Bilkent University

Department of Computer Engineering

CS319
Object Oriented Software Engineering

Analysis Report

Crazy Space War

Group Members

Ömer Hancı	21001218
Muhammed Emin Öztürk	20901349
Erdinç Albayrak	21202312
Balanur İçen	21101621

Repository Name: crazy-space-war

Assistant Professor Can Alkan

October 30, 2014

This report is submitted to the Department of Computer Engineering of Bilkent University in partial

Table of Contents

1.Introduction

We have decided to design simplified version of the space war game which we called crazy space war. The aim of the game is destroying aliens by controlling a spaceship and to be survived until end of the game. We can also obtain some bonus features by colliding some special flags, which call power up. The way to win this game is destroying several types of aliens. The game will be a desktop application and will be controlled via keyboard. This report includes an overview of the game and explains functional & non-functional requirement and system model.

2.Overview

Crazy Space War is a kind of arcade game, which is uncomplicated to play because just controlled via keyboard. Yet it is very addictive. The player controls a spaceship along screen from down to up and right to left. The game includes several levels that we have to complete to win the game. In order to complete a level, user has to destroy all of the aliens in a level. Some levels contain certain number of aliens however other levels have time limit that you have to survive up to level is completed. The goal of the game is to move up the levels without losing all energy.

Lets describe ways in order to destroy enemies' spaceships in game. As first step the User collides spaceship with enemies so as to explode enemy's spaceships. In this case our spaceship also loses its life energy. Other way is to fire them with our weapons. As noted above the game contains several kind of power up, which appears randomly on the screen. Some of power up can increase our weapon destruction power. Power-ups make the game more enjoyable than you expected.

The game has sound and graphic effects to emphasize certain events like dying or passing the level. We also will add several in-game music.

2.1.Start

The user can start new game as a single player. However, he/she can change to multiplayer choice from setting menu. The game provides user to

change game setting, view help and obviously the choice of quit game any time she/he wants.

2.2.Gameplay

The player uses keyboard to play the game. The arrow buttons and space button are needed during the game. Arrows buttons are used to direct moving on the screen. Space button is used for firing to kill enemies target. In the case of multiplayer choice, instead of arrow keys the letter button “W”, “A”, “S”, ”D” on the keyboard are used by second player. Second player uses tab button for firing

2.3.Levels

The game has 10 different levels and 3 different difficulty degrees i.e. easy, medium and hard. The difficulty can be set at change setting submenu before starting playing game. The levels get harder from first to last. End of the some 2-3 levels special aliens space ship is appeared. The user must defeat this type of enemies so as to complete level successfully and continue the other levels. At each level there is different background that represent emotion of the game.

2.4.Aliens & Spaceships

Depends on the level, different number of aliens appears and at end of the level, boss alien comes. The boss alien is five times larger than other types of alien and have more health energy so it is hard to defeat boss aliens than the others. To defeat boss aliens, power-ups play crucial roles.

2.5.Power- Ups

The power-ups are the essential part of the game that makes it so fun. The player will never know when power-up come. Also the player cannot know exactly what type of power-up coming. So it makes game more enjoyable. As it

mentioned above, the level of degree triggers the difficulty of game. Especially at higher level, power-ups are indispensable requirement to complete level with high score.

In the game, there are seven types of power-ups.

2.5.1.Power-Ups "Armor"

After taking this special type of power-ups our spaceship is not be affected by enemies bomb. During 45 seconds spaceship have armor that protect player from enemies bomb so you can ignore the enemies fires.

2.5.2.Power-Ups -Attractive "Gun Power"

This makes our spaceships gun more destructive. After taking this special power ups during 60 seconds player can easily destroy aliens.

2.5.3.Power-Ups- Freeze Aliens

The type of power ups "Freeze Aliens" freezes all aliens. It makes them disable to fire and move. This feature will be active during 15 seconds. This power up provides important chance to destroy all aliens at short time.

2.5.4.Power Ups-Double Gun

This power-up presents extra gun to users spaceship. Then our spaceship can fire along two different directions so that destroy two aliens at the same time. This special features will be active until the end of level.

2.5.5. Power Ups- Mirror of spaceship

After gaining this power ups user's spaceship shadow has appeared at its symmetric coordinates with respect to y-axis. The shadow spaceship imitates users spaceship and move wherever our spaceship goes.

2.5.6.Power Ups-Block Hole

This power ups creates black hole so as to pull aliens into hole. When aliens approach black hole, immediately disappeared because of

black holes gravity. On the screen, block hole is represented by special images. It will be active during 20 seconds after being created.

2.5.7.Power Ups-Speed Up

This type of power up increases spaceship's speed as approximately 2 times. The spaceship move has been more flexible after taking this power up. It can easily escape from enemies bomb, handle time limit which is requirement some level.

3.Functional requirements

-The user must be able to start a new game.
-The game must provide multiplayer support for two people.
-The user must be able to pause during gameplay indefinitely.
-The user must be able to change sound level and music volume during a pause.
-The game must score the player for every completed game.
-The user must be able to view a list of high scores.
-The game must provide a tutorial containing information about power-ups, aliens and controls.

4.Non-functional requirements

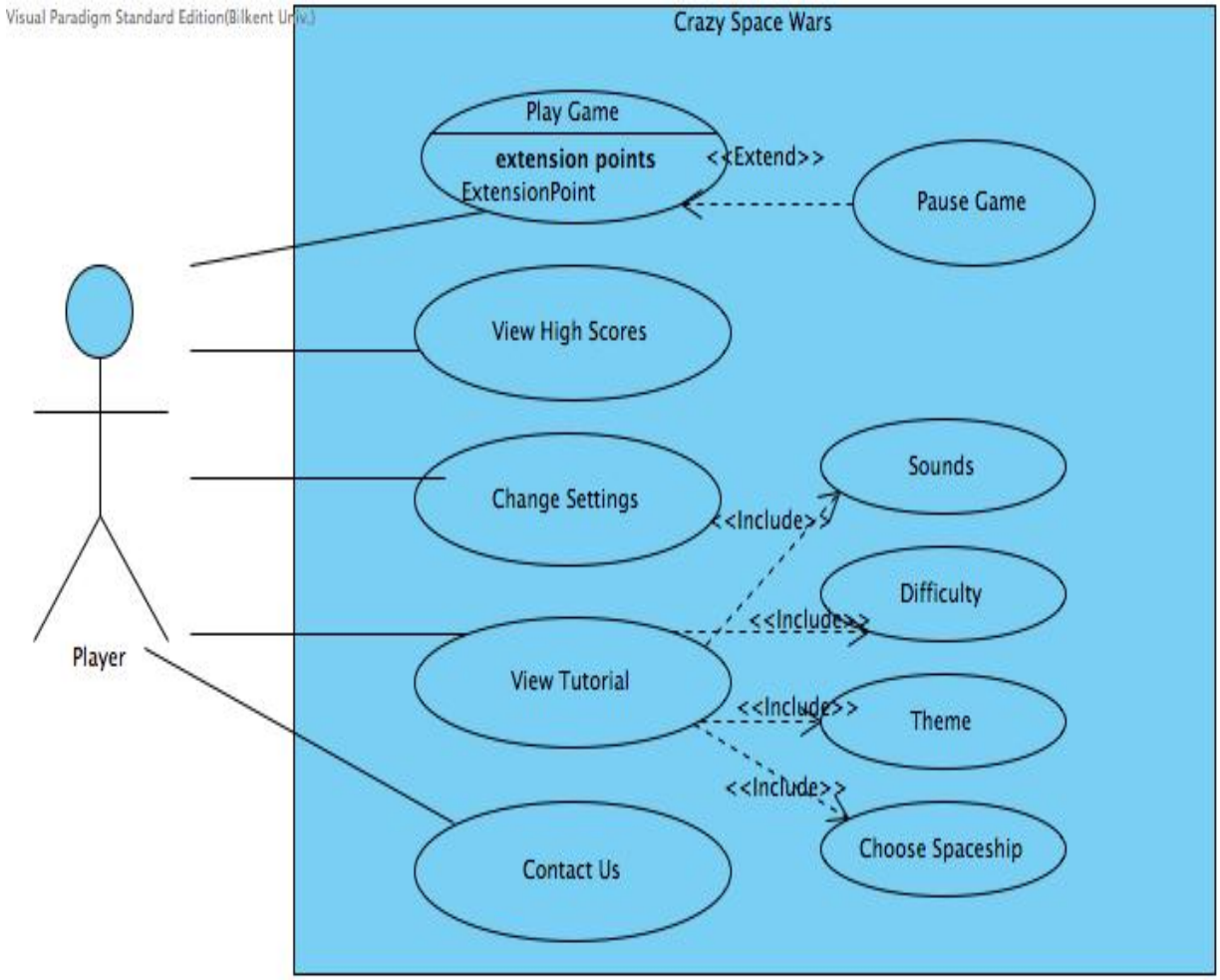
-The game must not require any documentation to play.
-The game must lose the current score in the event of a failure.
-The game must register user actions and produce output within 0.5 seconds.
-The game must allocate less than 16 MB of space during runtime.
-The game must run on every operating system.
-The game must not require an installation.

5.Constraints

-The implementation language must be Java.

6.System Model

6.1.Use Case Model



6.1.1.Play Game

Use Case Name: Play Game

Primary Actor: Player

Stakeholders and Interests:

-Player aims to complete levels and make highest score by killing enemy spaceships.

-System keeps the score of the Player.

Pre-condition: After player selects the Play Game, s/he chooses single/multi player options.

Post-condition: If player gets a higher score than ones in the high score list, System adds new score in the list.

Entry Condition: Player selects "Play Game" button from Main Menu.

Exit Condition:

- ✓ Player chose to exit the game.
- ✓ Player lost the game (no more lives or time up).
- ✓ Player completed all levels successfully.

Success Scenario Event Flow:

1. Player chose single/multi player option.

2. Player starts the game.

3. Game is started from the first level.

4. Player plays until the level goals are completed. Goals maybe completing some time without getting killed or shoot specific number of enemies according to level.

5. System constructs the next level.

6. Player starts playing next level.

*Player repeats the steps 4 – 6 until all levels are completed or Player loses all lives.

7. System displays the score of the player. If high score is enough to get into high score list, user enters name.

8. System returns to Main Menu.

Alternative Flows:

A. If player desires to pause the game at any time:

A.1. Player selects "Pause" button.

A.2. System shows a list of options during pause time. This pause game screen blocks all view of game to prevent extra thinking time.

B. If player desires to exit the game and return to main menu:

B.1. Player selects "Pause" button.

B.2. System shows a list of options during pause time which includes "Exit and Return Screen".

B.3. Player selects the "Exit and Return Screen".

B.4. System goes to 7th step of success flow.

6.1.2.Pause Game

Use Case Name: Pause Game

Primary Actor: Player

Stakeholders and interests:

-Player wants to pause to game for a while.

Pre-condition: Player must be playing the game.

Post-condition: Player returns to game without losing any information in the game.

Exit Condition: Player selects back from the options given for her/him.

Success Scenario Event Flow:

1. Player selects Pause from Game Screen.

2. Pause window is displayed to player.

3. User selects back to return to the game where s/he left.

6.1.3.View High Scores

Use Case Name: View High Scores

Primary Actor: Player

Stakeholders and Interests:

-Player wants to see highest 5 scores with player names.

-System shows the list containing top five scores.

Pre-conditions: System keeps records of top five scores with player names.

Post-condition: -

Entry Condition: Player selects "View High Scores" from Main Menu.

Exit Condition: Player selects "Back" to return Main Menu.

Success Scenario Event Flow:

1. System displays top 5 scores with player names.

Alternative Flows:

A. If player desires to return main menu at any time:

A.1. Player selects “Return to Main Menu” button.

A.2. System displays Main Menu.

6.1.4.Change Settings

Use Case Name: Change Settings

Primary Actor: Player

Stakeholders and Interests:

-Player can turn sound on/off

-Player can choose the difficulty level.

-Player can choose theme.

-Player can choose spaceship.

Pre-condition: Game settings are set as default. If Player changes game settings, adjusted settings will be saved and used by system.

Post-condition: Settings are updated.

Entry Condition: Player selects “Settings” button from menu.

Exit Condition: Player selects “Back to Menu” to return menu.

Success Scenario Event Flow:

1. Player presses “Settings” button to make changes about game settings.

2. Game settings are displayed to Player in “Settings”.

3. Player adjusts settings according to his desire.

4. System updates game settings successfully.

5. Player returns to the main menu by selecting “OK”.

Alternative Flow of Events:

A. Player does not change any settings (go to step 5).

6.1.5.View Tutorial

Use Case Name: View Tutorial

Primary Actor: Player

Stakeholders and Interests:

- Player opens guide and learn how to play the game from instructions.
- Menu shows a tutorial that explains how to play game, its features, rules and it gives useful tips.

Pre-conditions: Player should be in Main Menu.

Post-condition:

Entry Condition: Player selects “View Tutorial” from Main Menu.

Exit Condition: Player selects “Back” to return previous menu.

Success Scenario Event Flow:

1. Player selects “View Tutorial” from Main Menu.

2. System displays a guide about game.

Alternative Flows:

A. If Player requests to return previous menu at any time:

A.1. Player selects “Back” button from “View Tutorial” screen.

A.2. Player returns previous menu.

6.1.6.Contact Us

Use Case Name: Contact Us

Primary Actor: Player

Stakeholders and Interests:

- Player wants to reach the contact information of developers of the project.
- System displays contact information of the developers.

Pre-condition: Player should be in Main Menu.

Entry Condition: Player selects “Contact Us” from main menu.

Exit Condition: Player selects “Back” to return previous menu.

Success Scenario Event Flow:

1. Player selects “Contact Us” from main menu.

2. System displays contact information of the developers.

Alternative Flows:

A. If player desires to return main Screen at any time:

A.1. Player selects to go back to return main menu.

A.2. System displays main menu.

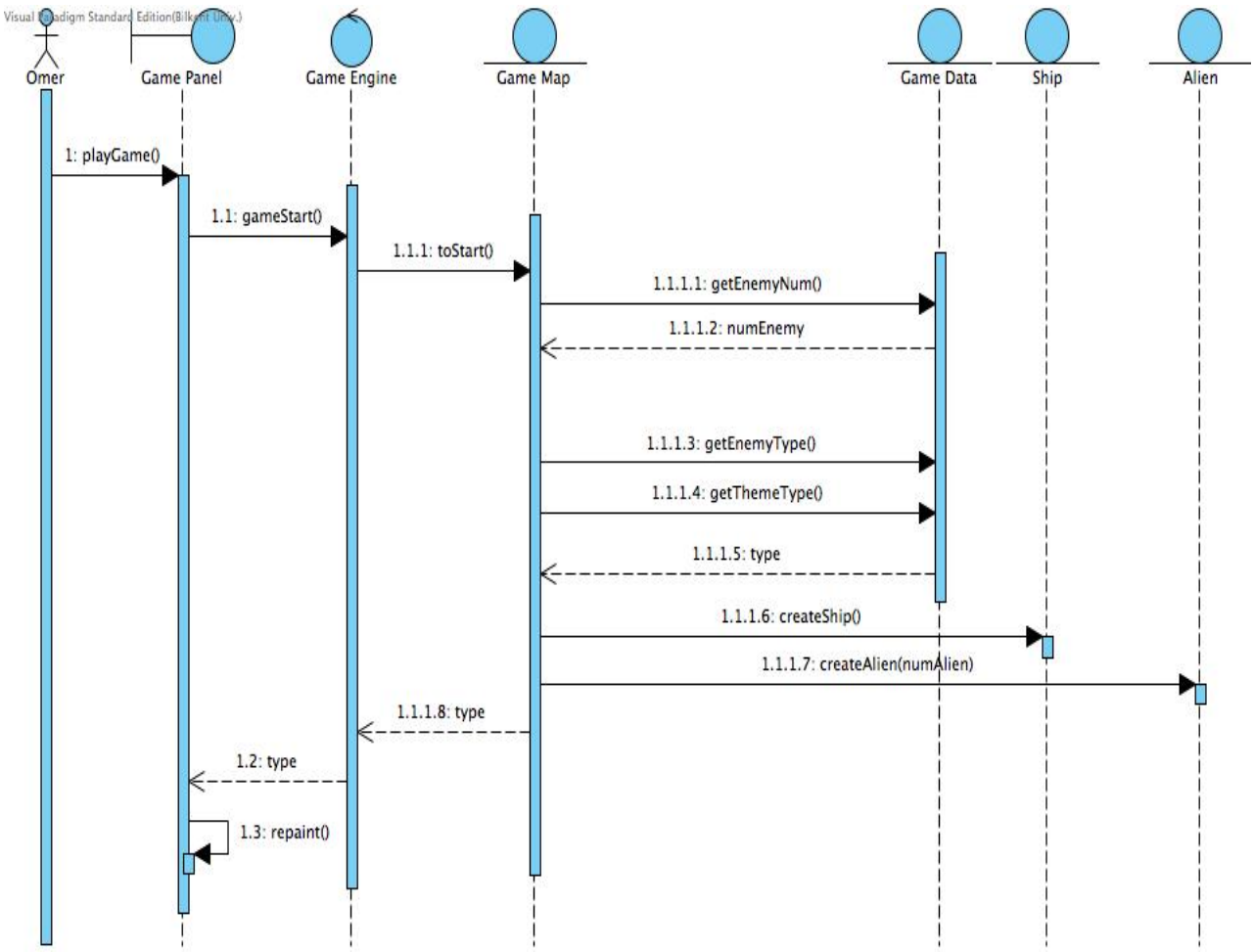
6.2. Dynamic Models

6.2.1 Sequence Diagram

6.2.1.1.Scenario Name : Start New Game

Scenario :

Omer has already downloaded this game and he is excited to try the game. So he selects "Play Game" from main screen. As a default setting choice "single" game starts. Omer wants to play alone so he doesn't change it to multiplayer from setting menu.



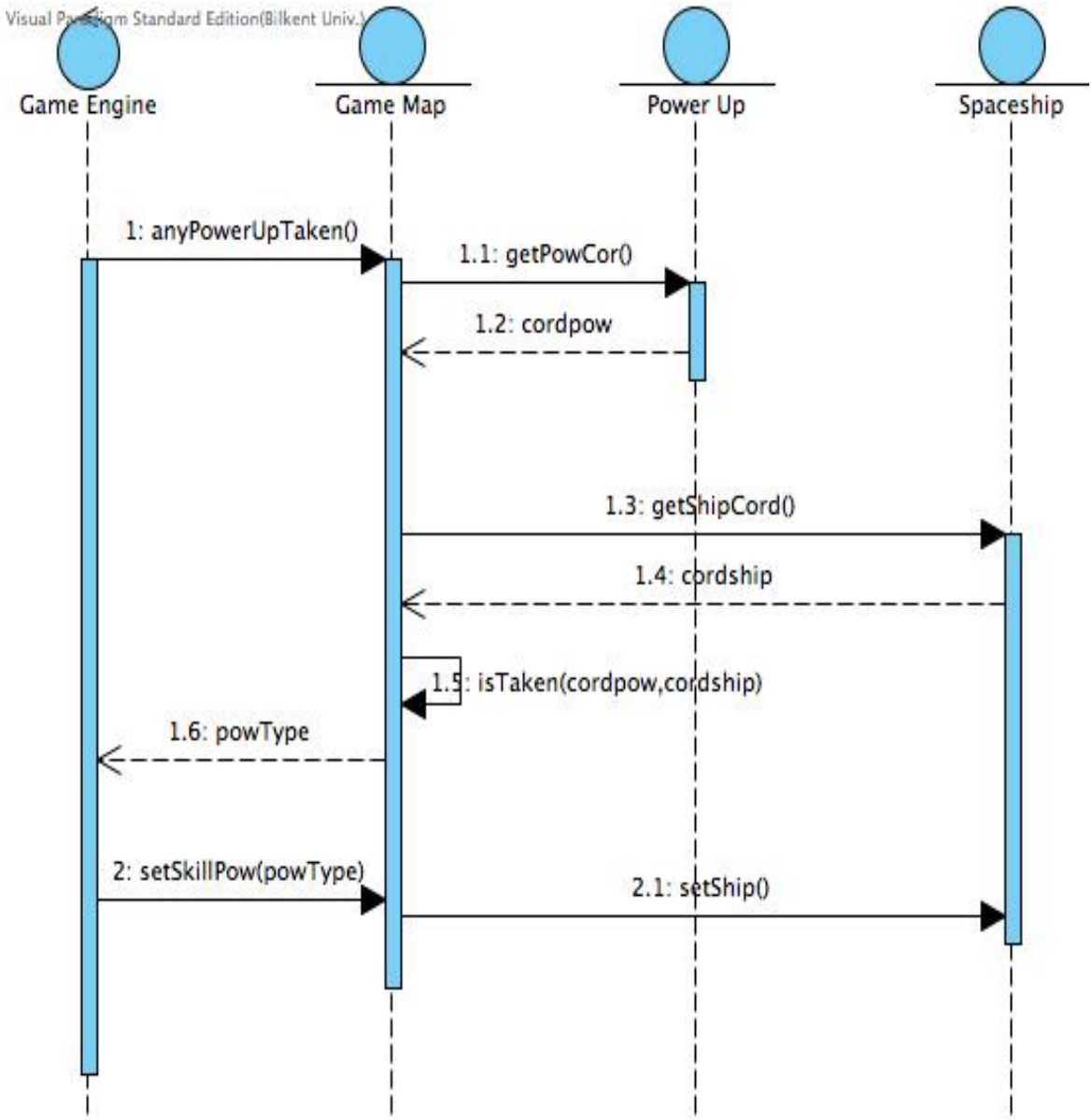
Description :

By selecting play game option Game Panel call to Game Engine that controls all models. Game Engine also calls Game Map to apply game data, which keeps information about level and setting. Game map creates aliens according to data taken from Game Data. After this process Game Panel display game.

6.2.1.2.Scenario Name : Collect Power Up

Scenario :

During the game , Omer have realized that power-up appeared. He intents to gain the power-ups to get high score and complete level. He goes to the direction where power-up is coming. And get it. According to power up type, spaceship has some features.



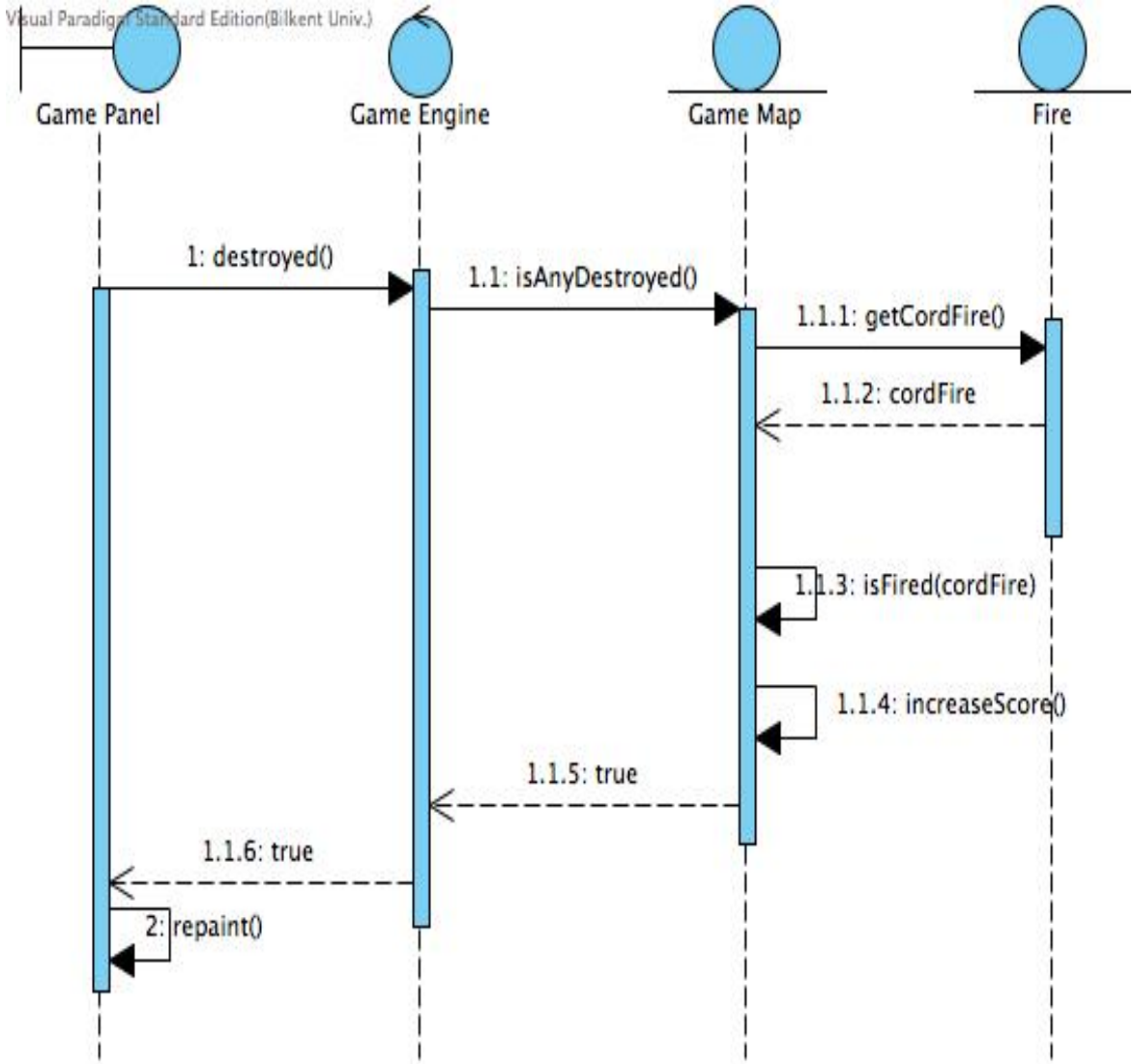
Description :

Game engine call Game Map by anyPowerUpTaken() method. Game Map take power up and spaceship coordinates. isTaken() methods in Game map checks whether power up is taken or not by comparing coordinate. If power up is taken, Game Map return power up type to Game engine. According to power up type, Game engine call Game Map so that Game Map can set spaceship.

6.2.1.3.Scenario Name : Destroy Enemy by firing

Scenario :

Omer is struggling all aliens at the game. There are two ways to kill enemies as explained above. Firstly, the user can choose to collide enemies and destroy them. However this method cause the user's spaceship lose health energy. Second way is focusing target and firing. For this reason, Omer prefers to fire.



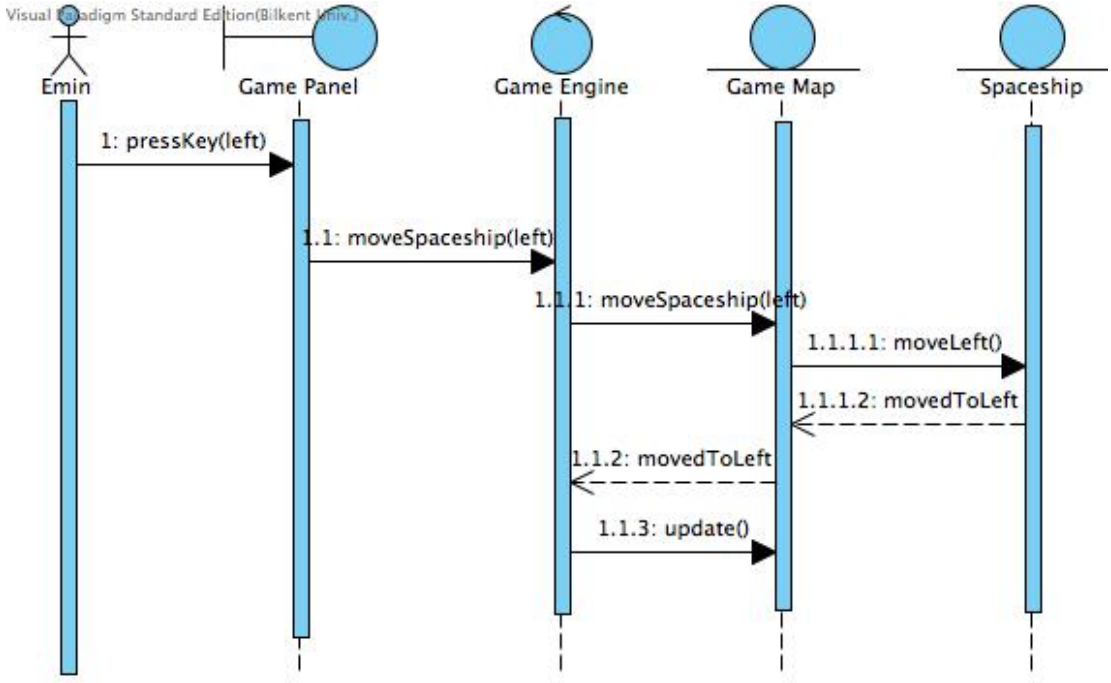
Description :

At any time there are so many bullets on screen, which are fired user's spaceship. In this scenario it is controlled that any enemy ship is destroyed from any bullet fired by users spaceship. Game Panel call Game Engine. Game engine applies Game Map via isAnyDestroyed method. Game Map describe fires coordinates and controls that any enemies ship are hit. Game Map class includes vectors which keeps all enemies object. Isfired method in Game map defines which aliens ship is destroyed. After this method, It triggers to increaseScor method to give point the user.

6.2.1.4. Scenario Name: Move Spaceship

Scenario:

Emin starts to game and tries to move by pressing arrow keys.



Description:

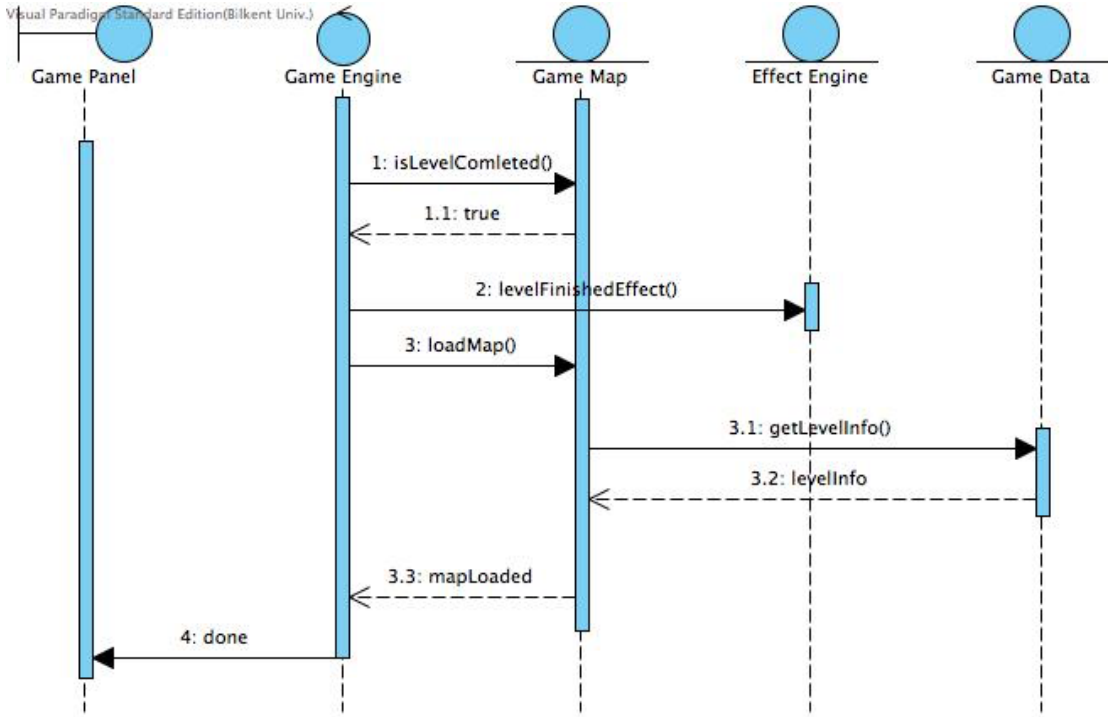
When the user presses to arrow keys, the signal goes to our boundary object which is Game Panel, Game Panel sends the message

to the controller which is Game Engine and Game Engine forwards this message to the Game Map, Game Map calls the moveLeft() method of the Spaceship object to change its coordinates. Then Spaceship changes its coordinates and returns that to Game Map, Game Map forwards that to Game Engine and Game Engine updates the map.

6.2.1.5. Scenario Name: Level-up

Scenario:

Emin is very successful while playing the game and finished one level successfully and goes next level of the game.



Description:

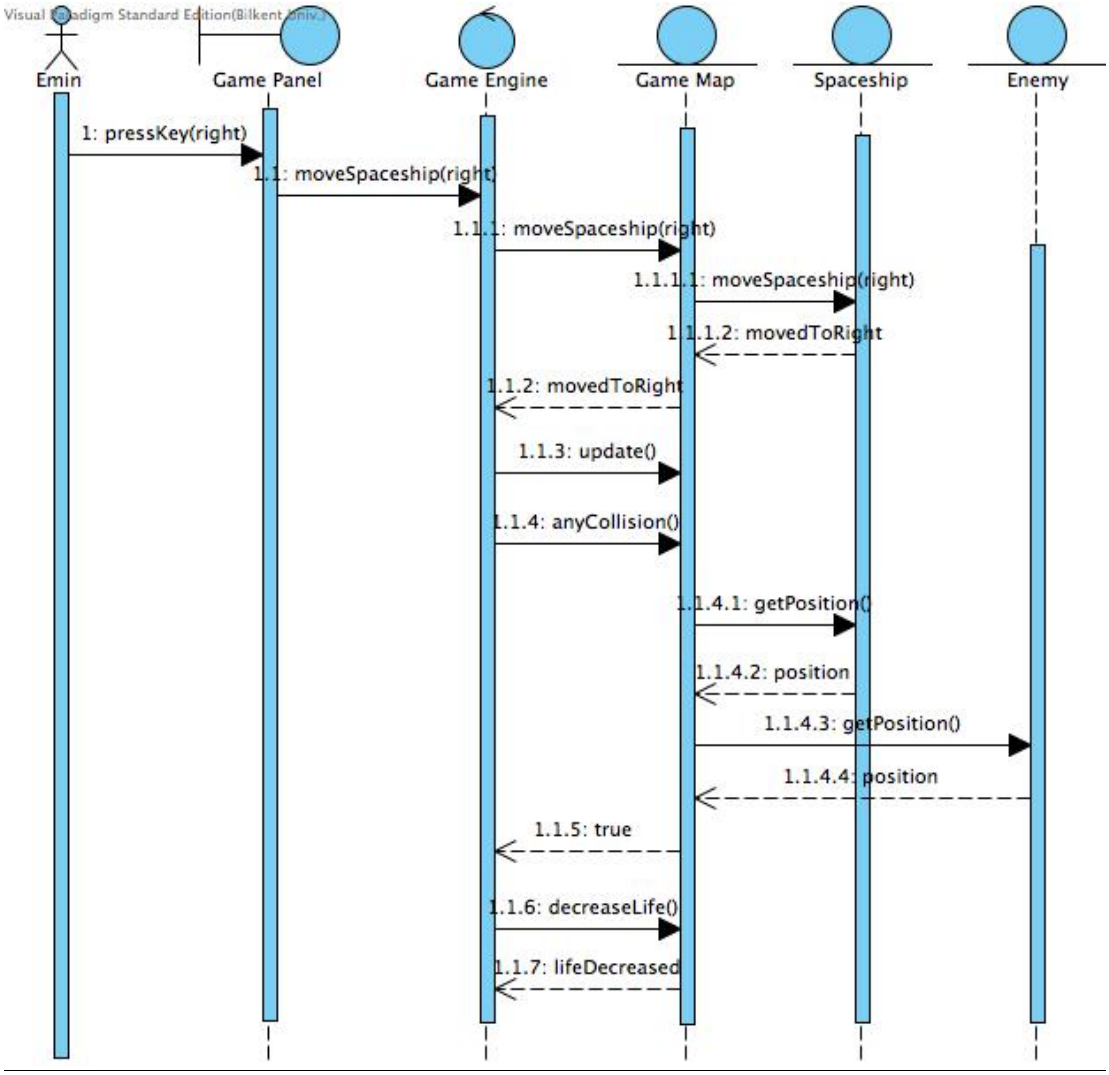
Game Engine controls the Game Map to learn whether the level is done or not. Because the level is up, Game Map returns true and then Game Engine calls Effect Engine to make the sound effect of ending the level successfully. Then calls Game Map to load new map, Game Map asks to the Game Data about new level information and calls the getLevelInfo() method and gets it then sends the signal of

mapLoaded to the Game Engine and Game Engine sends the signal "done" to the Game Panel.

6.2.1.6. Scenario Name: Die From Collision

Scenario:

Emin is brave warrior in the game, but one of the bullets of the enemies collides with the Emin's spaceship and Emin loses one of his lives.



Description:

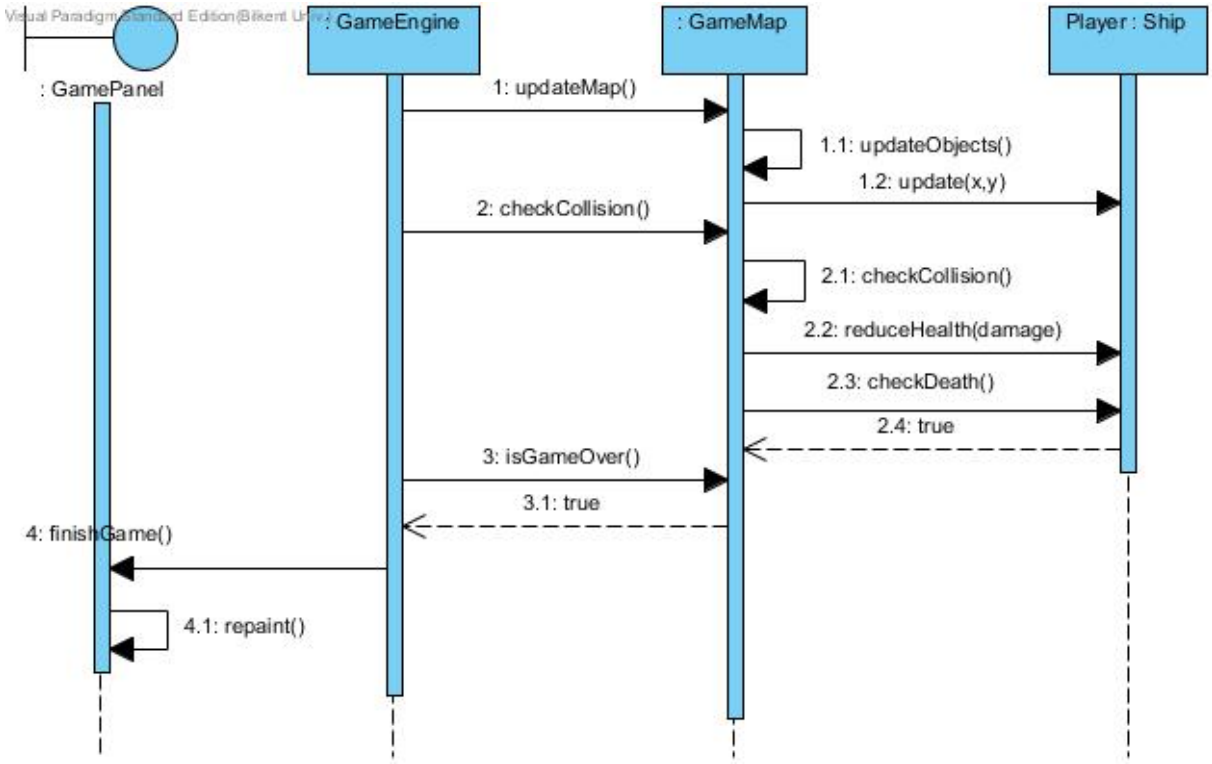
Emin presses to arrow key and moves after movement, Game Engine checks for any collision by calling the Game Map with anyCollision() method. Game Map gets positions of Spaceship first and then get positions of enemies and if it finds a match sends true to

the Game Engine. Game Engine calls decreaseLife() method and Emin's one of lives is decreased.

6.2.1.7.Scenario Name :Die from Projectile

Scenario :

While Erdinç is playing the game, his ship is hit by an enemy projectile. The ship is out of energy and is destroyed. Then the game is over and the session is finished.



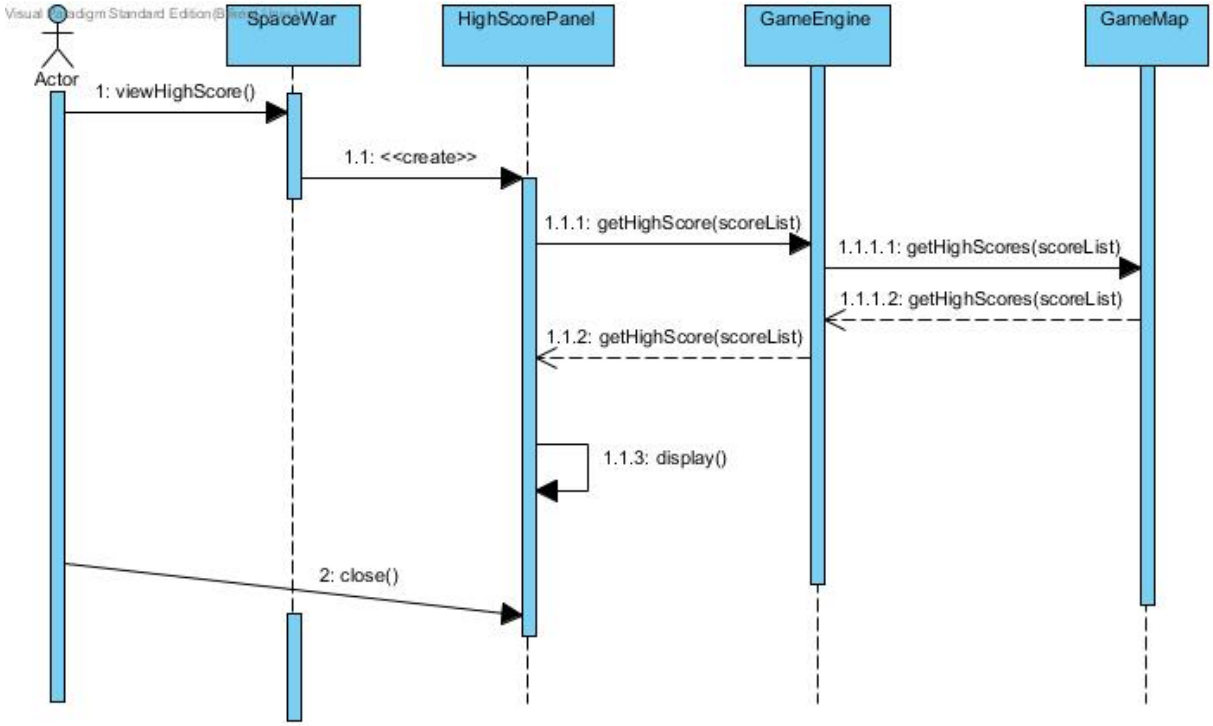
Description:

GameEngine periodically commands GameMap to update the map and check for collision between all instances of GameObject. When a collision is detected, GameMap handles the collision. Here the player is hit by an enemy bomb so GameMap reduces the Ship object's health. After this, GameMap asks whether the Ship object is out of health. If the Ship object returns true, the game is over. GameEngine is notified that game is over. GameEngine notifies boundary objects that game is over.

6.2.1.8.Scenario Name : View High Scores

Scenario :

Erdinç wants to view the high scores list. Erdinç clicks the High Scores button in the main menu. The game displays the high scores and Erdinç stays in this screen for a while. Then he clicks the exit button to go back to the main menu.



Description:

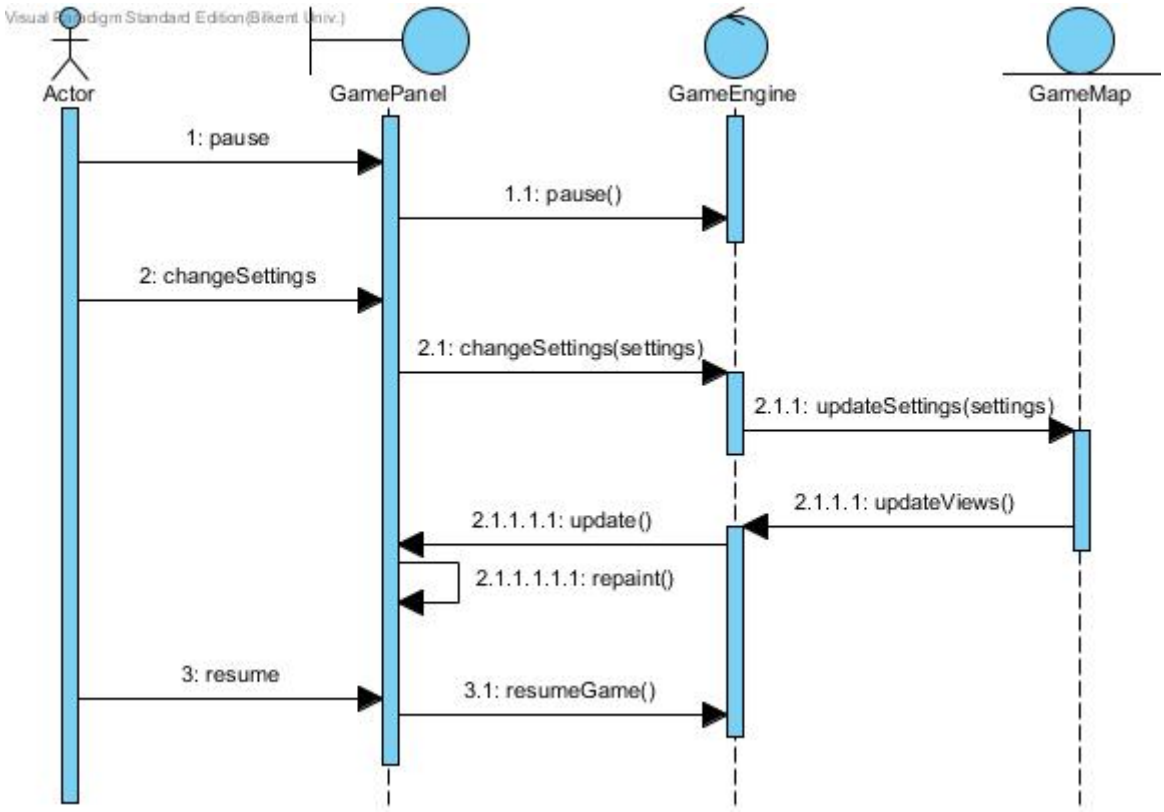
When the user clicks High Scores in main menu, SpaceWar creates an instance of HighScoresPanel. HighScoresPanel asks GameEngine to get the list of high scores, and GameEngine asks GameMap for the list. When GameMap fetches and delivers the list to GameEngine and GameEngine to HighScoresPanel, HighScoresPanel is complete and displays relevant information. When the user is done, the user closes the window and SpaceWar takes control.

6.2.1.9.Scenario Name : View High Scores

Scenario :

While Erdinç is playing the game, he pauses the game. The game displays the pause menu and Erdinç chooses to change the

settings. After Erdinç changes the settings, game takes him back to the pause menu. Next, he clicks Resume Game and continues to play the game.



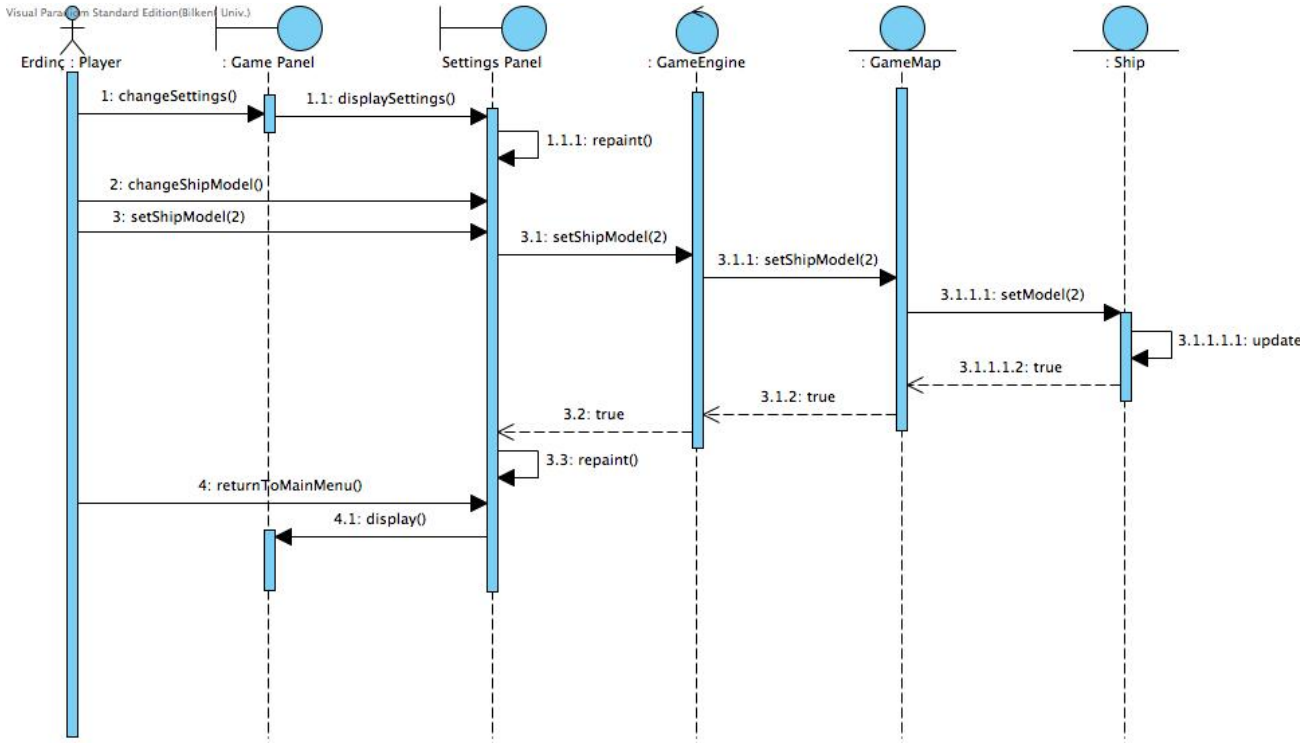
Description:

The user clicks the hotkey for pausing the game. GamePanel tells GameEngine to pause the game so the GameEngine starts to wait until the next order from user. The user changes settings from the pause menu and submits the settings form. GamePanel delivers the form to GameEngine to change the settings. GameEngine commands GameMap to update the settings according to new information. When GameMap is done, it notifies GameEngine. Then GameEngine notifies boundary objects to update. Next, user clicks resume so GamePanel tells GameEngine to resume the game.

6.2.1.10.Scenario Name: Change Settings

Scenario:

In this scenario, player Ayşe has already started the game and looks at the main screen. Ayşe wants to change ship model from game settings; she selects change settings from the main menu. System display settings menu. Ayşe selects change ship model option. System show possible ship models. She selects one of them. System updates the ship model. She returns to main screen by selecting return to main menu.



Description:

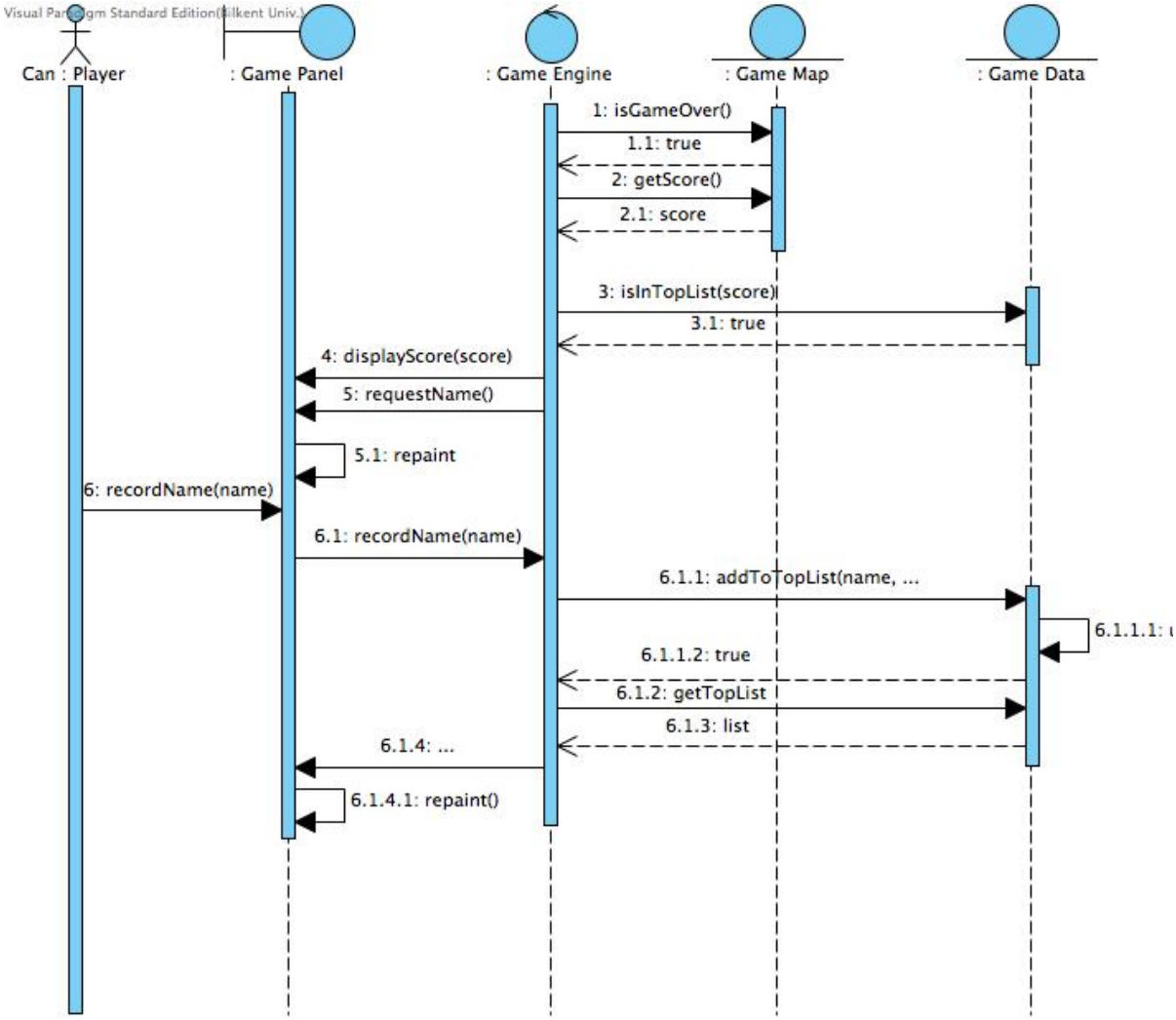
Ayşe sends a request to Game Panel object by invoking the method changeSettings. Then, Game Panel invokes displaySettings method of Settings Panel. Settings Panel invokes repaint and displays settings menu. Ayşe requests to change ship model and calls changeShipModel method of Settings Panel. Settings Panel repaints and shows available ship models. Ayşe selects 2nd model and calls setShipModel method. Settings Panel calls setShipModel of Game Engine. Then, Game Engine calls setShipModel of Game Map. Finally, Game Map invokes setModel of Ship. Ship calls update method. Invoked setShipModel methods return true until Settings Panel. Ayşe

calls returnMainMenu method from Settings Panel. Main menu is displayed.

6.2.1.11.Scenario Name: Make High Score

Scenario:

In this scenario, Can has already finished the game. System displays her score on the screen and check whether it's in the top 5 list. System confirms that she made the top list. System asks for her name for record. She enters her name. System updates the ranking with her name and score. Then, system displays updated list on the screen.



Description:

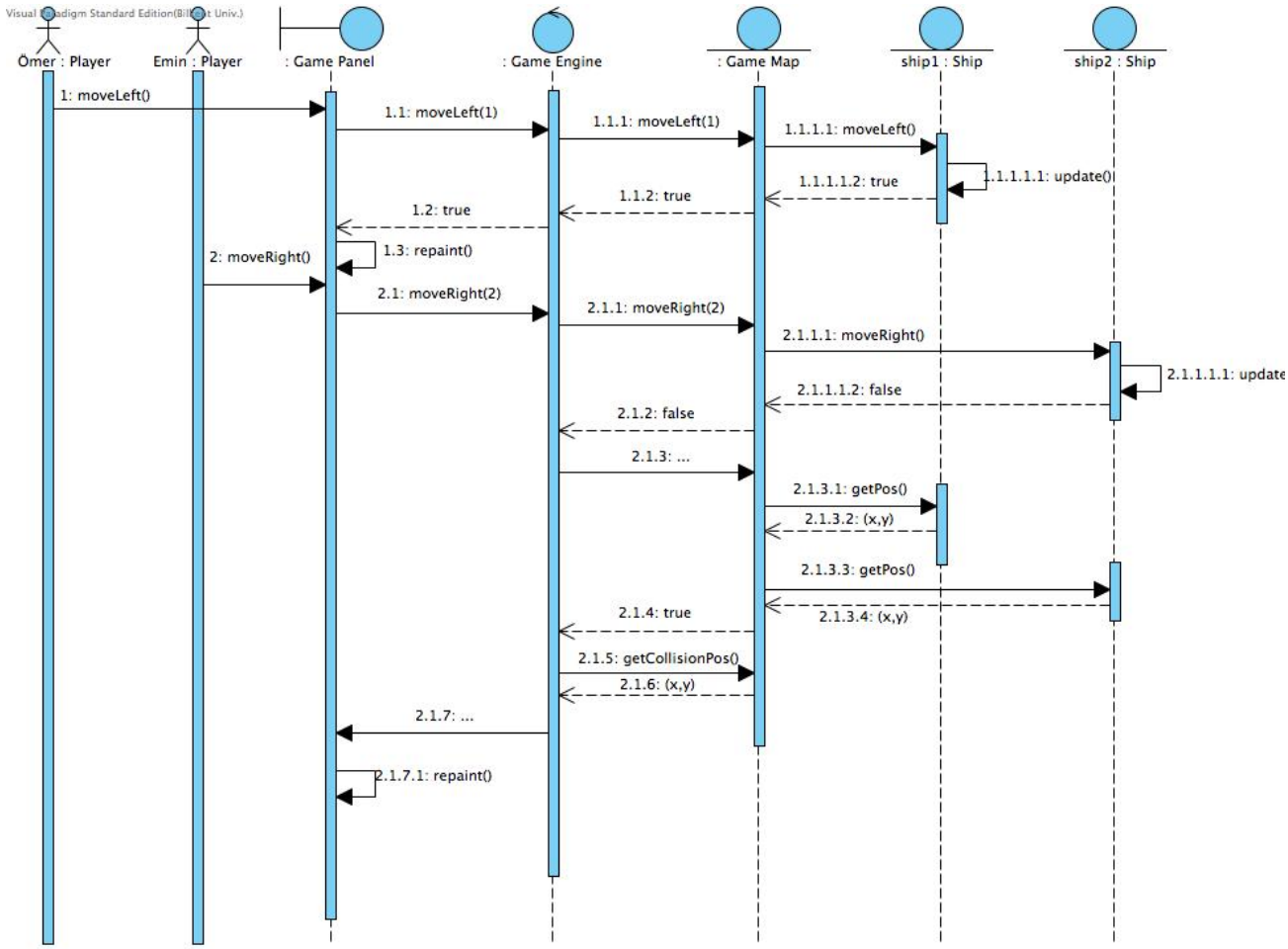
At some point of the game, Game Engine invokes isGameOver method of Game Map and it returns true. Then, Game Engine invokes

getScore method. Game Map returns score. Game Engine invokes isInTopList method of File Manager. File Manager returns true. Then, Game Engine calls displayScore of Game Panel. And Game Engine also invokes request Name of Game Panel. Game Panel self-invoke repaint method. Then, Ayşe enters her name. Then, Game Panel calls recordName method of Game Engine with the parameter name. Game Engine calls addToTopList of File Manager with parameters name and score. Then, File Manager self-invokes updateList method with name and parameter and return true to Game Engine. Game Engine calls getTopList of File Manager. File Manager returns top list to Game Engine. Game Engine calls displayList of Game Panel with the parameter top list. Finally, Game Panel self-invoke repaint method.

6.2.1.12.Scenario Name: Multi Player Mode – Ship Collision

Scenario:

While playing the game in multi-player mode, Ömer moves his ship to left. System updates its position. Then, Emin tries to move his ship to right. Then, system notifies a collision and display a collision effect. None of the players loses their energy or get any points.



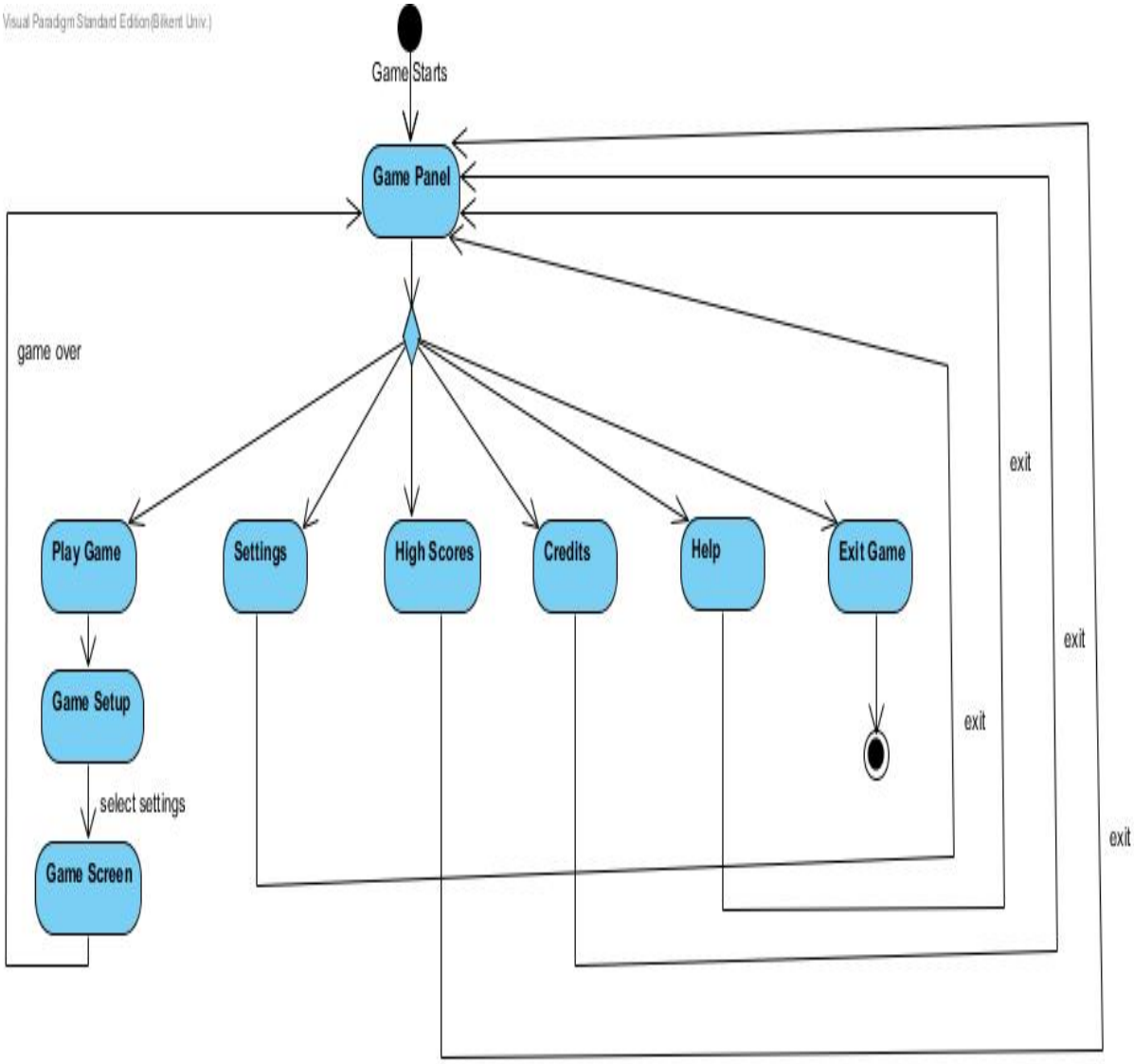
Description:

Player Ömer calls moveLeft of Game Panel. Game Panel calls moveLeft method of Game Engine with parameter 1. Game Engine invokes moveLeft of Game Map with parameter 1. Game Map calls moveLeft of ship1. Ship1 updates and return true. Then, Emin calls moveRight of Game Panel. Game Panel calls moveRight method of Game Engine with parameter 2. Game Engine invokes moveRight of Game Map with parameter 2. Game Map calls moveRight of ship2. Ship2 updates and return false since the position is already occupied. Then Game Engine calls shipCollision method of Game Map. Then, Game Map invokes getPos methods for both of the ships and compare their positions. Game Map returns true to Game Engine. Game Engine calls getCollisionPos method of Game Map. Game Map returns position of collision. Game Engine invokes displayCollisionEffect of

GamePanel with this position. Finally, Game Panel self-invokes repaint.

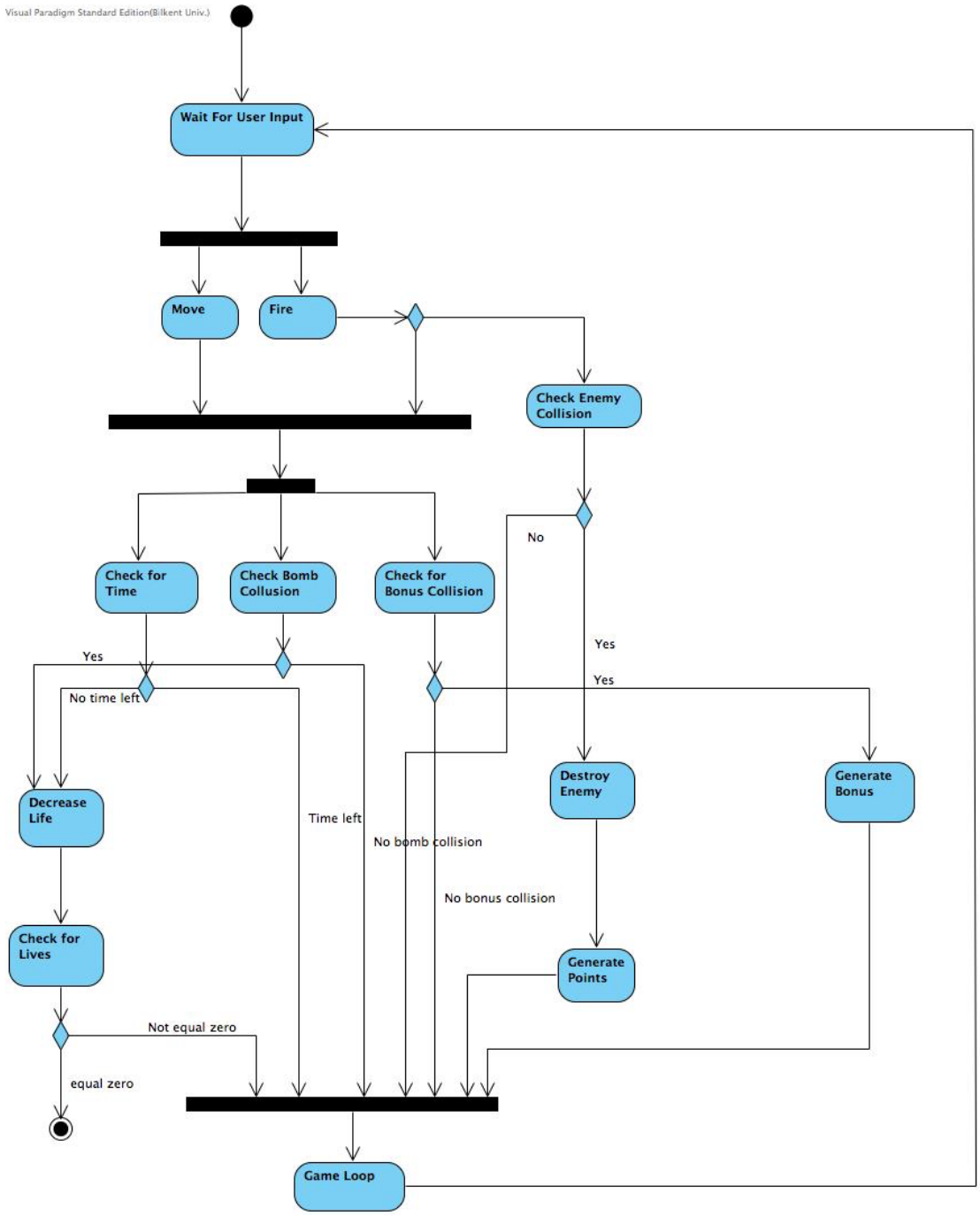
6.3. Activity Diagram

6.3.1.User Interface Navigational Path



Game Panel is the main menu where the user can view the state of the game. All of the interfaces come back to Game Panel when their function is over. Play game button opens another panel where the user chooses settings such as difficulty and spaceship icon. Once the user submits the settings, game begins.

6.3.2. Play Game

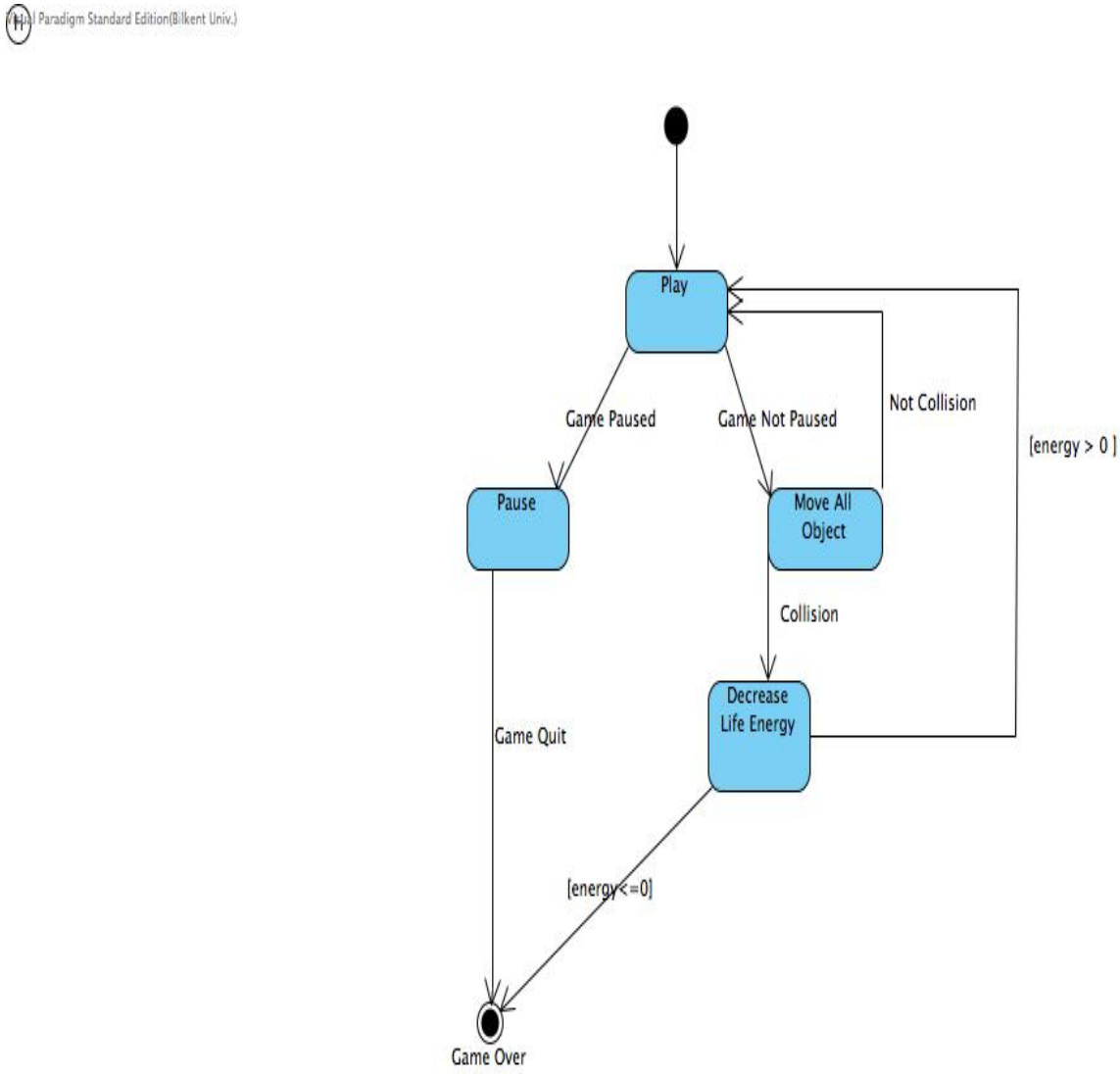


When the game starts it waits for user input, user can move and fire with pressing keys, after these inputs if user fires, game checks whether the fire is reached to any enemy or not, if it reaches to one of the enemies, it destroys the enemy and generates point for user. Also if user fires or moves, the game checks for time, checks for bomb collision and bonus collision. If the time is up or any bomb is received from the enemy, user loses one of its lives, if its lives are over the game is over. If any bonus is received the bonus will be generated and

the game will be continued. After all of these if the player did not die, the Game loop will be active again and wait for user input.

6.4. State Diagrams

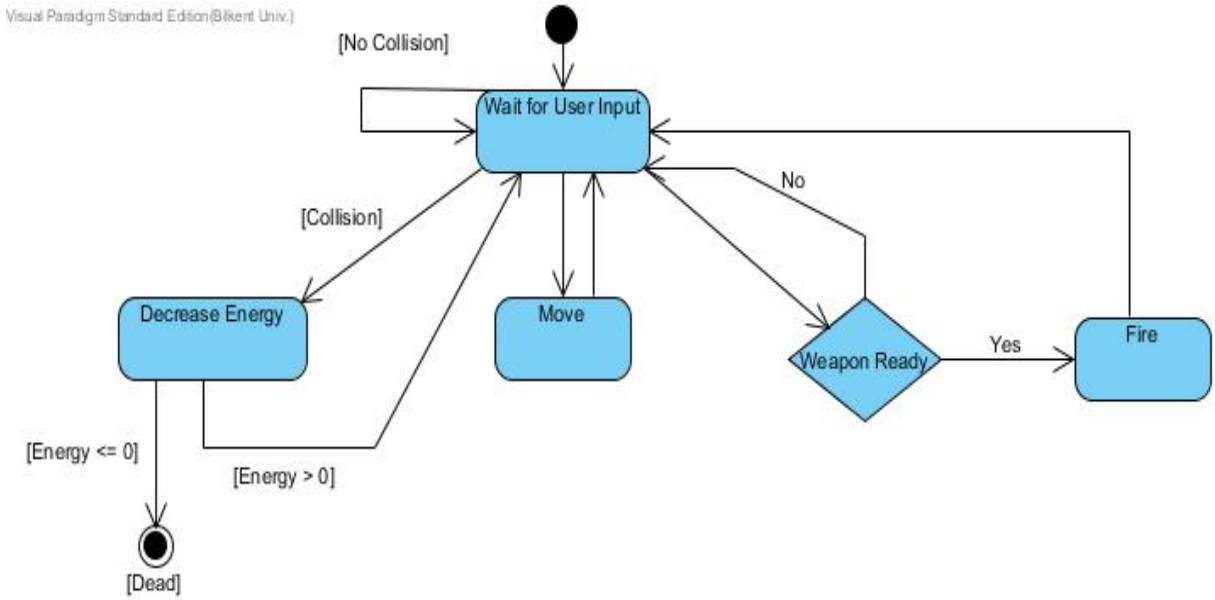
6.4.1 Play Game



While inside the loop, it is in Play state. Unless the game is paused, all movable objects are moved. Then possible collisions of spaceship with the aliens are checked. If collision is happened , spaceship loses its energy.The game loop continues in this manner as long as the spaceship is still alive (not all of its life energy are lost). Note that a paused game is either resumed or the user quits the current

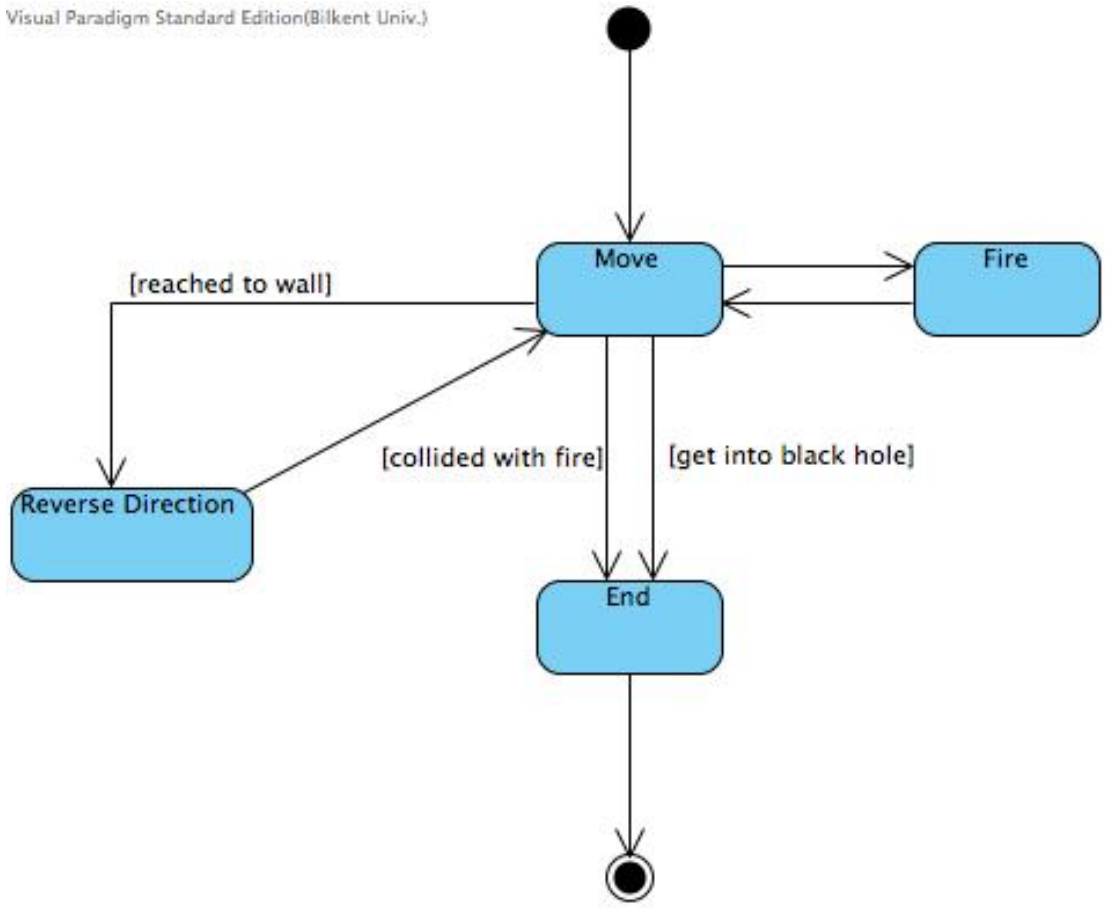
game.

6.4.2.Ship



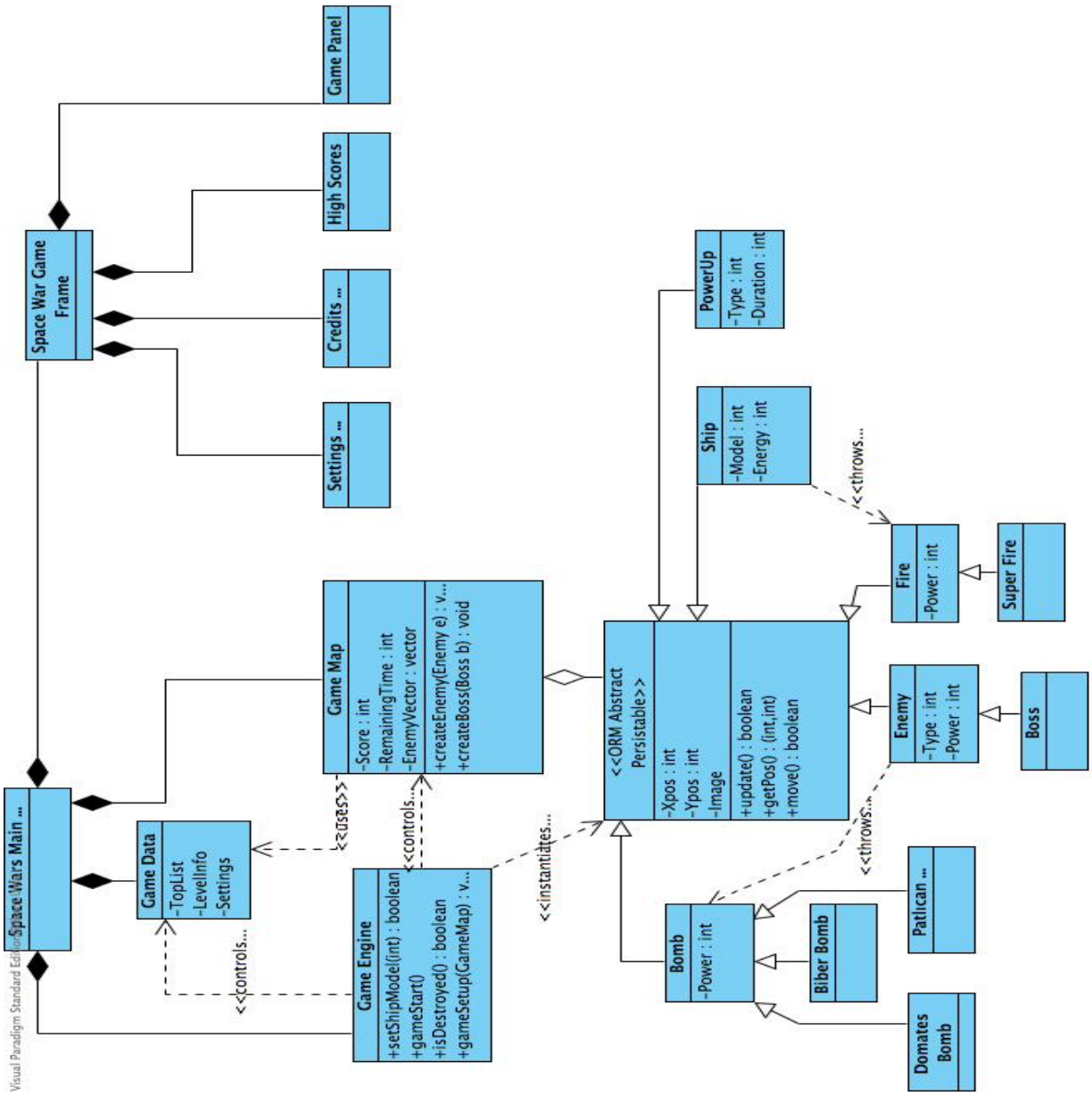
Ship object periodically checks for collisions and decreases energy if it is hit by a projectile. Then it checks its energy and decides whether it is dead. Besides this, ship object can move with user input and fire if its weapon is ready.

6.4.3. Enemy



Enemy always moves and periodically fires. If it reaches the wall moves in other direction. If it collides with a fire or get into black hole(which is a power-up of spaceship) it dies and gets to end state.

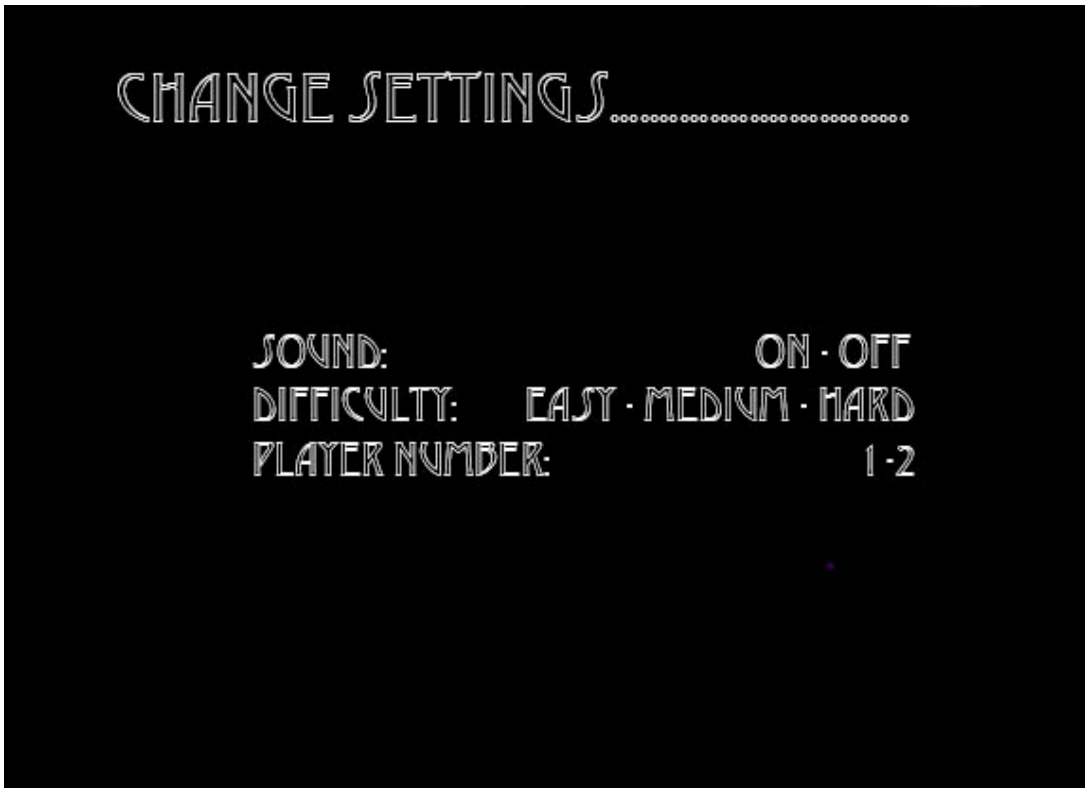
6.5. Object and Class Model



6.6. User Interface

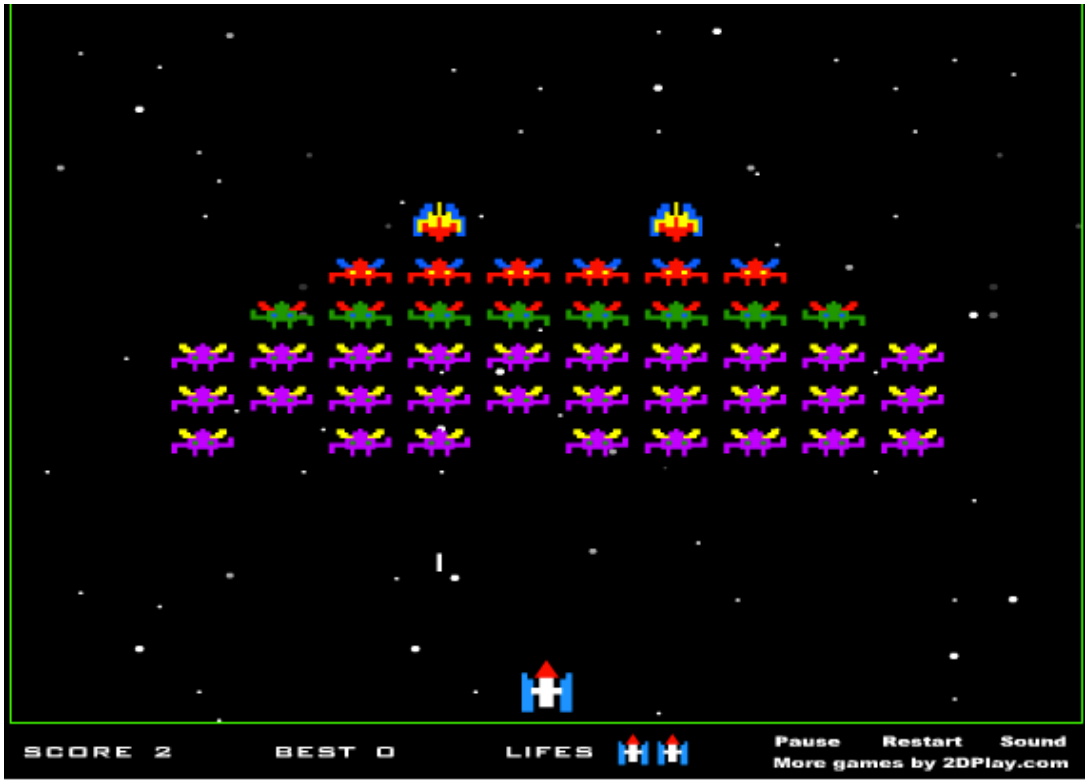
6.6.1. Main Menu

6.6.2. Change Settings



6.6.3. View High Score

6.6.4. Play Game



References

[1] http://www.oyunkrali.net/oyna/oyun/618/galaksiler.oyun