
Bilkent University

Department of Computer Engineering

# CS 319 Object Oriented Software Engineering Project

*Overtake: A New Twist on a Classic Car Game*

## Analysis Report

Bilal SIRAJ, Umut AKÖS, Oğuz LİV, Mithat ORHAN

Course Instructor: BORA GÜNGÖREN

Progress Report

July 14, 2017

# Analysis Report

*Overtake: A New Twist on a Classic Car Game*

## 1 Introduction

Approximately ten years ago, basic java games used to be very popular and most people liked playing those games. If we look at people's hobbies and the game market now, there is not much difference compared to past. For that reason, we decide to develop a java game called Overtake. The main objective of the game is to survive for as long as possible while avoiding oncoming enemy cars on a large highway. However, escaping enemies is not enough to capture the player’s attention so we are planning to develop our game by using special features. These features will excite the player and develop a desire to achieve the highest score. Overtake will be a desktop application and will be controlled using the mouse and the left and right keys on the keyboard.

### 1.1 Overview

Overtake is a bird's eye view game and it is quite easy to play and fun at the same time. The player selects their car from a variety of options and hits the play button to start playing. The player's car moves to one direction in the highway which has 3 lanes and other cars are driver's enemies because they come from opposite direction and driver must escape prevent crashing them to protect his/her car. However, driver should also escape other obstacles like oil spills and pot holes. At the same time, he/she should find power-ups to  increase life time.

### 1.2 Gameplay

The player will just need a keyboard to play the game. Left and right keys are enough to move the car and mouse will be used when exiting and pausing the game. At the beginning, the game ,is not full screen but covers most of the screen. If the player wants to play full screen, game allows him/her. There will be a pausing key on the keyboard. So if the player wants to pause the game, pressing it will be enough and he/she can continue whenever he/she wants.

### 1.3 Leveling

Actually, Overtake does not have specific levels. The game will not allow the player to choose the level but it is a score game so the difficulty will be based on score and determined by it. That is to say, if the player reaches a specific score for example 100,

after this score, the number of enemy cars, obstacles and game speed increase and every certain score which we determine can be seen as levels of Overtake. However, we will not have an option to save the game so if the player exits the game, everything will be deleted and if his/her score is the highest, it is the only information that is saved. Therefore, player can see his/her the highest score.

## 1.4 Player's Car

The car is the only tool that the player can control during gameplay. But at the beginning of the game, we will present some cars which can be selected by the player. These cars may have different features, colors and car brands but generally they have similar speed and endurance. After starting the game, user can not change his/her car.

## 1.5 Obstacle Types

Obstacles are designed to be fun because while player tries to escape the enemy cars, at the same time, he/she should look out for obstacles on the road because they make it harder for the user to achieve higher scores and player should distinguish between obstacles and power-ups. Otherwise he/she may miss them. Overtake has 3 different obstacle types and they have different difficulty for the player. Although no obstacle can kill the user's car, but they make it harder for the user's car to avoid enemy cars.

***Pot holes:*** *When the player drives over a pot hole, they lose control of the car for a few seconds.*
***Oil spills:*** *Car moves right and left beyond the driver's control fora few seconds*.
***Water puddle:*** *Car slows down for a few seconds but does not stop*.

## 1.6 Power-ups

Power-ups are an essential part of the game to make it fun. The player will never know where power-ups will show up so if he/she needs power-ups, he/she may have to change his/her lane in the highway to take them. It will be more risky for player but we know, that is what makes it enjoyable. Overtake has different types of power-ups and all of them provide different advantages for the user.

***Boost:*** *Car speed increases and score rises faster.*

***Shield:*** *Even if the player hits the other cars and obstacles, the player is not become damaged. This effect lasts for a short time.*
***Score Multiplier:*** *The player's score rises twice as fast for a short period of time.*
***Juggernaut:*** *The player's car explodes enemy cars on contact. This effects lasts for a short time.*

# *2 Requirements Specification*

## 2.1 FUNCTIONAL REQUIREMENTS

### I. Play Game

Overtake is based on a simple car game which is enriched by some arcade features in order to entertain the user. Initially, user has 3 lives, and appears on a 3 lane road. Randomly generated other enemy cars eventually appear on the road. Main purpose of the user is escaping from enemies by using left and right arrow and getting the highest score. User gets points for each second passed alive. Also player will encounter some extras as obstacles and power ups, such as water puddles, pot holes etc.

### II. Change Settings

The user has the chance to change their car. There are multiple choices for the user's cars in terms of design of the car. There are no extra's about selecting car by us.

### III. High Score

Player can see previously recorded highest records, and user cannot see his previous records unless high score is among in the highest scores.

### IV. Pause Game

The game can be stopped by the user, in the pause state, game stops until user starts it again.

## V. Credits

Gives information about the creators of the program.

## 2.2 Non-Functional Requirements

Graphics of Overtake will be designed by project workers themselves so that there will be no incompatibility in terms of collision mechanics. Moreover original designs will be much more attractive and appealing. There will be also image usage in terms of car modeling for better visual experience.

In order for players to have a better experience, our key listeners' delays will be reduced. In other , players will not have issue with control mechanism and frame drop rate since the project workers' designs about low response time.

Main strategy on designing will be based on motto, "Easy to play, hard to master ". Thus, beginners will not have any problems because of simple interface and game experience, on the other hand, masters will not be bored because game design will be much complicated as game level increases.

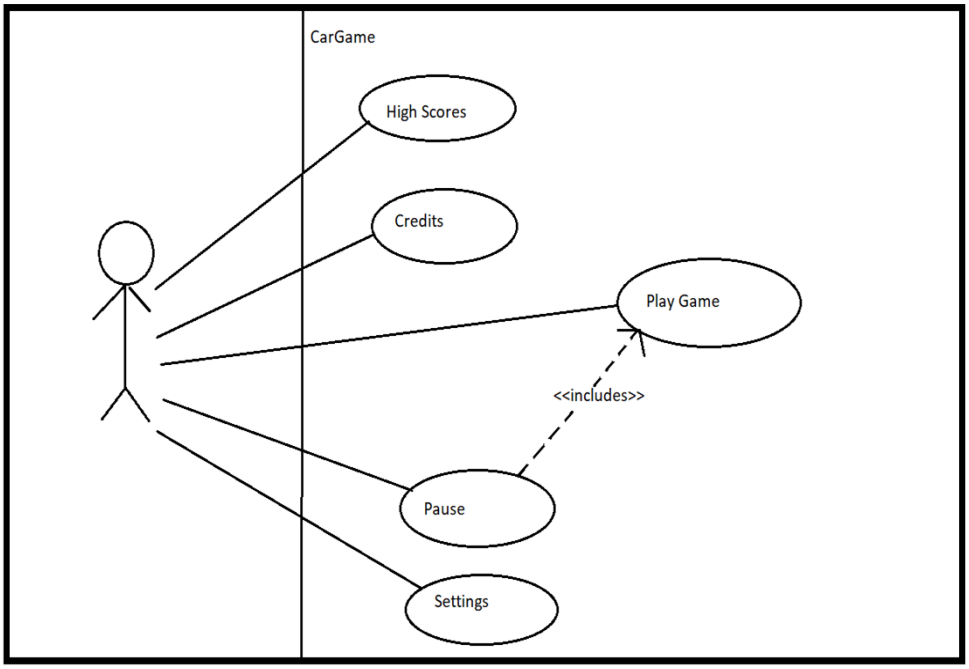
## 2.3 Pseudo Requirements

The language Java will be used for implementing Overtake. For graphical designs Java's Swing library will be used. By this case the project will not be "fabricated".

# *3 System Models*

## 3.1 Use Case Model

This section provides information about the main use case model of Overtake. Detailed use cases are included below.




1. Use Case Name : Show High Scores

   1.1. Actor : User

   1.2. Flow of events :

      1.2.1. User clicks "High score" button.

      1.2.2. System shows the top scores.

      1.2.3. User returns to main menu.

   1.3. Entry Conditions

      1.3.1. User clicks "High score" button at Main Menu

   1.4. Exit Conditions :

      1.4.1. User clicks "Main Menu" button

   1.5. Quality Requirements : NONE

2. Use Case Name : Pause Game

   2.1. Actor : User

2.2. Flow of events :

2.2.1. User clicks "Pause" button during the gameplay

2.2.2. System stops running.

2.3. Entry Conditions

2.3.1. User clicks "Pause" button during the game.

2.4. Exit Conditions

2.4.1. User clicks "Continue" button OR

2.4.2. User closes whole program

2.5. Quality Requirements : NONE

3. Use Case Name : Change Settings

3.1. Actor : User

3.2. Flow of events

3.2.1. User clicks "Settings" button.

3.2.2. System shows the modifiable preferences.

3.2.3. User can change the car in terms of color, shape.

3.2.4. System saves modifications

3.2.5. User returns main menu

3.3. Entry Condition

3.3.1. User clicks "Settings" button during

3.4. Exit Condition

3.4.1. User clicks "Main Menu" button

3.5. Quality Conditions : NONE

4. Use Case Name : Play Game

4.1. Actor : User

4.2. Flow of Events :

4.2.1. User send message to operating system in order to play game

4.2.2. System opens the game

4.2.3. Main Menu opens.

4.2.4. User starts playing the game

4.2.5. System keeps running the game until user lose his all healths

4.2.6. If user's score is in top five score, program records the score as High Score.

4.2.7. User returns to Main menu.

4.3. Alternative Flow of Events :

4.3.1. User decides directly close the game ( go step 5.2.7 )

4.4. Entry Conditions

4.4.1. User runs the game.

4.5. Exit Conditions

4.5.1. User closes the program.

4.5.2. User loses all health points.

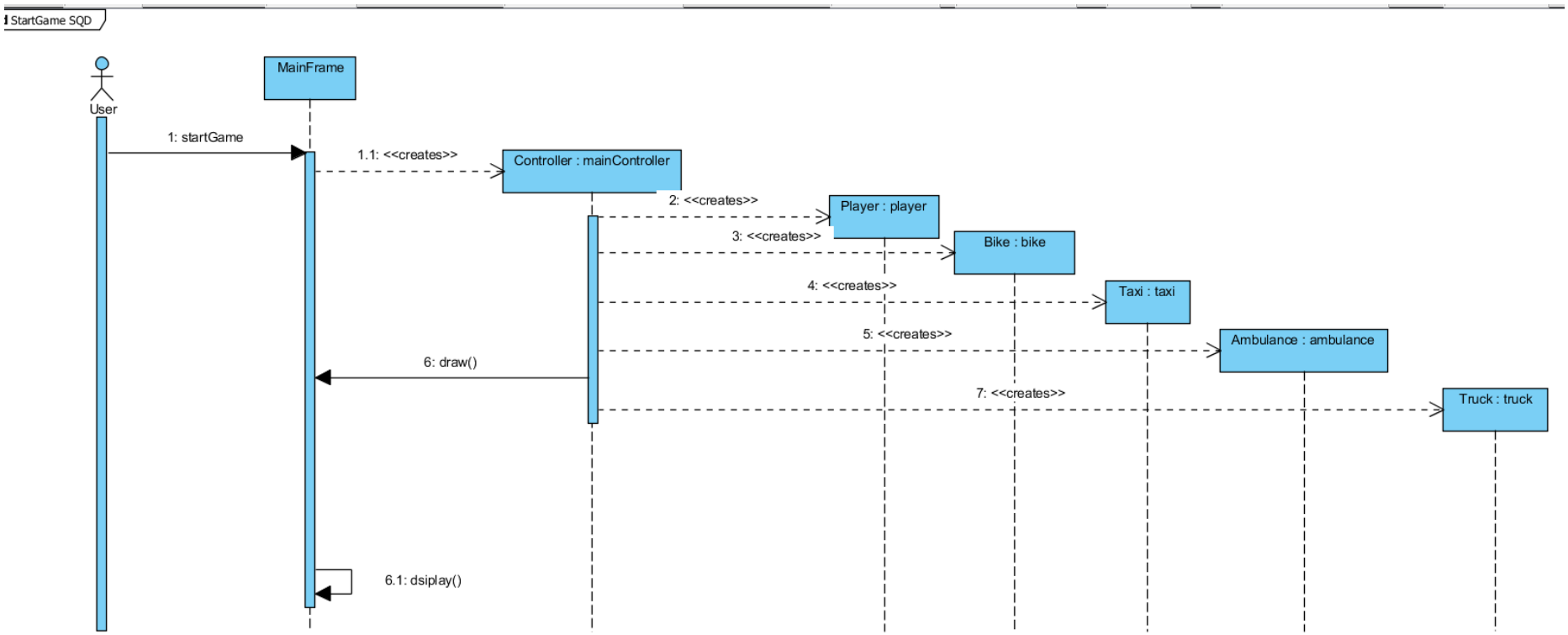
## 3.2 Dynamic Models

This section provides detailed information about Overtake by illustrating some core functionality using sequence diagrams. These include how the game, when initialized, is created along with the necessary objects, and how collisions are handled.

## I. Start Game

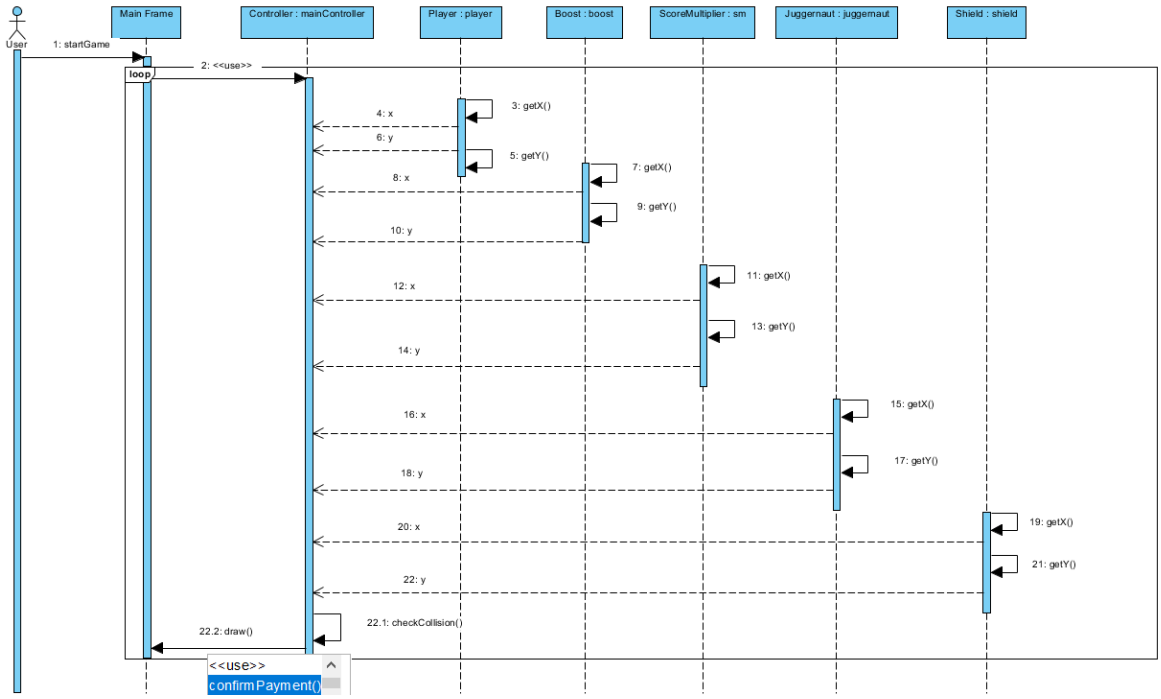
**Scenario:** Player invokes proper method to initialize Main frame of the game, which is the main menu of the game. When main frame is initialized, player clicks start game button in order to initialize a game. After initialization of the game, Main Frame creates a Controller object, which mainly controls the game logic. Once the controller object is

created, a bike, taxi , ambulance and truck objects are created as enemies by the controller, and a player object is created by the controller as well. After all enemy and player objects have been created, those objects are drawn on to main screen by the controller. Lastly, Main Frame invokes display method in order to represent graphics on the screen.


StartGame SQD
User
MainFrame
1: startGame
1.1: <<creates>>
Controller : mainController
2: <<creates>>
Player : player
3: <<creates>>
Bike : bike
4: <<creates>>
Taxi : taxi
5: <<creates>>
Ambulance : ambulance
6: draw()
7: <<creates>>
Truck : truck
6.1: dsiplay()

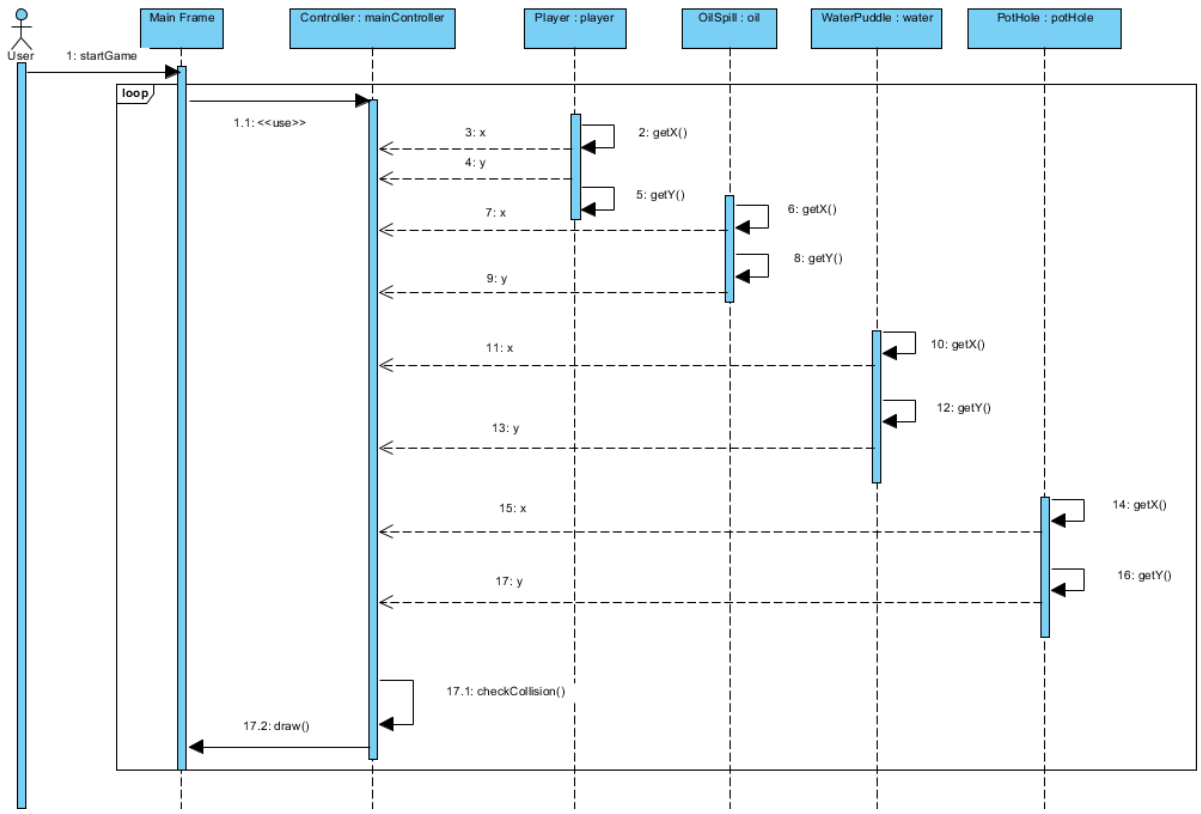
## II. Power Up Logic

**Scenario:** User request to initialize Main frame of the game. Assuming previous diagram performed. After the game is initialized on the screen, in other words, user starts playing, system enters game play loop which performs game logic. Assume that user moves the player icon on the screen in order to escape from enemies. In the game, we assumed that all power ups created after game initialized and all four powers can be on the screen at the same time. If system generates a power up on the main frame, controller must know the coordinates of emerged power up. When user's x and y coordinates matches with the power up's x and y coordinate, user powers up according to which power up that user collided with. If a user powers up, controller draws the power up effects on the main frame.



## III. Obstacle Logic

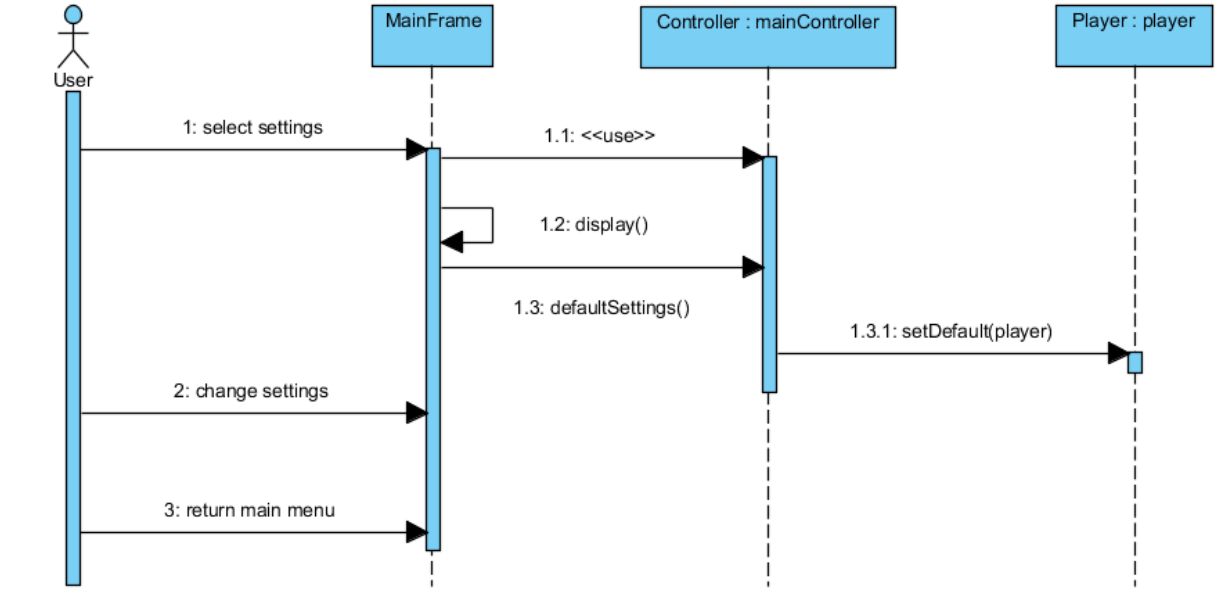
**Scenario:** User request from operating system to invoke the game. Assuming Start Game case diagram performed. After game initialized on the screen, obstacles generated randomly as time goes by according to game. There are four obstacles for the game, and they can be on the screen at the same time. Obstacles are generated randomly by the system on the screen. When an obstacle or obstacles emerge on the screen, controller must know generated obstacles coordinates in terms of x and y. If player's coordinates match with the obstacle's coordinates, user will be affected according to obstacle that user collides with. Controller draws the effects of the obstacle on the player to the main screen.



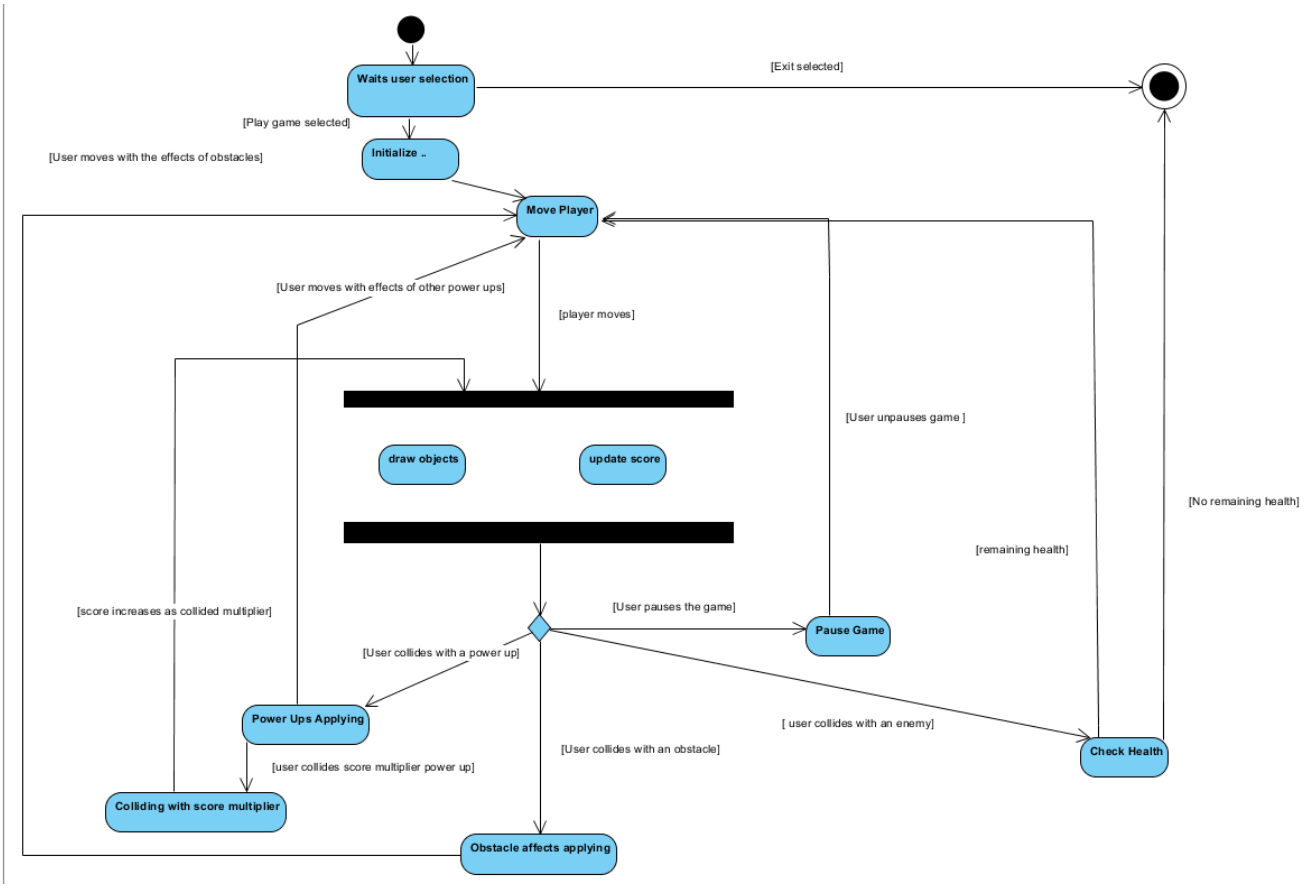

## IV. Change Settings

**Scenario:** User requests game invocation from operation system. Assuming Start case diagram performed. After initialization of the Main frame on the screen, user selects

settings on the main menu which is shown by Main Frame. User selects settings on the main menu. Then, system initialize the players current preferences as defaults. User can change the icon of the player from this screen. After returning main menu, system records alterations on the player.



# V. Activity Diagram




When game is run by operating system, program awaits user to receive a signal. If received signal is exit game, then program closes. If user's signal is play game, then program start initializes every aspect of the game; player, controller, obstacles, power ups, enemies and so on. After initialization, enemies start coming against the player, and the player tries to avoid colliding with enemies. If player collides with an enemy, player's health point reduces. As player increase its score during the game, frequency of emerging enemies and obstacles will increase, but power ups will remain with the same frequency as game starts.

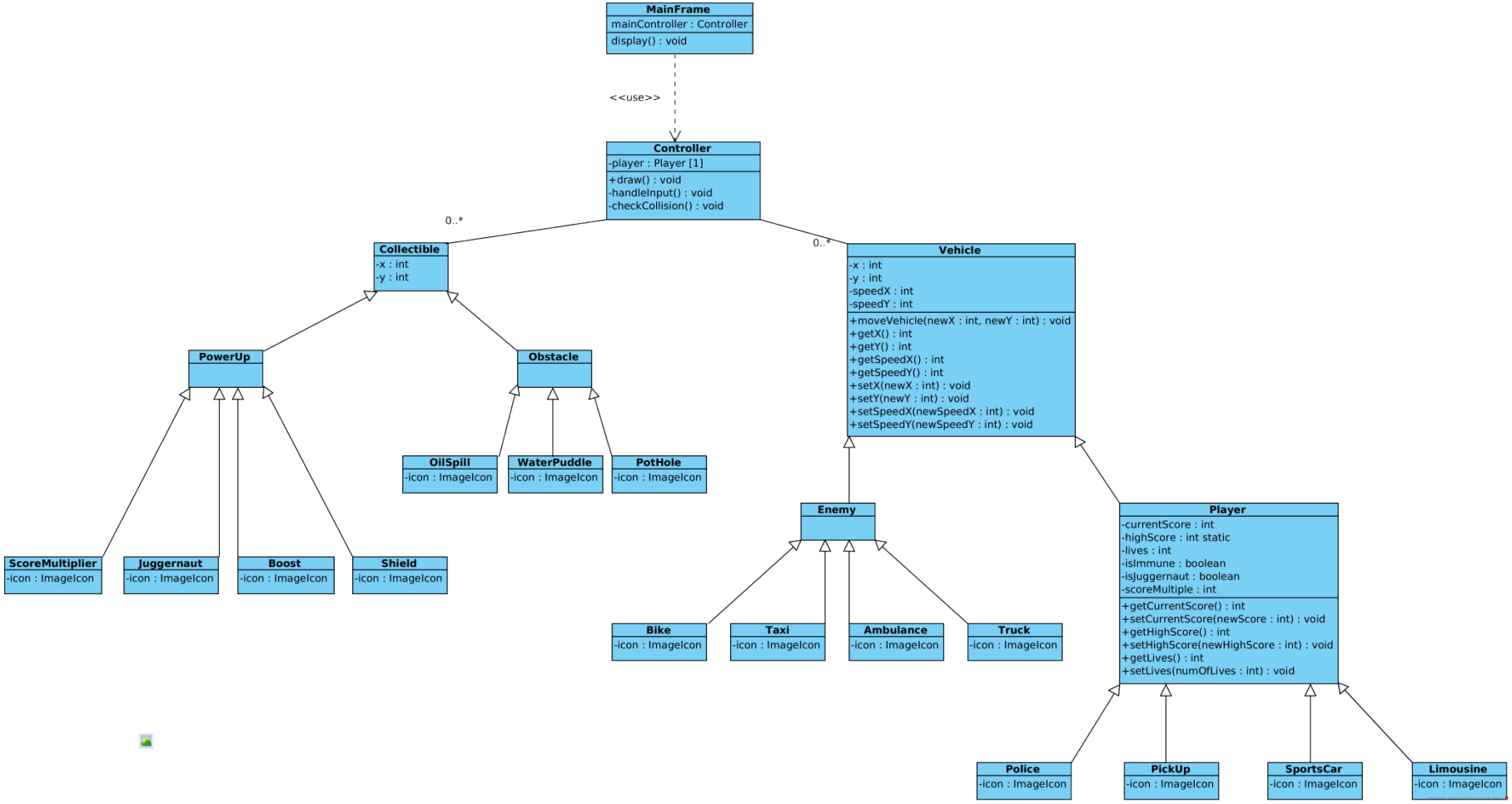
Also, power ups and obstacles are generated by the system on the screen with random places. If user collides score multiplier power up, increasing rate of the score will be multiplied by 2. Collision with other power ups will not affect the score increase directly, but gives advantage to user, such as shield, which ignores one collision with an enemy,

juggernaut, which destroys an enemy after collision, and boost, which makes player move faster in the vertical axis.

Obstacles are like power ups in terms of initialization, emerging on the screen, but they are disadvantages for player. If player collides with water puddle, movement of horizontal axis would deviate from normal horizontal movement as moving not with certain units. Oil Spill has similar aspects, but the movement deviation is much bigger than water puddle. Pot Hole is different kind of obstacle, if player collides with a pot hole, player loses control of the car for a few seconds.

The game will go on as long as player keeps its health points, if player have no health points left, then system reports that game is over and then returns user to the main menu. Also user can quit program by directly closing it.

## 3.3 Object and Class Model

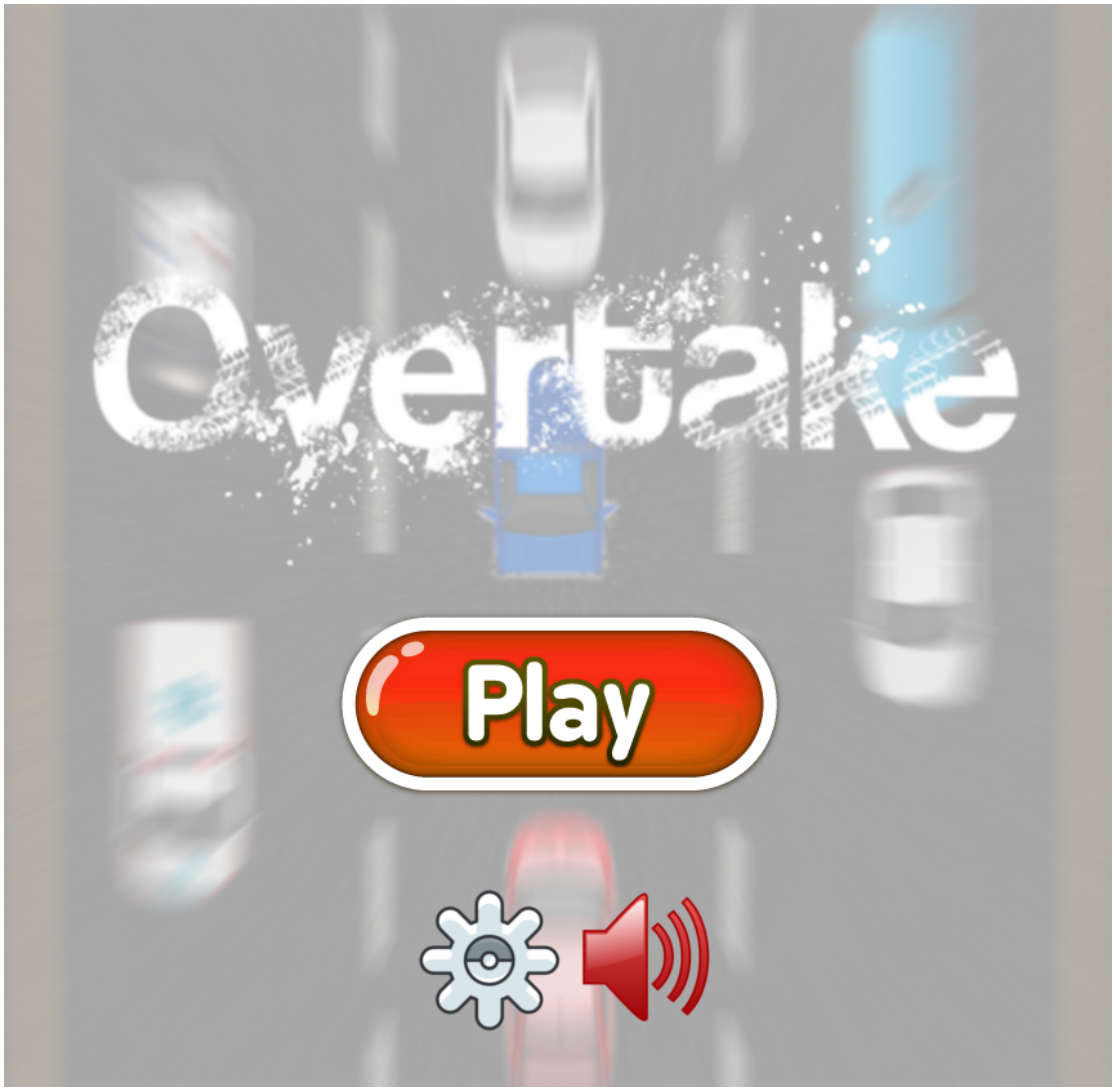


The above diagram illustrates the current class design for the game. It consists on 23 classes. This design is still subject to change. Here is an outline of each class's purpose:

The current design is an attempt to implement the Model-view-Controller architectural pattern. The top most level in the program is the **MainFrame** class. It serves as the view for the game, using the data it gets from the next class, the **Controller**. As the name suggests, this is the controller for the game, in which all game logic is implemented. It is by far the most complex and essential piece of the whole program. It handles object creation, collision tracking, score keeping and everything that goes into the overall experience of the game. Following the Controller class, is a hierarchy of model classes. The **PowerUp** and **Obstacle** classes are generalized under the Collectible class. Which brings us to our next class, the **Vehicle** class. This was a necessary abstraction when designing the game as it models any object that moves, namely the player's vehicle and the enemy vehicles. To distinguish between the two **Player** and **Enemy** classes were added. The Player class contains all instance variables to keep track of the player's car position, score, health and anything else associated with the player. The bottom most level which models the different types of collectibles and types of vehicles, exists mainly to hold the different graphics they represent.

## *4 User Interface*

### Main Menu



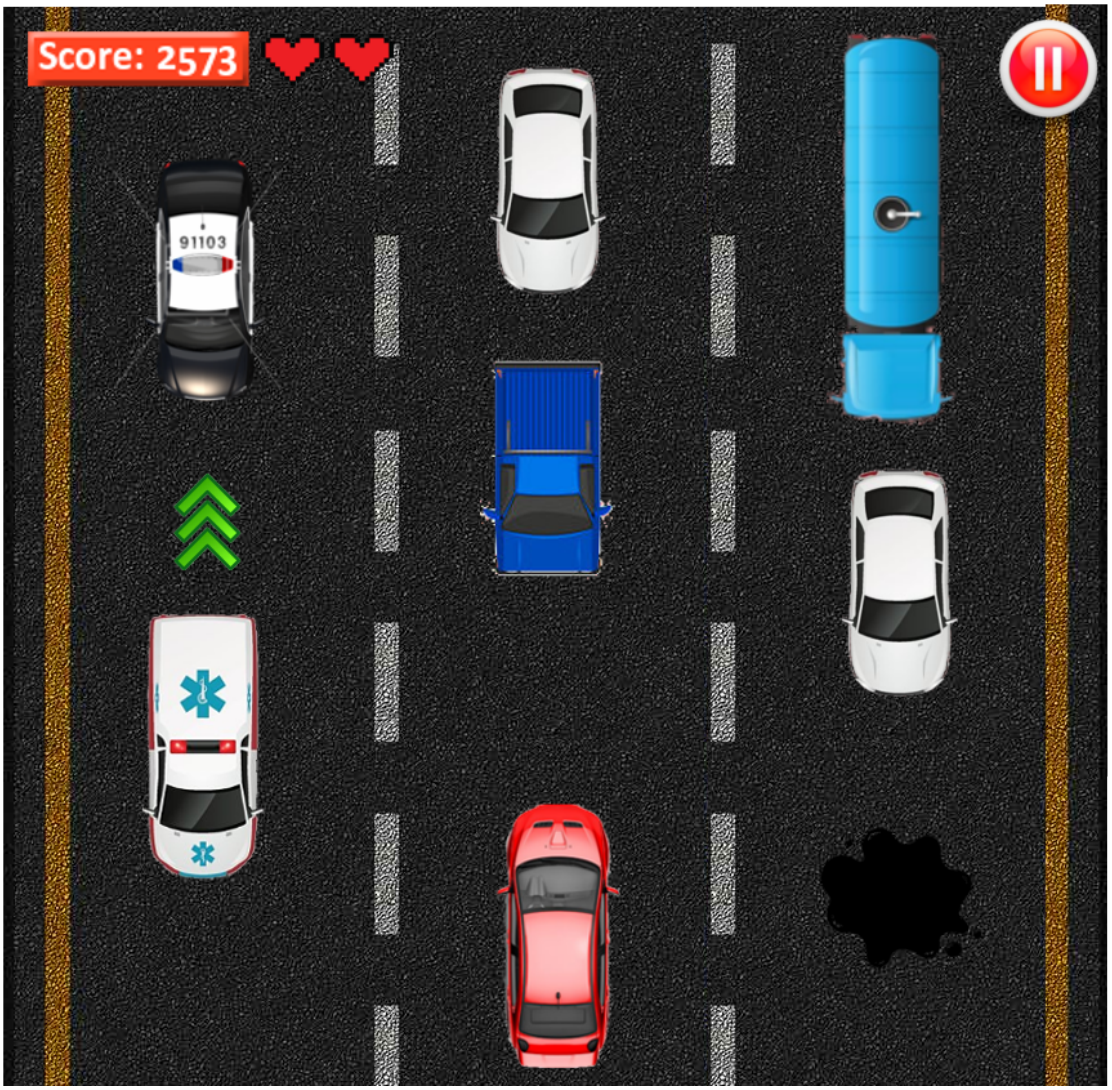

When the program is launched, the user is greeted with the screen above, containing the game logo, a play button to start a game, a gear icon to access the settings and a volume logo to turn on or off the music.

## Selecting a vehicle

When the user presses the gear icon on the main menu screen shown above, they are taken to the screen below, in which they can cycle through their vehicle options using the green left and right arrows. Once they have chosen one, they press the select button and are returned to the main menu.

## Gameplay

When the user presses the play button shown on the main menu screen above, a game is started and the challenge begins. Below is an image of what the gameplay experience might look like.




The red car in the middle lane is the player's vehicle. The player is moving straight into oncoming traffic. To the top right corner is the pause button, and on the top left are the

score and remaining lives indicators. Also in the image are an oil spill obstacle and a boost powerup.