  
  
  
  
CS 319 - Object-Oriented Software Engineering  
Analysis Report  
**SURVIVOR**  
Group 2-G  
Metehan Kaya

Mehmet Taha Çetin  
Ferhat Karaca  
Emre Yiğit Kuzhan

1. **Introduction**

We decided to design and implement a game named Survivor that based on reaching destination point in shortest time period. Actually, we are influenced from Mario which was very popular game a long time ago. But we decided to design many different properties from Mario like environmental obstacles.

This report aims to explain and show the game’s elements, rules and requirements. Also, it contains System model.

1. **Game Overview**

The game flow basically goes to finish a racecourse without dying. We will have 4 different environments; Earth, Ocean, Atmosphere and Space. In this case, speed of player will be affected from the environment. The Game’s environments include different obstacles which belong that environments in real life. For example, in space there will be no gravity therefore player should use keyboard more sensitively. Player will have lives that he/she can use for more game. Some obstacles can kill player such as animals in earth. In such a situation player will lose game. Player will have oil tank that will help to go further and if player cannot reload oil tank player will lose the game again.

During game, user can stop the game with pause menu. At the end of environmental part of game, if player finishes up the environment by coming destination point, user will open next environment which is in order like Earth, Ocean, Atmosphere, and Space. User has to be successful in all these environments to finish the game.

* 1. **Gameplay**

User will use →, ↓, ↑, ← on keyboard to play game. For the pause menu, ESC button on keyboard will work. When user lose 3 life game will ask for another game, therefore user will use mouse out of game, just in menu options (end of game or pause).

* 1. **Levels**

There will be 5 different levels which are different environments. The level system comes from their hardness of obstacles environment to environment.

* 1. **List of Components**
     1. **Components on Earth**

Harmless components: small animal (mosquito, bee, etc.), gross, quicksand…

Dangerous components: big animal (dog, wolf, etc.) …

* + 1. **Components in Ocean**

Harmless components: small fish, stream, moss, waste…

Dangerous components: big fish, lava…

* + 1. **Components in Atmosphere**

Harmless components: little bird, wind, parachutist…

Dangerous components: plane, bigger bird, tornado…

* + 1. **Components in Space**

Harmless components: low gravity, portal, electric and magnetic fields…

Dangerous components: alien…

* 1. **Character**

Character’s appearance changes according to environment.

1. **Requirements**
   1. **Functional Requirements**

* User will be using character by →, ↓, ↑, ←.
* User can understand game rules by option ‘help’.
* Pause menu can stop the game.
* Game will save itself user finishes up the level or exited from the game.
* Player has to finish a level within a limited sources/time.

**3.2 Non-functional Requirements**

**3.2.1. Extensibility**

Survivor’s source code should be arranged in way that it should allow to changes and adding new features. For instance we may change the obstacles in a specific map or we may add different sounds into the each map in order to make Survivor more enjoyable. These cases should be in consideration during the development process of the game.

**3.2.2 Ease of Use**

Survivor should be easily learn and used by user. If user encounter with a problem during the game, he/she should get information about the Survivor. Therefore, it should be an option that contains basic information about game. Survivor should include the “how to play” option in order to provide user ease of use. The main menu should be used with mouse and game controls should be direction buttons of the keyboard like most of other games on market.

**3.2.3 Graphics**

Visual element of game, such as obstacles and players’ icon should be easily distinguishable from the user.

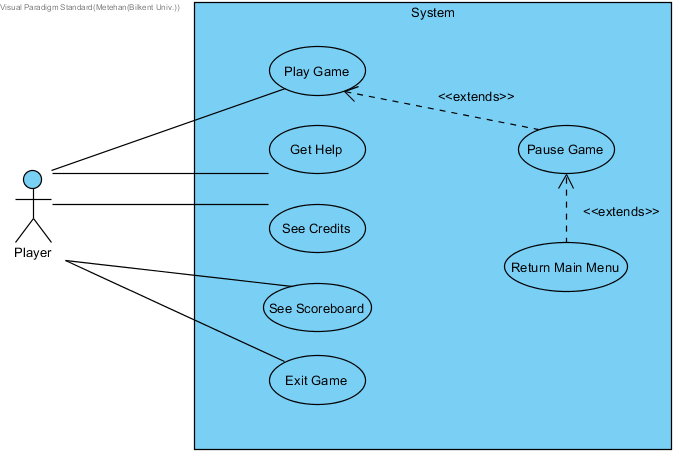
* + 1. **Performance**

Since the game requires to sensitive control because of the side effects of environments, Survivor should be response the sensitive control in order to run way from the obstacles. Response time from the keyboard is considerably important because the player would not be expecting any delay after pressing direction buttons of keyboard. Otherwise, player may crash with the obstacles.

**3.3** **Pseudo Functional Requirements**

* Game will be implemented in Java.
* Eclipse will be used for IDE.
* Some images will be created in Photoshop CS4.
* Map Editor is going to help us for 4 environments.

1. **System Models**
   1. **Use Case Model**



**<<Use Case #1>>**

Use Case name: Play Game

Participating Actors: Player

Entry Condition: Player opened the game and is main menu

* Exit Condition

1. Player has completed all the environmental levels successfully or
2. Player lost all of its life points basically it is oil tank or
3. Player choose return main menu option in pause menu

* Main Flow events

1. Player chooses ‘start game’ option on main menu
2. System creates environmental level
3. Player plays the game and finishes successfully all of the levels
4. If player finishing time period is appropriate for high score table, game asks for name
5. Player enter the name
6. Player returns menu or replay game

* Alternative Flow of events

→If user’s time is not enough for top scores player return main menu or replay game directly without asking name

→Any time player can exit to main menu

→If player run out of oil or killed by an obstacle game will Show message in step 6.

**<<Use Case #2>>**

Use Case name: Pause Menu

Participating Actors: Player

Entry Condition: Player is in game

* Exit Condition:

→Player choose to return main menu option or

→Player return the game

* Main Flow events

1. Player press to ESC
2. Game stops
3. Player returns to game by pressing ESC key again

* Alternative Flow of events

→Player can choose to return main menu

**<<Use Case #3>>**

Use Case name: Return Main Menu

Participating Actors: Player

Entry Condition: Player is in pause menu

* Main Flow events

1. User clicked return main menu option on Pause menu
2. Game save this level condition to computer
3. Game return to main menu

**<<Use Case #4>>**

Use Case name: Get Help

Participating Actors: Player

Entry Condition: User is in main menu

* Exit Condition:

1. User user press ‘return main menu’ button

* Main Flow events

1. Player choose help option on main menu
2. Game shows game rules and how to play
3. Player returns to main menu

**<<Use Case #5>>**

Use Case name: See Scoreboard

Participating Actors: Player

Entry Condition: User is in menu

* Exit Condition:

1- User press ‘return main menu’ button

* Main Flow events

1. Player choose to high scores option in main menu
2. Game shows high scores that are done before which are created in a file before
3. Player returns to main menu

**<<Use Case #6 >>**

Use Case name: Exit

Participating Actors: Player

Entry Condition: User is in menu

* Exit Condition:

1. User press ‘yes’ or ‘no’ button

* Main Flow events

1. Player choose to exit from the game
2. Game asks for sureness of player
3. Game is closed after yes option

* Alternative Flow of events

→ If user says no game returns to main menu

**<<Use Case #7>>**

Use Case name: See Credits

Participating Actors: Player

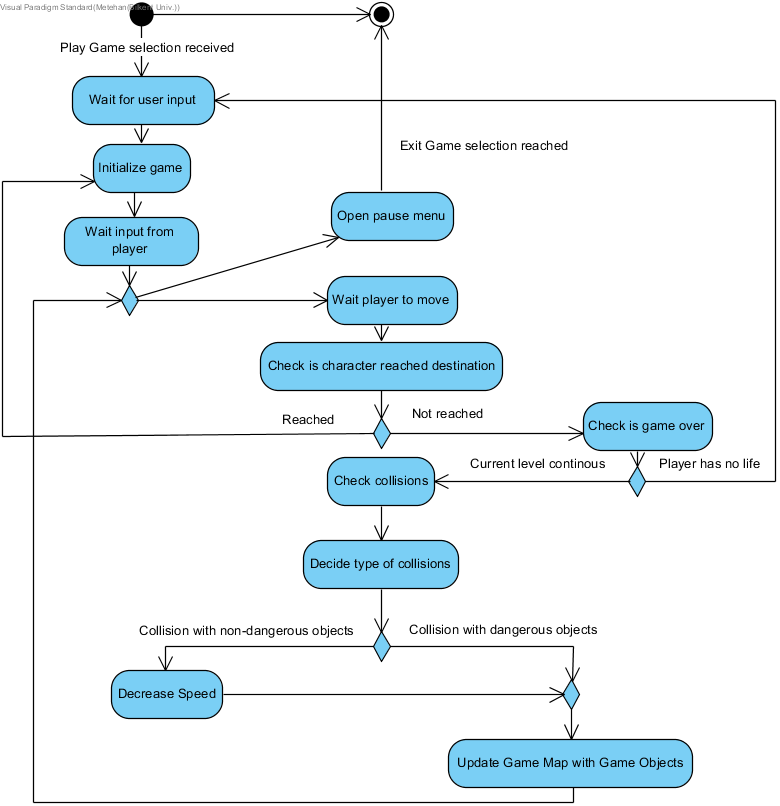
Entry Condition: User is in menu

* Exit Condition:

1. User press ‘return main menu’ button

* Main Flow events

1. Player choose to ‘credits’ option
2. Game shows its credits
3. Player returns to main menu by pressing ‘back’ button.
   1. **Dynamic Models**
      1. **Activity Diagram**



Activity diagram shows the main flow of gameplay. For each movement of player, first, it checks if character reached the destination point. Then, it checks for any collisions which have 2 kinds. According to these, it makes some changes like decreasing speed or losing a live.

**4.2.2 Sequence Diagrams**

**Scenario1- Play Game**

Alper wants to play the game and opens the game.

He selects play game option.

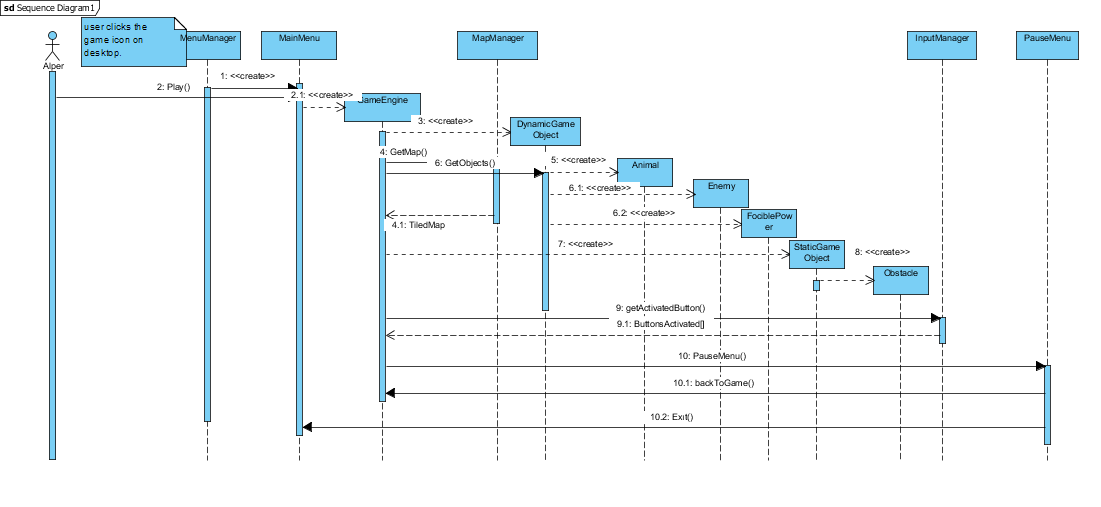
He plays game and wants to stop playing for 5 seconds.

He presses esc button and goes into pause menu.

He wants to keep going on his game and chooses continue option

After a while, he wants to exit from the game and presses esc button.

Then chooses exit from game option and come back to the main menu.



**Scenario2- View Credits**

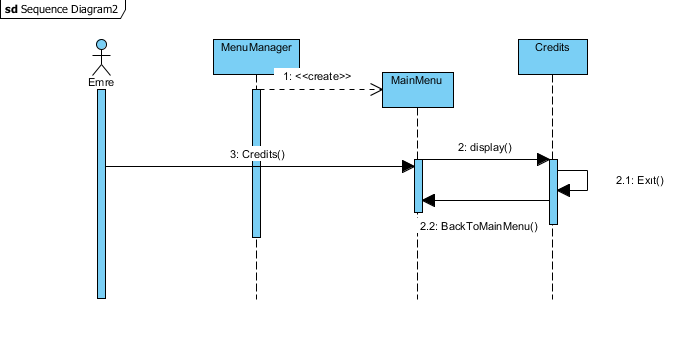
Emre wants to see the this awesome game’s credits.

He opens the Game.

He chooses Credits Option in the screen of main menu.

He reads and learn credits and chooses exit option.

He finally come to main menu.

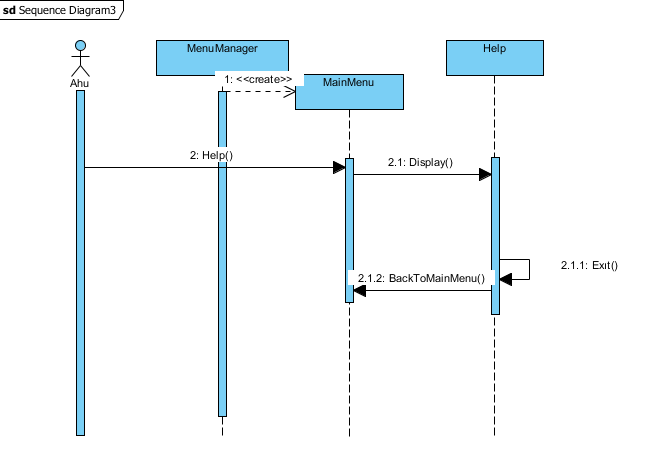


**Scenario3-View Help**

Ahu wants to learn how to play this game.

She enters the game.

She chooses Help option and gets into help screen.

She chooses exit and comes back to main men

**Scenario4-Show High Scores**

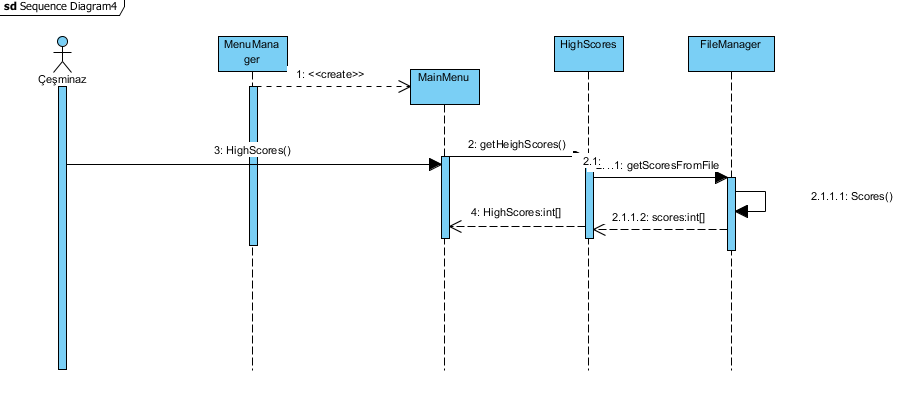
Çeşminaz wants to see best scores done in her computer before.

She enters the game.

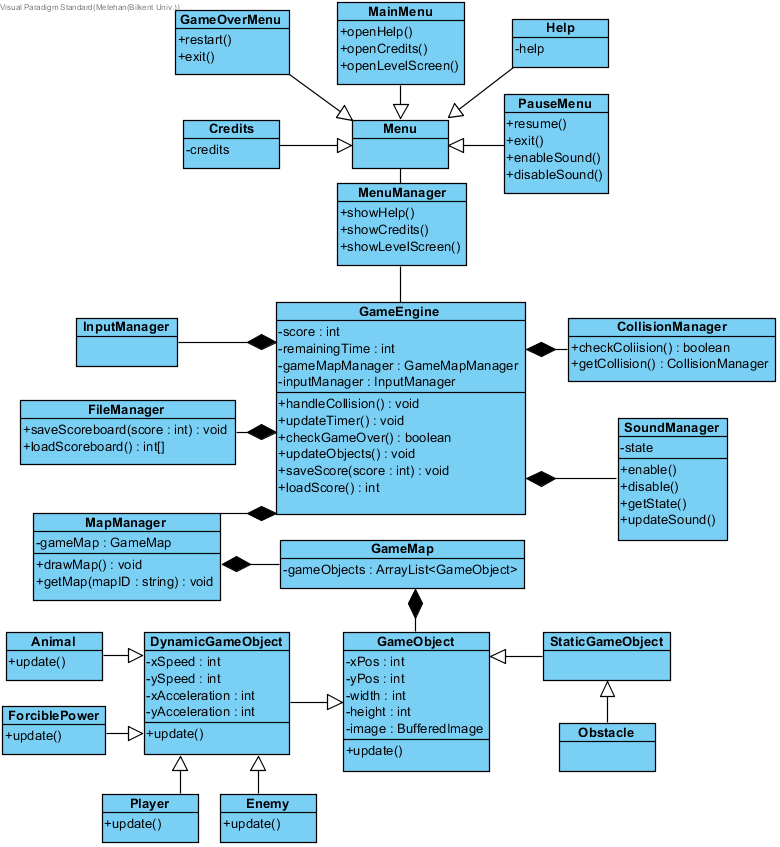
She chooses the heigh scores option on main menu.

She learns high scores and presses back button.

She comes back to main menu

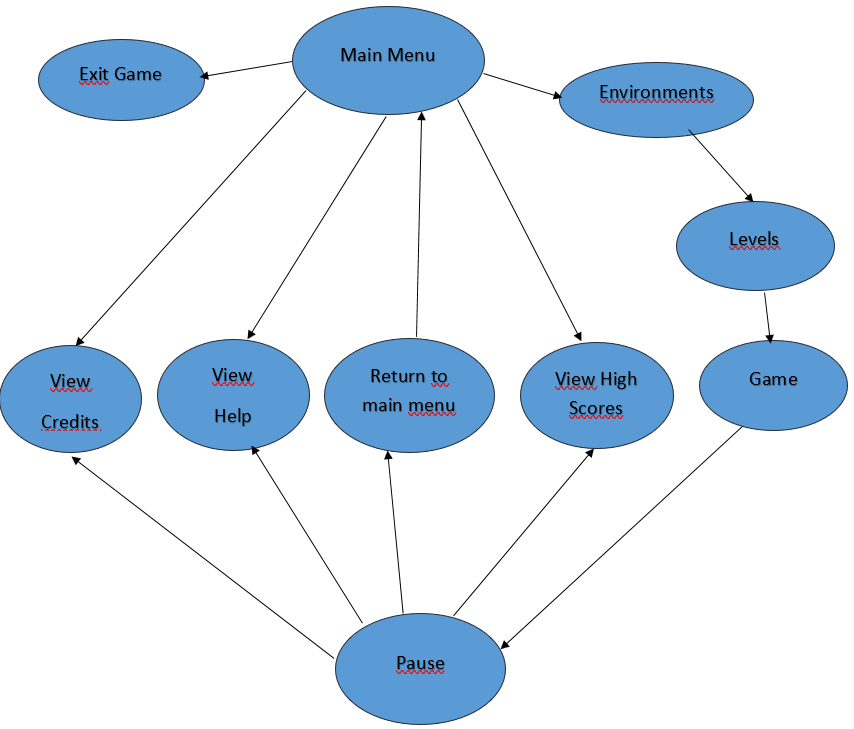


**4.3 Object and Class Models**

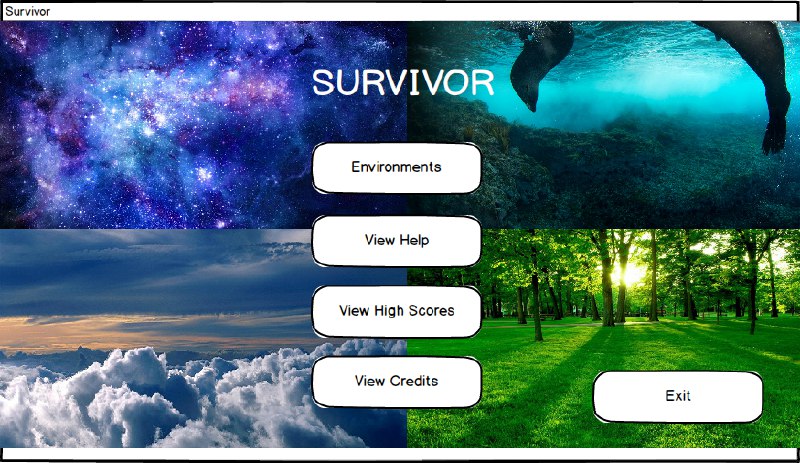


In project’s design, “MainMenu” is the first class that will be constructed by the main method. At the centre of the design, “GameEngine” deals with all complexity. Before this class, “Environments” and “MapLevel” are constructed. Also, “FileManager” that is for scoreboard and save files, “InputManager” which is for getting input, and all classes related to components are linked to “GameEngine”.

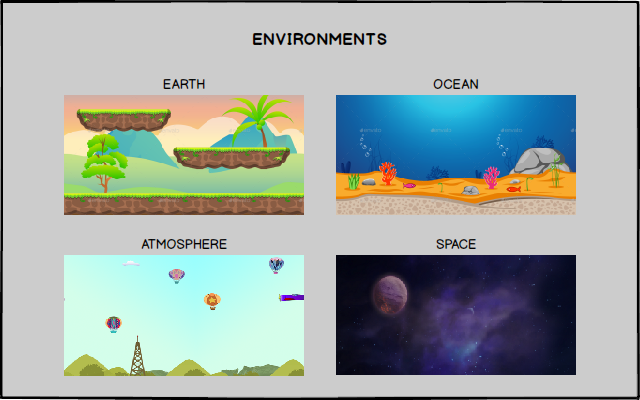
* 1. **User Interface – Navigational Paths and Screen Mock-ups**



**Main Menu**: When user runs the game, this menu will appear. In this menu the user can choose environments, to view help, to view high scores, to view credits, to exit game.



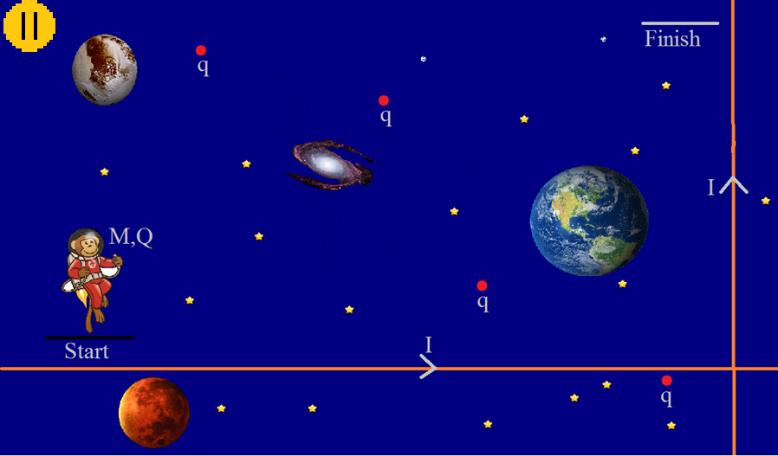
**Environments**: Four environments (earth, ocean, atmosphere and space) are shown if “Environments” button is pressed.



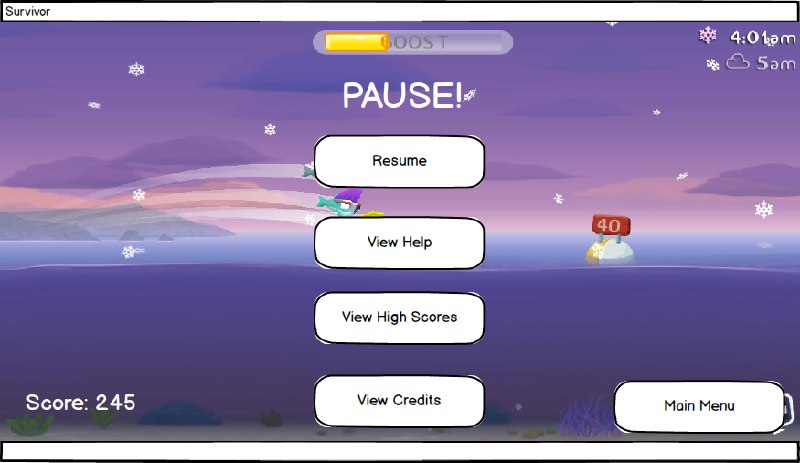
**Levels**: A map of levels are shown when any environment is chosen.



**Game**: When any levels is chosen, game starts.



**Pause Menu**: When the player types “ESC” on keyboard, pause menu will appear. This menu contains credits, help, high scores, resuming the game and an option to return the main menu.



**View Help**: If the user chooses help, some useful contents about the game and instructions are shown.



**View High Scores**: When “High Scores” button is pressed, game will show the user scoreboard which contains top 10 scores with player names. Initially, the scoreboard has 10 player names and some smooth numbers as scores.



**View Credits**: If the player selects “Credits” button, team id and names of project developers will be shown.



**Exit Game**: If the user chooses “Exit Game” button, program terminates.

1. **Conclusion**

In this report, we prepared our analysis report in order to design and implement an arcade type game called as Survivor. This report contains two main part. First part related with requirement specification. Second part consist of system model of the project.  
For the first part, we tried to examine possible requirements which a player could perform in a Survivor game. In this project design we plan to meet all these requirements. After we determine the requirement specification, we were ready for the system model  
Second part of the analyse report related with system model. This part formed the main part of our report. System Model includes four subsections. These are use case model, dynamic models, class model, user interface.

The dynamic model of the project contains activity diagrams. This diagram shows our game play basically. It has activities of user and the obstacles. In our class diagram, we tried to create in the best way since we will implement the project accordingly. We spent some time thing possible relations of classes.

Fourth part of analyse report is user interface and navigational path diagram. We tried to show this part through using mock-ups and charts. We formed a navigational path using our use cases

To sum up, we tried to create good analyse report in order to guide us in the implementation process.