Bilkent University

Object Oriented Design

IMPRISONMENT

Analysis Report

İdil Ağabeyoğlu 21300761
Evren Ergen 21300781
Ismail Kerimov 21300355
Sinan Öndül 21200962

Progress Report
February 20,2016

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Object Oriented Design course CS319/1.

Table of Contents

Table of Figures

1. Introduction

The game of "Imprisonment" is an action game that is inspired by "Volfied" which created by DOS game. DOS game can be considered as the ancestor of the video games.

In the game, the user will be able to control a dot and the goal of the user is to complete the map by drawing quadratic shapes until the game is won. However, the monsters in the map makes it easier said than done. If monsters hit the dot or the line that is drawn by the dot the user will lose 1 life span.

Although ”Imprisonment“ is an inspired game, it contains many different features from its inspiration such as new bonuses, new maps and new sound effects. In addition the game will be designed as a desktop application that can be controlled by mouse and keyboard.

(https://www.youtube.com/watch?v=vlYYXHICUis)

2. Proposed System

The monsters will randomly move around the screen and the player, a dot, will try to fill the screen from edges by drawing squares. While it is trying to it, if any of monsters hits the player, the game is lost. However as a difference of the existing game,Volfied, our game entertains the users with new maps, new and various bonuses and also new sound effects.

2.1 Overview

In the game, while the masters who tries to catch the dot and causes to lose the game, a dot which is kind of trapped in a frames is going to try the fill the screen by drawing quadratic shapes and at the same time by running from monsters. Size of the frame will get smaller with each quadratic shapes including rectangles and squares and in the limited space within the monsters will give a challenging game to the users. In order to win the game the size of the frame must decrease %80 so basically if the

user can keep continue to play until %20 of the map without losing he/she will win the game. In addition the user will have 3 lives to be successful at the game.

2.1.1 Features

2.1.1.1 Dot

Dot is the feature which is in the con troll of the user while playing the game. The colour of the dots changes according to the difficulty of the levels regarding as in the easy part the dot will have the colour of blue, in the medium part the dot will have the colour of green and in the hard part the dot will have the colour of red. The direction of dots will be limited in the edge of the frame in addition it will have only the ability to move left, right, up and down.

2.1.1.2 Monsters

Monsters will randomly move around in order to eliminate the user from the game and its move limits when it touches to wall. Additionally if the monster hits the dot or the line which is drawing by the dot, the user will lose one of its lives and the line will be removed from the frame. In each level the speed of the monster will change according to the difficulty of the level.

2.1.1.3 Bonuses

There are 3 types of bonuses which can be collected by the dot.

- Lives: In that bonus the dot will have extra lives to continue the game without losing the game.

- Freeze time: When this bonus is collected the monsters will freeze for 3 seconds and the dot will have chance to play the game without any threats
- Slow time: This bonus provides monsters to slow down for 5 seconds and with the decreasing speed of the monsters the user will have chance to see the moves of the monsters more efficiently and act accordingly.
- Monster Destroyer: This bonus helps the dot in order to deal with the monsters. This bonus enables the player to choose one monster that will be removed from the map.
- Map Shrinker: This bonus decreases the threshold of the map that the user should satisfy in order to win the game by %5.

2.1.1.4 Wall

The shape and the size of the wall is unstable regarding with the acts of the dot. It will decrease in each quadratic shape that is drawn by the dot and will work as a mirror in the aspect of reflecting the monsters and changing its directions in a straight line by horizontally and vertically.

2.2 Functional Requirements

- The user will use the keyboard and the usage of the mouse will not be included in the game except selecting the options in the main page such as play game, settings.
- The user will be allowed to pause and then resume the game.

2.2.1 Play Game

The player tries to fill the screen up to %80 without being caught by monsters that are randomly moves in unfilled screen area. Player must not die three times in order not to lose the game. While there is monsters and a player, there will be random bonuses in the game so that the player can boost his/her dot up and makes easier to win. Also the bonuses will be in the unfilled area of the screen and when the player covers the bonus into the filled area, the bonus is collected.

When the player finishes a hardness level, he/she can select others such as easy, medium and hard. With the more hardness, the faster monsters and slower dot make the player's job harder. This increases the eye and hand coordination while offers much more fun.

2.2.2 Settings

Some features of the game's control is given to user. User will be able to change the some settings of the game and these changes are as follows

- Change the background image
- Turn down and turn up the music

2.2.3 Help

Although "Imprisonment" is user friendly and easy to understand and play by the user the game will include an help ingredient that shows the overview of the game and answers of the questions that might be frequently asked as a text document.

2.2.4 Exit

While the program is running the player may desire to not pause the game but quit the game and this operation is done by pressing the exit button. The user may not be satisfied with his/her current success such as user might have only 1 life span while only the %10 percent of the map is accomplished so in that case the user might want to exit the game and start a new one.

2.3 Non-functional Requirements

2.3.1 Game Performance

The performance of the imprisonment is the high priority for overall design. The game will have a short response time with 3ms delay for the user to play. Additionally the inspiration of the game is based on the old games, thus the system requirements will be tried to keep minimum.

2.3.2 Graphical Smoothness

Although the game bases on the old games, the graphical interface the proper animations and smooth graphics are aimed in order to make the game more exciting and giving pleasure from the interfaces to the users and this will be accomplished by trying to keep the response time of the graphics as low as possible.

2.3.3 User-Friendly Interface

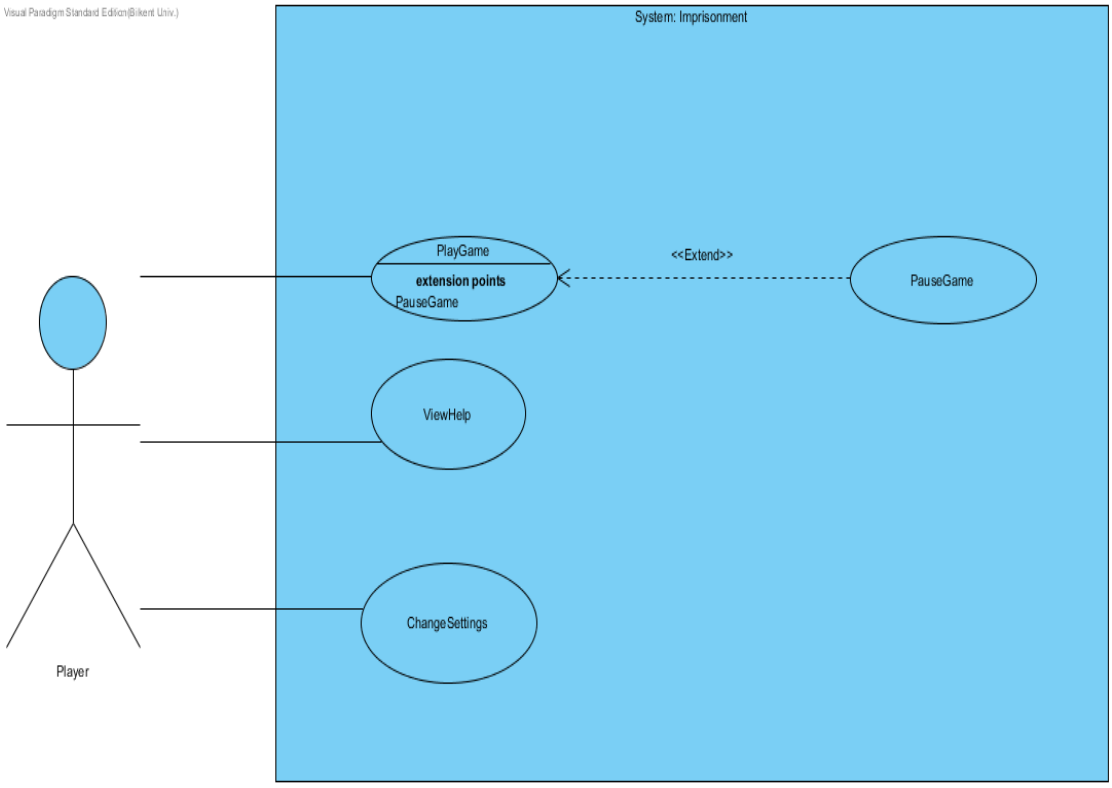
User friendly interface is an essential attribute to make the game attractive for the users. The users should be able to play and understand the game easily and they ought to feel comfortable by facing with minimum difficulty. Because of the reasons

which is specified above the benefits which is given to user such as bonuses are maintained to the users.

2. 4 Constraints

- The game will be implemented in Java and will be able to played in all operating systems that support Java Language.
- The game will be played in 40 fps(frames per second).

2. 5 Use-Case Model



Illustrates the Imprisonment's use case model: Figure 1.

2.5.1. Use Case Descriptions

Use Case #1

Use case name: PlayGame

Participating actors: Player

Entry condition: The game must be open and it will be on main menu.

Exit condition:

• Player covers the map's at least %80.

• Player loses lives level.

• Player chooses to exit to main menu.

Main Flow of Events:

1. The user opens the game through executable.

2. Player selects the game difficulty.(Easy-medium-hard)

3. Player wins.

4. System displays the time has passed of the player.

5. The exit music starts to play.

6. The game returns to the main menu.

Alternative Flow of Event:

• Player loses. (go step 4)

• Player exits the game before it finishes. This cause the progress will be lost.(go to step 6)

Use Case #2

Use case name: PauseGame

Participating actors: Player

Entry condition: A new game must have been started by the the player pressing the Play Game button.

Exit condition:

- Player chooses to exit to main menu
- Player decides to continue to play

Main Flow of Events:

1. Player chooses to play a new game
2. When he needs to take a break from the game, he clicks the pause game button.

Alternative Flow of Event:

- Player clicks the pause game button.
- Terminates the new game to go to main menu.

Use Case #3

Use case name: ViewHelp

Participating actors: Player

Entry condition: Player must be on the Main Menu and select "View Help"

Exit condition: Player selects back button to return the Main Menu

Main Flow of Events:

1. Player opens the View Help from the Main Menu
2. Player reads the texts which give instructions about how to play the game and its purpose
3. Player closes the View Help and returns back to Main Menu

Alternative Flow of Event:

- Player may not read the documents and return to Main Menu directly.

Use Case #4

Use case name: ChangeSettings

Participating actors: Player

Entry condition: Player must open executable and press Change Settings button.

Exit condition: Player closes the Change Settings button to go back to the Main Menu

Main Flow of Events:

4. Player opens the change settings menu
5. Player changes the Sound settings
6. Player closes the change settings menu

Alternative Flow of Event:

- Player may not change the sound settings and close change settings menu

2.6. Object Model

2.6.1. Class Diagrams

Imprisonment consists of 17 classes.

GameManage class is responsible for the main management of the game and it interacts with different classes. It stores the time and also has 4 operations as gameLoop(the continuity of the game), refresh(updates the position of objects and status of the game).

CollisionManager class reports the collision types whether it is wallCollided or bonusCollided and also make 2 objects to collide with each other.

InputManager class manages the inputs from the user that is given by buttons.

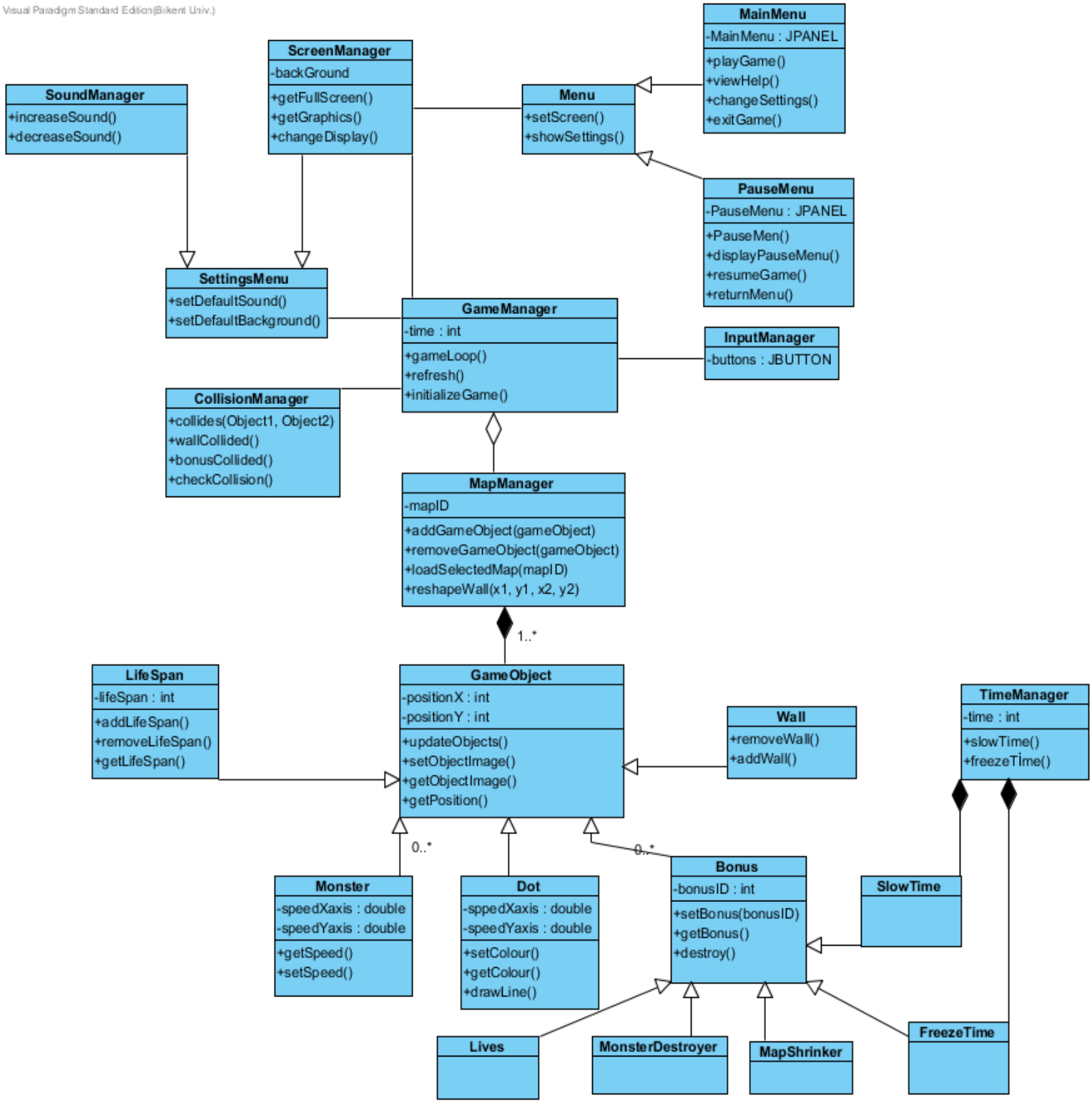
ScreenManager class stores background and by cooperating with related classes such as Menu(Main Menu and PauseMenu) it handles the user interface.

SoundManager class controls the volume of the sound.

SettingsMenu class is inherited setDefaultSound and setDefaultBackground to ScreenManager and SoundManager.

MapManager stores a mapID, handles with the addition and removal of the objects, reshapes the wall and loads the map that is selected by the user.

Furthermore, GameObject that consists of 5 different types of objects (LifeSpan, Monster, Dot, Bonus, Wall) deals with setting and getting the images of the objects and also updates them. In addition Bonus class interact with TimeManager for the implementation of the bonuses related with time such as freezeTime and slowTime.



UML Class Diagram: Figure 2.

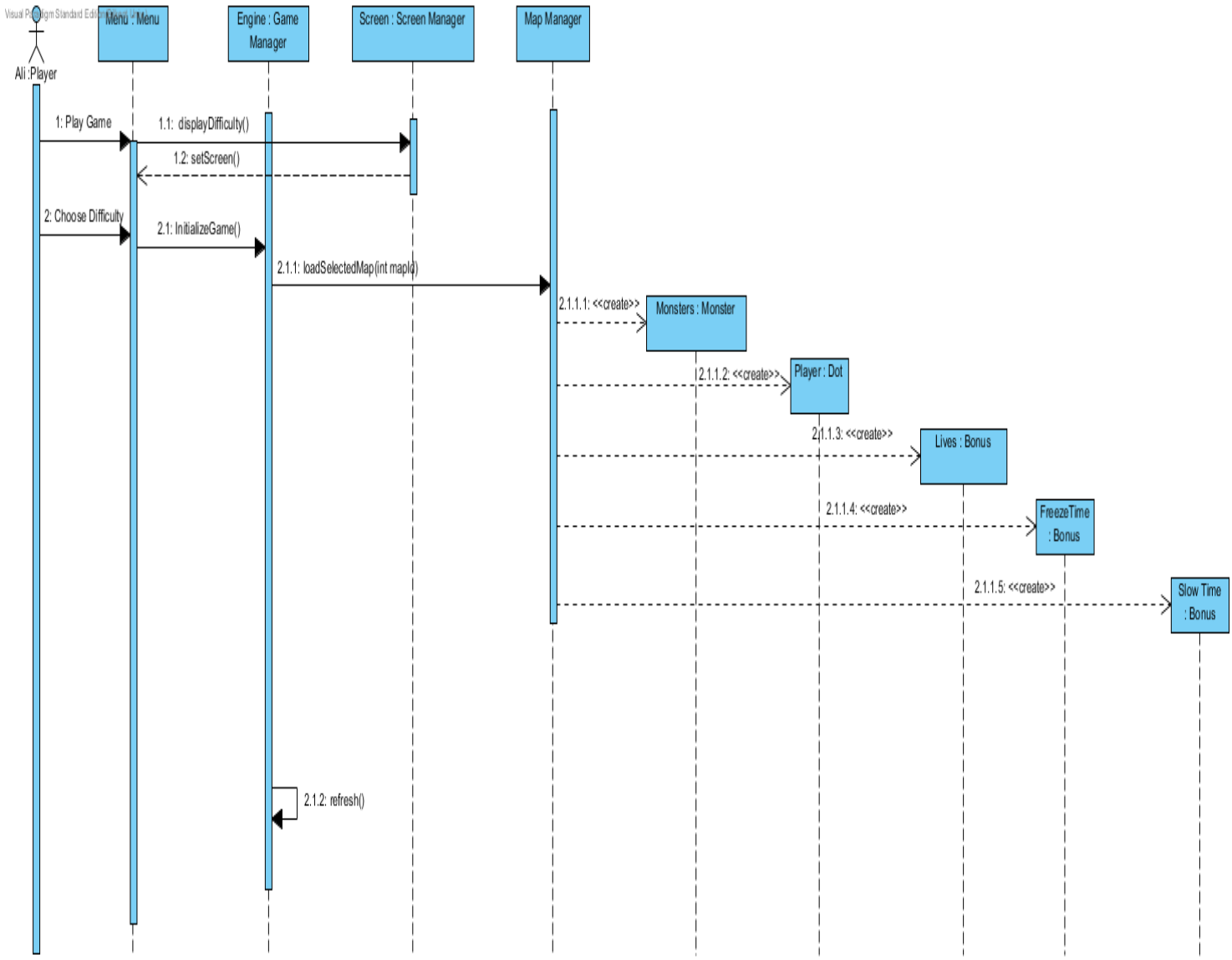
2.7 Dynamic Models

In this section of the Analysis Report, detailed information will be given about ***Imprisonment*** with Sequence Diagrams, showing important scenarios of the game. The character *Dot* and the monsters are the most important objects so they will be described in detail. Particularly, States of *Dot* will be described in state diagrams. During gameplay, system will be having some background activities and they will be described in *Activity Diagrams* section of *Dynamic Models.*

2.7.1 Interaction Between Objects and Classes

2.7.1.1 Start Game

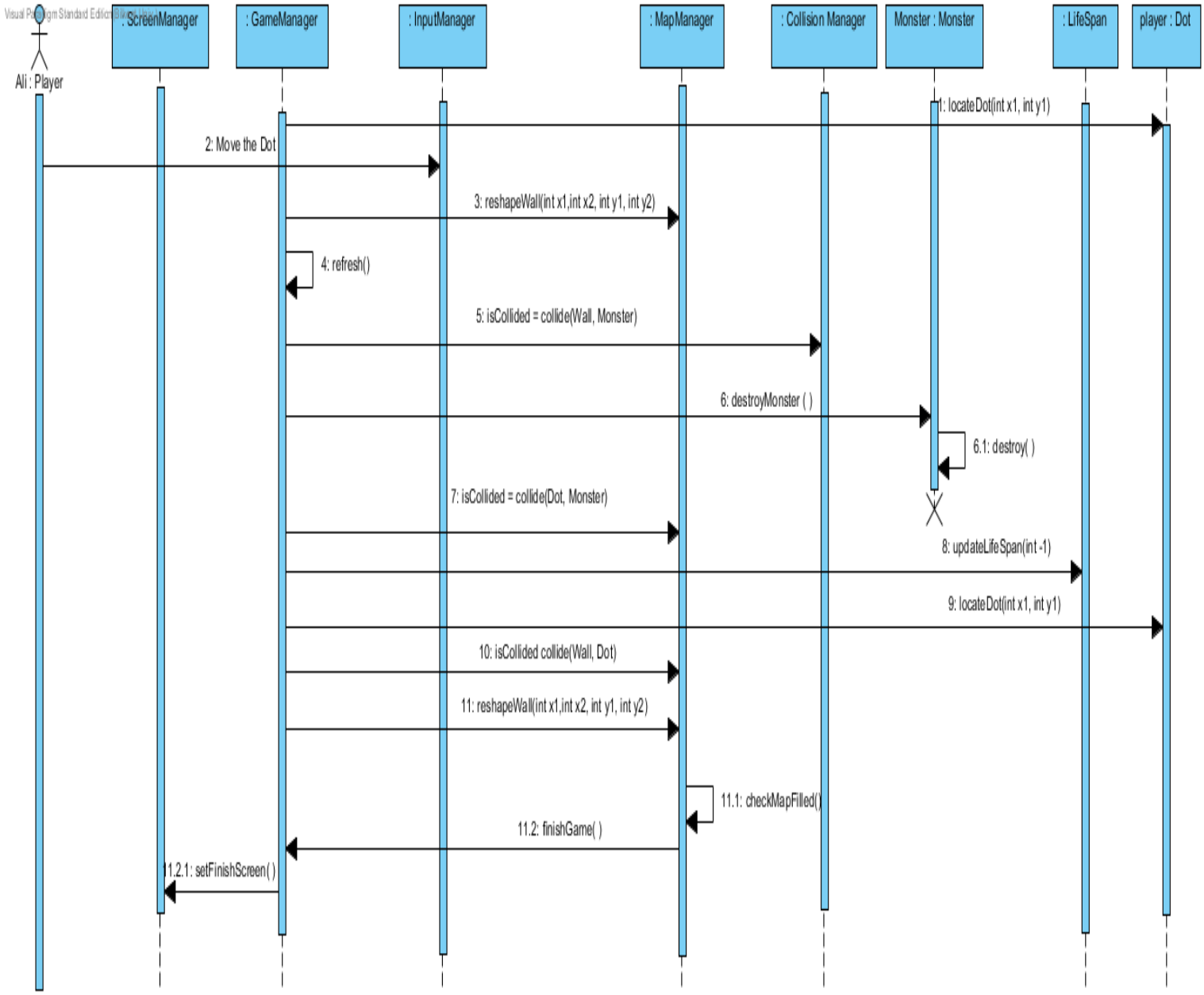
Scenario: Player Ali presses to play game by a play button. System corresponds and brings up three difficulty choices, namely easy, medium and hard. Ali chooses his desired difficulty level. Then the system loads the specified difficulty level's map, monsters, bonuses and the dot. After creating the objects of the game, system puts those objects onto desired locations specified as default at the map. Finally, system refreshes the frames and wait for player Ali to play the game.



Start Game Sequence Diagram: Figure 3.

2.7.1.2 Play Game

Scenario: Player Ali starts to play the game. Since the map is ready for play, Ali moves the dot on the screen. Ali manages to draw a polygon between walls, the walls is reshaped and collisions is checked. Any monster collided with the surrounded area, it is destroyed. Before Ali manages to draw a polygon, a monster hit Ali or the line is being drawn, one of Ali's lives is lost. Ali finishes the game by surrounding at least 80% of the map and wins.



Play Game Sequence Diagram: Figure 4.

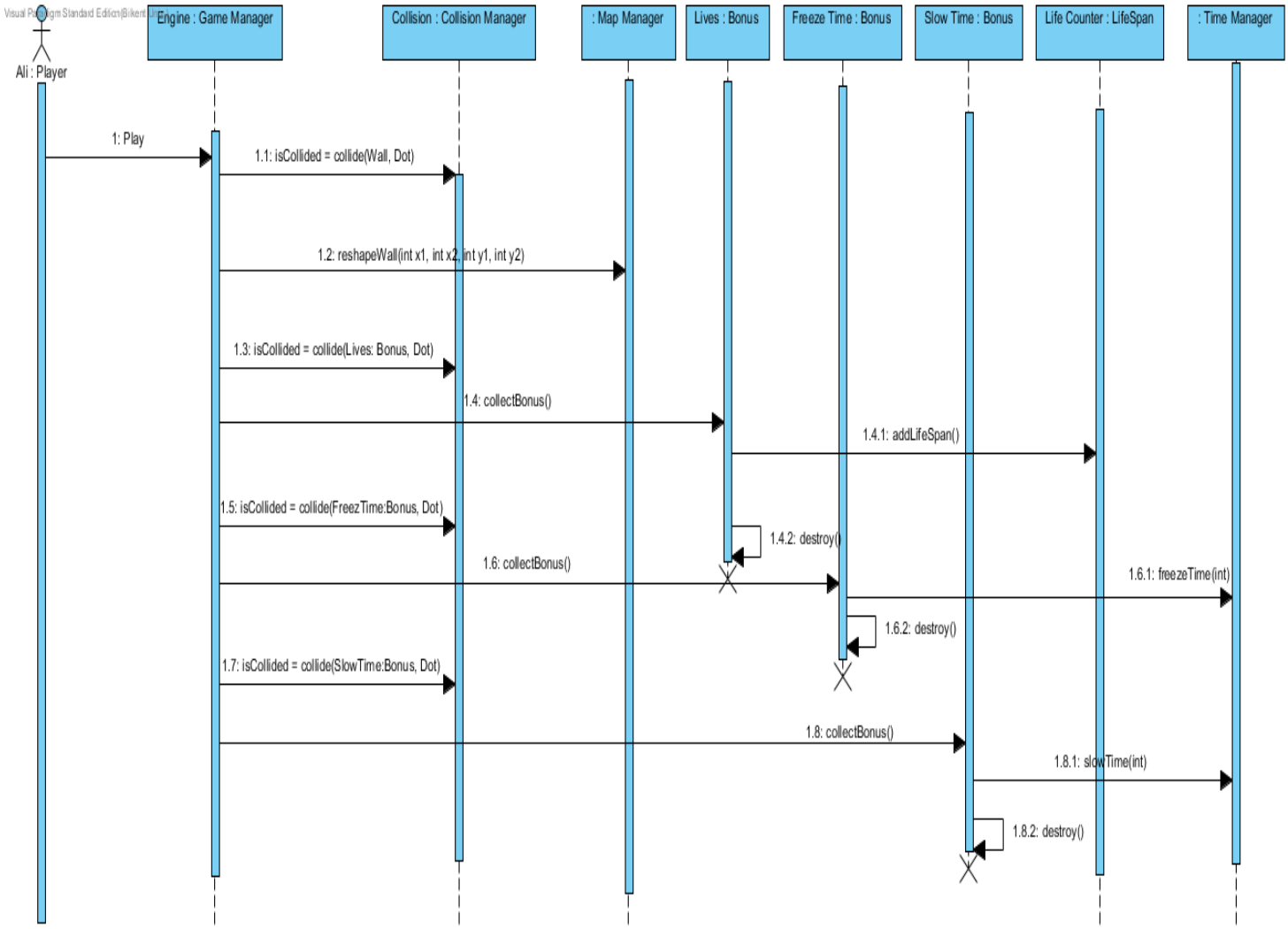
Alternative Flow of Events

Ali has lost all of his lives before surrounding the required percent of the screen to win the game. Monsters consumes all the lives and Ali loses the game.

2.7.1.3 Bonus Managing

Scenario: Player Ali plays the game. System creates the bonuses and places them onto the map on initializing. Ali wants to collect a bonus and covers it into the wall.

System recognizes if the bonus is covered or not. Then system gives the dot, player, the corresponding bonus. Ali plays with the bonus until he loses his life or finishes the level.

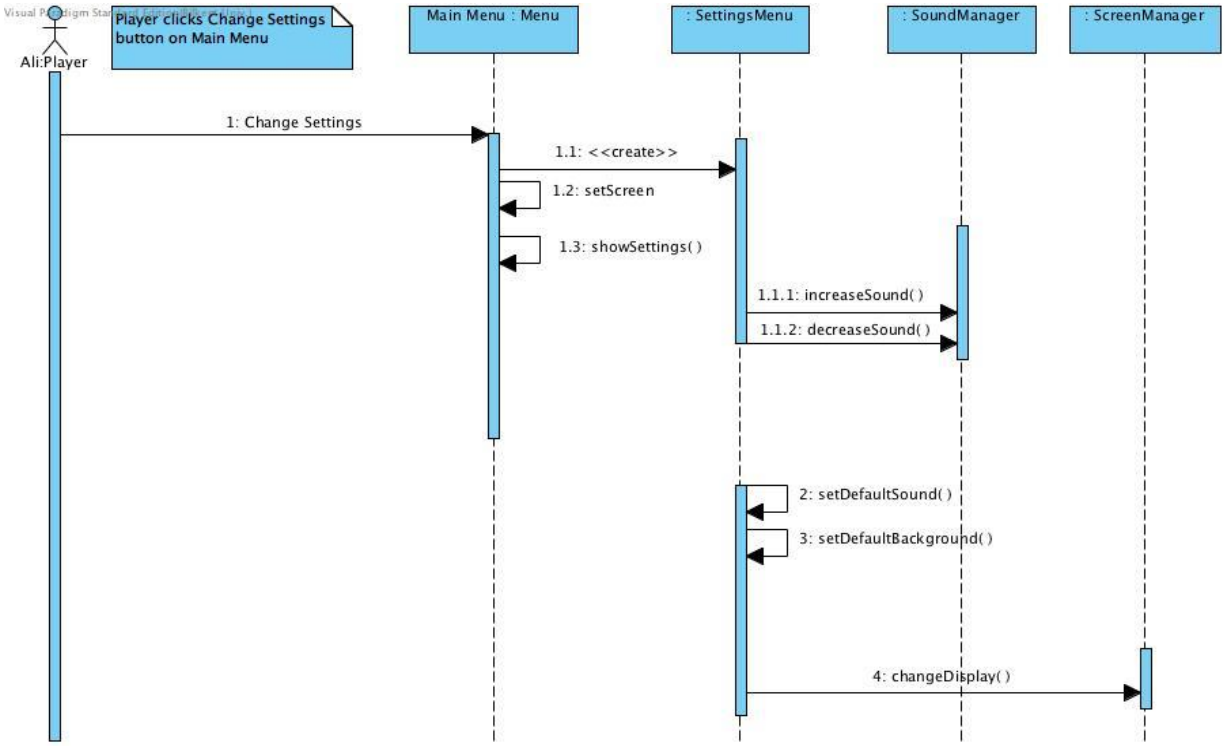


Bonus Managing Sequence Diagram: Figure 5.

2.7.1.4 Change Settings

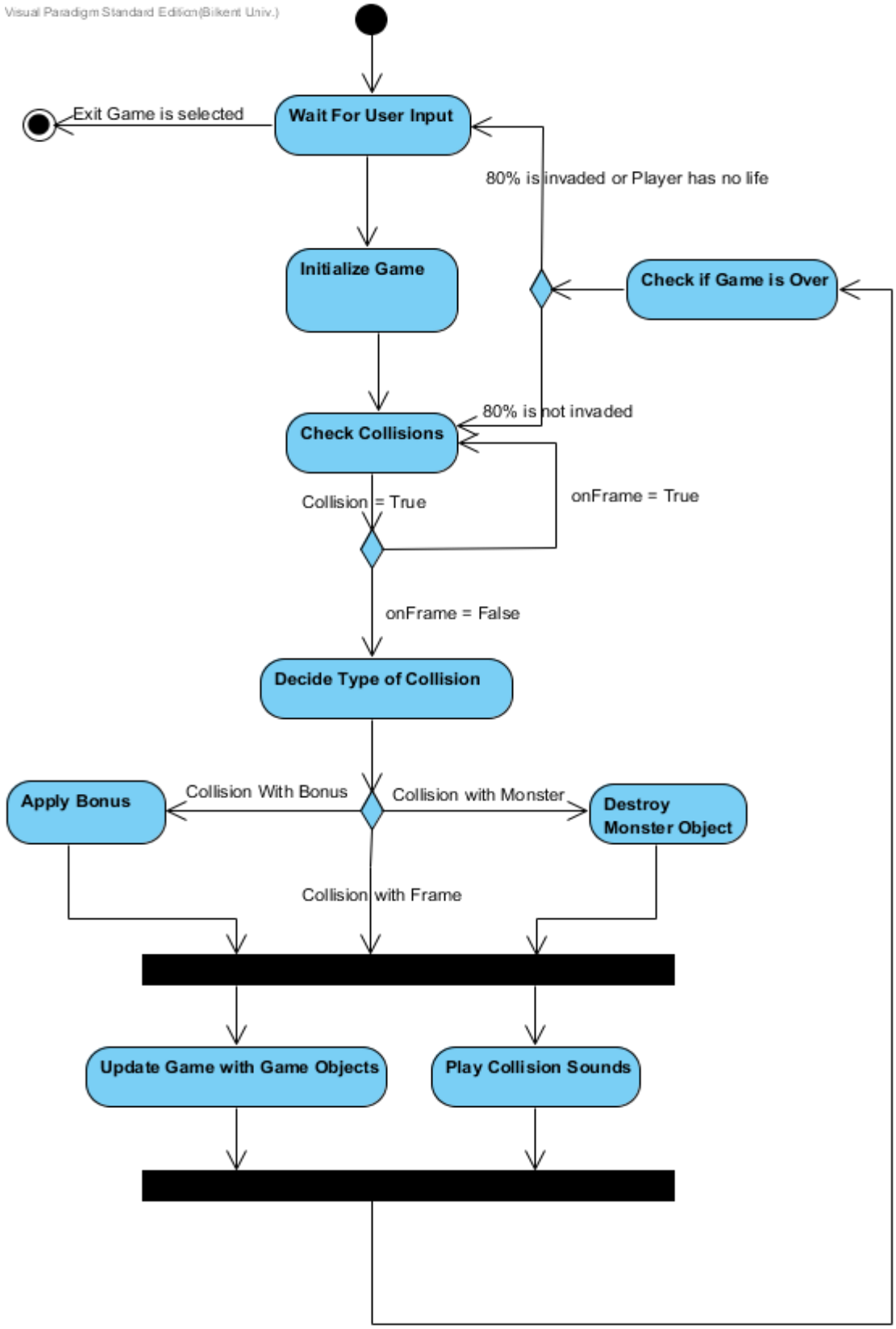
Scenario:

Player Ali wants to change the game settings.When he presses the “Change Settings” button he sees features given to him that he can change. Furthermore, when Ali changes the settings of the sound and makes them default as he wanted when he presses the back button System updates the default settings.



Change Settings Sequence Diagram: Figure 6.

2.7.2 Activity Diagram



Activity Diagram: Figure 7

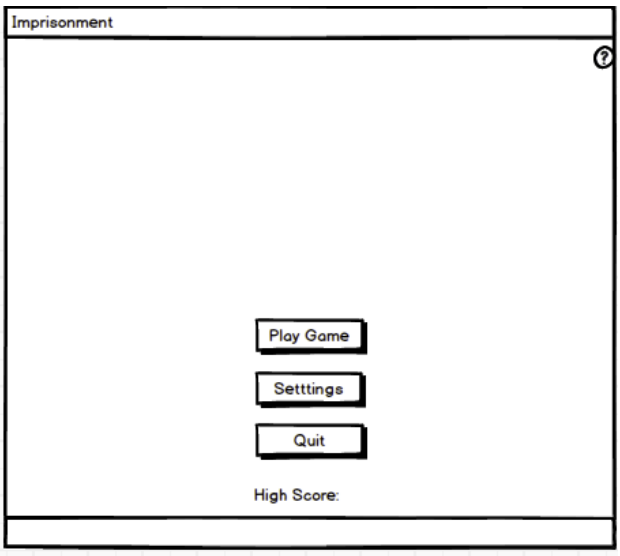
When the player selects *Play Game* option, system will initialize map by instantiating game objects such as *Dot, Monsters, Bonuses* and it will load the pre-selected map.

- When the preparations are finished, the system will wait for user input to start Dot's movement.
- The system will check onFrame, if *Dot* is on the outer frame, *onFrame* will be TRUE and all the incoming collisions will be ignored.
- If onFrame is False and *Dot* collides with a monster, the system will make a decision to end the game or to decrease Life.
- If also there is no life, the system will end the game, showing the score.
- If the player has life, the system will check whether there is a collision. When a collision is detected, the system will decide the next operation to perform according to the type of collision.
- If it is a collision after onFrame=0, the player will invade the surrounding area.
- The system will check (%) of the invaded land after every collision, if the area invaded is more than or equal to 80, the game is successful and player wins.
- If it is a collision with a bonus, the system will apply bonus features to the game objects.
- The system will accordingly update the map objects after collisions and play collision sounds according to the type of collision.
- After every collision, system will check whether it is a collision with a monster and if the player has life.
- If there are lives left, the system will stay inside the game loop and look for next collision.

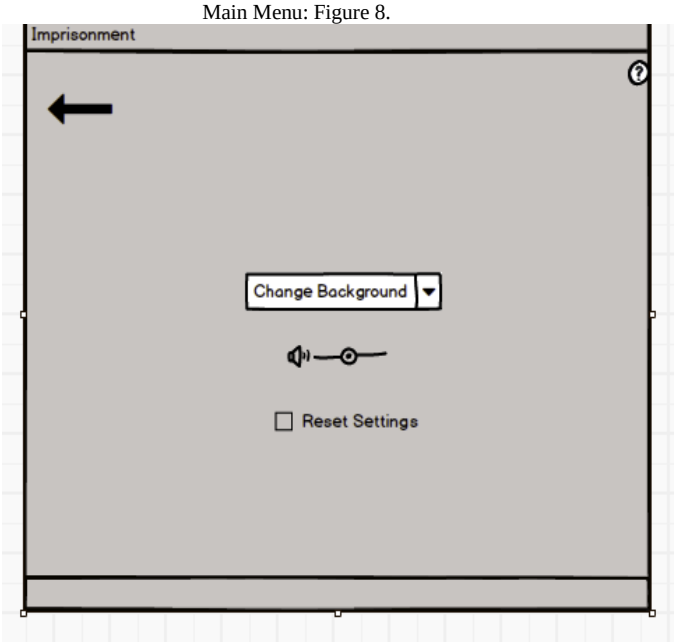
- The system will check (%) of the invaded land after every collision, if the area invaded is more than or equal to 80, the game is successful and player wins.

3. User Interface

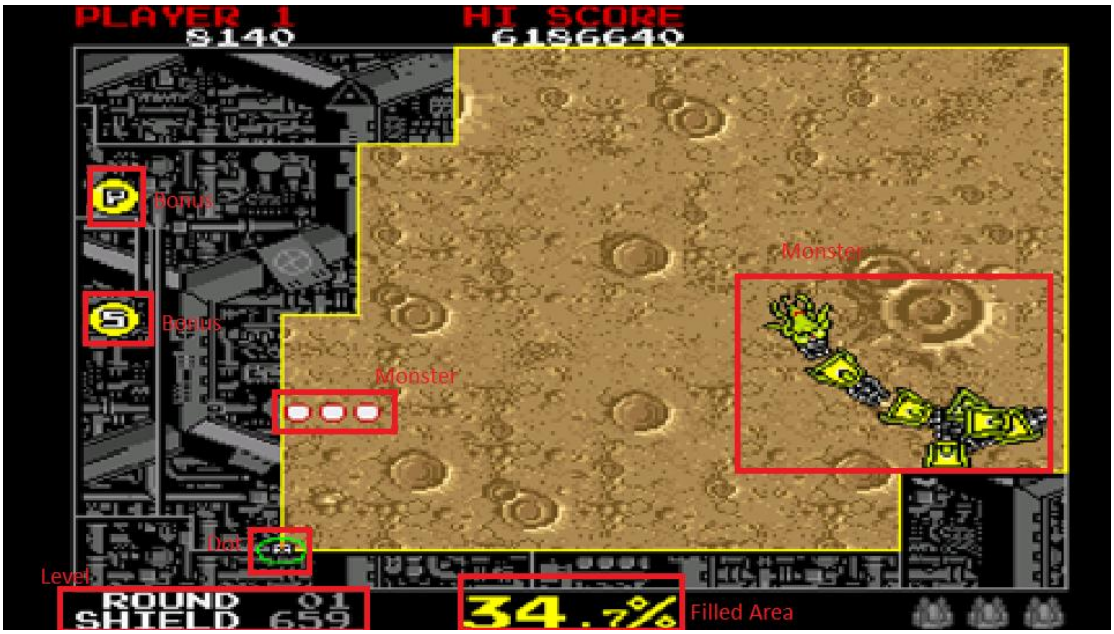
3.1 Screen Mock-ups



Main Menu: Figure 8.



Change Settings: Figure 9.



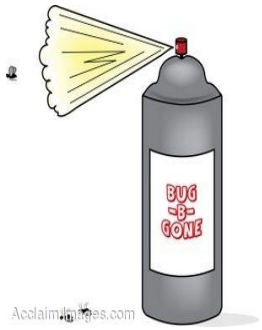
In-Game Screenshot (https://www.youtube.com/watch?v=vlYYXHICUis): Figure 10.

3.2 Bonuses


Freeze Time: Figure 11.


Lives: Figure 12.

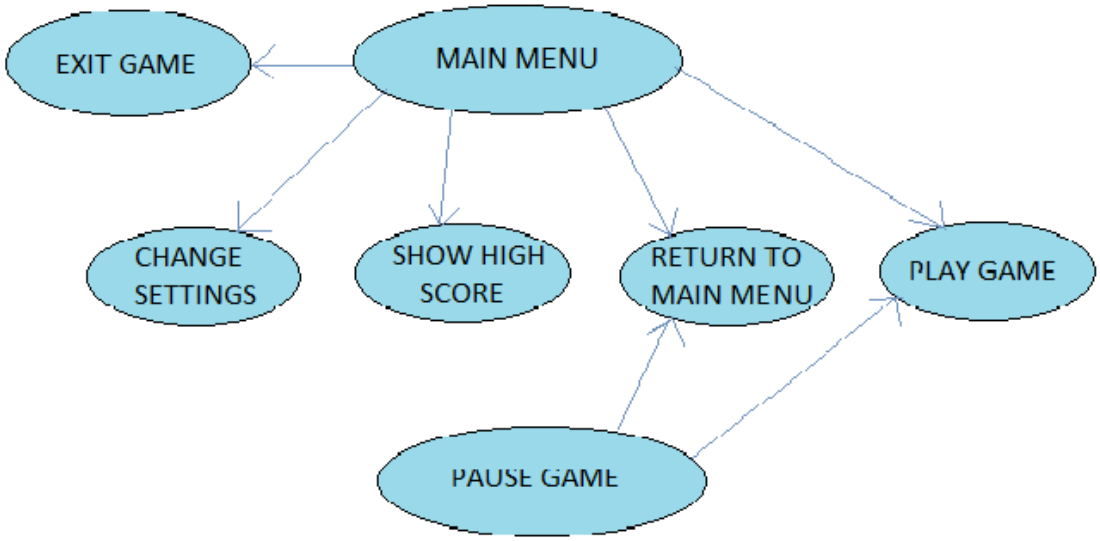

Slow Time: Figure 13.



Monster Destroyer: Figure 14.

Map Shrinker: Figure 15.

3.3 Navigational Path



Navigational Path: Figure 16.

4. CONCLUSION

In this report, in order to design and implement arcade game called ***Imprisonment***, we have analyzed requirements and system models and created an Analysis Report.

This Analysis Report consists of two main sections, which are;

1. **Requirement Specification**
2. **System Models**

- For the first section, **Requirement Specification**, we tried to think of all the possible actions that can be done by the player and all the possible requirements which a player can perform in a *Volfied* like arcade game. These were the main foundation of our thoughts for both functional and non-functional requirements. In Design Report, we are looking forward to fulfill all the requirements we have analyzed in this Analysis Report
- We have spent a considerable time trying to think of requirement specification, since it is one of the most important parts of designing an Object Oriented Program. After specifying all the requirements that a player can expect from this game, we began to draw our attention System Requirements.

As first section of the project, we decided the use cases by determining the requirements, actions and uses that should be covered in the project for the needs of the players and by considering the game features.

Our Dynamic model includes sequence diagrams and activity diagrams. In the 4 sequence diagrams that we have, we tried to demonstrate the possible interactions

between the system of the game and the player, from the scenarios that had been illustrated in the use case part. Our activity diagram indicates game play of our implementation. It stands for the actions of the game objects and components.

Our class diagram, represents the implementation of real life world onto computer world. Thus by assimilating the importance of class diagram we spent so much time and effort to think the possible classes and the interactions between them.

The last but not least part of our analysis report is user interface and navigational path diagram. By aiming a user friendly interface in our game we used the mock-ups simple and fancy as much as we can. Additionally, by using our use cases, we created a navigational path.

References

1. http://www.abandonia.com/en/games/811
2. http://farmville.wikia.com/wiki/File:Baby_Turtle-icon.png
3. http://www.donaldrussell.com/benefits-of-shock-freezing
4. http://www.iconarchive.com/tag/lifesaver
5. http://worldartsme.com/bug-spray-clipart.html#gal_post_11408_bug-spray-clipart-1.jpg
6. http://tr.depositphotos.com/2149756/stock-illustration-cute-vacuum-cleaner.html