
Bilkent University

Department of Computer Engineering

# Object Oriented Software Engineering Project

*Crossit!*

## Analysis Report

Umutcan Aşutlu, Sarp Saatçıoğlu, Figali Taho, Utku Uçkun

Supervisor: Uğur Doğrusöz

Oct 29, 2016

# Table of Contents

# Table of Figures

# Analysis Report

*Crossit!*

# 1. Introduction

The project we are going to develop is a game we decided to name “Crossit!”. It is a Frogger type of single player arcade game in which the player tries to successfully cross roads where vehicles pass continuously without getting hit [1]. It will be a desktop based game.

The game will generate infinitely many stages and each stage will be more difficult than the previous one. Player will try to go as far as possible. Player’s score will be stored locally.

In this report, we are going to give a general overview of our project, the gameplay and the objects of the game. The functional, non-functional requirements, some scenarios and the use cases of the game will be described. Following that we are going to describe the game objects and class diagram, some sequence diagrams of common scenarios, state diagram of player and activity diagram of the system, as well as talk about and illustrate the user interface of the project.

# 2. Proposed system

## 2.1 Overview

### 2.1.1 General explanation of the game

In this section we are going to provide a general outline of the gameplay. Next, we are going to describe the levelling up and scoring system of the game, as well as the objects that will “flavour up” *Crossit!*.

*Cross It!* is a single player arcade game in which the player crosses a road in which different types of vehicles pass. In our implementation of game we will have features that include variety in power-ups and 3 kind of vehicle that are bus, car and motorcycle. The player will be granted with 3 lives at the start of the game. Player will try to finish the current stage that he is in. Each stage will have 10 lines in which the different objects will be placed randomly according to the stage number. Each stage will get more and more difficult in terms of what vehicles in the road will be. Amongst the objects included which will be discussed with more details in the following sections, there will be coins, which the user can collect which he/she can later use to purchase items, i.e different outfit or hats which can be used for changing the look of the player.

The game will be controlled through the keyboard buttons up, down, left, right. M button of keyboard will mute the game sound. At any time while player is playing the game, he/she will be able to check his/her current score. Player can also view the highest scores won on that machine in the high score page. The theme of the game can be changed from the settings in the menu. There will be two themes,

a white and a dark theme. Moreover, the user will be able increase or decrease the sound of the game from the settings. If player is confused he/she can check help documents to learn about how to play the game and content of the game.

### 2.1.2 Game progress

After passing all the roads in a stage, the player will enter the next stage which will be harder than previous one. The difficulty will be set by increasing the speed of vehicles. In each stage there will be more vehicles on the lines so it will be more difficult to pass the roads. In every stage there will be 2 coins and 1 mystery box placed randomly. When the player gets hits by any kind of vehicle, player will be respawn at start of the current stage and lose one health point. If player runs out of health points the game will be over. The player will be asked to save the current score he/she made by entering his/her name, and then he/she will be prompted to play again or go to main menu.

### 2.1.3 Scoring system

Calculation of the score will depend on the stage number and time it takes the player to finish the stage. If a stage is completed in less than 30 seconds, *time_bonus* will be 100 points, if it is completed in less than 1 minute, *time_bonus* will be 50 points and if it is completed in more than 1 minute, the *time_bonus* will be 25 points. Total score will be calculated by summing every stage score which is going to be computed as *stage_number* * *time_bonus*.

### 2.1.4 Collectibles or mystery boxes

During the gameplay, in every stage there will be some random collectibles which will contain random effects. Player will not be able to learn whether it is a power-up or a power-down until he/she picks it. The picking action will be done by walking top of the box on the road. The chance of getting positive effect will be higher than getting a negative, to make it more fun and motivate the player to pick up the mystery boxes.

**- Slow the Traffic:** The speeds of vehicles will be slower during that stage.

**- Shield:** In duration of shield effect, player will not lose health point for getting hit by a vehicle just for one time and then shield will disappear.

**- Extra Life:** Player will immediately gain one health point.

**- Magnet:** During the magnet effect all the coins which are in that stage will be collected immediately.

**- Teleportation:** Teleportation will automatically send player to end of the stage.

**- Faster Traffic:** This effect will increase the speeds of vehicles during the stage.

**- Reverse Control:** Controls will be reversed for 10 seconds, e. g. when player hits the key to going to forward direction will be resulted with going backward.

**- One less life:** Player will immediately lose one health point.

### 2.1.5 Vehicles

To make the game more interesting and challenging three types of vehicles will be used. At all moments during the game, all vehicles will have the same speed.

**- Motorcycle:** Motorcycles will have one unit length and one unit width in roads.

**- Car:** Cars will have two units length and one unit width in roads.

**- Truck:** Trucks will have three units length and one unit width in roads.

### 2.1.6 Hats

Player can equip his/her character with different hats for visual variety.

## 2.2 Functional Requirements

### 2.2.1 Play Game

After player opens the application he/she will be able to start the game and play it.

### 2.2.2 Change Settings

Player will be able to change settings of the game like setting the sound level. Player will be able to change the look of his/her character. Player can also change the theme of the game.

### 2.2.3 Buy Hats

Player can purchase different hats for his/her character with the coins collected in the game.

### 2.2.4 View High Scores

Player can view the highest 20 scores achieved on that specific computer. Player can see a sorted list from highest score to lowest containing scores and name of players achieved that scores.

### 2.2.5 Save Highscore

While playing the game, if player runs out of lives and his/her current score is in top 20 high scores, he/she can save that score with his/her name to the high score list.

### 2.2.6 Play Again

While playing the game, if player runs out of lives he\she can chose to start the game from first level.

### 2.2.7 View Help

Player can view a help document to understand the game better. In help menu there will be information about mystery box effects and vehicles, and help will include basic instructions of the game.

### 2.2.8 View Credits

The user will be able to view the credits. In the credits there will be the names of the developers of the game and their contact information.

## 2.3 Non-functional Requirements

### 2.3.1 User friendly interface

The game will have user-friendly interface. Objects and things will be clear. The game will contain objects of matching color according to each of the two themes, light and dark.

### 2.3.2 Action Delay Time

The action delay time will be low and the performance and the flow of events of the game will catch the player into the game.

### 2.3.3 Game Concept

Understanding and apprehending the game will be easy. The features of the game like the mystery boxes and the vehicles will be not complex, and the user will be able to grasp these easily and play along.

## 2.4 Pseudo Requirements

- The game will be implemented in java.System Models.
- Photoshop will be used for designing the objects of the game.

# 2.5 System Models

### 2.5.1 Use-Case Model and Scenarios

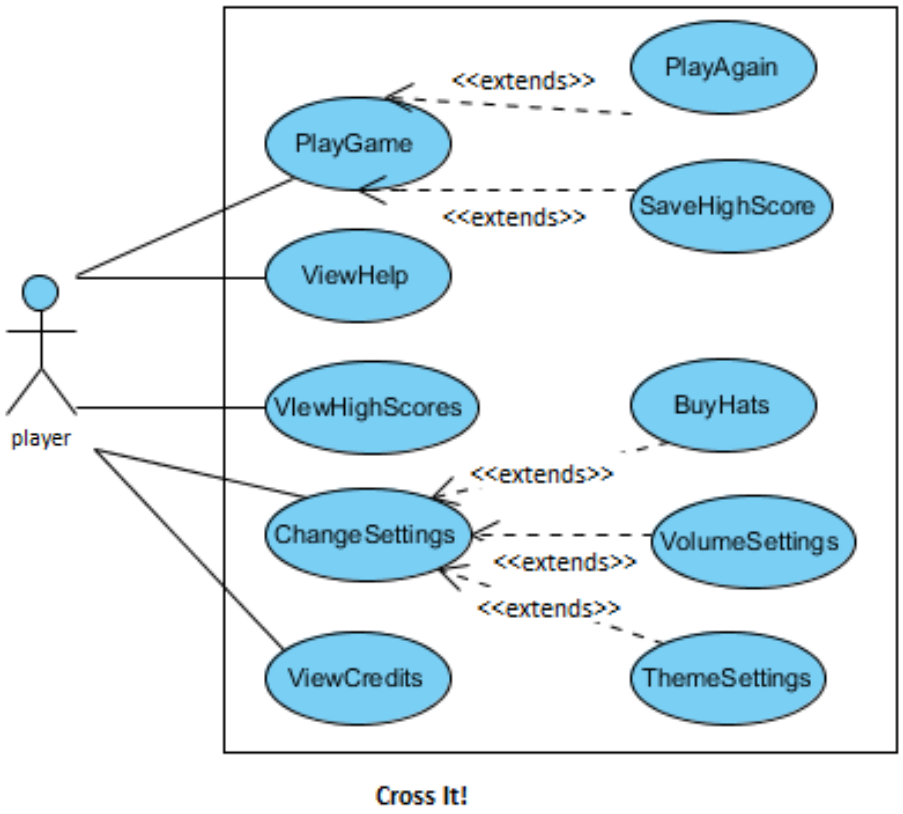


**Figure 1**. Use case diagram.

**1. PlayGame**

**Use Case Name:** PlayGame
**Primary Actor:** Player
**Stakeholders and Interests:**
- Player wants to have fun time and aims to break his record.

**Pre-conditions:** System saves the settings and implies them to game.
**Post-condition:** If player beats one of the high scores, the record will be saved.

**Entry Condition:** Player starts the game.
**Exit Condition:** Player closes the game.

**Success Scenario Event Flow:**

**1-** Player starts the game.
**2-** System will set the game according to settings.
**3-** Player goes through roads to finish-up stage while avoiding vehicles.
**4-** Player picks up some mystery boxes along the way.
**5-** System provides the objects in the road.
**6-** Player finishes the stage and goes to the next one.
**7-** System makes every stage harder than previous one.
**8-** Player eventually will get hit by a vehicle.
**9-** System reduce health point.
**10-** Player loses all his health points.
**11-** The system ends the game.
**12-** System records the score to high score list.
**13-** Player choose replay button.
**14-** System goes back to start.

**Alternative Flows:**

**1-** Not beating high score:

**1.A.** Player starts the game.
**1.B.** Player loses all of his health points before achieving a score that is higher than at least one of the high scores.
**1.C.** The game ends without recording session.

**2-** Closing during game:

**2.A.** Player starts the game.
**2.B.** While he is playing on, he closes the game.
**2.C.** System will not record anything and shut itself down.

**2. ViewHelp**

**Use Case Name:** View Help
**Primary Actor:** Player

**Stakeholders and Interests:**

- Player wants to obtain an information about general concept of the game.
- Player learns how to play the game.

**Pre-conditions:** Player should be in the game.
**Post-condition:** -

**Entry Condition:** Player selects “Help” button.
**Exit Condition:** Player selects “Back” button.

**Success Scenario Event Flow:**

**1-** Player enters the “Help” section.
**2-** System opens up help segment and shows icons and rules from the game.
**3-** Player learns how to play the game.

**4-** Player presses back button and exits.

**Alternative Flows:** -

**3. ViewHighScores**

**Use Case Name:** View High Scores
**Primary Actor:** Player

**Stakeholders and Interests:**

**-** Player wants to see recorded high scores.

**Pre-conditions:** Player should be in the game.
**Post-condition:** -

**Entry Condition:** Player selects “High Scores” button.
**Exit Condition:** Player selects “Back” button.

**Success Scenario Event Flow:**

**1-** Player presses “High Scores” button.
**2-** System opens up high scores segment.
**3-** Player looks at highscores that have been achieved previously.
**4-** Player presses back button and exits.

**Alternative Flows:**

**1-** If there is no high score:

**1.A.** System returns a text that says there is no high score.
**2.B.** Player presses back button and exits.

**4. ChangeSettings**

**Use Case Name:** Credits
**Primary Actor:** Player

**Stakeholders and Interests:**

**-** Player wants to change settings of the game, like sound on/off or changing outfit.

**Pre-conditions:** System sets previous settings, if it has not been changed before, sets default settings to game.
**Post-condition**: Game settings are changed.

**Entry Condition:** Player selects “Settings” button.
**Exit Condition:** Player selects “Back” button.

**Success Scenario Event Flow:**

**1-** Player presses “Settings” button.

**2-** System opens up settings menu.
**3-** Player changes his/her hat by purchasing it with coins for his/her character.
**4-** Player changes settings of the game and presses save button.
**5-** System saves the changes.
**6-** Player presses back button and exits the setting menu.

**Alternative Flows:**

**1-** Player goes back without making any changes.
    **1.A.** Player selects "Back" button.
**2-** Player want default setting of the game:
    **2.A.** Player presses "Default" button.
    **2.B.** System sets every setting to default.
    **2.C.** Player selects "Save" button.
    **2.D.** System updates the settings.
    **2.E.** Player selects "Back" button and exits the setting menu.
**3-** Player goes settings to change hat.
    **3.A.** Player selects a hat to purchase.
    **3.B.** System checks players credit.
    **3.C.** Request of player gets deny because of insufficient credit.
    **3.D.** Player exits settings without buying anything.

**5. ViewCredits**

**Use Case Name:** View Credits
**Primary Actor:** Player

**Stakeholders and Interests:**

**-** Player wants to see credits which includes information about game and developers of the game.

**Pre-conditions:** To see credits, player must be in the game.
**Post-condition**: -

**Entry Condition:** Player selects "Credits" button.
**Exit Condition:** Player selects "Back" button.

**Success Scenario Event Flow:**

**1-** Player presses "Credits" button.
**2-** Sees all information that has been included in credits.
**3-** Credits ends
**4-** System goes back to start.

**Alternative Flows:**

**1-** Player goes back without waiting the end of credits.
    **1.A.** Player selects "Back" button.

### 2.5.2 Object and Class Model

- **MainMenu**

Main menu will be an important boundary object which help player to navigate in game.

- **Setting/Setting Manager**

Setting object will display settings where user can change game sound, theme etc. Setting manager will apply these settings and will save them to a file on client's PC.

- **2.5.2.3 HatMenu/HatManager**

HatMenu will display purchased and purchasable hats for user to interact with them. HatManager will execute the player's decisions.

- **GameEngine**

Game engine is the control object responsible from play game function of the game. It will keep track of the player's life, money, collectible's it picked up, total score, current stage and such. It is responsible for managing gameGrip.

- **GameGrid/GUI**

GameGrid is responsible for managing the gameMatrix it contains. GameMatrix is a 2 dimentional gameObject array which will hold all gameObjects such as vehicles, roads, coins, mystery boxes and player's character. GameGrip updates locations of these objects and detects collisions or mystery box pick ups. GUI is the object responsible for drawing the gameMatrix so player can display the game.

- **GameObject**

GameObject's will represent traffic in our game. Each GameObject will have a location and an image. Some GameObjects will be able to contain others (such as RoadPart can have a vehicle in it).

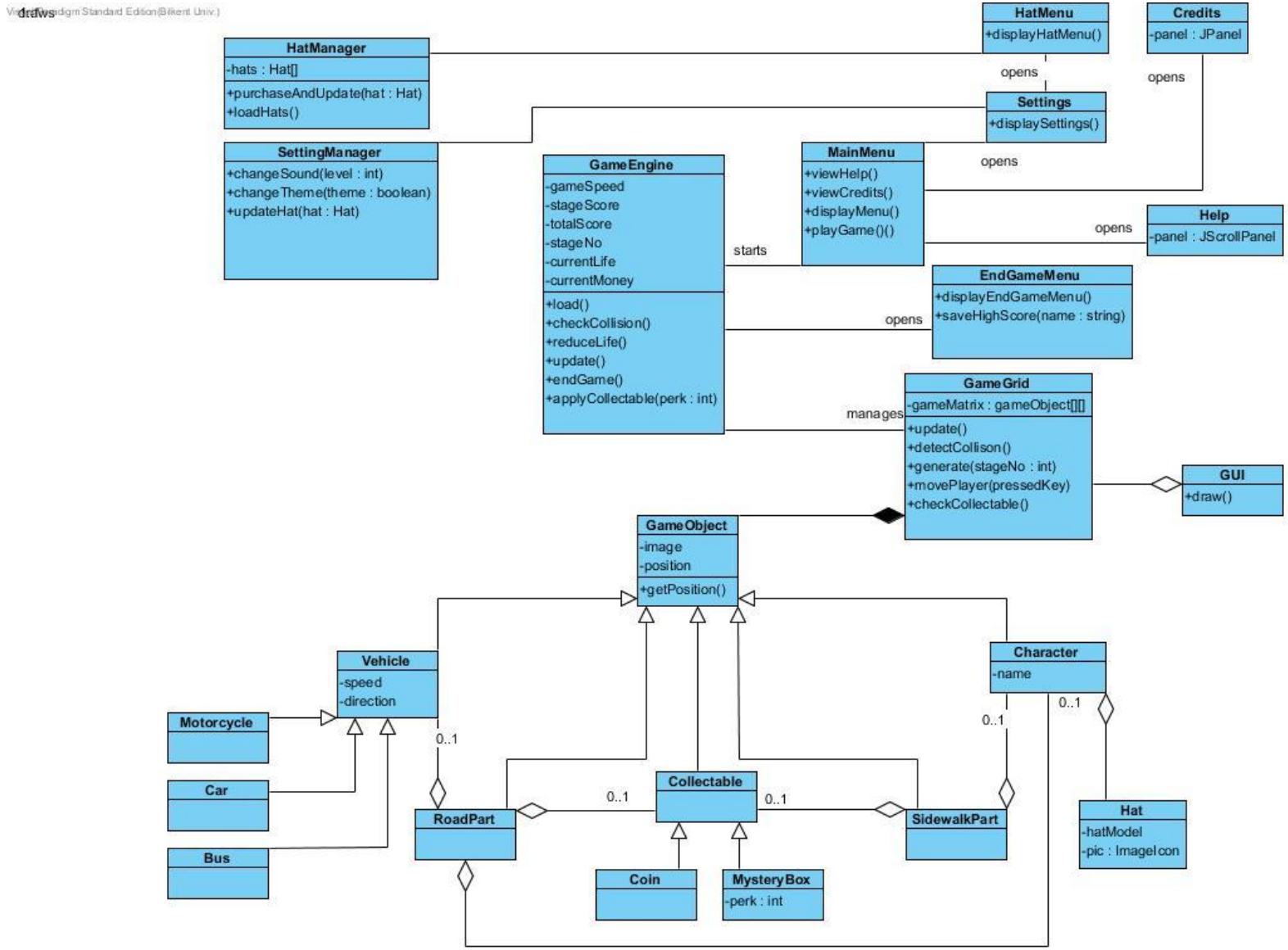


**Figure 2**: Objects and Class Diagram.

### 2.5.3 Dynamic Models

## Sequence diagrams

- **Play game and grab a magnet collectible**

  Cengiz has opened the game and interacts with the screen to start the game. The game starts at first stage. Afterwards Cengiz chooses to move up. Since his new position doesn't collide with any vehicle, but with a magnet collectable, his money amount increases and the coins in the field are not in the game field anymore. The cars keep moving.

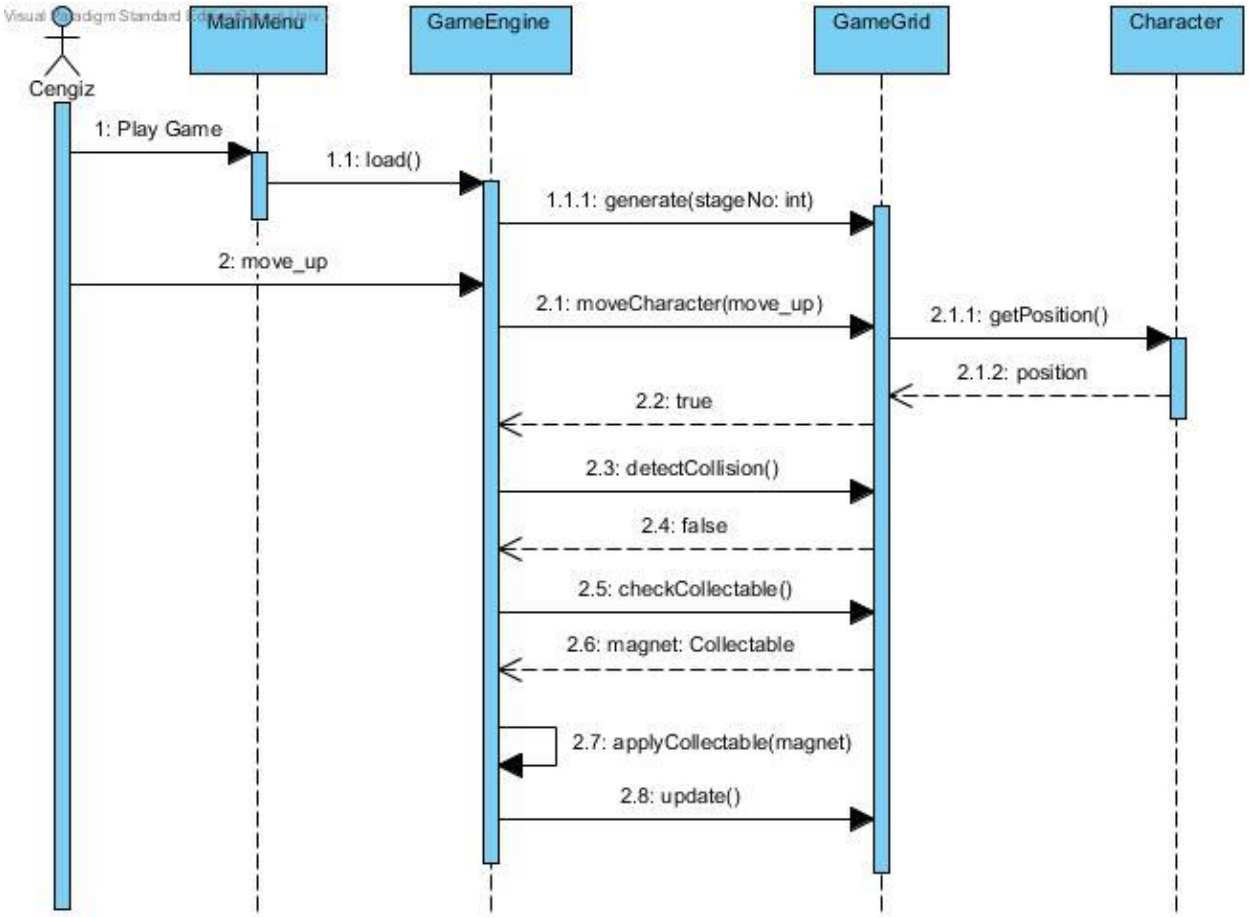


**Figure 3**: Sequence diagram for in game collection of magnet

- **Change sound, theme and hat**

Player has opened the game, and he has chosen the settings in the main menu. He then changes the volume of the game. The volume is updated. He then changes the theme of the game, which can be either dark or white. Then he decides to shop for a hat. If he has enough money he purchases the hat he chose, which is what happens in this case. The purchased hat is applied on the player and the player approves on the change.

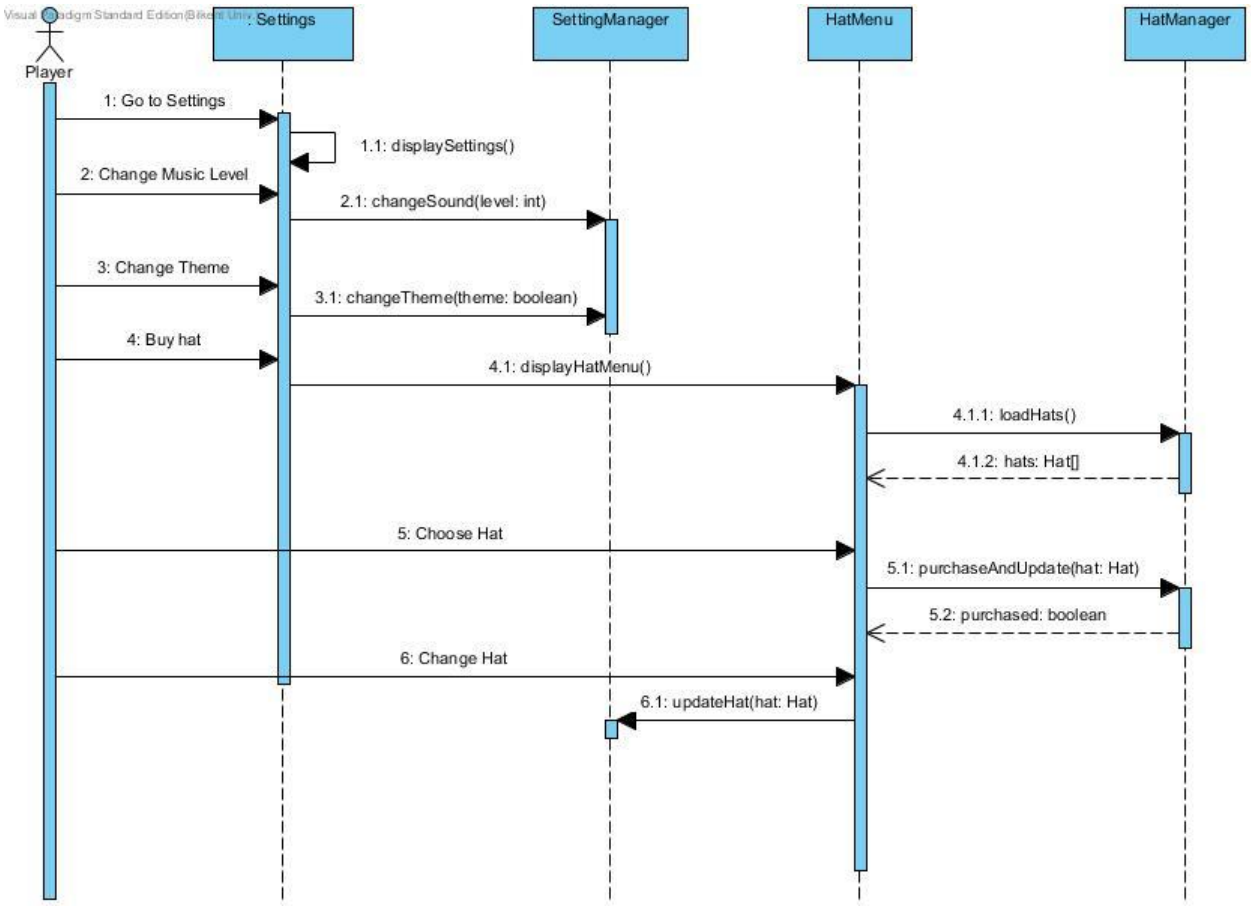


**Figure 4**: Change of settings sequence diagram

- **End game with no lives left**

In this scenario, Cengiz presses down, and his game character gets hit by a vehicle. A life point is reduced from Cengiz and since he has no live points left, the game terminates. Since, Cengiz scored a high score he is prompted to enter his name in the death menu, and game saves his score.

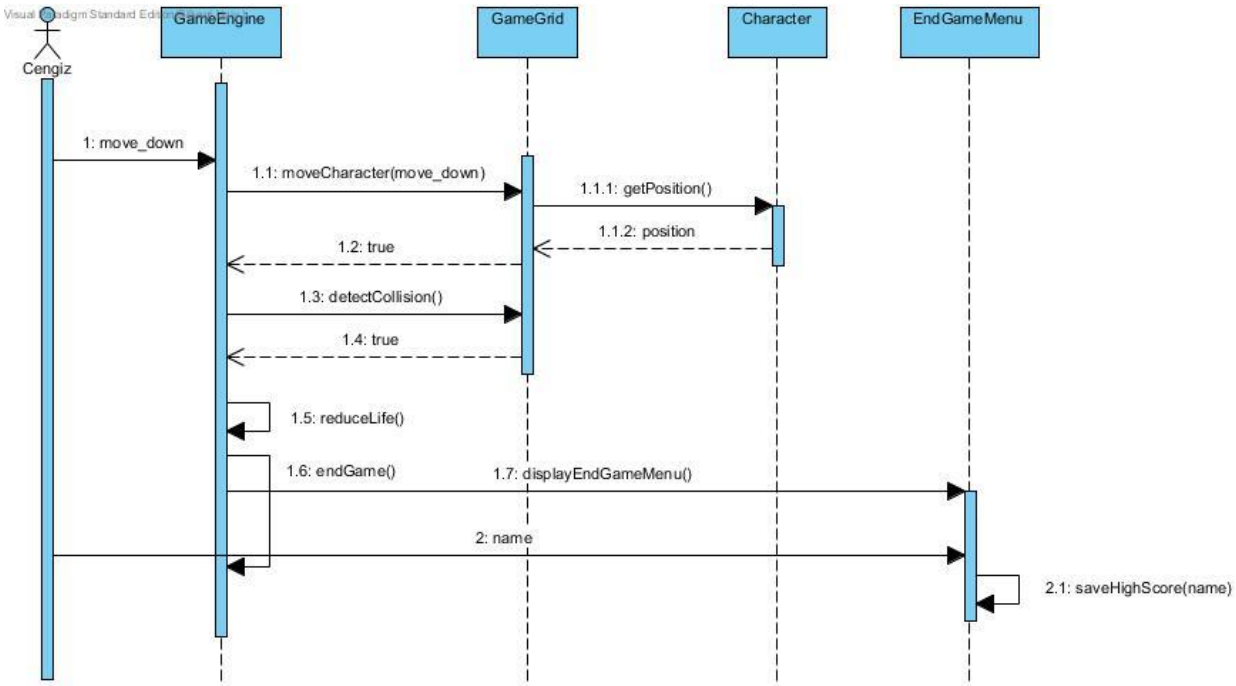


**Figure 5**: End of game sequence diagram

**Activity diagram**

Activity diagram represents the activities with respect to the possible conditions during the regular gameplay.
After starting the game, system checks for collisions:

- If there is a collision, it checks whether the game is over or not (meaning that player has no lives left)
- If not, system checks for collectibles and if there is a collectible, it applies that5 collectible to the game.

If game is over, system updates the high scores depending on the score of current session and restarts the game flow; otherwise, system updates the game stage and restarts the activity flow.

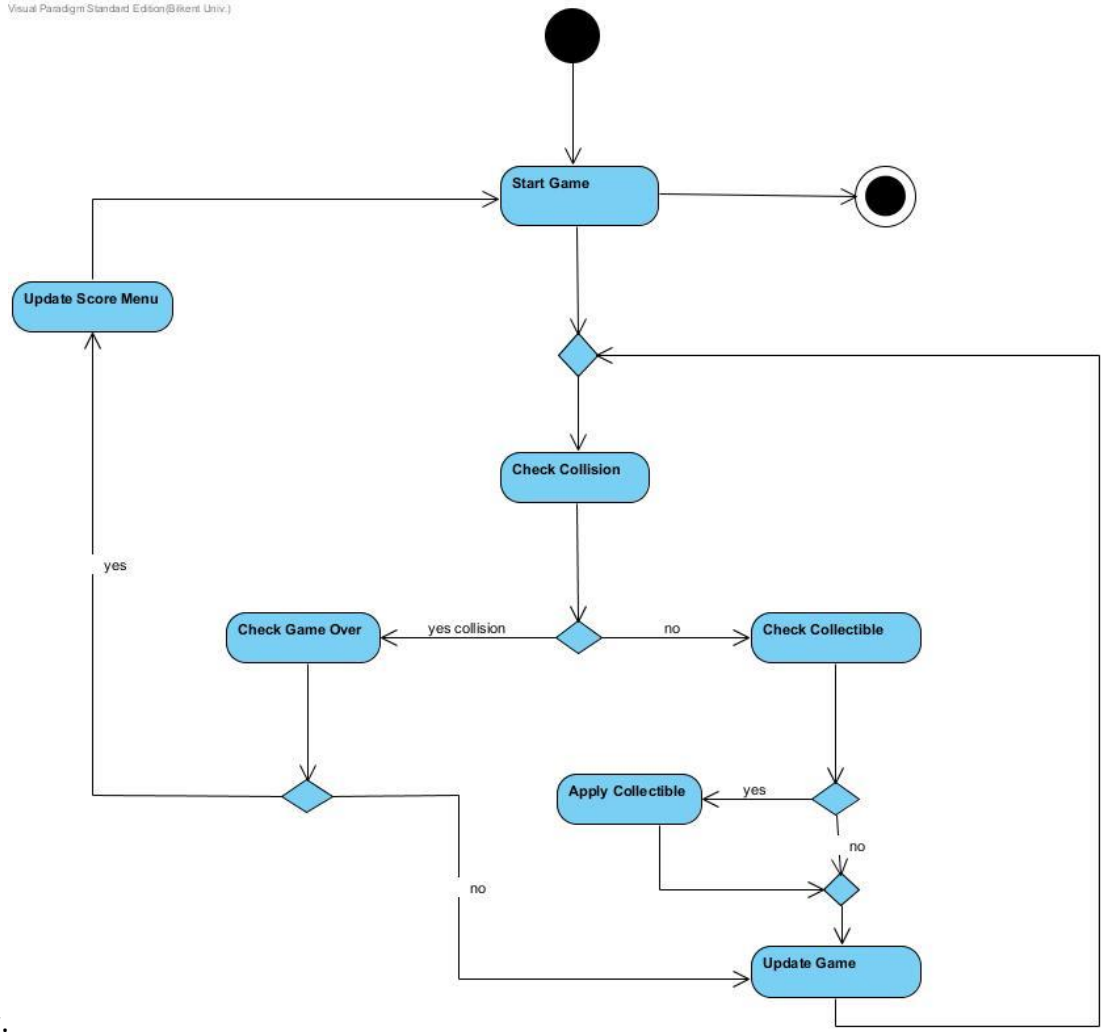


**Figure 6**: Cross-It activity diagram

### State Diagram of the *Character* object

There will be four states of the Character object, a default state in which the character is alive and can move up, down, left or right, or anywhere in the grid if it got hold of a teleportation collectible, an invincible state in which the player can collide with the vehicles and not die, a state in which it gets hit, and a death state. If the user grabs the Shield mystery box/collectible, then he or she can be invincible until time of shield expires. If the character gets hit a check is made on his life counts. If there are zero lives left, the player enters the death state, in which it also stops existing. If there are lives left, the player has the default state.

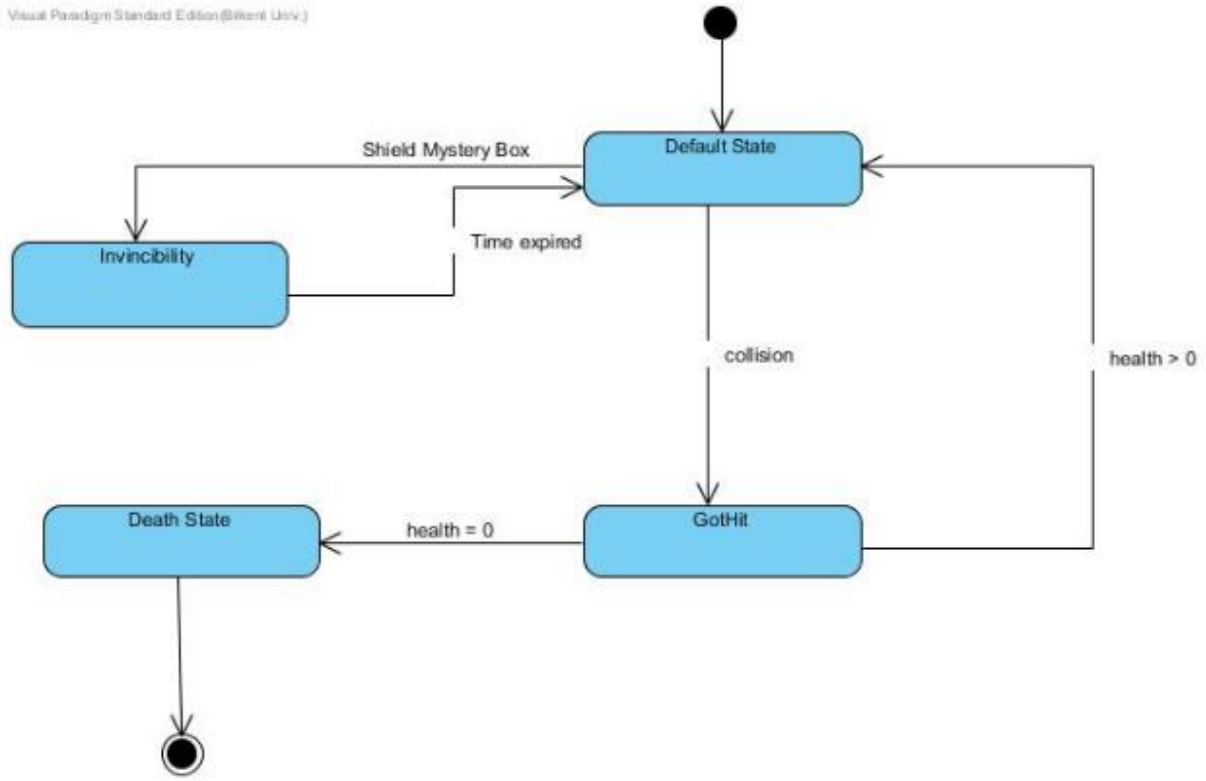


**Figure 7**: State diagram of Character Object

### 2.5.4 User Interface

**Navigational Path**

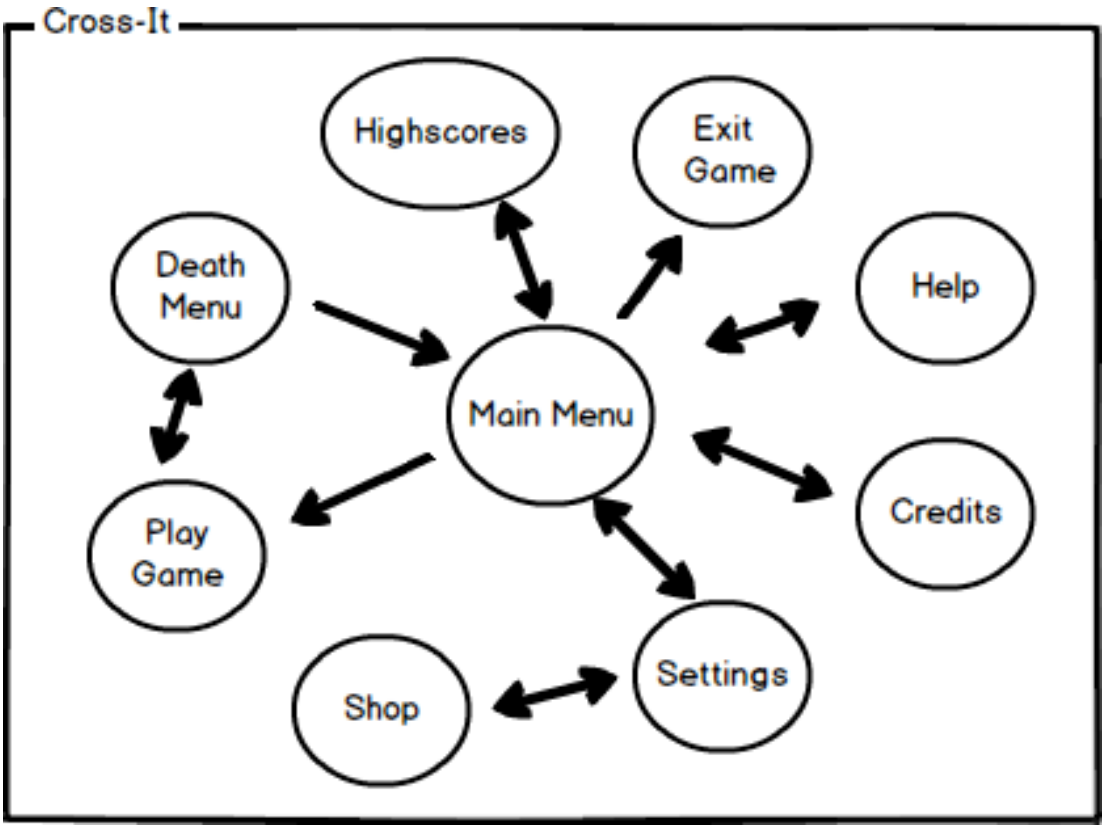


**Figure 8**: Navigational path

**Menus**

**Main menu:** This is the main menu and the starting screen of the game. You can go every section via main menu. Player can start the game by pressing new game button. Because system doesn't let saving the game, everytime player should start a new game. "?" symbol leads to help menu.


**Figure 9:** Main menu

**Help:** The help segment provides general information about the game like concept of the game, and control keys of the game.

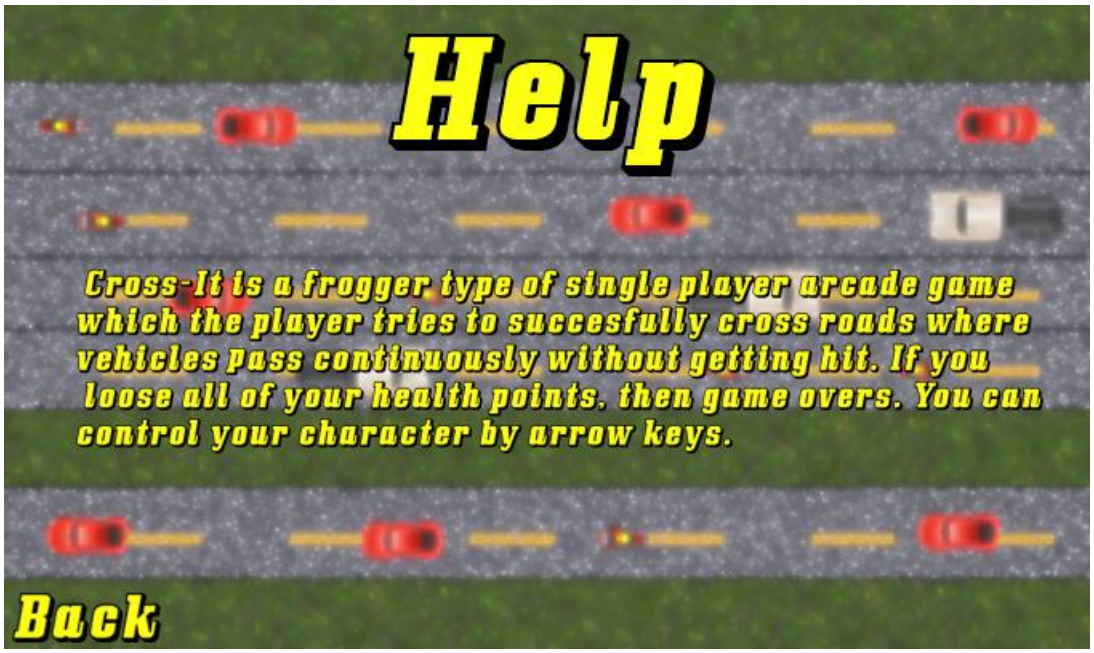


**Figure 10:** Help menu

**Highscore:** This is the window that demonstrates high scores that have been recorded in that machine. It can hold to 10 people's high score. It updates when one of the previous scores has beaten.


**Figure 11:** Highscore menu

**Credits:** Credits is for providing external game information. It provides information about developers of the game and version check of the game.


**Figure 12**: Credits menu

**Settings:** Settings menu is where player can customize his/her game experience. Player can set the music volume, can change the theme, access to shop and changing his/her outfit that previously have bought.

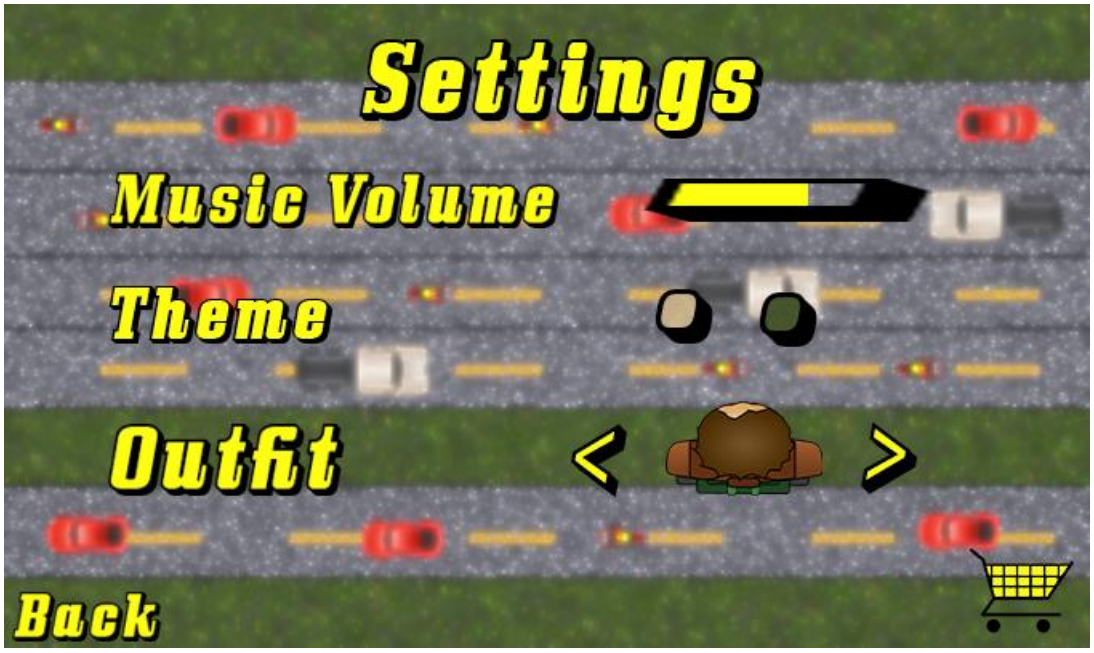


**Figure 13**: Settings menu

**Vehicles:** [2]

**Car:** The first type is a car. Cars will have 2x1 unit size.

**Figure 14**: Car type

**Truck:** Another type of vehicle. Truck is slightly bigger than car. It has 3x1 unit size.

**Figure 15**: Truck type

**Motorcycle:** The last type is motorcyle. It has 1x1 unit size.

**Figure 16**: Motor type

**Player**

General style of the player will be this. However; there will be hats that player can provide with coins that have been collected by playing game, player can customize his character.

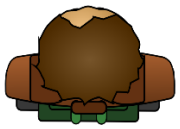

**Figure 17**: Player image

**Coin**

Coins are collectable items of the game. Every stage there appears two coin that player can shop with later on.

**Figure 18**: Coin image

# 3. Conclusion

This report explains the Cross-It game as an arcade game to be available as a desktop application. The report aimed to analyze the flow of the game as it happens in real world.

For this purpose, we have tried to make our diagrams easy to read and follow through, as well as tried to determine a user interface with pleasant visuals of objects in the game(i.e. vehicles) and mock-ups of screens and menus of the game. We prepared this analysis report as a guidance for us on design and implementation stages as a presentation and layout of our idea of Cross-It.

# 4. References

[1] "Classic Frogger Game" http://www.froggerclassic.appspot.com/ [Accessed, 15 Oct 2016].
[2] These are references of all graphical models we have used. All of them have public free license type:

- **The vehicles:** http://opengameart.org/content/free-top-down-car-sprites-by-unlucky-studio

- **Motorcycle:** http://publicdomainvectors.org/en/free-clipart/Vector-clip-art-of-top-view-of-man-on-motorbike/27380.html
- **Grass Texture:** http://opengameart.org/content/grass-textureseamless-2d
- **Wooden box for mystery box:** http://free-texture-site.blogspot.com.tr/2010/10/free-wooden-crate-texture.html
- **Coin:** http://opengameart.org/content/coins-animation
- **Sand texture for theme 2:** http://opengameart.org/content/generic-tileable-sand-texture
- **Player model before edited:** http://opengameart.org/content/top-down-shooter-character
- **Health:** http://opengameart.org/content/pixel-hearts

[Accessed 23.10.2016].