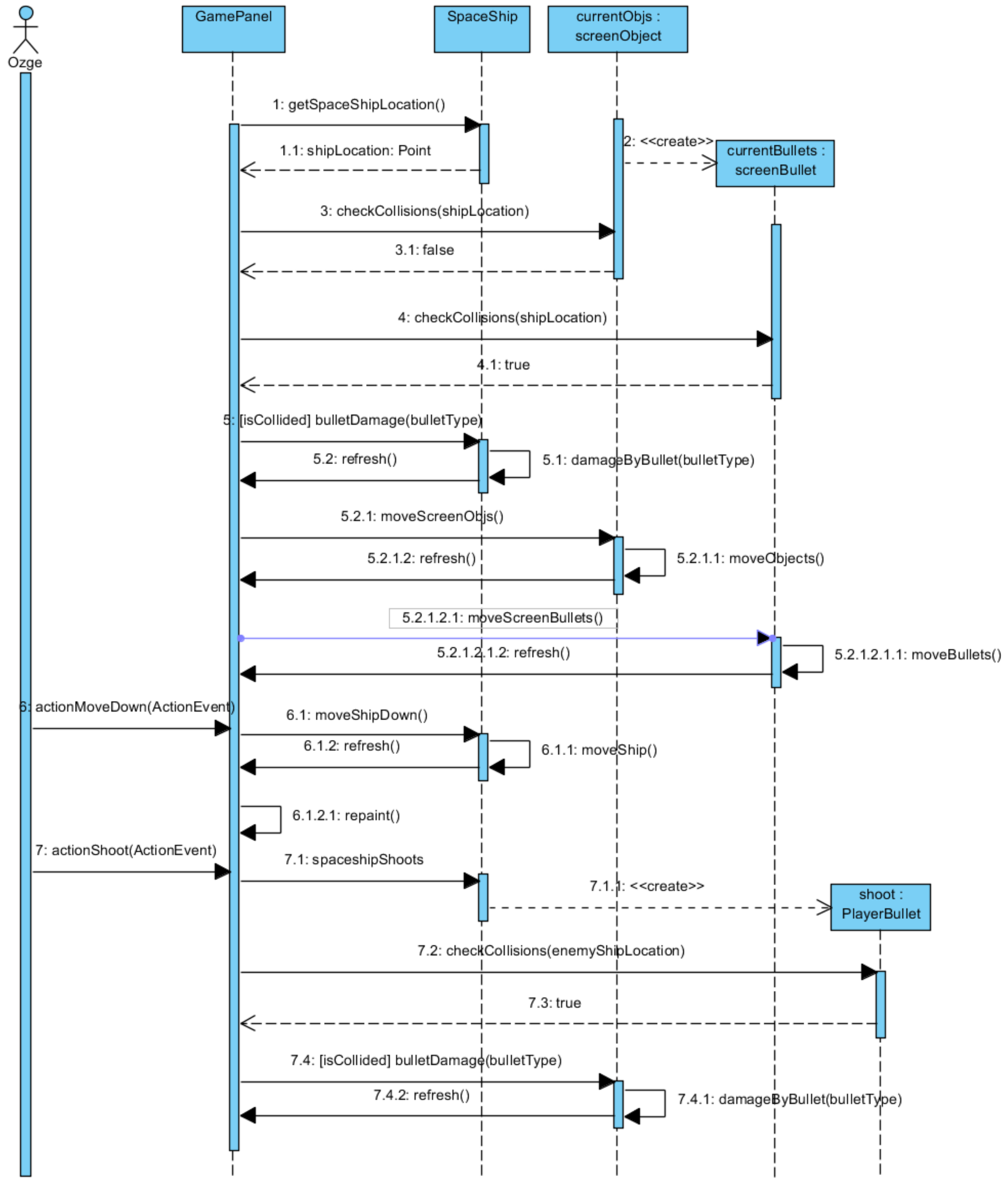

Ozge
GamePanel
SpaceShip
currentObjs : screenObject
currentBullets : screenBullet
1: getSpaceShipLocation()
1.1: shipLocation: Point
2: <<create>>
3: checkCollisions(shipLocation)
3.1: false
4: checkCollisions(shipLocation)
4.1: true
5: [isCollided] bulletDamage(bulletType)
5.1: damageByBullet(bulletType)
5.2: refresh()
5.2.1: moveScreenObjs()
5.2.1.1: moveObjects()
5.2.1.2: refresh()
5.2.1.2.1: moveScreenBullets()
5.2.1.2.1.1: moveBullets()
5.2.1.2.1.2: refresh()
6: actionMoveDown(ActionEvent)
6.1: moveShipDown()
6.1.1: moveShip()
6.1.2: refresh()
6.1.2.1: repaint()
7: actionShoot(ActionEvent)
7.1: spaceshipShoots
7.1.1: <<create>>
shoot : PlayerBullet
7.2: checkCollisions(enemyShipLocation)
7.3: true
7.4: [isCollided] bulletDamage(bulletType)
7.4.1: damageByBullet(bulletType)
7.4.2: refresh()

Figure 12: Sequence Diagram for Shooting War Scenario

Scenario #3: Changing Settings

Player Ozge request to start a game by pressing the “New Game” from Main Menu. Assuming that the steps of the Start Game sequence diagram are performed, Ozge wants to change the game setting as she desires. In order to do that Ozge pauses the game by pressing Esc key from the keyboard. Small SettingPanel shows up on the game screen and game is stopped. Ozge changes sound and music volumes by sliding bars on this panel. New volume settings are arranged by the SettingPanel. After that Ozge clicks to “Return to Game” button and game continues.

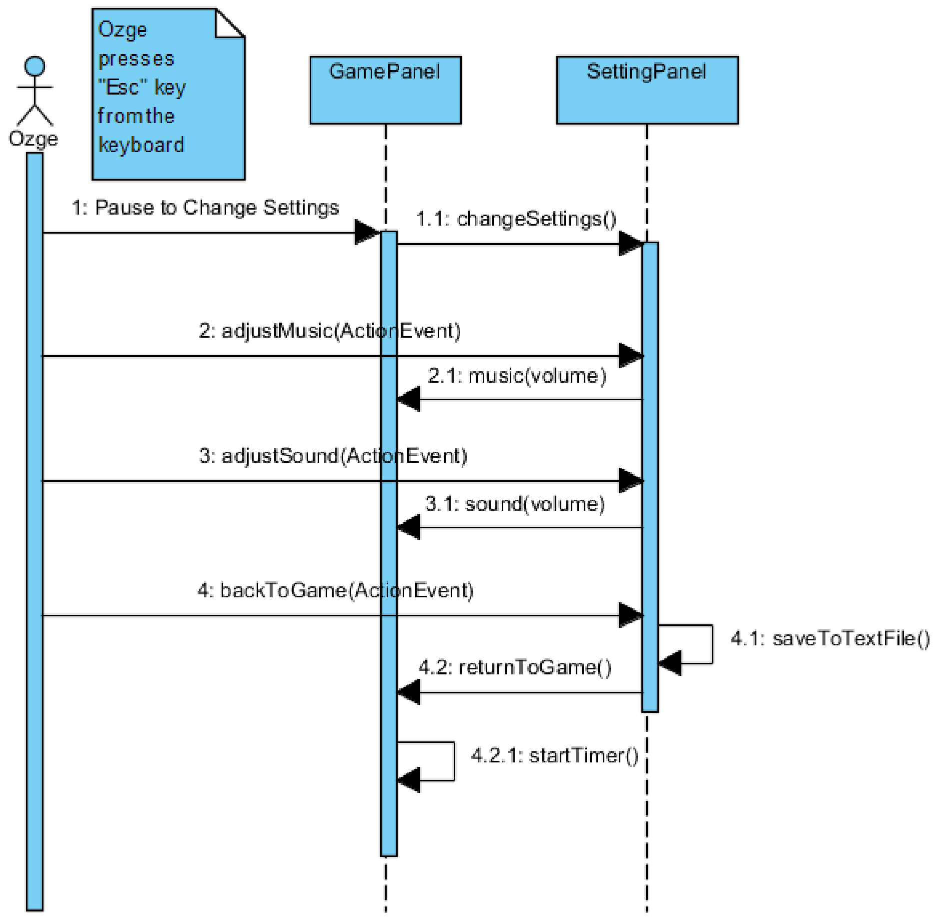

Ozge presses "Esc" key fromthe keyboard
Ozge
GamePanel
SettingPanel
1: Pause to Change Settings
1.1: changeSettings()
2: adjustMusic(ActionEvent)
2.1: music(volume)
3: adjustSound(ActionEvent)
3.1: sound(volume)
4: backToGame(ActionEvent)
4.1: saveToTextFile()
4.2: returnToGame()
4.2.1: startTimer()

Figure 13: Sequence Diagram for Changing Settings Scenario

Scenario #4: Entering Station

Player Saner request to start a game by pressing the “New Game” from Main Menu. Assuming that the steps of the Start Game sequence diagram are performed, Saner's spaceship runs out of fuel and his shield is damaged during the game. Saner presses down button from the keyboard and spaceship enters the station in order to repair his shield and load fuel. Game is paused when spaceship is in the space station. Then it gets an extra bullet from space station which is specific to the current level. After a certain period spaceship leaves the station and game continues.

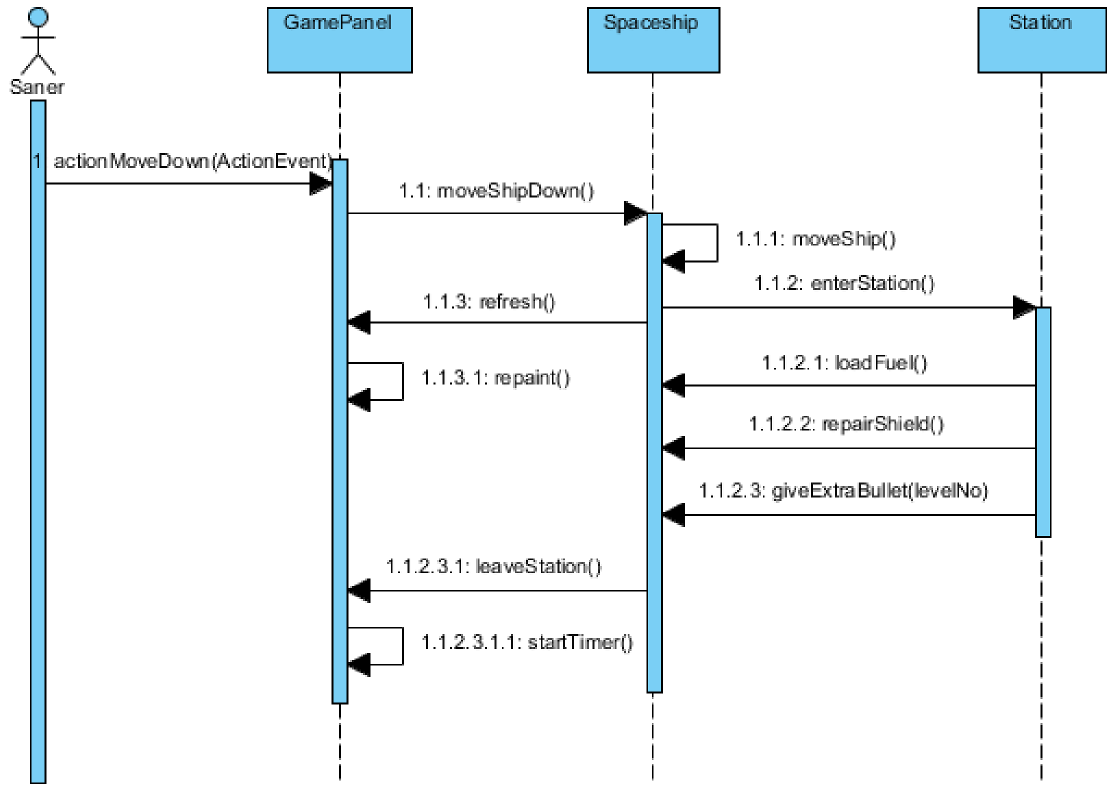


Figure 14: Sequence Diagram for Entering Station Scenario

4.Conclusion

In this report, we created our analysis report with the purpose of designing and implementing our space game Armageddon. This report has two main parts. First one is requirement specification and the second part is the system model.

During the requirements specification part, our goal was to determine and examine all possible requirements a player could perform while playing a space game. With this ideology we decided on our functional and non-functional requirements. We know how important is the requirement specification part, therefore we spend enough time on acknowledge all possible requirements. Our aim is to fulfill them during our project. After the requirement specification part we started to work on our system model.

The system model of our project model includes four parts which are Use Case Model, Dynamic Models, Class Model and User Interface.

While deciding use cases we tried to acknowledge which uses we need for our game and which actions we can indicate as our use cases.

Our Dynamic model consists of sequence diagrams, state chart diagram and activity diagram. Sequence diagrams enabled us to show the interaction between the player and the system. While showing these interactions we focused on very crucial parts of the game. In our state chart diagram, we aimed to indicate all possible states of ships' fuel and shield. Our activity diagram shows our flow of the game. Knowing that our class diagram will have an important role for our design and implementation process, we worked hard for forming the most efficient class diagram and tried to indicate all possible classes and their relationships with each other.

Last part of of our system model was the User Interface and navigational path diagram. We aimed to make our interface user friendly as much as possible therefore we tried to make our mock-ups simple and nice. Then using our interfaces we formed the navigational path.

To conclude, we aimed to form the most efficient analysis report and took it very seriously since we know the fact that a good analysis report will be a good guide for us in the future processes(design and implementation). Therefore we took it very seriously.

5.References

- Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.