Bilkent Üniversitesi

CS319 Term Project

Section 3

Group 3J

Galaxy Invader

Analysis Report Iteration 1

Group Members:

1. Umer Shamaan
2. Tanay Toksoy
3. Osman Can Yıldız
4. Cemre Osan
5. Denizhan Yağcı

Contents

5.5 User interface and Screen Mock-ups

5.5.1 Menu

5.5.2 How to Play

5.5.3 Settings

5.5.4 Credits

5.5.5 Game Frames

6. Conclusion

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 in order to fight back. Galaxy Invaders is an arcade style space shooter game inspired by Defender. The role of the player is to shoot down enemy spaceships in order to advance and finish the game.

We decided to work on this game 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 given 3 spaceships as health. The player will be able to move the ship with w, a, s, d keys and they will make the ship go up, left, down and right respectively.

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. Occasionally, the invaders will try to abduct humans and the player will get bonus points by saving them. However, shooting them will inflict a severe point penalty. There are three types of enemies, which behave in a different manner ie. one tries to shoot down the player, the other tries to abduct humans and the third type tries to slam into the player ship. Defeating a certain amount of players will culminate in a difficult boss battle.

2.2. Enemy Types

2.2.1 Basic Enemy

Basic enemy type shoots orange laser to player's spaceship. They can also move up and down.

2.2.2 Kamikaze

Kamikaze does not shoot but they move aerobically with the purpose of colliding with the player's spaceship.

2.2.3 Boss

The boss come up at five-k levels. Boss moves slowly but it has high damage and it can shoot multiple fire.

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 most slowest type of bullet and it has low damage.

2.3.2 Laser

2.3.2.1 Player Laser

Player laser is a blue laser that faster than basic bullet and it has more damage than basic bullet. 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. If this laser hits the player, the player's spaceship is destroyed and the player loose one health.

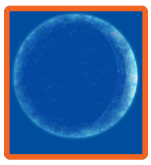
2.3.3 Missile

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 but it is able to destroy the bosses in seconds.

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.

2.4.3. Health Power-up

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

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. Play Game:

The player will start the game by clicking on this button. The player controls a spaceship. Enemy spaceships will randomly appear from both sides of the screen. The player will be able to control the craft both vertically and horizontally. He/She will fire by pressing spacebar. The amount of the enemy ships will increase over time. The main objective is to avoid taking damage and destroying a specific number of enemy ships and then defeating a boss.

3.2. 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.

3.3. Settings:

Here the player can control volume and turn game music on or off.

3.4. Credits:

This section will display the name and the main roles of the developer.

4. Nonfunctional Requirements

4.1. Game Performance:

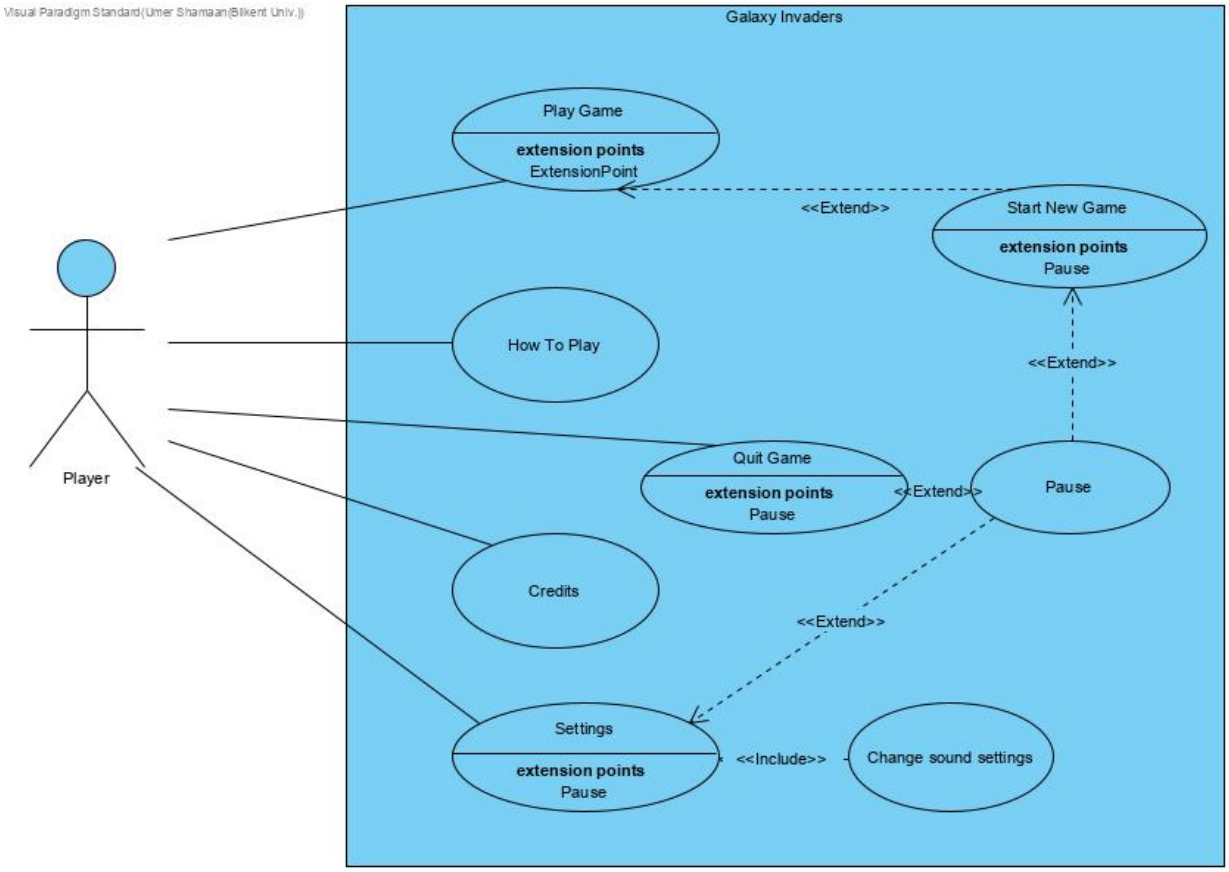
As the game is very light it is intended to work on most of the computers containing a below average gpu and at least 20Mb of ram and 30Mb of disk space. Moreover it will work on Windows XP and above.

4.2. Interface:

The interface is very user-friendly and the controls are simple and easy to learn. The minimum age required to play the game is 5.

5. System Models

5.1. Use Case



UseCase1:

Use Case: Play Game

Primary Actor: Player

Pre-conditions:

- Player must be in main menu

Entry Conditions:

- Player chooses the play game option

Exit Conditions:

- Player clicks on Quit Game
- Player presses escape key

UseCase2:

Use Case: Credits

Primary Actor: Player

Pre-conditions:

- Player must be in main menu

Entry Conditions:

- Player chooses the Credits option

Exit Conditions:

- Player presses escape key
- Player clicks on the back icon

UseCase3:

Use Case: How To Play

Primary Actor: Player

Pre-conditions:

- Player must be in main menu

Entry Conditions:

- Player chooses the How to Play option

Exit Conditions:

- Player presses escape key
- Player presses the back icon

UseCase4:

Use Case: Setting

Primary Actor: Player

Pre-conditions:

- Player must be in main menu

Entry Conditions:

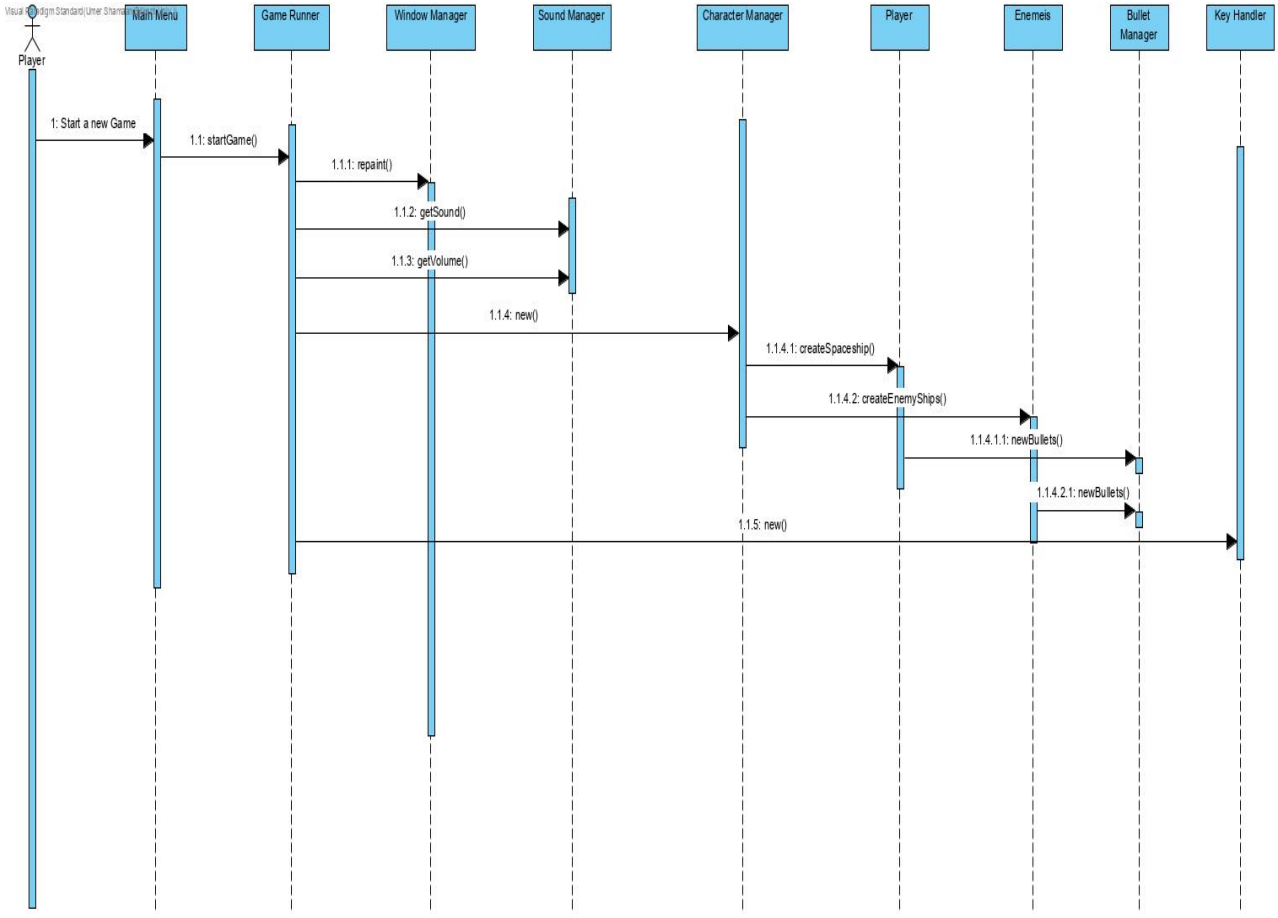
- Player chooses the Settings option

Exit Conditions:

- Player presses escape key
- Player presses the back icon

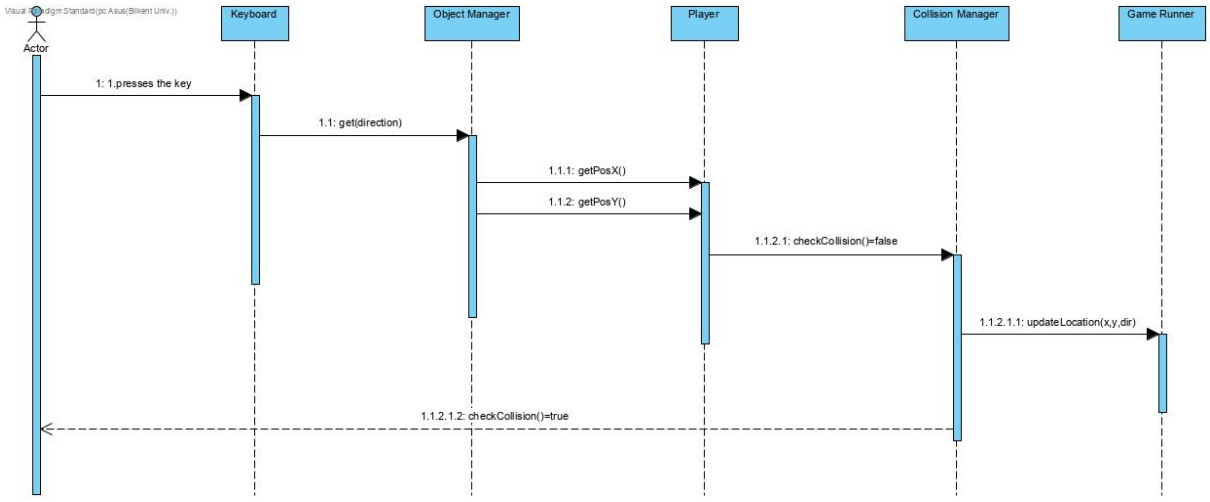
5.2. Sequence Diagrams

5.2.1. Start Game



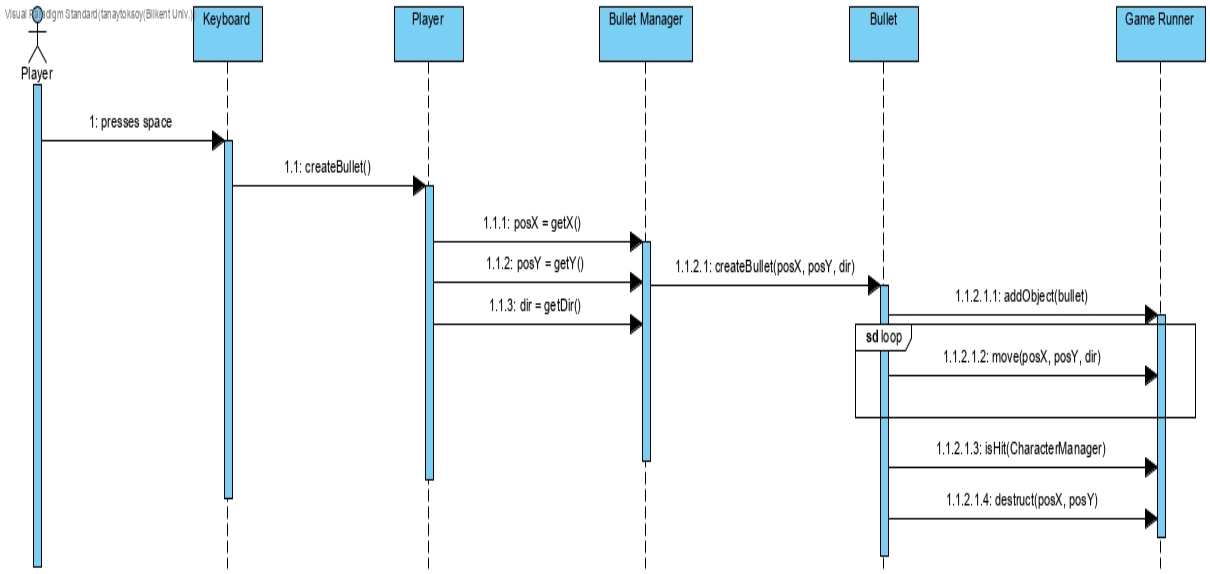
When the player starts the game by clicking on the play game icon on the main menu, the game is initialized by the game runner. Sound and the game window are created by the sound manager and window manager respectively. The game runner also creates the level manager which in turn forms the game objects including the player ship, enemy ships and bullets.

5.2.2. Aircraft Movement



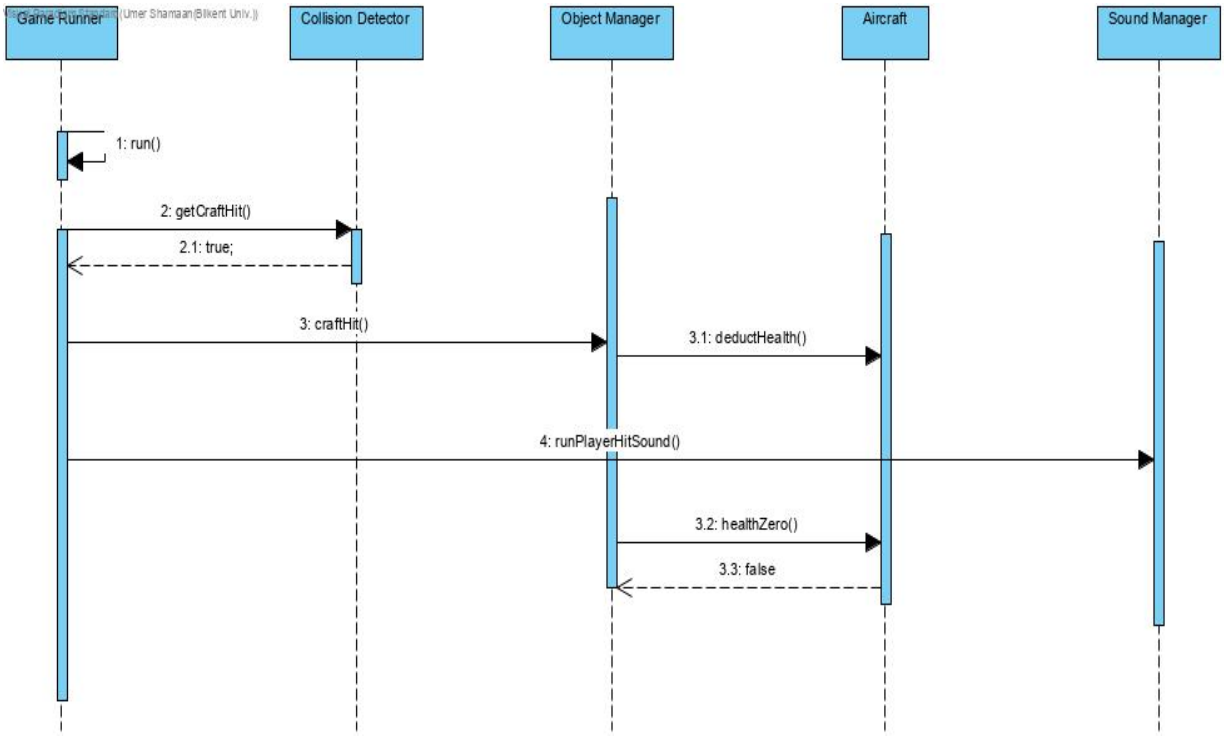
When the player presses keyboard buttons assigned for movement, object manager gets the spaceships coordinates, then checks if the new coordinates crosses with the other objects coordinates. If they do not collide, collision manager allows spaceship to move, otherwise asks for another input.

5.2.3. Firing weapons



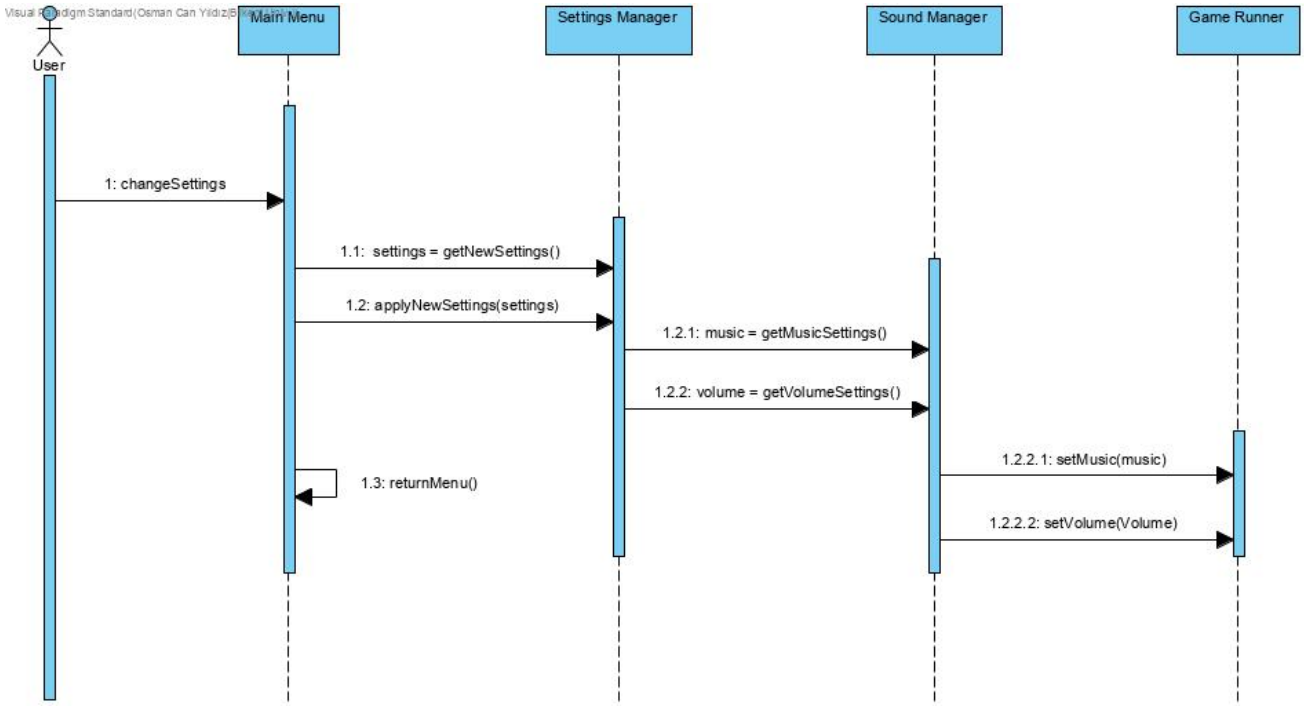
When the player presses the space button while the game is running, the information will be sent to the bullet manager which will create a bullet according to the position and the direction of the aircraft. Then the game runner will add the bullet object to the map.

5.2.4. Getting hit by Bullet

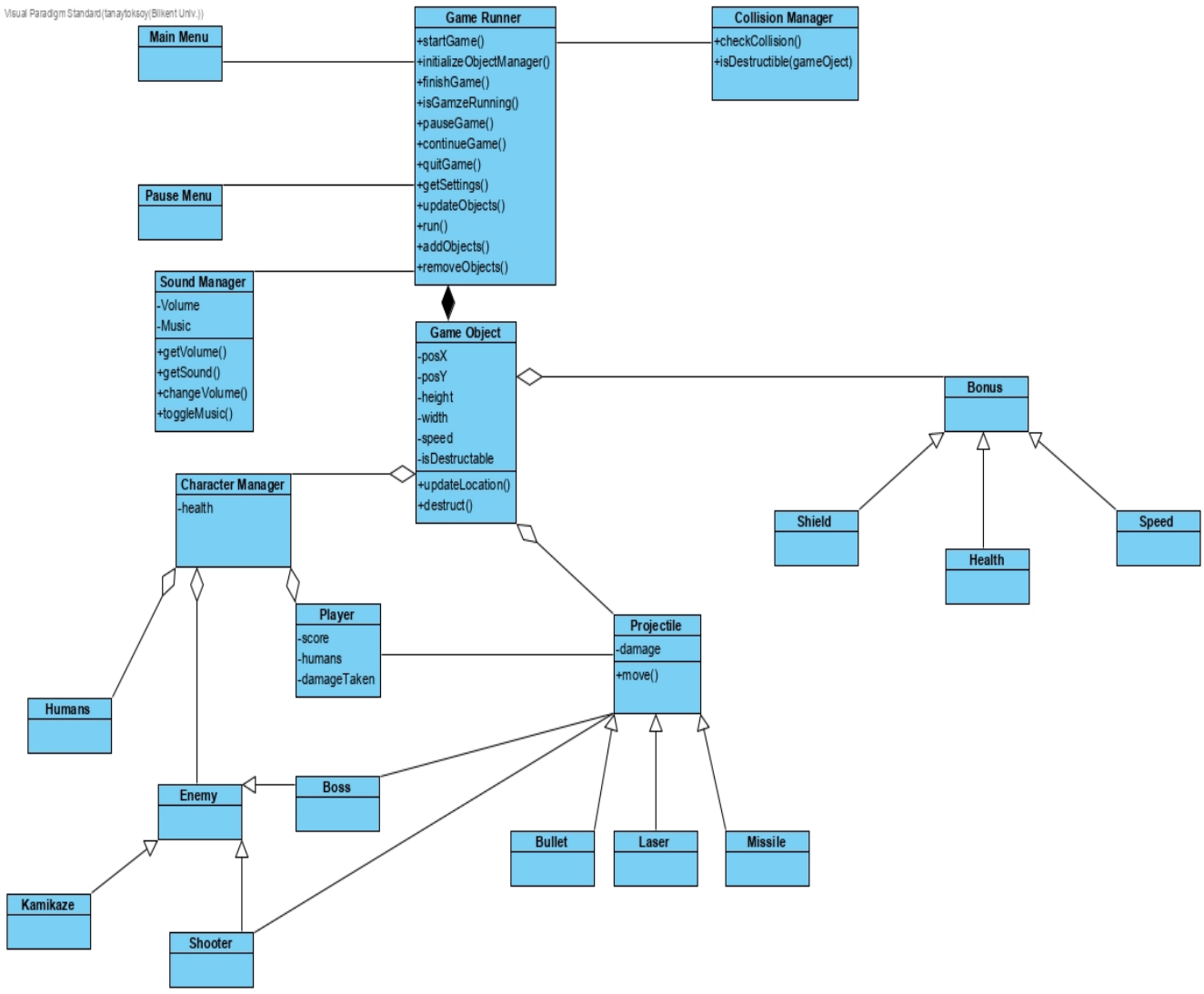


The collision detector determines whether the player craft has collided with a projectile. If so, the object manager deducts a certain amount of health from the player and the sound manager plays an explosion sound.

5.2.5. Sound Settings



5.3. Class Diagram



MainMenu: The main menu class provides a user interface where the player can start a game when they click on new game, or see credits or change sound settings.

PauseMenu: This class also provides an interface when the user goes to pause menu after pressing escape key. Here the user can quit game, change sound settings or resume game.

SoundManager: This class is responsible for all of the sounds of the game including game music and effects such as firing and explosions. It also enables the player to change the volume or turn the music on or off.

Collision Manager: This class constantly checks if there are any collisions between the objects for example if the enemy ships are hit by the player bullets. It then checks if the collided object is destructible and whether if it is out of health and then instructs the object manager class to remove the item from the map.

GameRunner: This is the most important class of the game. It is initialized as soon as the player initiates the application. It creates a new frame as the player starts the game and manages the other classes such as Sound Manager and Game Objects according to the data it gets from those classes.

GameObject: This class is responsible for initializing and removing game objects and storing their common attributes such as coordinates and size.

CharacterManager: This class manages the characters which have specific behaviours. Such as enemy bots and players.

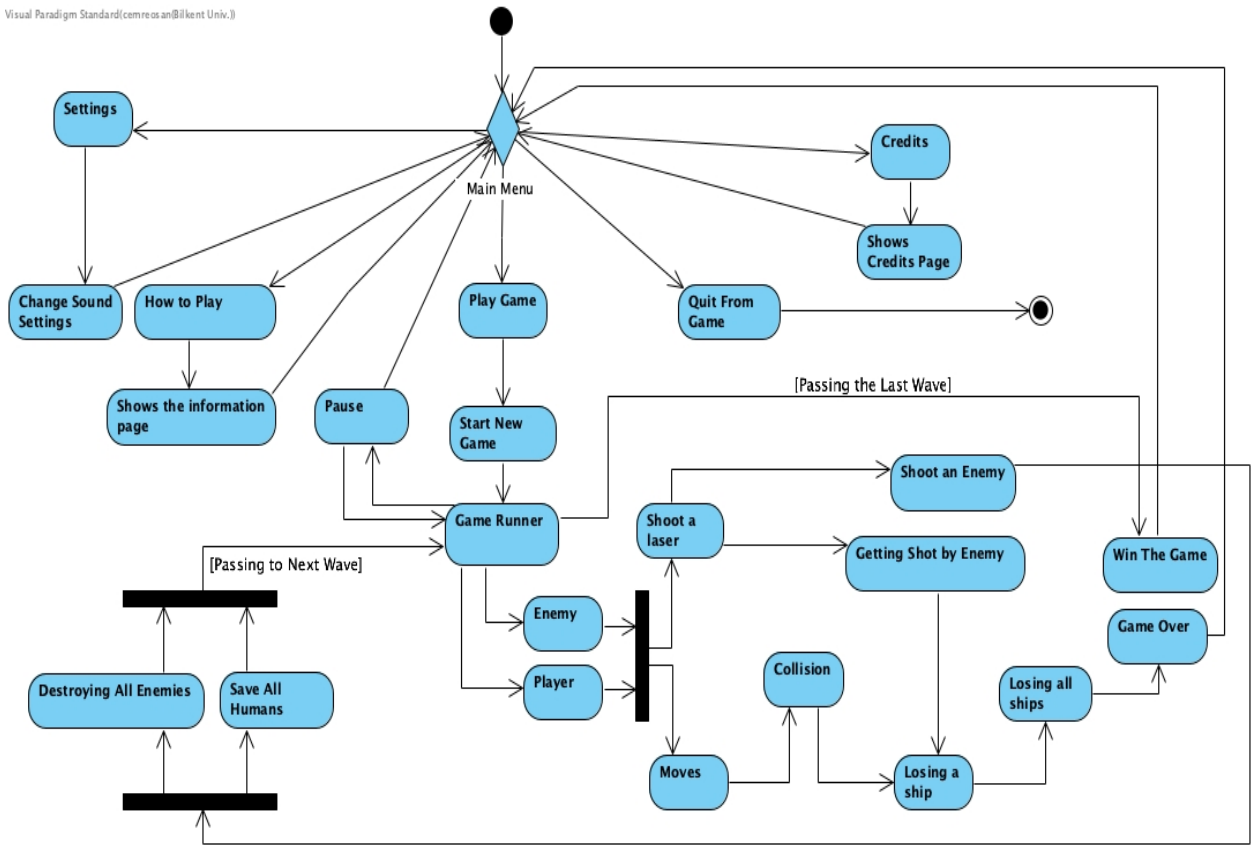
Player: This class represents the spaceship controlled by the player.

Enemy: This represents the enemy bots. There are three subclasses for this class which represent three types of enemy with seperate behaviors and roles.

Projectile: This class is initialized when the player presses spacebar or at random times from the enemies. There are three subclasses of it which have different properties such as speed, penetration and damage.

Bonus: This subclass of GameObject is initialized randomly when an enemy bot has been destroyed. It consists of three types of upgrade classes.

5.4. Activity Diagram



The activity diagram above represents the series of actions in the process of this game. The game process is modeled with all detailed stages. In this way, all the action stages of our software program can be followed.

First, the initial node presents the start point of using this game program. As the second point, decision node represents the main menu page of the game. There are five options which are “Play Game”, “Credits”, “Quit From Game”, “How to Play” and “Settings”. “Credits”, “How to Play” and “Settings” as buttons in the page refer the user to the their related pages and then

to return to the main menu page again is possible. Moreover, “Quit From Game” action is to exit from the game program.

The “Play Game” action stage begins with by pressing “Start New Game” button. Therefore, “Start New Game” activity is passed to and playing game part starts in “Game Runner” activity stage. In this step, user as a player having three ships has two options which are moving and shooting a laser to the enemies. Player has a chance to do this actions at the same time. Enemy actor and its all doings belong to the system mechanism. Like player, enemy can move and shoot a laser. When they moves and a collision happens between them, player loses one of its three ships. Furthermore, when the player is shot by enemy, one more ship is lost. If all ships are lost, the game is over and the user is returned to the main menu page again. However, if the player shoots to enemies and succeeds to destroy all enemies, one of the two conditions for passing to next wave is done. “Save All Humans” activity represents the other condition which is done, if the player succeeds to save all humans. After these two activities are completed, the player can pass to the next wave of the game. This action follows “Game Runner” activity stage in this diagram. If all wave sections are completed, the player wins the game. In this stage, program is in the “Win The Game” activity stage. Then, the next stage is “Main Menu” that is a decision node. In addition, while playing the game, pause button option lets the user stop the game. This action is represented by “Pause” activity in this diagram. In that stage, the user has two options which are returning to play game or closing the game page played then returning to the main menu page.

5.5. User interface and Screen Mock-ups

5.5.1. Menu

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

5.5.2. 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.

5.5.3. Settings

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

5.5.4. Credits


GALAXY INVADERS
CREDITS
OSMAN CAN YILDIZ
UMER SHAAMAN
TANAY TOKSOY
CEMRE OSAN
DENIZHAN YAGCI
GO TO MAIN MENU

5.5.5. 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.

6. Conclusion

In this report, we have analysed, designed and implemented a new kind of classic Defender or space shooter game which is called Galaxy Invaders.Our reports consist of Functional Requirements, Non-Functional Requirements and System Models. In functional requirements part, a player can perform all the functions which is explained in that part. Secondly,in Non-Functional Requirements, our predictions about performance and easy

learning interface are mentioned. Lastly, system models of our game consist of use case model, Dynamic models, Activity Diagram, Sequence Diagrams, Class and Object model and User interface- Navigational Paths and Screen Mock-ups.As a result, we have created this analysis report to make our job easier in the implementation stage.