BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS319
Object Oriented Software Engineering

Requirement Analysis Report
Project "Rush Hour"

Group 2.C Not So Oriented

Deniz Dalkılıç
21601896
Ahmet Ayrancıoğlu
21601206
Kaan Gönç
21602670
Ali Yümsel
21601841
Sina Şahan
21602609

Table of Contents

1. Introduction

Rush Hour is a sliding block logic game, which is considered as one of the best logic games of all time. It provides a fun way to improve one’s problem solving, sequential thinking and logical reasoning skills by associating it with many people’s real life struggle, the “Traffic”. In real life there are several different versions of the game Rush Hour, which include different scaled maps and vehicle packs. Our game will be based on the original Rush Hour game, however with our own additions and special features to make the players have a better experience in virtual life when compared to playing the game in real life.This report consists of an overview, the requirements, UML models and mock-up designs for our own version of the game.

2. Overview

Rush Hour is a totally self-directed logic game which consists of a 6 by 6 playing board, having an exit opening on the right edge, an escape car and different sized vehicles which are used as obstacles preventing the player's car to move on the board. The main goal in the game is to lead the red car to the exit by moving the other vehicles on the map out of the way. In the original version of the game there are 40 challenges, ranging in difficulty, which will push the player to think strategically before making a move. Our game will also consist of 40 levels, however the difficulty will be a bit harder than usual. In every 10 levels as these levels will have a time limit which the player will play as an ambulance, a police car or a fire truck having a limited time to accomplish its task as a similar case when compared to real life. We think that such an addition to the original game would provide the players a better environment for improving their skills as they will actually have the chance to solve a problem under stress and pressure which occurs frequently in real life. Additionally, in the later processes, if we are able to construct different levels we are planning to increase the board size to 7 by 7 or 8 by 8 and include non movable objects to our game so that the game will include more challenges to solve and keep the player playing it. Moreover, our game will have a reward system as the players will collect stars according to how efficiently they pass the level and be able to use those stars to acquire new vehicles and themes to use on the desired levels. For example, our game will have a “Safari” theme as an addition where the players will try to move their safari cars to the exit by moving wild animals out of the way. As a distinction, when compared to the real life game which is played by real sliding blocks with no sound effects, our game will have a sound system which we think will create a better gaming experience for the players. The players will be able to decide which theme and sound package they want in the game freely (including mute option) as well as choosing their own car by using the stars they earned. During the gameplay, the players will be able to pause or exit the game which will automatically cause the game to save the latest progress of the player so that the player could come back another time and load their previous game. There won’t be a highscore system, however the game will keep a track of the number of moves the player did during a level, so that he could return back to a previous level to find a better solution and break their own record which would reward them with new stars. In terms of the gameplay, players will use a mouse click to choose a vehicle on the map, and the “W”, “A”, “S”, ”D” keys to move the chosen vehicle through the grids. The vehicles will display a smooth animation during their sequence between the grids. Whenever there is an empty road between the player’s car and the exit grid, the car will automatically move towards the exit and the player will pass on to the next level. Finally, whenever a player needs help he will apply to our help panel which will have all the necessary information about the game.

3. Functional Requirements

- **Opening the Game:**
  When the user starts Rush Hour, the main menu must appear on the screen, allowing them to create/select user, play the last saved level, open levels screen, change game settings, get help about the game and access to their progress details.

  1.1. Create/ Select User:
  Upon pressing the “Create/Select User” button in the main menu, the user should be able to access to “User Selection” screen in which the user is able to create a user profile or select an existing user profile from the list. Each user should have their own progressions of the games saved on these profiles.

  1.2. Play the last saved level:
  The players will play the last saved level in the exact state which the user left the level at when they interact with the play button on the home screen.

1.3. Open levels screen:

Upon pressing “Levels” button in the main menu, the user is be able to access to the “Levels” screen in which each available level is represented with a different button. Player should be able to interact with any button that is not blacked out, which means it is not yet unlocked for that user, upon pressing any of the available level buttons, the game screen should be loaded with the selected level in the start state of the level.

1.4. Change Game Settings:

Upon pressing the “Settings” button in the main menu, the “Settings” screen should appear. In the “Settings” screen, players should be able to adjust the volume of the game sound and change the theme of the game. If the users want they can also look at the “Credits” section.

1.5. Get Help:

Upon pressing the “Help” button in the main menu, the user must be directed to the Help screen in which there is a picture that shows the user how to play the game and informs him about the controls.

1.6. Access Progress Details:

The user will access his own progress details such as how long it took for him to pass a specific level or how many stars he earned after each level to understand how efficiently he solved the level.

Playing the Game:

User starts the game at the start position of the level in a map filled with cars that blocks the exit that the user is trying to reach. If the level is a time challenge level, a countdown is started as the game starts.

1. Controls:

The user must be able to select which car to move by using a mouse click and move the selected car with the keyboard buttons W, A , S ,D.

2. **Pause/Resume the Game:**
   The user should be able to pause the game by pressing the pause button in the “Game” screen. When the game is paused, “Pause” menu should appear, in which the user can choose to access the settings, exit game or return to the currently paused game. Additionally, if the level is a time challenge level, countdown should stop while the pause menu is on.

3. **Save the Game:**
   The user will be able to choose to save the current progression of the level when exiting. Saving operation must store the changes made to the level so that the user can load the level and play the level where they left of.

4. **Collect stars:**
   The player will collect stars from levels by completing the level. The stars are given to the user based on how the user performed in that particular level. User can use the stars collected on the levels to unlock new themes for the game.

5. **Changing In-game Settings:**
   The user will be able to change the theme and turn the sound on/off during the game.

- **Game Elements:**

  1. **Units:**
     The game consists of vehicles that block the way of the player and can be moved by the player. The vehicles can vary in size and color. The vehicles can only move on their own axises, they are not able to rotate.
     a. **Car:**
        The car is a 2x1 vehicle that can be moved by the user.
     b. **Truck:**
        The truck is a 3x1 vehicle that can be moved by the user.
     c. **Player Car:**
        The player car is a special kind of car that the user can again choose and move. However, this player car has to be moved to the exit location on the map in order for the level to be completed.

2. **Map:**
   The game is played on a 6x6 map that is filled with vehicles and has 2 special locations marked on it.
   a. **Start Location:**
      The player car starts the level from this location which is at the left most of the map.
   b. **Exit Location:**
      The player car has to reach this location in order to finish the level.

- **Rules:**

1. The user can only move one vehicle at a time.
2. Vehicles can only move 1 full block or multiples of 1 full block.
3. The level will be finished when player car reaches the exit location marked on the map.
4. The user will get stars based on the users performance on that particular level. The less moves the user makes the more stars will the user collect.
5. If the level is a time challenge level, the game will end when the countdown reaches 0, and the level has to be played again.

4. Nonfunctional Requirements

Quality Requirements

Supportability:

- Game must be implemented with an extendible design, especially the game engine and how it handles the game loop, so that new features, game objects and graphical updates can be added without changing the existing parts.
- The system should allow multiple people work on the implementation at the same time by providing a maintainable structure in order to not have any difficulties in understanding the existing code segments, implementation of features and the design choices.
- Object oriented design concepts must be employed to make the game easily extendable even though some of the concepts could be considered not necessary at this state of the game.

Usability:

- Players must be able to understand the main features of the game easily.
- Players must be able to interact with the user interface without having any difficulty and without being given any extra information about how to use the interface..
- User interface should be provided with an attractive design to the players.
- No unnecessary information should be provided to the user as it can cause confusion about the game.
- The players must be able to get help whenever they want with a help menu that is easy to follow (i.e images instead of long descriptions).
- The players should be able to feel the metaphors of real world effectively such as hearing car engines and horns while playing a level.

Reliability:

- The game must be able to store multiple player profiles to ensure that each player can save their progress of the game separate from each other.
- Any crashes during the gameplay should not effect the game data so that when a crash happens the player should be able to continue to his game from where he left.
- The system should automatically save the progress including the settings adjusted the player as a backup if the player quits the game without saving his progress, or in case of a crash.
- The system should be able to provide a local backup system so that the players do not worry about having an internet connection.

Performance:

- The system should be able to support smooth animations which means the game should be able to run at 60 frame per second.
- The system should be able to define each input separately in the same frame when multiple inputs are given by the player at the same time.
- Progress saving should consume a small amount of time.
- Transition between panels must be fluent.

Pseudo Requirements

Implementation Requirements:

- All code related to the project must be written in JAVA programming language
- Detailed comments should be provided to explain methods and complex code segments when necessary.

Interface Requirements:

- Data is exported/imported into the system through local files and directories

5. System models

5.1. Use case model

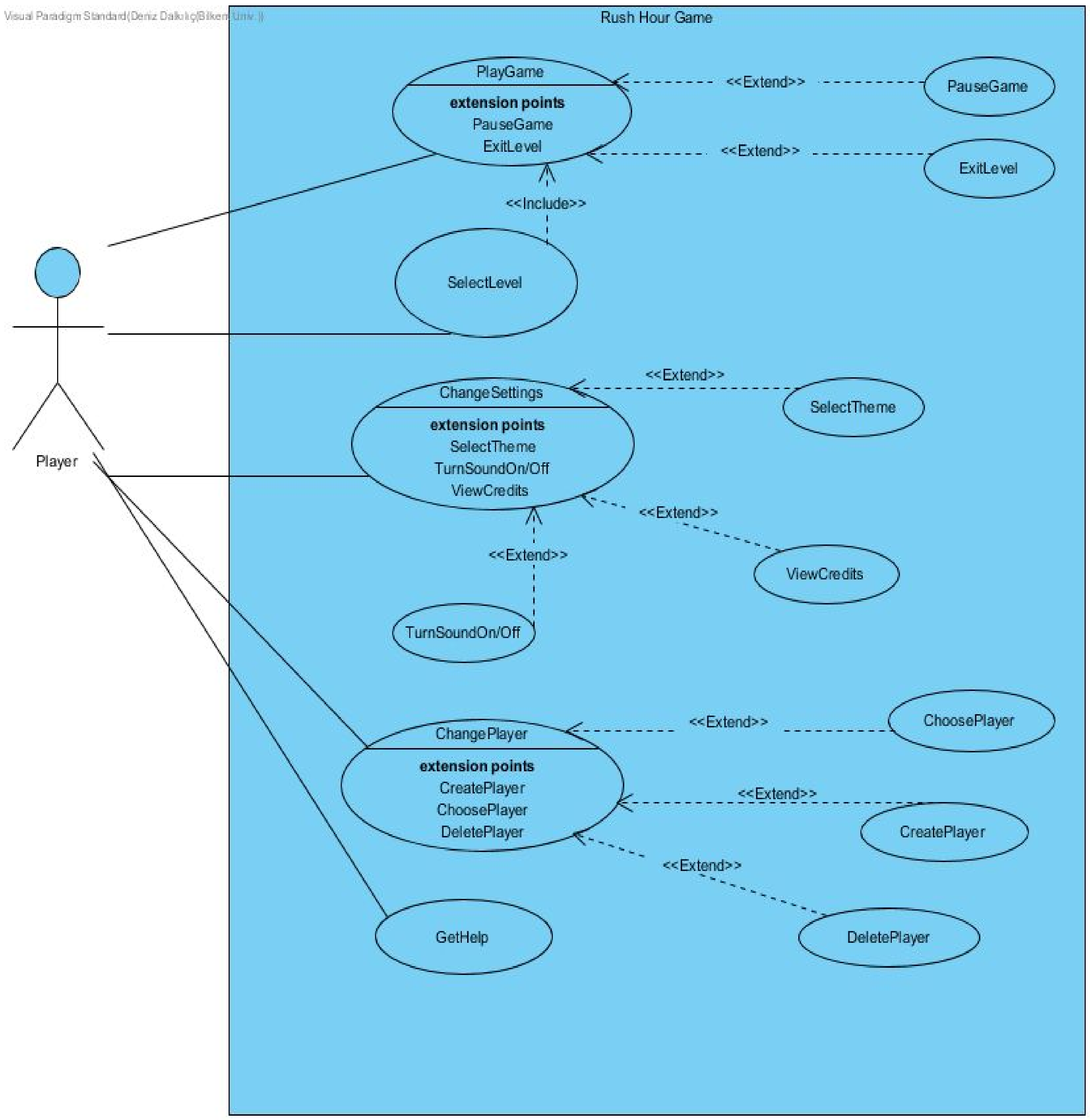


Figure 1 - Use Case Diagram

Textual Use Case Descriptions

Description of Play Game Use Case

Name: Play Game

Participating actor: Player

Entry condition:

- Player has accessed to the game menu by opening the game or by using “return menu” feature from other panels such as settings, help or choose player.
- Player interacts with the PlayGame button

Or

- Player has selected a previous level from the “Levels” menu

Exit condition:

- Player has completed all the levels or chose to exit the current level to return back to the main menu

Successful Scenario Event Flow:

- Game is initialized by using the currently selected level
- Player passes the level
- Next level is loaded
- The last two steps are repeated until all the levels are completed
- Credits are displayed
- System returns to the main menu

Exceptional Scenario Event Flow:

- Game is initialized by using the currently selected level
- Player plays the level
- Player can not pass the level
- Player chooses to exit the current level
- System saves the current progress
- System returns back to the main menu

Special Requirements

- Player can pause the level
- Player can exit the level

Description of Select Level Use Case

Name: Select Level

Participating actor: Player

Entry condition:
- Player has accessed to the game menu by opening the game or by using “return menu” feature from other panels such as settings, help or choose player.
- Player has selected the Levels option

Exit condition:
- Player has chosen an available level from the levels menu

Successful Scenario Event Flow:
- Levels screen is displayed
- The levels that player passed are set as unlocked and others are set as locked
- Player chooses a level from the list of unlocked levels
- The system continues with PlayGame use case

Exceptional Scenario Event Flow:
- Levels screen is displayed
- The levels that player passed are set as unlocked and others are set as locked
- Player does not choose a level
- Player exits the level screen by pressing the back button
- The system returns to the main menu

Special Requirements
- Select level use case includes PlayGame use case which automatically directs the player to the game scene after a level is selected

Descriptions of Change Settings Use Case

Name: ChangeSettings

Participating actor: Player

Entry condition:
- Player has accessed to the game menu by opening the game or by using “return menu” feature from other panels such as game scene, help or choose player.
- Player interacts with the Settings option

Exit condition:
- Game settings are updated according to the adjustments of the player

Event Flow:
- Settings panel is initialized
- Settings are displayed according to the last adjustments of the player, if none they will be set as default
- Player adjusts the settings according to his own desire or presses the default button
- Player presses the save button
- Settings are adjusted to their selected values

Exceptional Event Flow:
- Settings panel is initialized
- Settings are displayed according to the last adjustments of the player, if none they will be set as default
- Player adjusts the settings according to his own desire
- Player does NOT press the save button
- Settings are NOT changed and kept as their previous values

Special Requirements
- Player can select a desired Theme
- Player can adjust the Sound level
- Player can view the credits of the game

Description of Change Player Use Case

Name: ChangePlayer

Participating actor: Player

Entry condition:
- Player has accessed to the game menu by opening the game or by using “return menu” feature from other panels such as game scene, help or choose player.
- Player interacts with the Change Player option

Exit condition:
- Player is changed, game progress and the settings are updated according to the selected player

Or
- A new player is created

Event Flow:
- Change Player panel is initialized
- Active players are displayed in a list as well as a button to create a new player
- Current user chooses a player from the player list or creates a new player
- Player is set to the chosen or the newly created player
- Player presses the save button

Special Requirements:
- Player can choose an existing player
- Player can create a new player with an non existing id
- Player can delete his own account

Description of Get Help Use Case

Name: GetHelp

Participating actor: Player

Entry condition:

- Player has accessed to the game menu by opening the game or by using “return menu” feature from other panels such as game scene, help or choose player.
- Player interacts with the Help option

Exit condition:

- Player has seen the help documentation

Successful Scenario Event Flow:

- Help panel is initialized
- Necessary information about the gameplay is displayed
- Player reads the document
- Player returns back to main menu

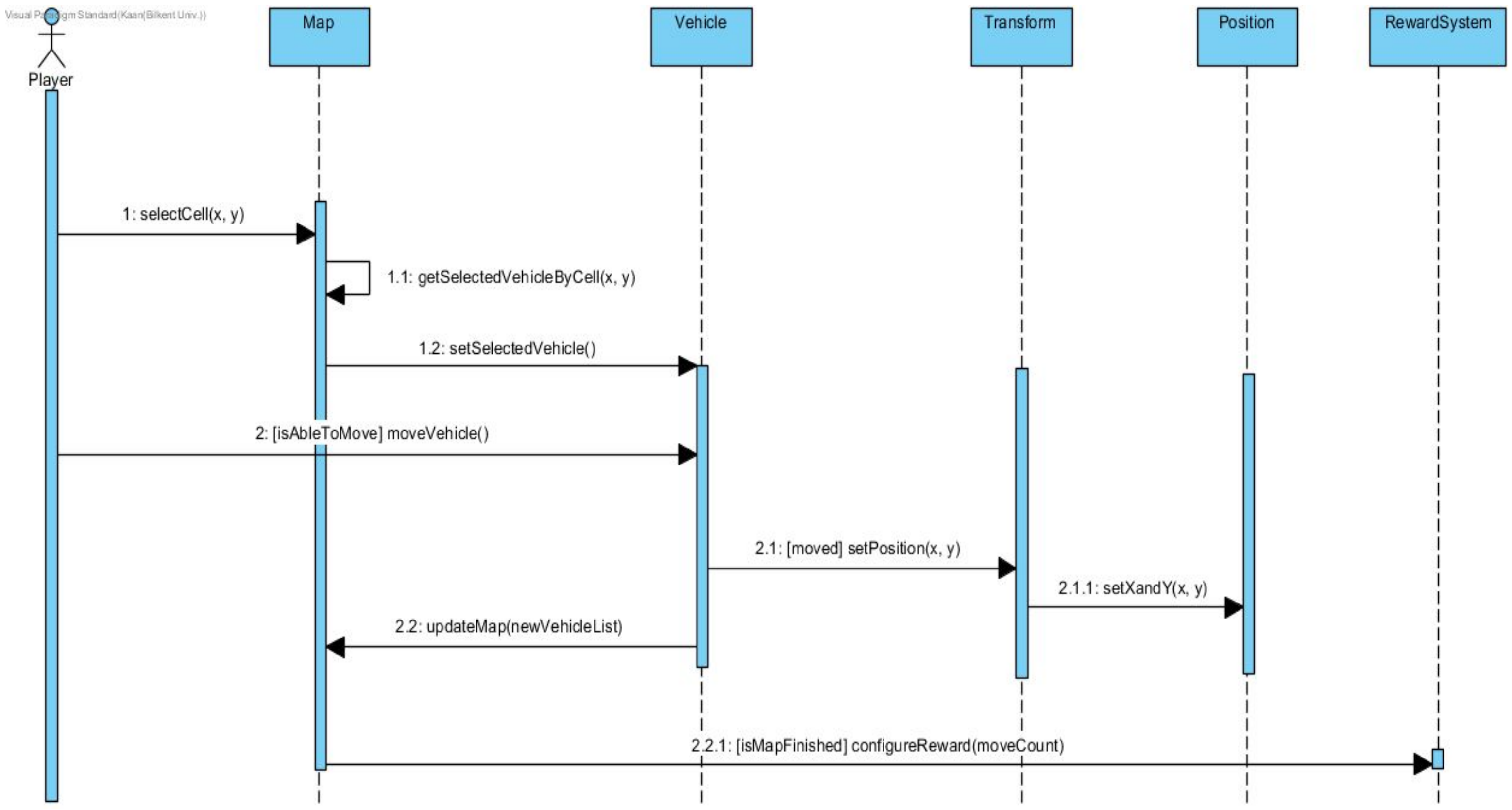
Special Requirements:

- none

5.2 Dynamic Models

5.2.1 Sequence Diagrams

Classic Mode



Visual Paradigm Standard(Kaan(Bilkent Univ.))
Player
Map
Vehicle
Transform
Position
RewardSystem
1: selectCell(x, y)
1.1: getSelectedVehicleByCell(x, y)
1.2: setSelectedVehicle()
2: [isAbleToMove] moveVehicle()
2.1: [moved] setPosition(x, y)
2.1.1: setXandY(x, y)
2.2: updateMap(newVehicleList)
2.2.1: [isMapFinished] configureReward(moveCount)

Figure 2 - Classic Mode Sequence Diagram

This is the sequence diagram of the gameplay for the problem domain. The player first presses a cell on the map. Then, the corresponding vehicle is found and set selected. After that, the player moves the selected vehicle if it is able to move. If the vehicle is moved, its position gets the new x and y values. After the movement, the map gets updated. If the map is finished after that, Reward System configures the reward.

Bonus Mode

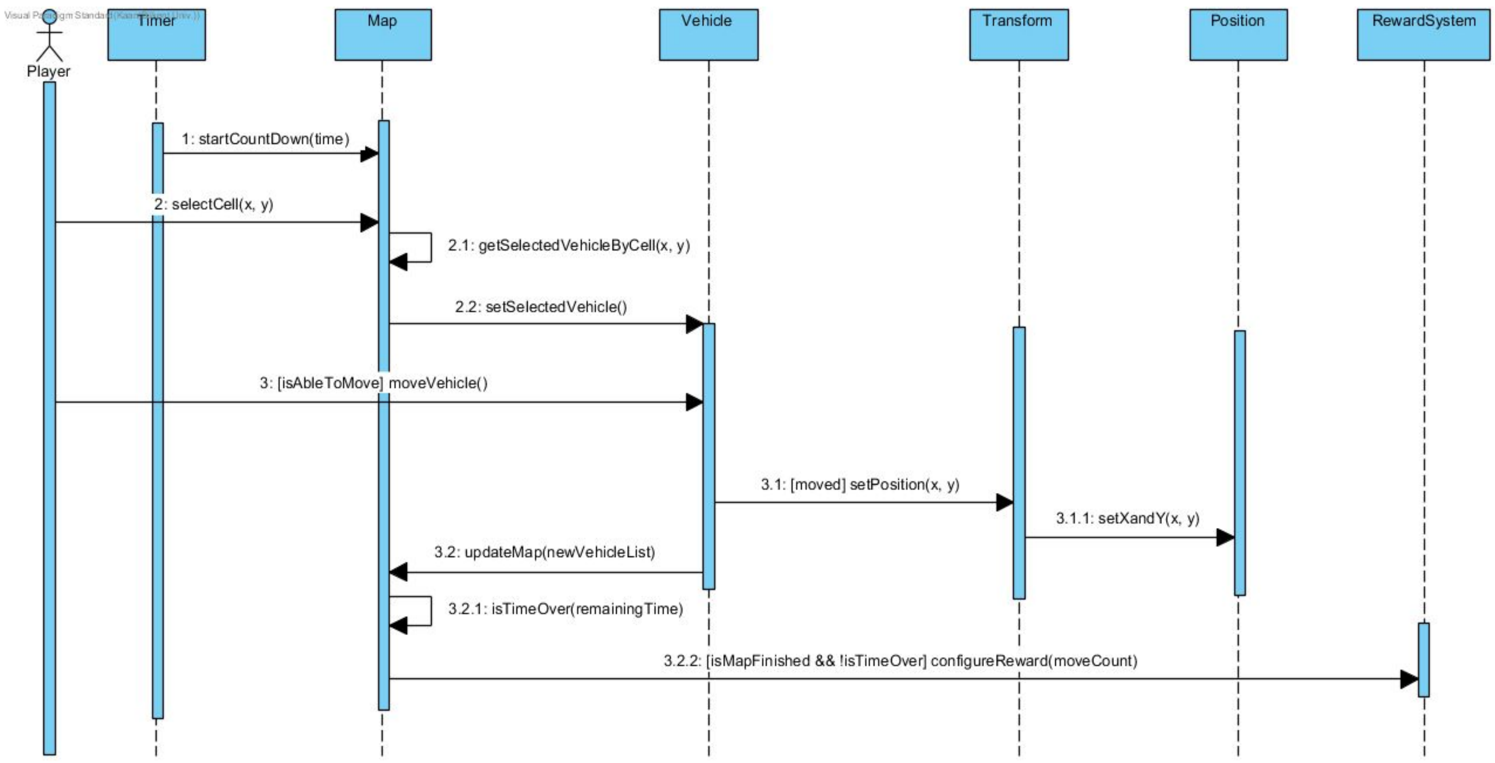
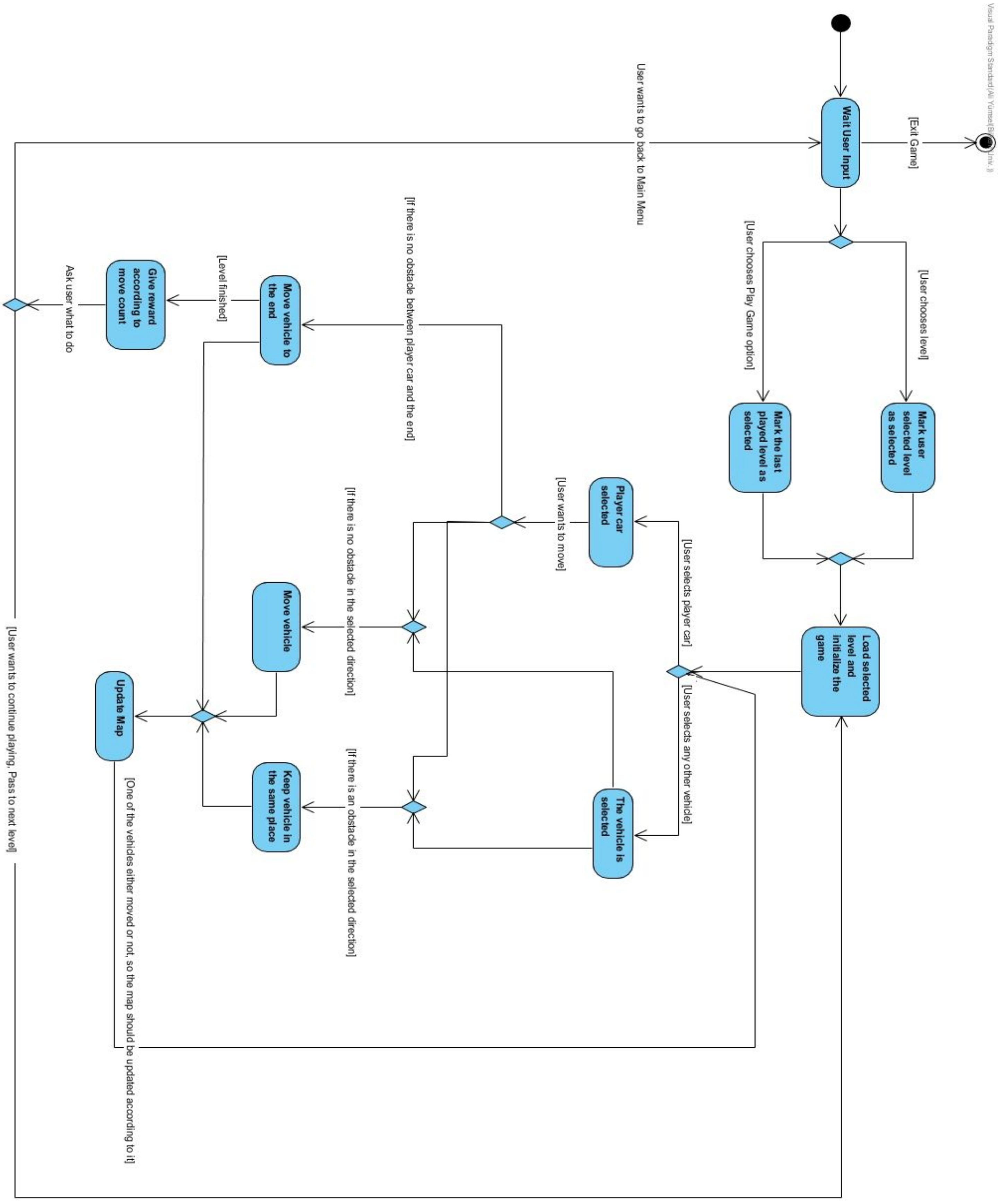

Player
Timer
Map
Vehicle
Transform
Position
RewardSystem
1: startCountDown(time)
2: selectCell(x, y)
2.1: getSelectedVehicleByCell(x, y)
2.2: setSelectedVehicle()
3: [isAbleToMove] moveVehicle()
3.1: [moved] setPosition(x, y)
3.1.1: setXandY(x, y)
3.2: updateMap(newVehicleList)
3.2.1: isTimeOver(remainingTime)
3.2.2: [isMapFinished && !isTimeOver] configureReward(moveCount)

Figure 3 - Bonus Mode Sequence Diagram

The gameplay sequences of bonus mode and classic mode are the same except for a timer which forces the player to finish the map in a specified time. Therefore, a countdown starts when the map is loaded. The level gets over if the remaining time is zero without giving the player any reward.

5.2.2 Activity Diagram



Wait User Input
[Exit Game]
User wants to go back to Main Menu
[User chooses level]
[User chooses Play Game option]
Mark user selected level as selected
Mark the last played level as selected
Load selected level and initialize the game
[User selects player car]
[User selects any other vehicle]
Player car selected
The vehicle is selected
[User wants to move]
[If there is no obstacle between player car and the end]
Move vehicle to the end
[Level finished]
Give reward according to move count
Ask user what to do
[If there is no obstacle in the selected direction]
Move vehicle
[If there is an obstacle in the selected direction]
Keep vehicle in the same place
Update Map
[One of the vehicles either moved or not, so the map should be updated according to it]
[User wants to continue playing, Pass to next level]

Figure 4 - Activity Diagram of the Game (For Readability Refer to Glossary)

5.3 Object and class model

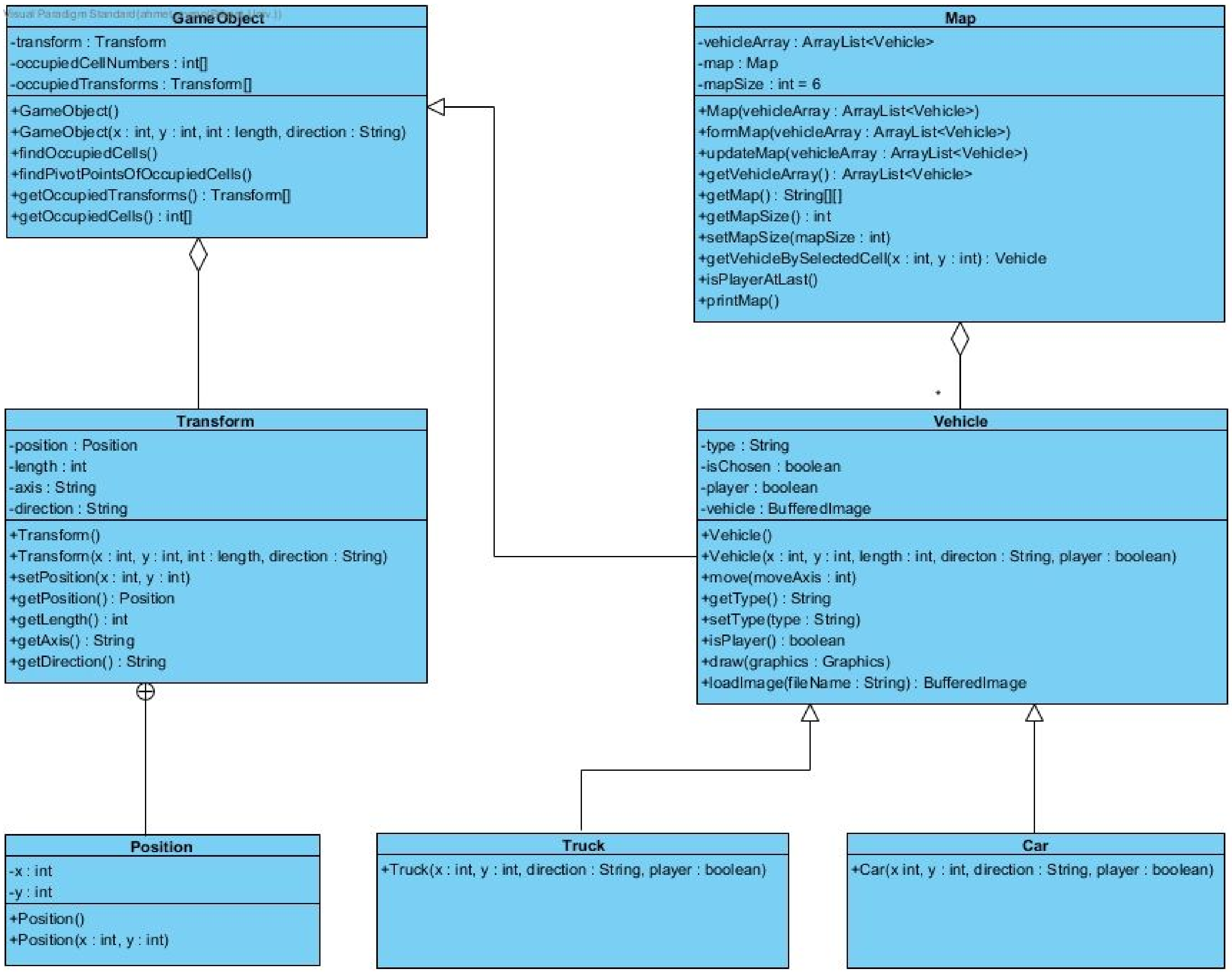

GameObject
-transform : Transform
-occupiedCellNumbers : int[]
-occupiedTransforms : Transform[]
+GameObject()
+GameObject(x : int, y : int, int : length, direction : String)
+findOccupiedCells()
+findPivotPointsOfOccupiedCells()
+getOccupiedTransforms() : Transform[]
+getOccupiedCells() : int[]
Map
-vehicleArray : ArrayList<Vehicle>
-map : Map
-mapSize : int = 6
+Map(vehicleArray : ArrayList<Vehicle>)
+formMap(vehicleArray : ArrayList<Vehicle>)
+updateMap(vehicleArray : ArrayList<Vehicle>)
+getVehicleArray() : ArrayList<Vehicle>
+getMap() : String[][]
+getMapSize() : int
+setMapSize(mapSize : int)
+getVehicleBySelectedCell(x : int, y : int) : Vehicle
+isPlayerAtLast()
+printMap()
*
Transform
-position : Position
-length : int
-axis : String
-direction : String
+Transform()
+Transform(x : int, y : int, int : length, direction : String)
+setPosition(x : int, y : int)
+getPosition() : Position
+getLength() : int
+getAxis() : String
+getDirection() : String
Vehicle
-type : String
-isChosen : boolean
-player : boolean
-vehicle : BufferedImage
+Vehicle()
+Vehicle(x : int, y : int, length : int, directon : String, player : boolean)
+move(moveAxis : int)
+getType() : String
+setType(type : String)
+isPlayer() : boolean
+draw(graphics : Graphics)
+loadImage(fileName : String) : BufferedImage
Position
-x : int
-y : int
+Position()
+Position(x : int, y : int)
Truck
+Truck(x : int, y : int, direction : String, player : boolean)
Car
+Car(x int, y : int, direction : String, player : boolean)

Figure 5 - Class Diagram of the Game (Problem Domain)

In our design, “Map” class is responsible for holding all the game objects that the player can act with such as vehicles which is represented by the “Vehicle” class. There is two type of vehicles in the game, car and truck. Each object in our design extends “GameObject” class which holds a “Transform” class for handling translations of position, rotation and scaling. As we are interested in the Problem Domain in the Requirements Analysis phase we did not include our Controller classes such as GameEngine, GameManager or MapController in our diagram as they are considered in the Solution Domain of our project.

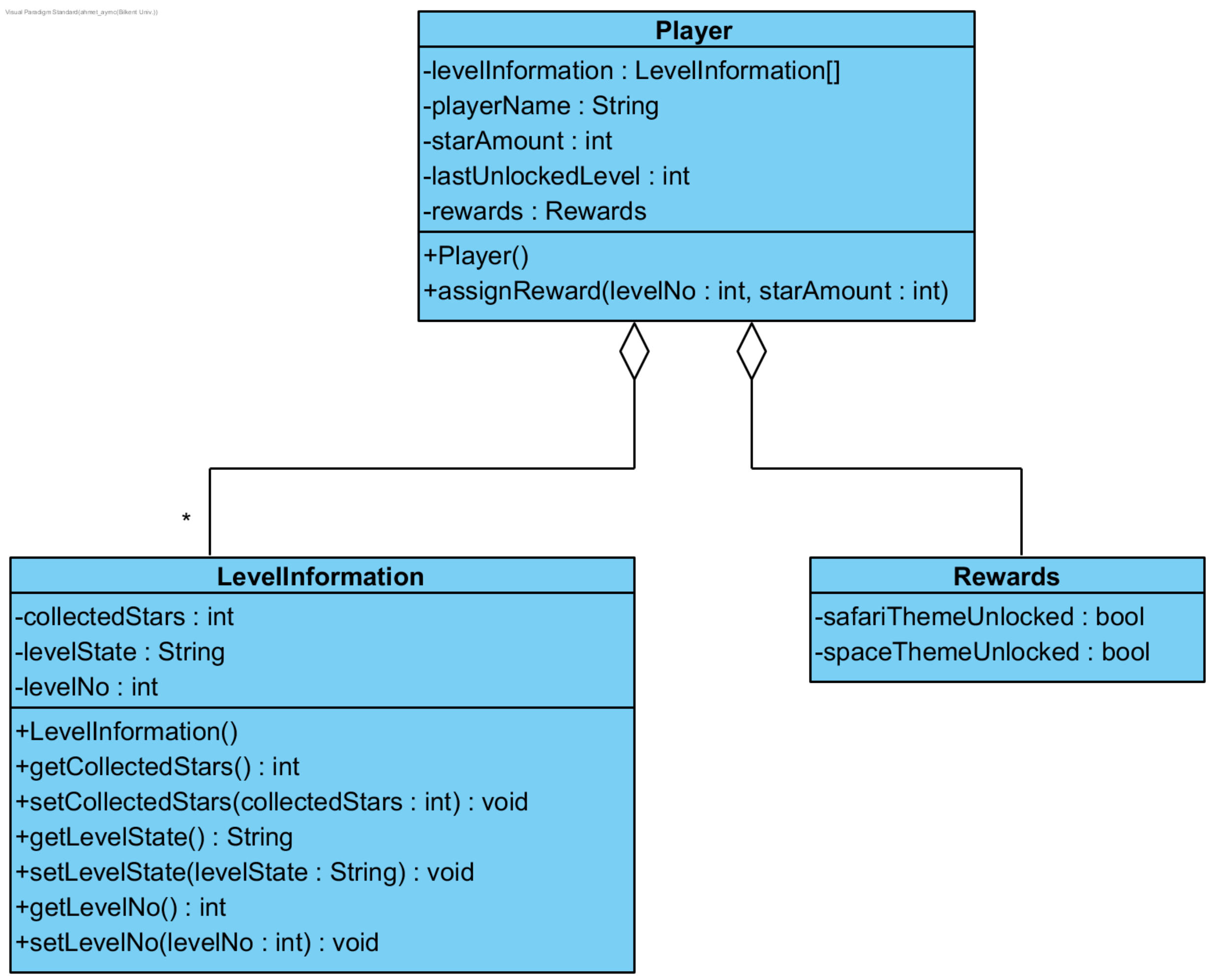

Visual Paradigm Standard(ahmet_ayrnc(Bilkent Univ.))
Player
-levelInformation : LevelInformation[]
-playerName : String
-starAmount : int
-lastUnlockedLevel : int
-rewards : Rewards
+Player()
+assignReward(levelNo : int, starAmount : int)
*
LevelInformation
-collectedStars : int
-levelState : String
-levelNo : int
+LevelInformation()
+getCollectedStars() : int
+setCollectedStars(collectedStars : int) : void
+getLevelState() : String
+setLevelState(levelState : String) : void
+getLevelNo() : int
+setLevelNo(levelNo : int) : void
Rewards
-safariThemeUnlocked : bool
-spaceThemeUnlocked : bool

Figure 6 - Class Diagram of Player

In our design “Player” class represent a single user account and holds all the information about that particular account. It holds information about the levels inside the “LevelInformation” class, such as the level state and the stars gotten from that level. Additionally, it stores the rewards that the user account has gotten so far in “Reward” class.

5.4 User interface - navigational paths and screen mock-ups

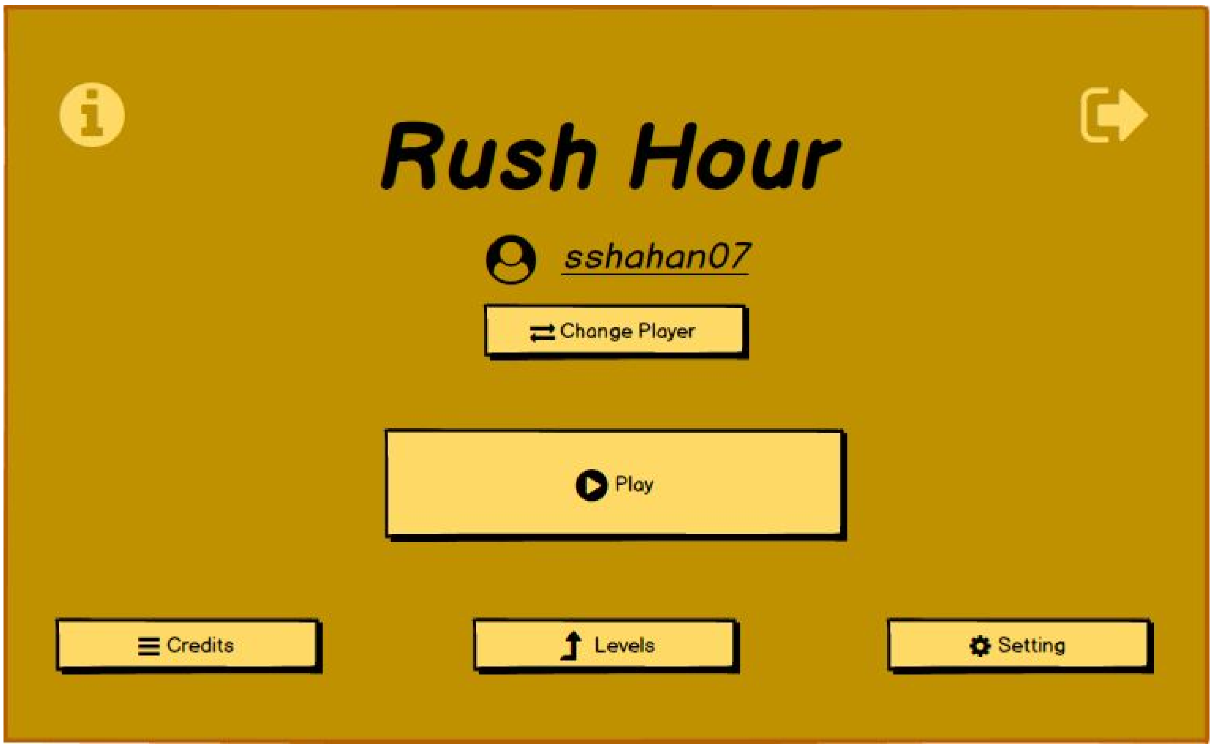


Figure 7 - Main Menu Screen

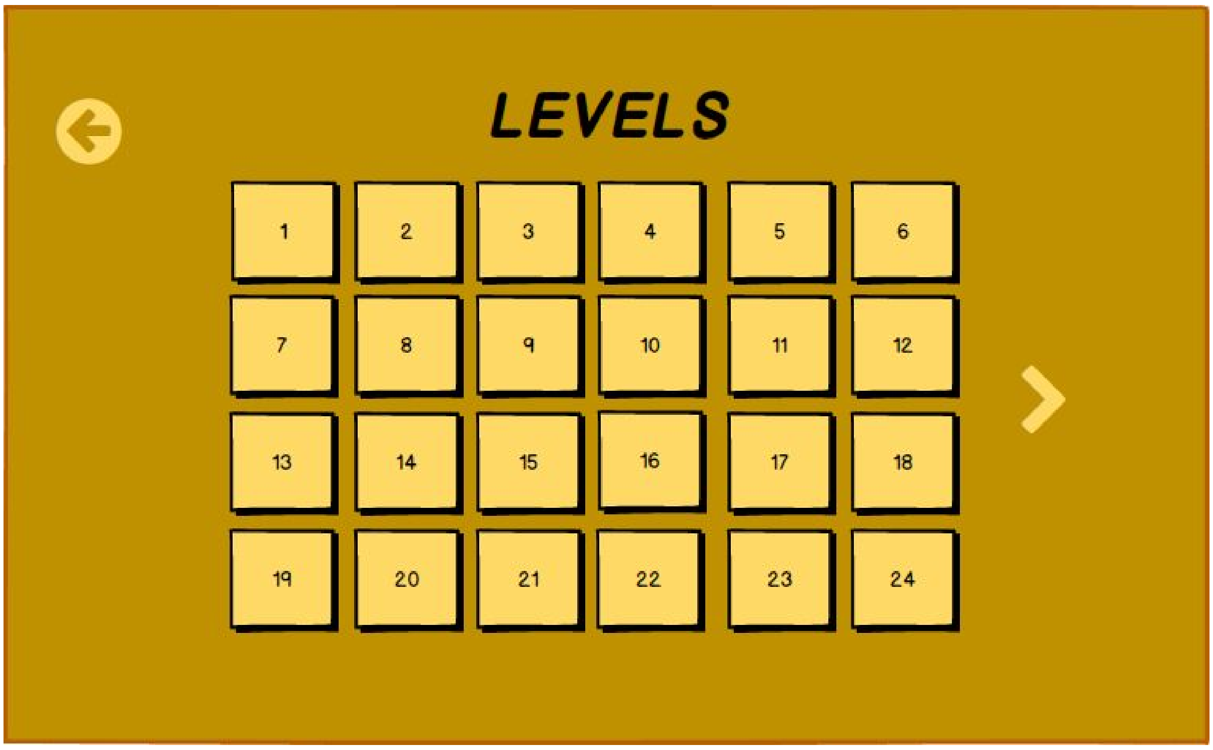


Figure 8 - Levels Screen

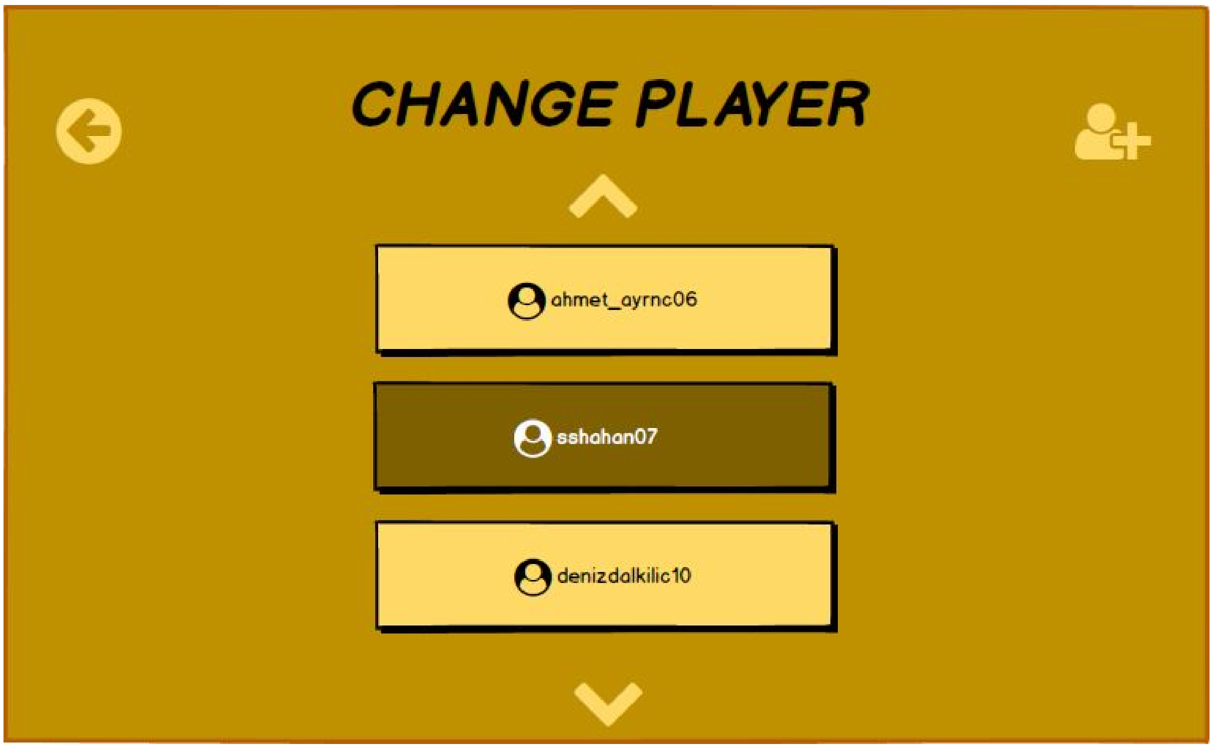


Figure 9 - Change Player Screen

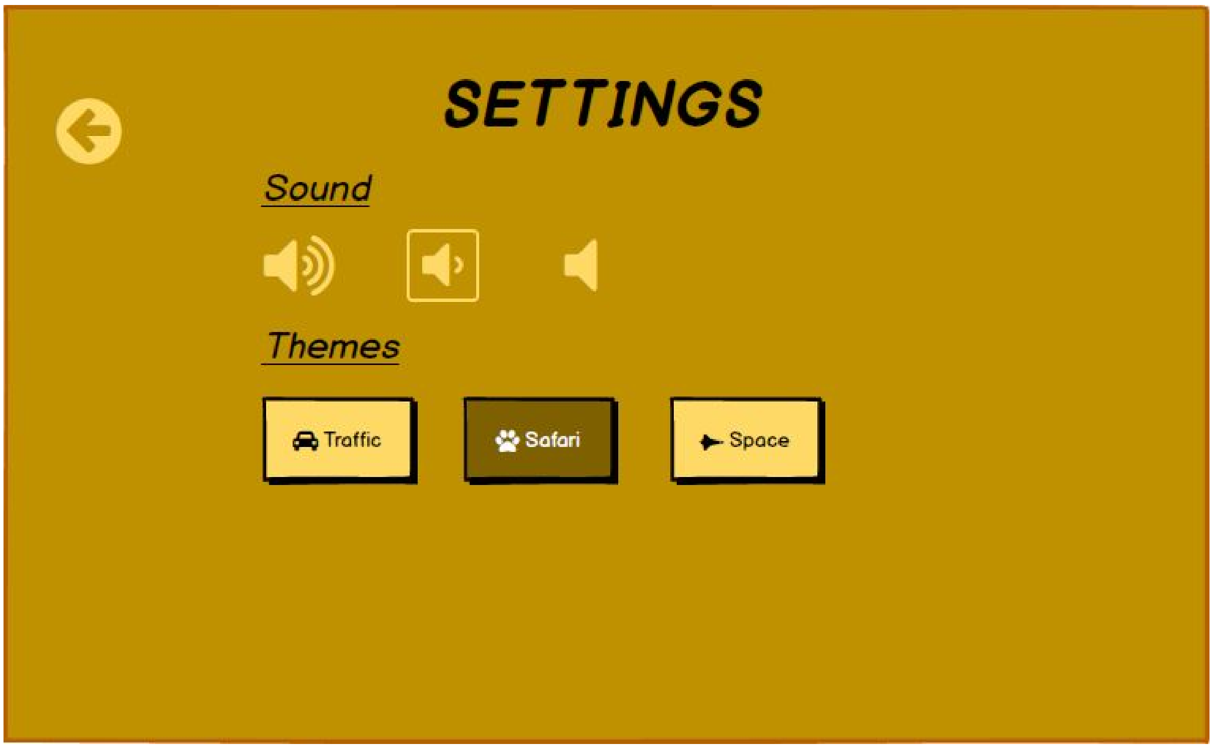
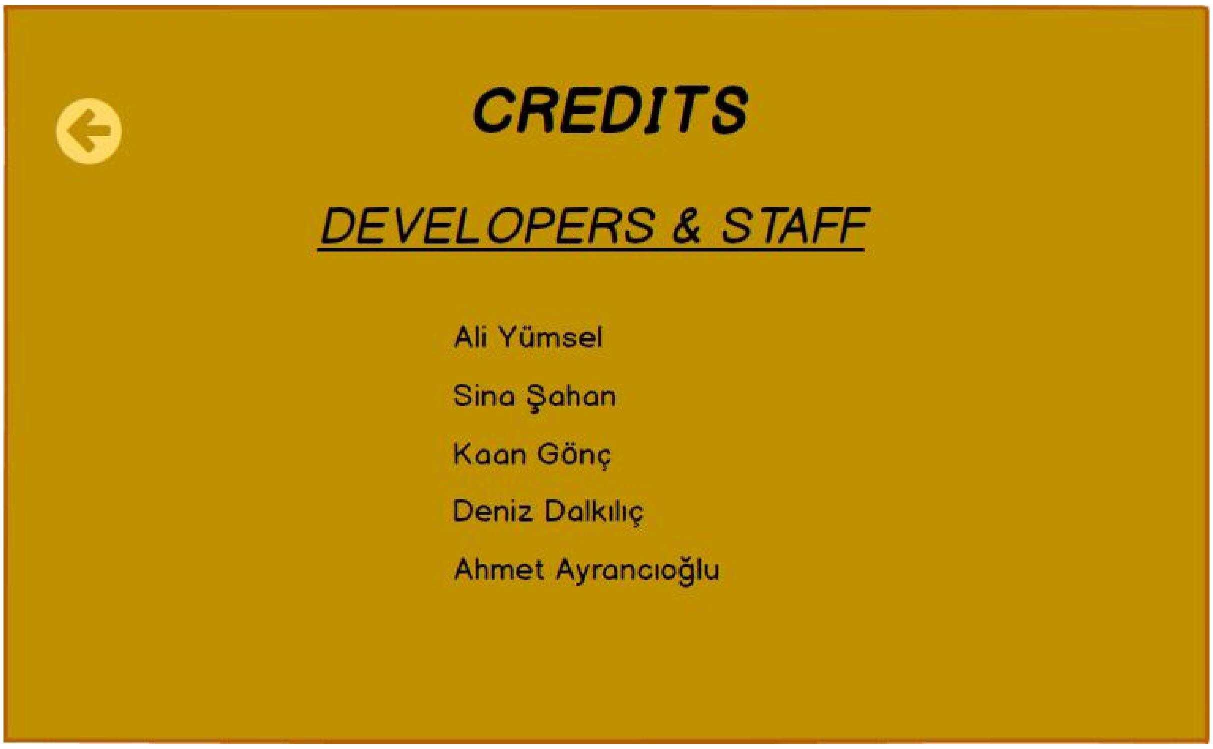


Figure 10 - Settings Screen


CREDITS

DEVELOPERS & STAFF

Ali Yümsel

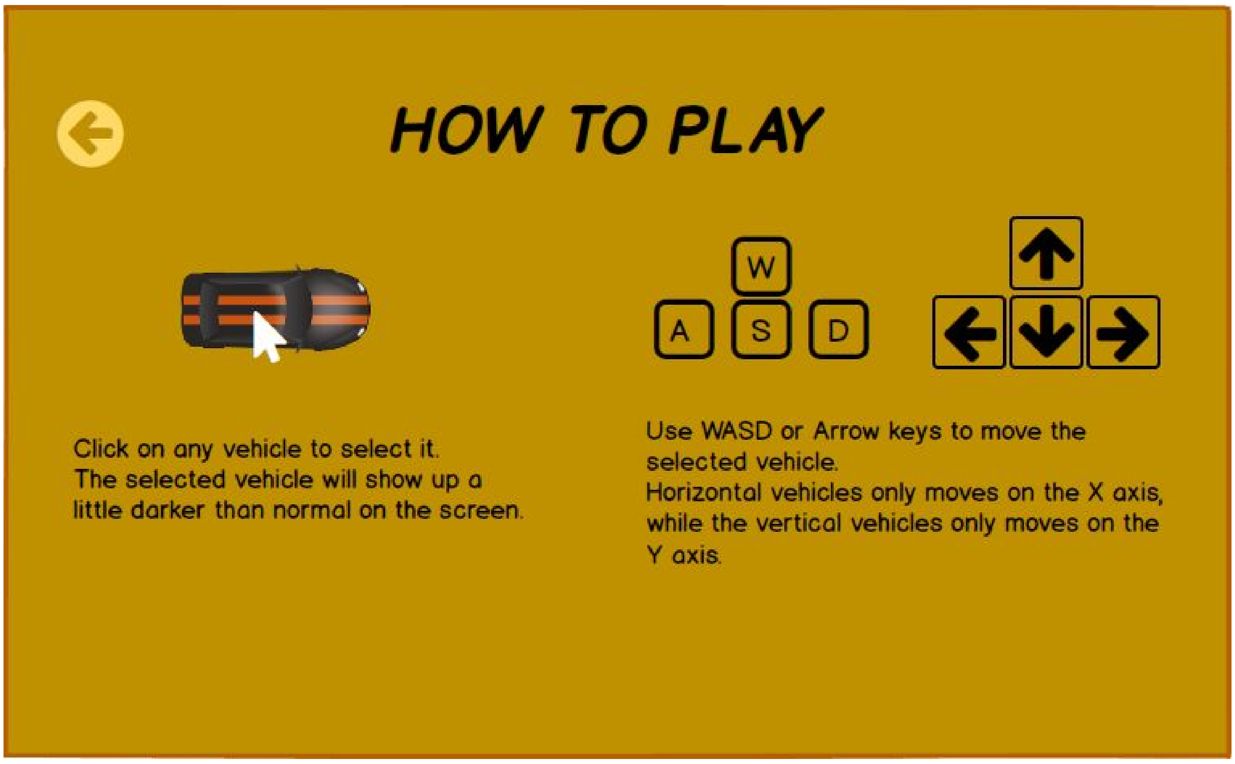
Sina Şahan

Kaan Gönç

Deniz Dalkılıç

Ahmet Ayrancıoğlu

Figure 11 - Credits Screen



Click on any vehicle to select it.
The selected vehicle will show up a little darker than normal on the screen.

Use WASD or Arrow keys to move the selected vehicle.
Horizontal vehicles only moves on the X axis, while the vertical vehicles only moves on the Y axis.

Figure 12 - How to Play Screen

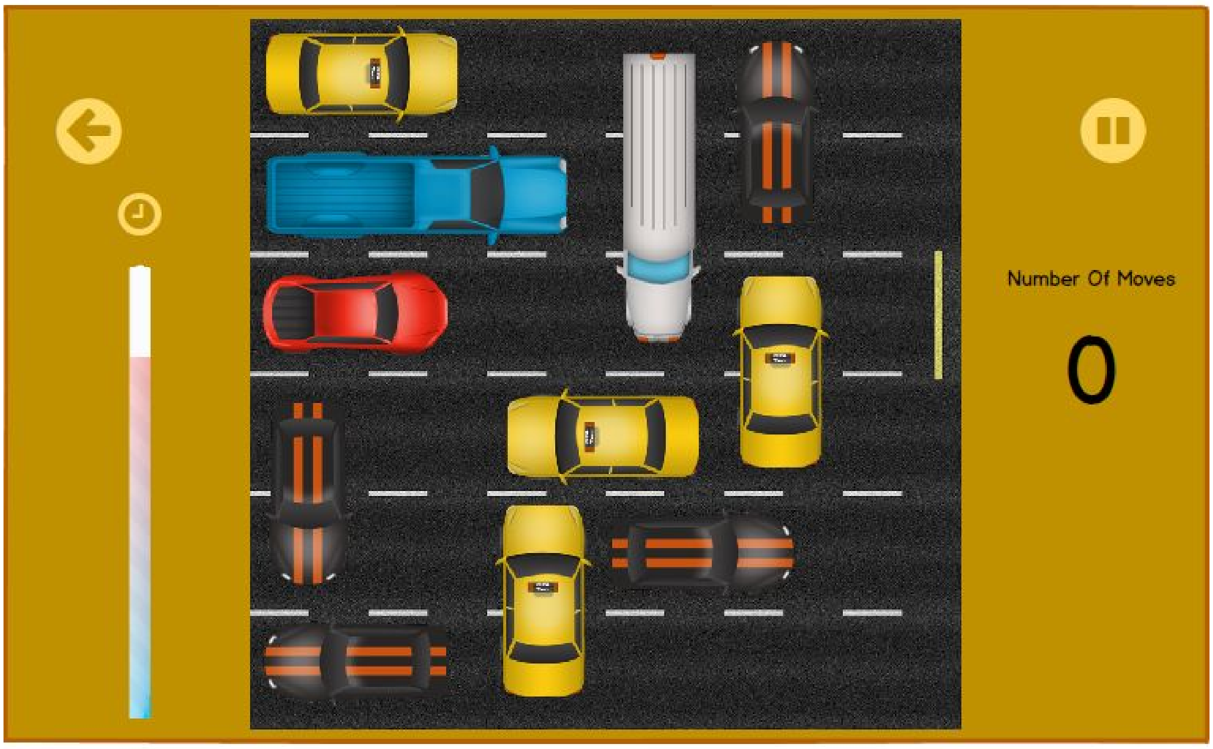


Figure 13 - Gameplay Screen of a Normal Level

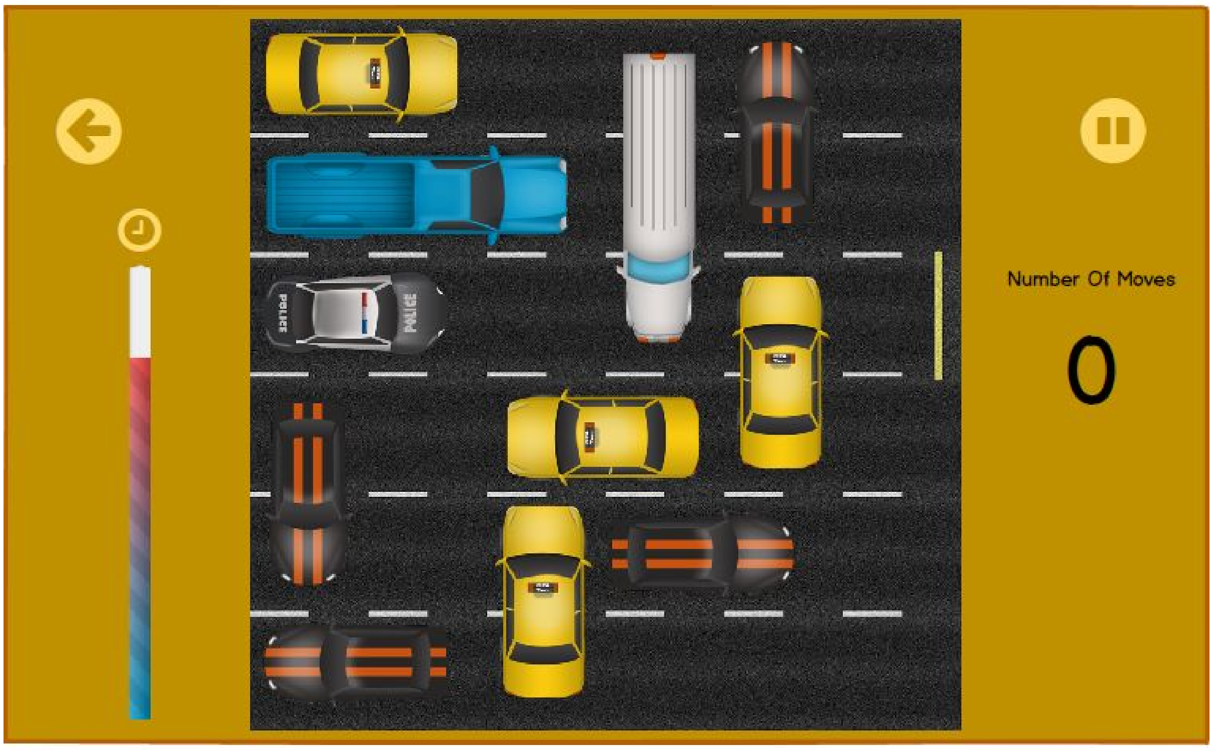


Figure 14 - Gameplay Screen of a Special Level

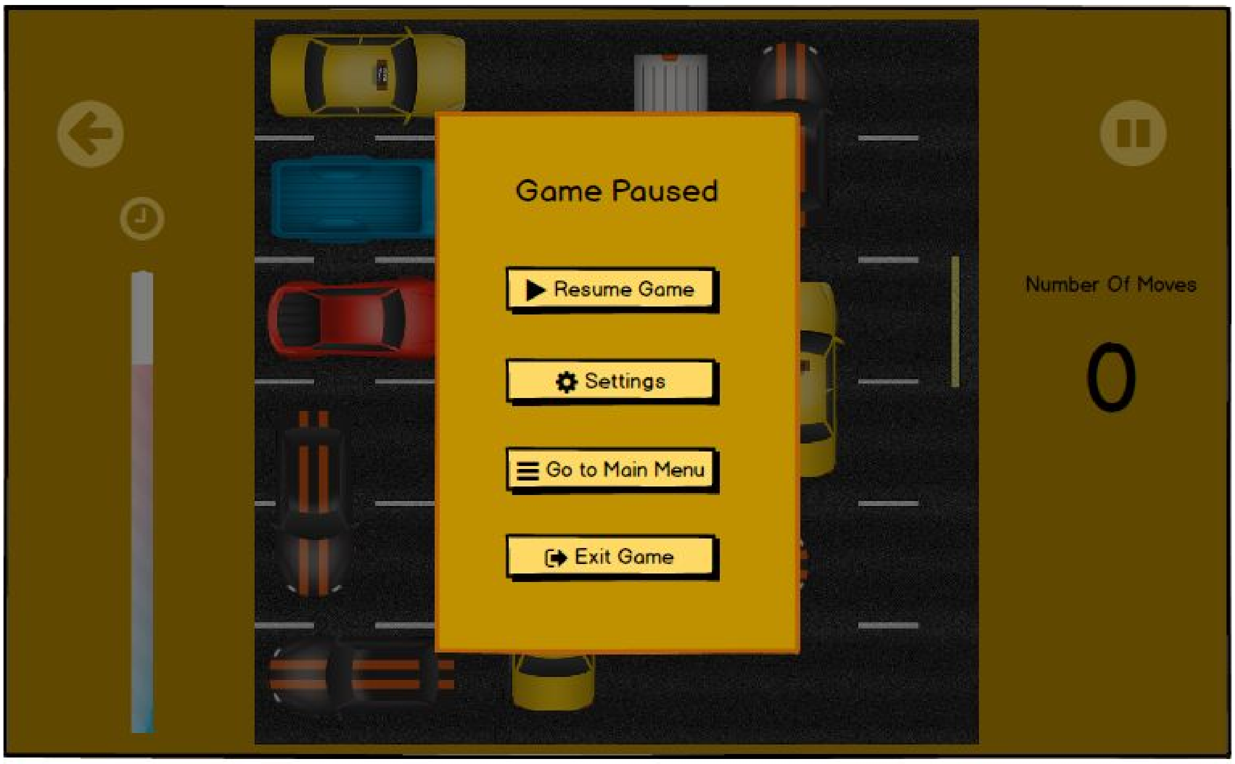
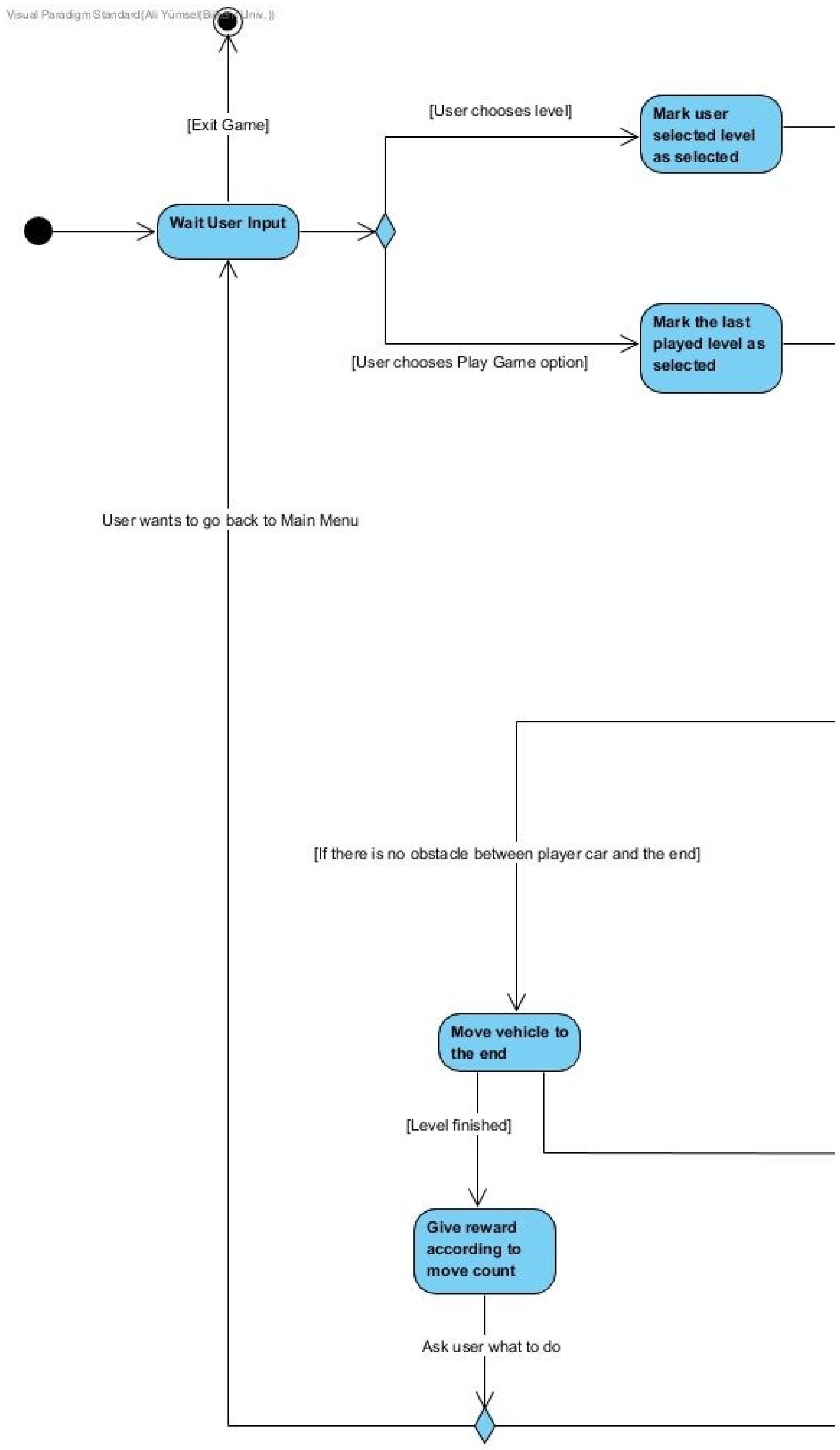


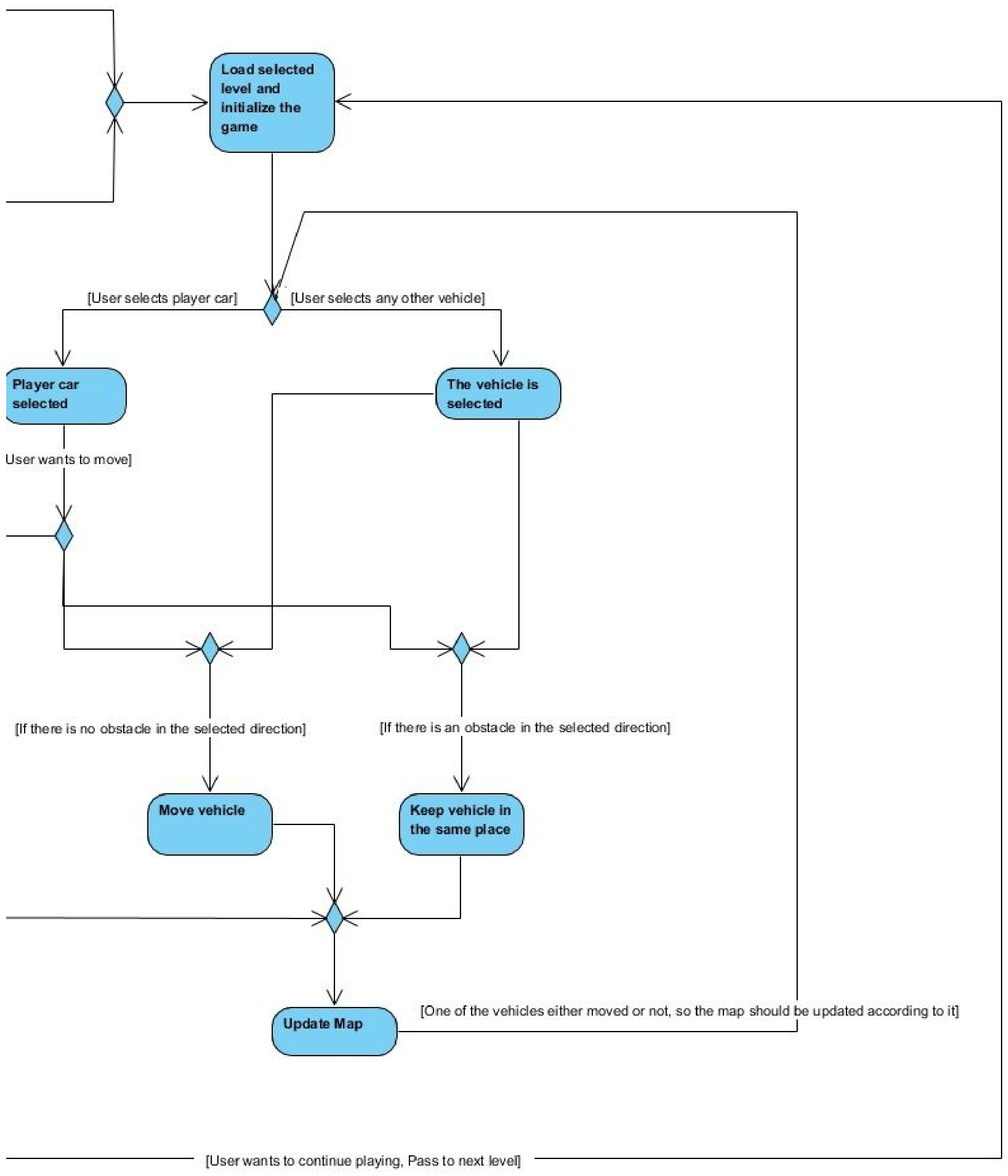
Figure 15 - In Game Pause Menu

6. Glossary

1. For readability purposes, the 2 part divided version of the Activity Diagram is added below.



Visual Paradigm Standard(Ali Yümsel(Bilkent Univ.))
[Exit Game]
[User chooses level]
Mark user selected level as selected
Wait User Input
Mark the last played level as selected
[User chooses Play Game option]
User wants to go back to Main Menu
[If there is no obstacle between player car and the end]
Move vehicle to the end
[Level finished]
Give reward according to move count
Ask user what to do


Load selected level and initialize the game
[User selects player car]
[User selects any other vehicle]
Player car selected
The vehicle is selected
User wants to move]
[If there is no obstacle in the selected direction]
[If there is an obstacle in the selected direction]
Move vehicle
Keep vehicle in the same place
Update Map
[One of the vehicles either moved or not, so the map should be updated according to it]
[User wants to continue playing, Pass to next level]

7. References

"Rush Hour." *Thinkfun,* https://www.thinkfun.com/products/rush-hour/
"How to Play Rush Hour", *Youtube*
https://www.youtube.com/watch?v=HI0rlp7tiZ0