BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS319
Object Oriented Software Engineering

Requirement Analysis Report
Iteration II
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. 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 of the game is to lead the player 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 as the player progresses through the levels. In every 10 levels, there will be a time limit and the player will play as an ambulance, a police car or a fire truck. 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. 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 themes and to unlock power ups to use on 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 on a 6x6 map, players will be able to add an empty grid to the desired location on the map as well as being able to transform a 3x1 object into 2x1 object. Such power ups will be earned according to how effectively they passed the previous levels. 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.

During the gameplay, the players will be able to reset or exit the game which will also 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

In this section, we will describe our game in 4 parts. Firstly, we will discuss the how to open the game and navigate the menus. Secondly, we will describe how to play the game. Thirdly we will describe about the elements of the game. FInally, we will list the rules of the game.

- **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 unlocked level:

The players will play the last unlocked 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. See Credits:

Upon pressing the “Credits” button in the main menu, the user must be directed to the Credits screen in which there is a list of developers that helped creating the game.

1.7. 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. **Using Powerups:**
   The user must be able to select powerups by pressing the power up button in the game panel. After pressing the power up button the user must be able to choose from 2 power up options, Form Extra Space and Change Vehicle.

   2.1. Form Extra Space
   After choosing the form extra space power up, the user must see all the available places that he can form extra space on and by pressing any of the available spaces, the user must be able to create a valid space at that position that any vehicle could move into.

   2.2. Change Vehicle
   After choosing the change vehicle power up, the user must see all the available trucks that he can change into cars and by pressing any of the available trucks, the user must be able to change the selected truck into a car.

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

4. **Reset Level:**
   The user will be able to reset the level to its original state by pressing the on screen button.

5. **Save the Game:**
   The game will automatically 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.

6. **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.

- **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 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:

- All classes used in the game must have a description explaining the overall function of the class. Additionally, all methods inside the game classes must have descriptions explaining the functionality of the method, a brief description of what the method returns and the meaning of parameters.
- The system should allow multiple people to work on the implementation at the same time by providing a maintainable and modular 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 editable even though some of the concepts could be considered not necessary at this state of the game.

Usability:

- 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. 95 out of 100 people should be able to start the game from the main menu when they open the game for the first time.
- 92 out of 100 people, who never played rush hour before, should be able to understand and play the game after seeing the tutorial in the help menu. The tutorial should be made using images that describe and show how tha game is played instead of long textual descriptions.

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. So 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 not take longer than 100 milliseconds.
- Transition between panels should not take longer than 200 milliseconds.

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.
- Java Swing must be used to create any panel in the project.

Interface Requirements:

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

5. System models

5.1. Use case models

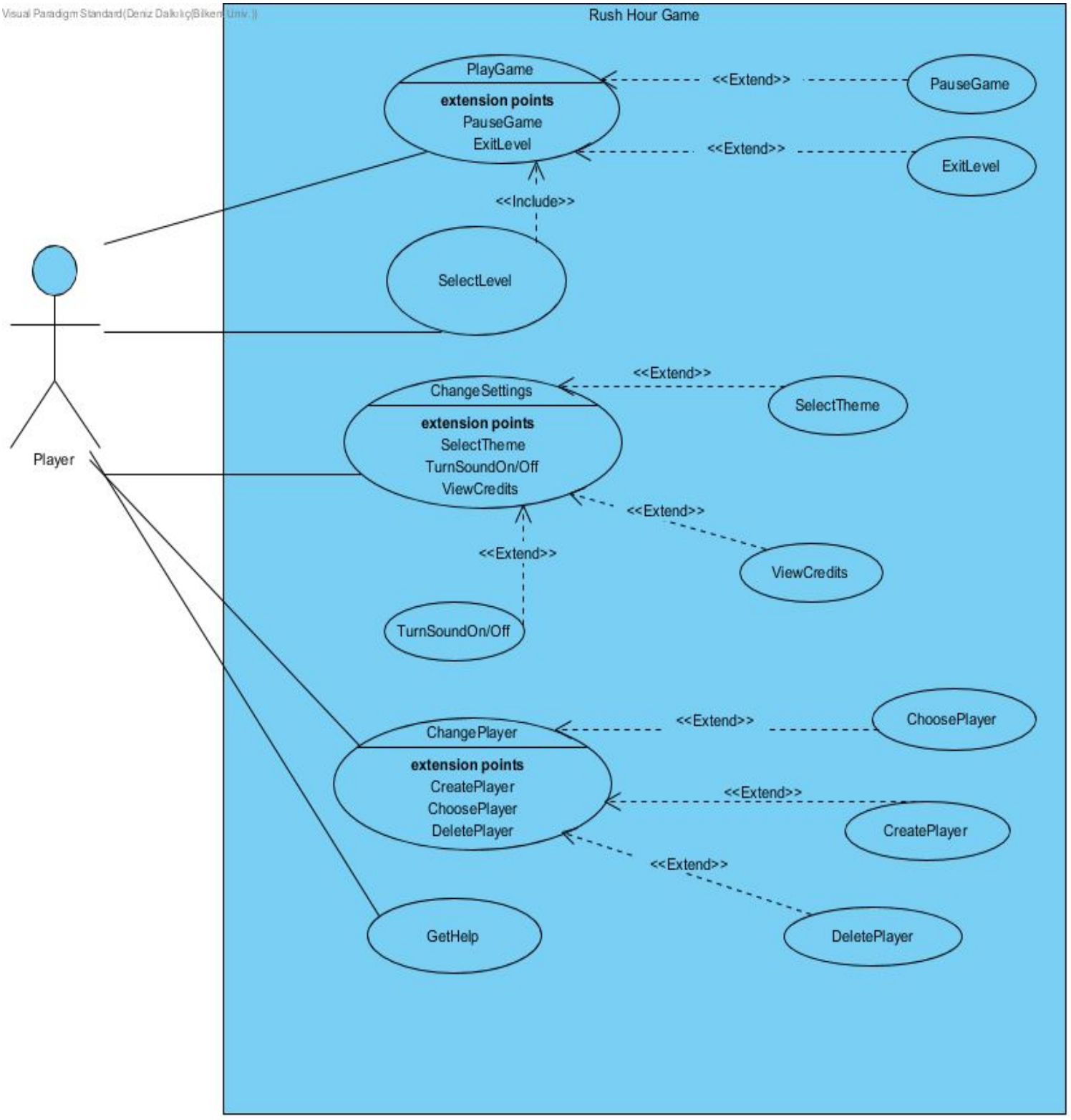


Figure 1 - Use Case Diagram

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
- 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 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
- Settings are adjusted to their selected values

Special Requirements

- Player can select a desired Theme
- Player can turn on and off both the sound effects and the theme music

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

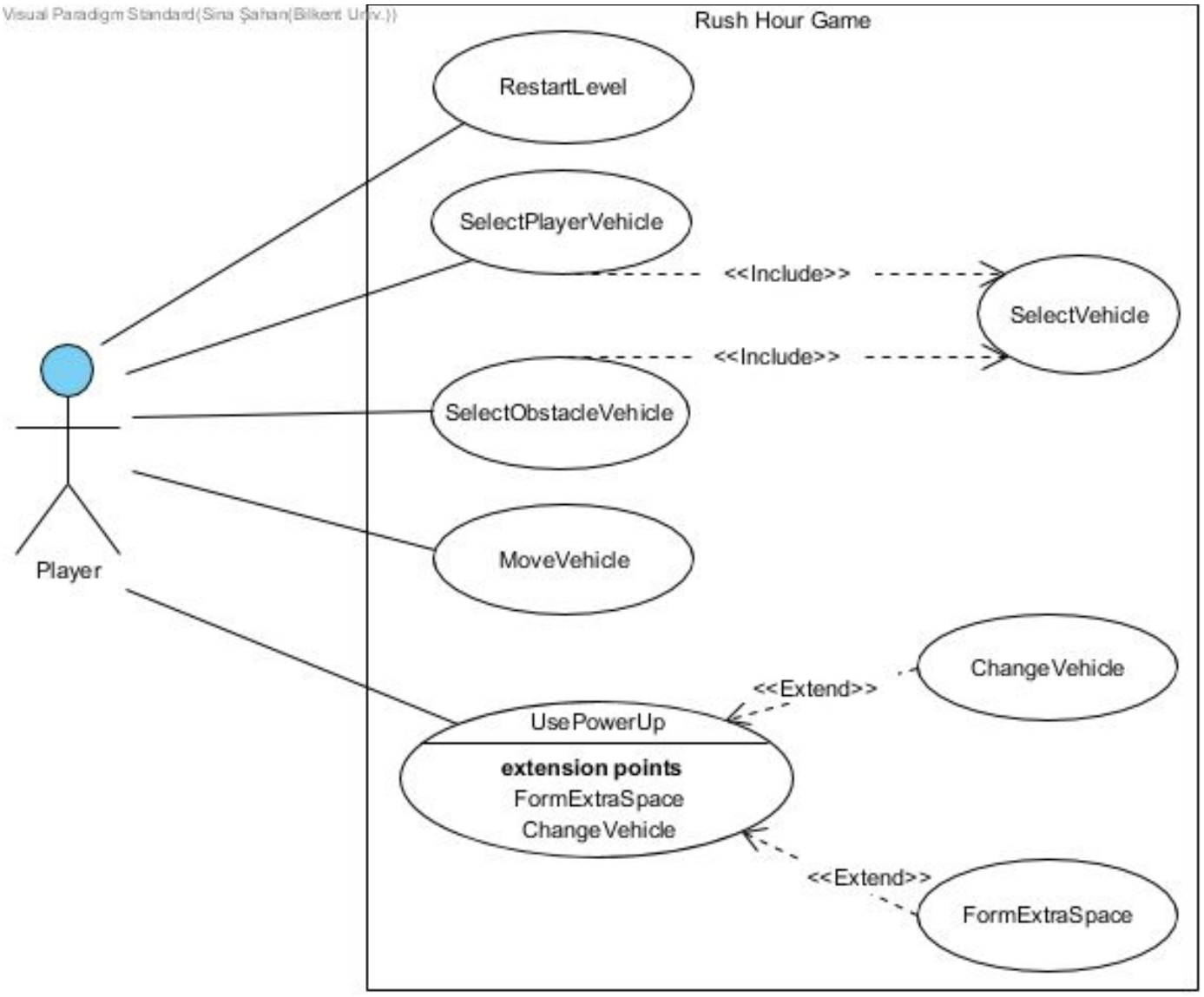


Figure 2 - Use Case Diagram (in game)

Description of Select Player Vehicle

Name: SelectPlayerVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.

Exit condition:

- Player is able to control the player vehicle by using the W,A,S,D keys.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player clicks on the player vehicle.
- The system identifies the player vehicle as selected.
- The player vehicle is highlighted.

Description of Select Obstacle Vehicle

Name: SelectObstacleVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.

Exit condition:

- Player is able to control the selected obstacle vehicle by using the W,A,S,D keys.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player clicks on the desired obstacle vehicle.
- The system identifies the obstacle vehicle as selected.
- The obstacle vehicle is highlighted.

Description of Select Vehicle

Name: SelectVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.
- Player has clicked the player vehicle with the mouse.

Exit condition:

- Player has clicked a vehicle with the mouse.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player clicks on the desired vehicle.
- The system identifies the vehicle as selected.
- The vehicle is highlighted.

Special Requirements

- none

Description of Move Vehicle

Name: MoveVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.
- Player has already selected a vehicle.

Exit condition:

- Player has successfully moved the selected vehicle.
  OR
- Player tried but could not move the selected vehicle as it is blocked.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player selects a vehicle.
- The player presses the movement keys to move the selected vehicle.
- The selected vehicle is moved in the wanted direction.

Exceptional Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player selects a vehicle.
- The player presses the movement keys to move the vehicle.
- Selected vehicle does not move, because the selected vehicle is blocked or the movement input is perpendicular to the vehicle.

Description of UsePowerUp

Name: UsePowerUp

Participating actor: Player

Entry condition:
- The level is loaded and the vehicles are shown on the map.

Exit condition:
- Player is successfully directed to the power up selection panel.

Successful Scenario Event Flow:
- The level is loaded and the vehicles are shown on the map.
- The player presses the power up button.
- The system activates the power up selection panel.

Description of FormExtraSpace

Name: FormExtraSpace

Participating actor: Player

Entry condition:
- The level is loaded and the vehicles are shown on the map.

Exit condition:
- Player has successfully formed a space on the map.

Successful Scenario Event Flow:
- The level is loaded and the vehicles are shown on the map.
- The player presses the extra space power up button.
- The system activates the power up usage.
- The player selects an occupied grid on the map to make it unoccupied.
- The selected grid is set to unoccupied.

Exceptional Event Flow:
- The level is loaded and the vehicles are shown on the map.
- The player presses the extra space power up button.
- The system informs the player that he does not have any power ups left.

Description of Change Vehicle

Name: ChangeVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.

Exit condition:

- Player has successfully transformed a 3x1 obstacle into a 2x1 obstacle.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player presses the change player power up button.
- The system activates the power up usage.
- The player selects an 3x1 obstacle on the map to use the power up on.
- Selected obstacle is changed to an 2x1 obstacle

Exceptional Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player presses the power up button.
- The system informs the player that he does not have any power ups left

Description of Restart Level

Name: RestartLevel

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.

Exit condition:

- The map and move count are successfully initialized to their original states.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player plays the level.
- The player decides to restart the level.
- The player presses the restart button.
- The map is reloaded according to its original state (no progress) and move count is initialized to 0.

Description of Change Obstacle Vehicle

Name: ChangeObstacleVehicle

Participating actor: Player

Entry condition:

- The level is loaded and the vehicles are shown on the map.

Exit condition:

- Player has successfully transformed a 3x1 obstacle to a 2x1 obstacle.

Successful Scenario Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player presses the change player power up button.
- The system activates the power up usage.
- The player selects an 3x1 obstacle on the map to use the power up on.
- Selected obstacle is changed to an 2x1 obstacle

Exceptional Event Flow:

- The level is loaded and the vehicles are shown on the map.
- The player presses the power up button.
- The system informs the player that he does not have any power ups left

5.2 Dynamic Models

5.2.1 Sequence Diagrams

Classic Mode

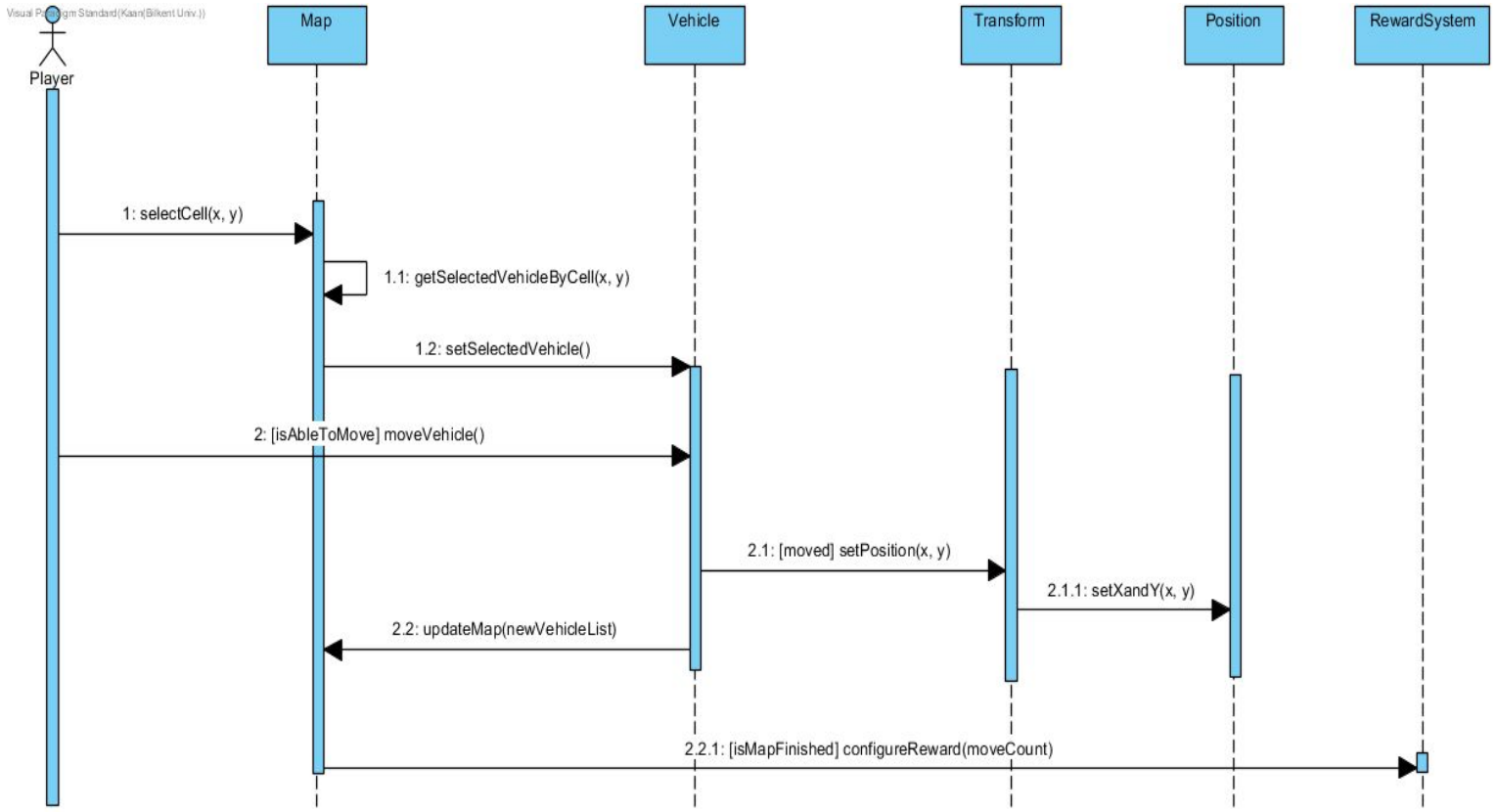


Figure 3 - Classic Mode Sequence Diagram

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 number of stars the player will earn.

Bonus Mode

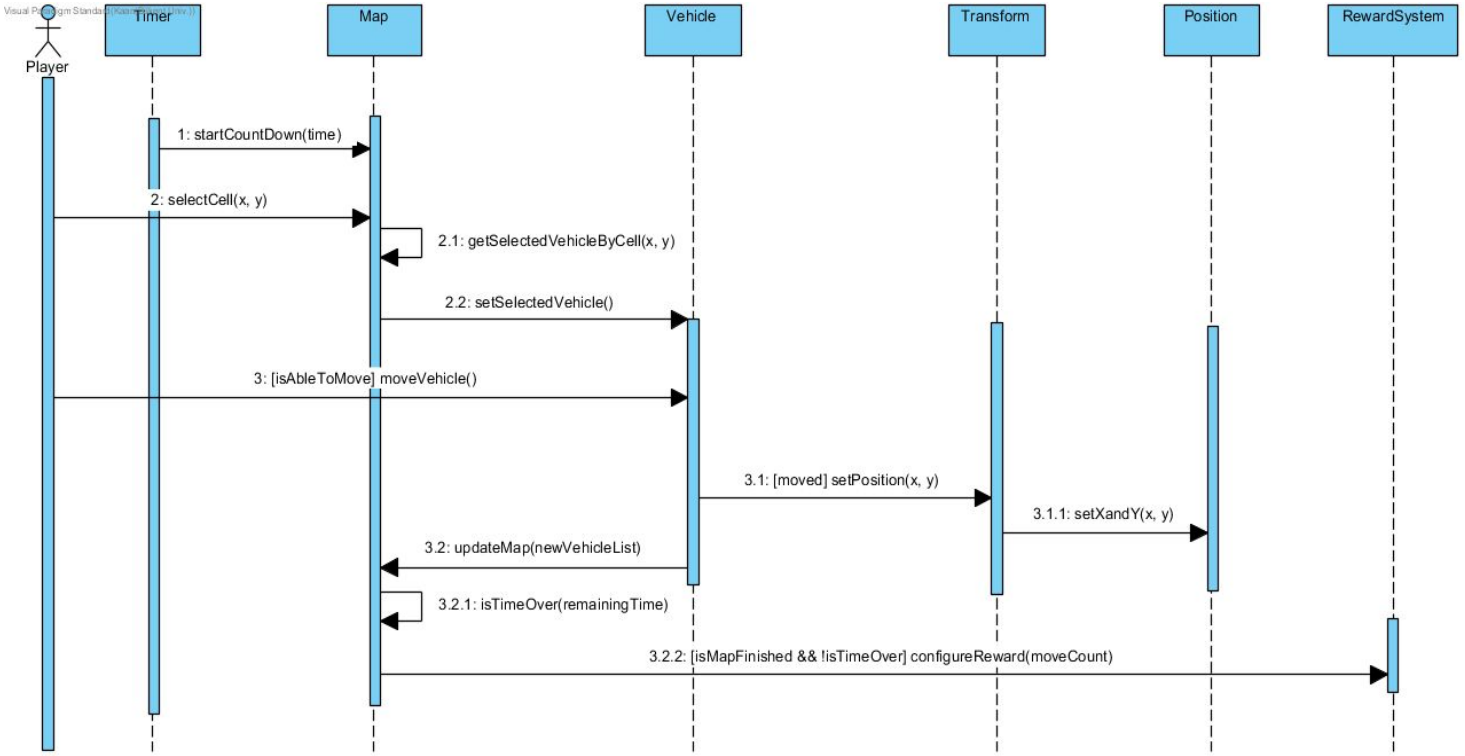


Figure 4 - Bonus Mode Sequence Diagram

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 stars.

5.2.2 Activity Diagram

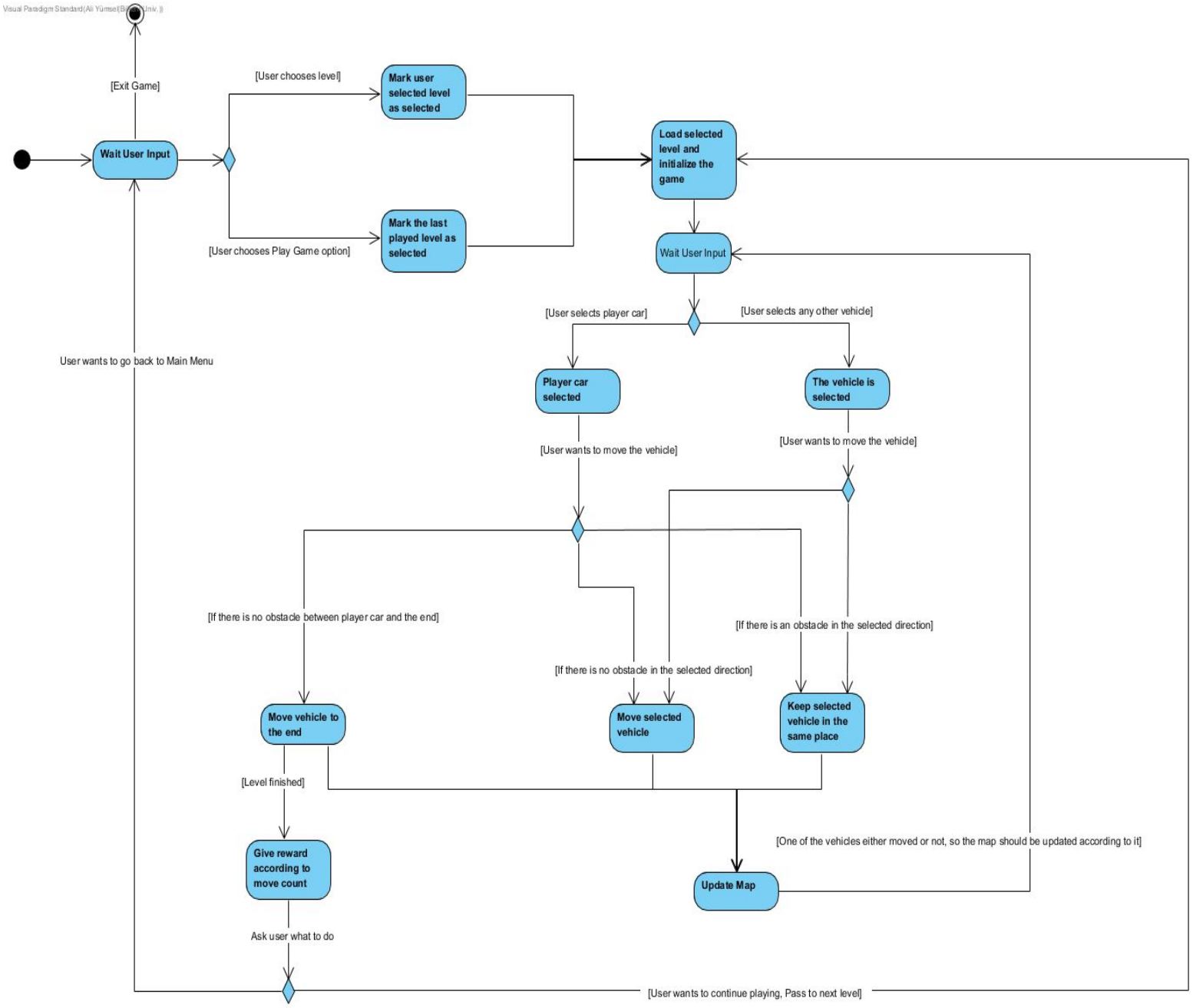


Figure 5 - Activity Diagram of the Game (For Readability Refer to Appendix)

The activity diagram of opening and playing a level.The game waits for player to select a level either from the main menu, or from the levels screen. When the user selects the level, the game is loaded with the selected level and the game starts to wait for user input to select a vehicle. After selecting a vehicle, the game checks if the selected vehicle is the player vehicle or not. Then the player can move the vehicle forwards or backwards if there is no obstacles. After moving the vehicle, the game updates the map and checks if the level is finished or not.

5.2.3 State Diagrams

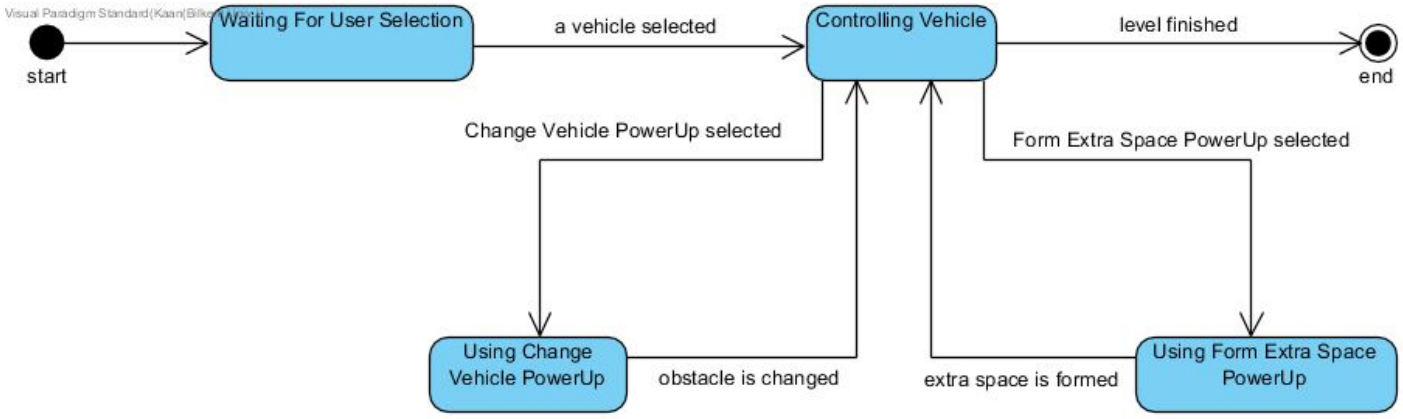


Figure 6 - Main State Diagram

This is the main state diagram of the game. When the game starts, it is waiting for a selection of a vehicle from the user. Then, the game continues with 'Controlling Vehicle' state which will be expressed detailly in Figure X. There are three potential outcomes of this state. If Change Vehicle Power Up gets selected, the state of using this power up occurs and finishes with an obstacle gets changed, then returns the 'Controlling Vehicle' state again. Else if, Form Extra Space Power Up gets selected, the state of using this power up occurs and finishes with an extra space gets formed, then returns the 'Controlling Vehicle' state again. Else if, level gets finished as an outcome, the state diagram ends.

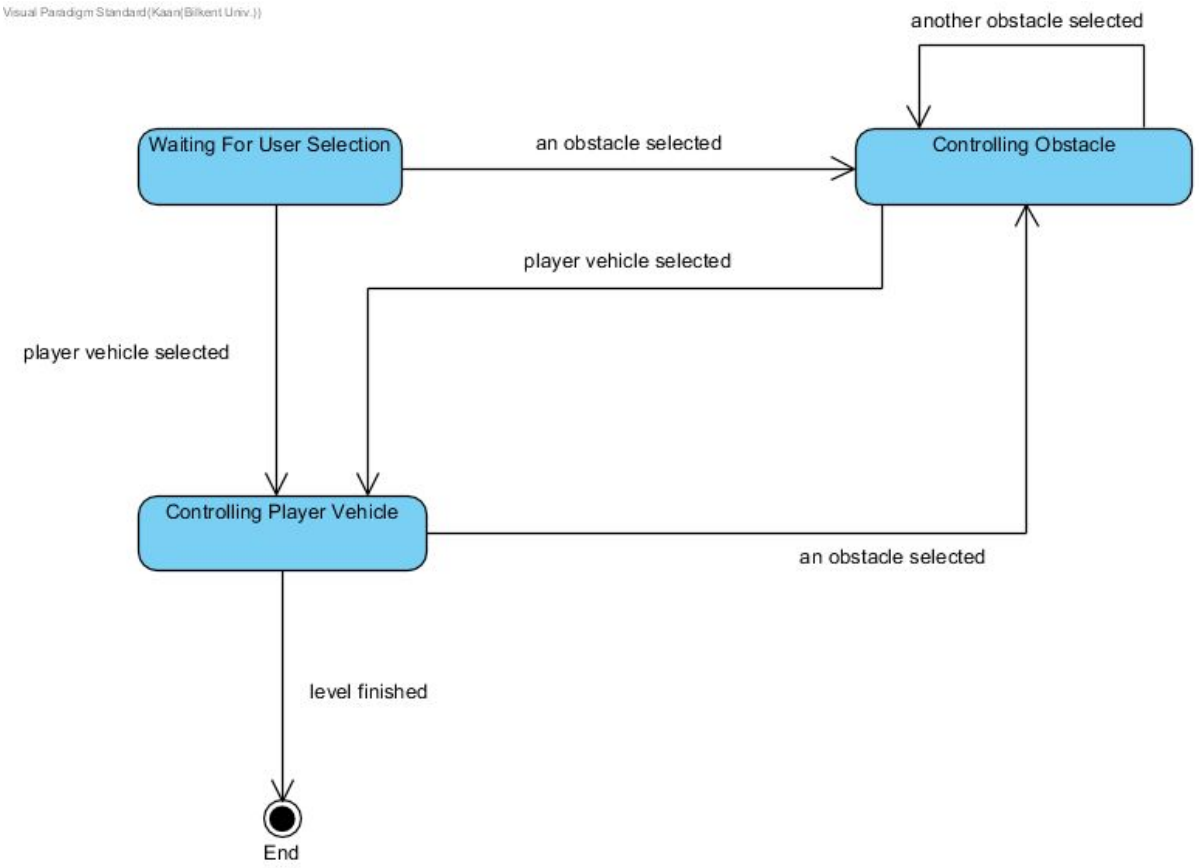


Figure 7- Controlling Vehicle State Diagram

This is the inner state diagram of ‘Controlling Vehicle’ state. The user has two options to get out of the ‘Waiting For User Selection’ state. If the user selects an obstacle to move, the game enters to ‘Controlling Obstacle’ state. Else if ‘ the user selects the player vehicle, the game enters to ‘Controlling Player Vehicle’ state. During the ‘Controlling Obstacle’ state, an obstacle is being moved. If another obstacle gets selected, ‘Controlling Obstacle’ state repeats itself for the new selected obstacle. Else if the user selects the player vehicle, the ‘Controlling Player Vehicle’ begins. In the ‘Controlling Player Vehicle’ state, the player vehicle is being moved. If an obstacle gets selected ‘Controlling Obstacle’ state begins. Else if, the player vehicle comes to the end of the map, the level finished and the diagram ends.

5.3 Object and class model

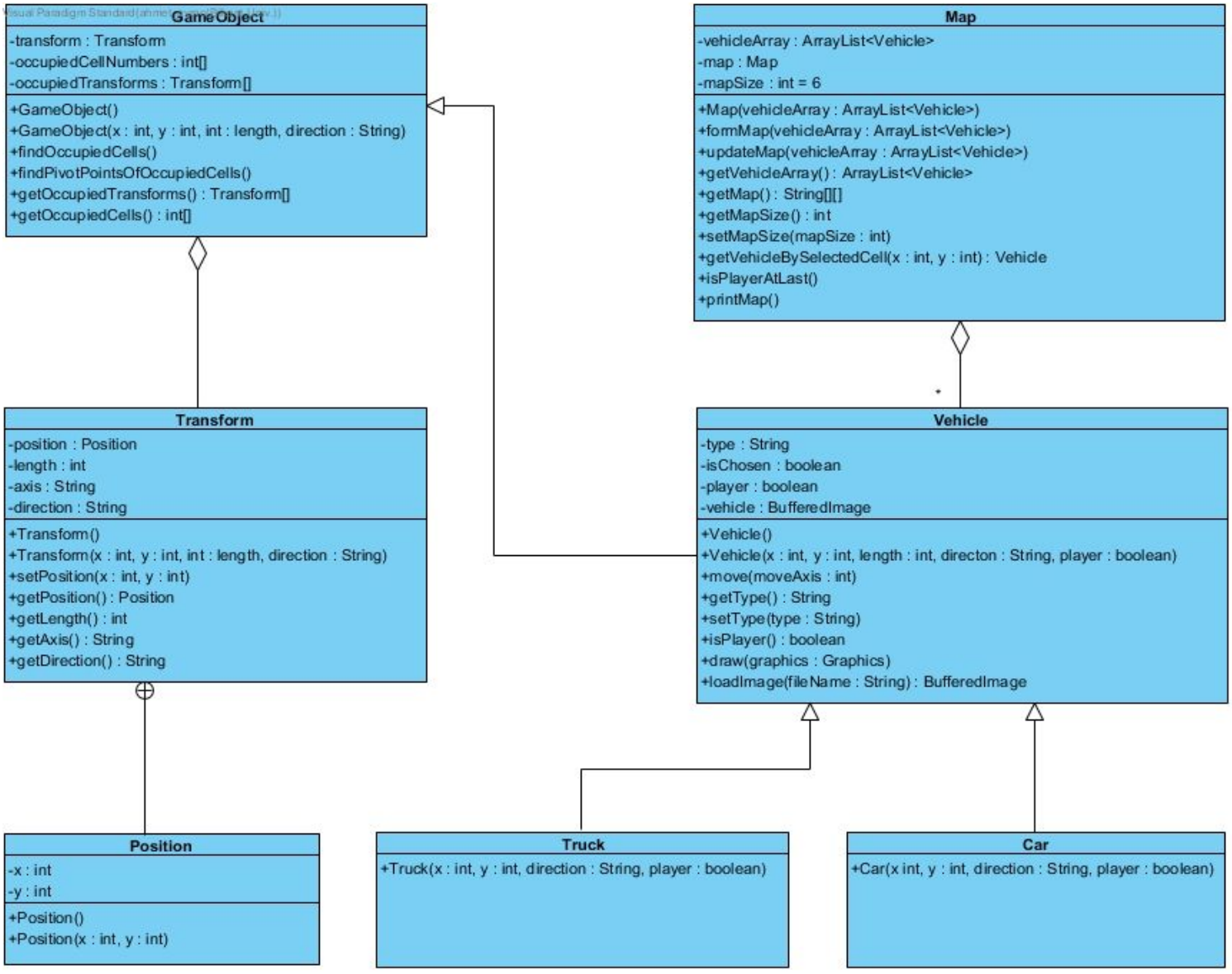


Figure 8 - Class Diagram of the Game (Problem Domain)

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.

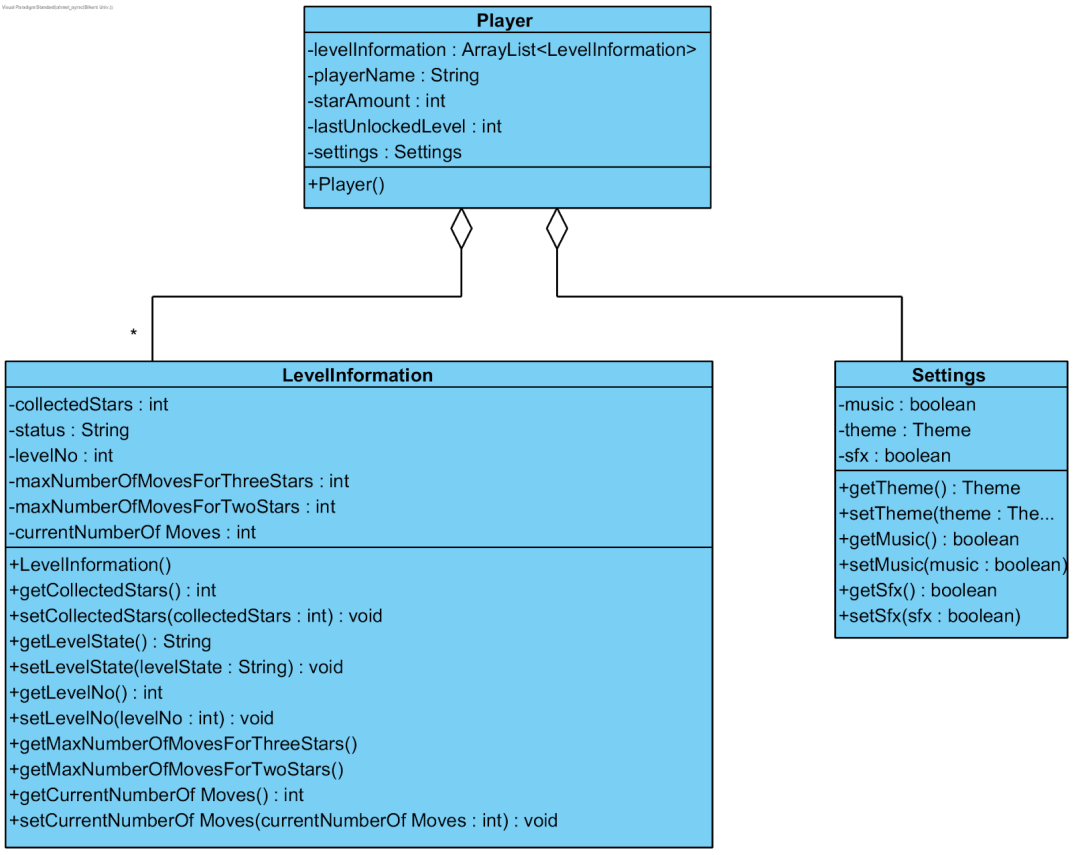


Figure 9 - Class Diagram of Player

In our design “Player” class represents 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 - Screen Mockups

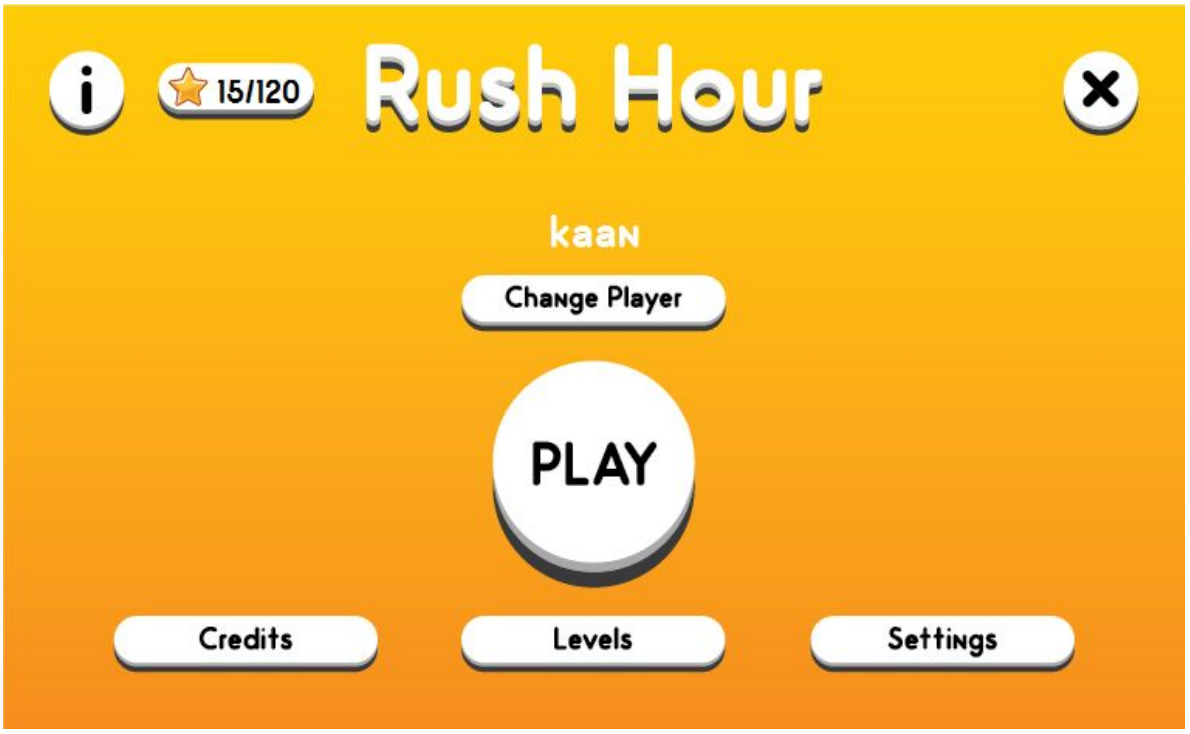


Figure 10 - Main Menu Screen

In the Main Menu screen, the player is able to open the last unlocked level by pressing the play button. The player can access the levels, credits, setting screens. Additionally, the player name is on top and and the player is able to change the player by pressing the change player. At top, the player can quit the game or get help when desired.

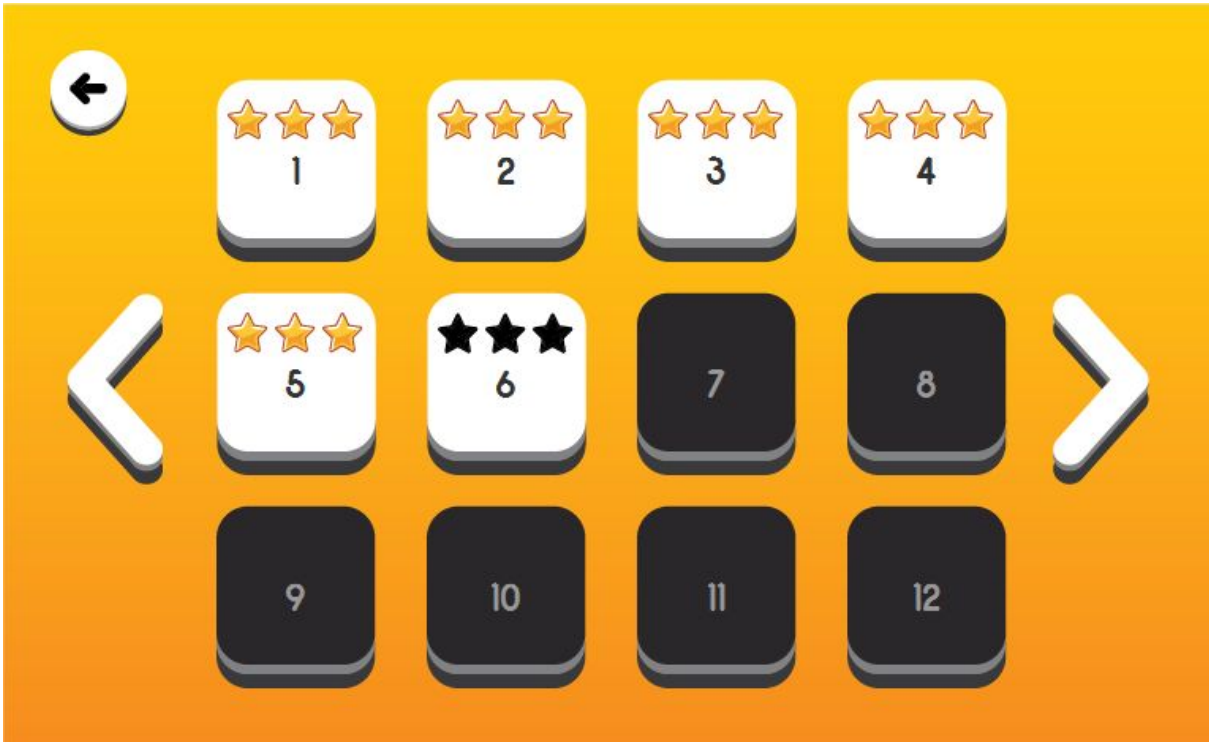


Figure 11 - Levels Screen

In the Levels Screen, the player can see available levels and the stars collected in those levels. The player can open any level by pressing the desired levels button. The player can also go back to the menu.

.

Figure 12 - Change Player Screen

In the Change Player Screen, player can see all the players, change active player, add and remove players.

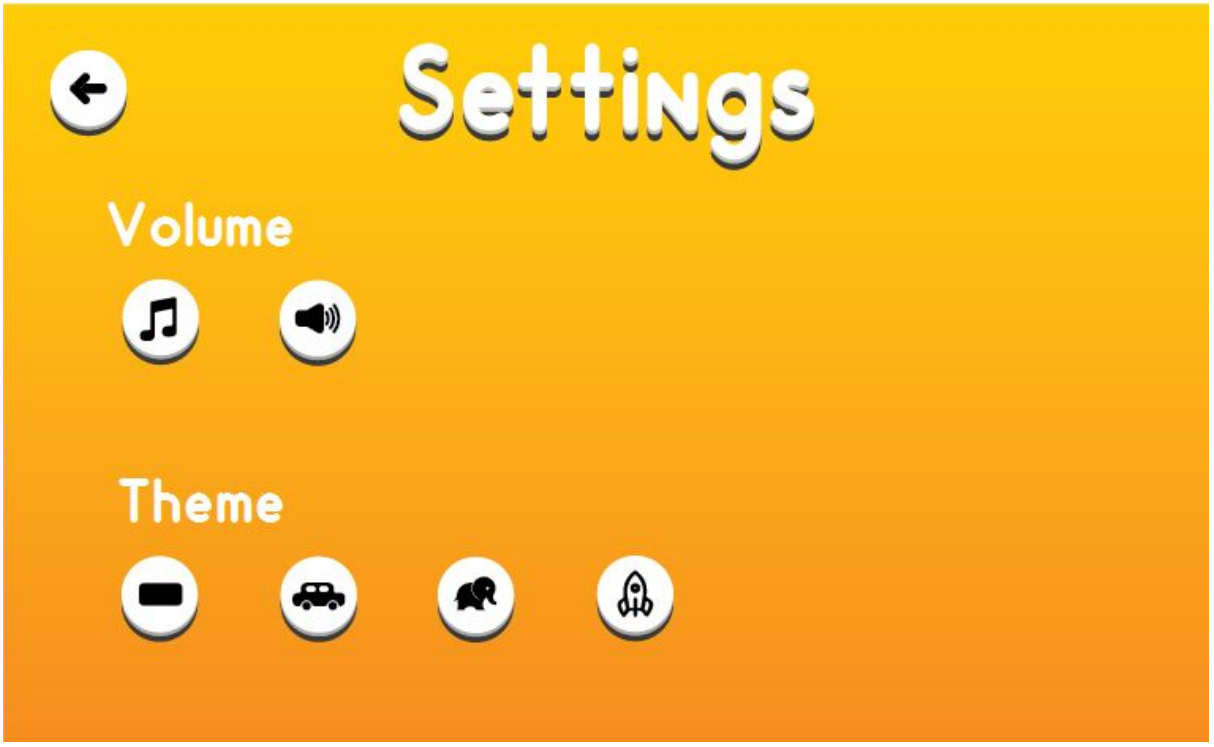


Figure 13 - Settings Screen

In Settings Screen, the player can turn the music and sfx on/off. The player is also able to change the themes which will be unlocked after playing through the levels. The player is also go back to the menu screen.

Figure 14 - Credits Screen

In the Credits Screen, the player can check the developers and the staff of the game. The player can again go back to main page.

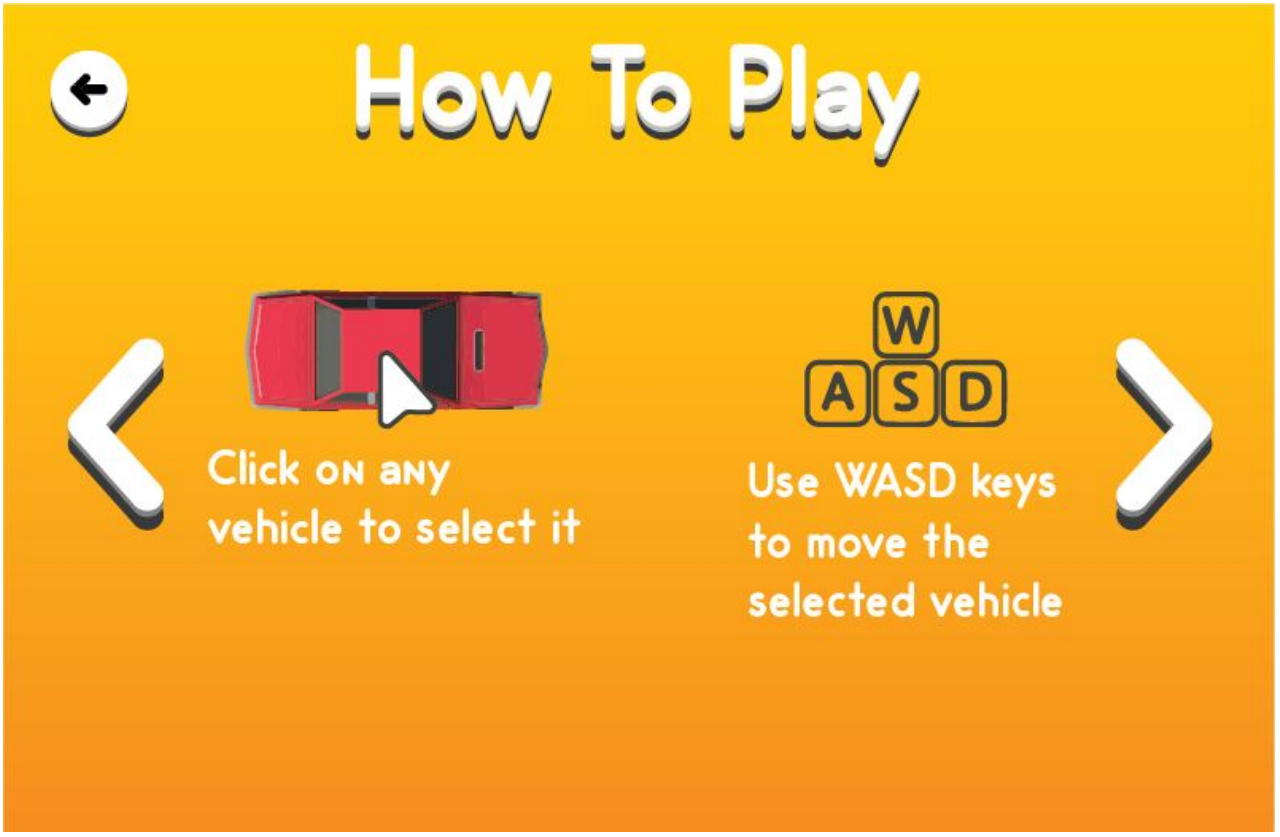


Figure 15 - How to Play Screen

In How to PlayScreen, the player can learn how to play the game by reading the available game manual and instructions. It is clearly demonstrated that the player should click to desired vehicle and could move to the regarding direction via A S W D which has same function with respected arrow keys. The player is also able to go back to the menu.

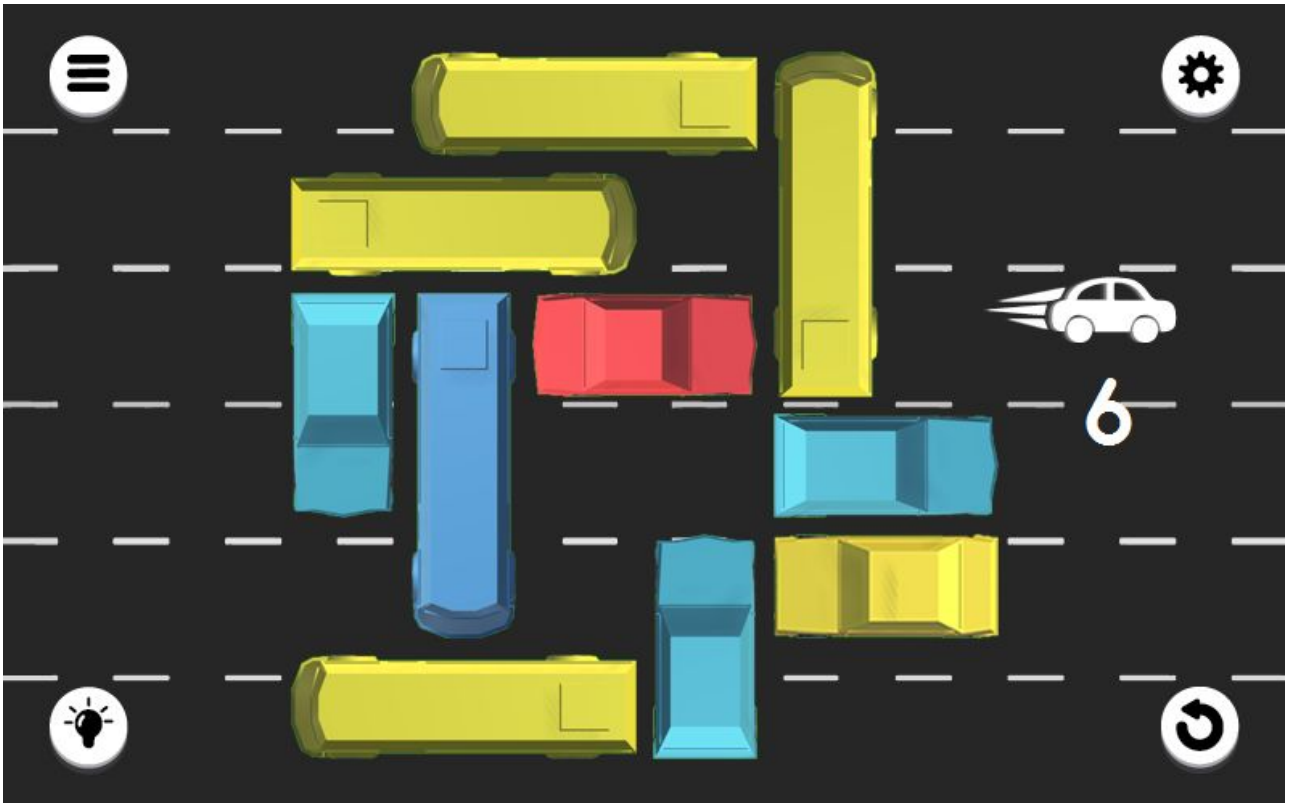


Figure 16 - Gameplay Screen of a Normal Level

The Gameplay Screen of a normal level is observable when the player decides to play normal level which has no time limit. The player can see his move count. The player can press the bottom left button to choose power up and use the selected power up. The player can press the restart button to reset the level. The player can press the settings button to change the settings of the game. Finally the player can press the main menu button to return to the main menu.

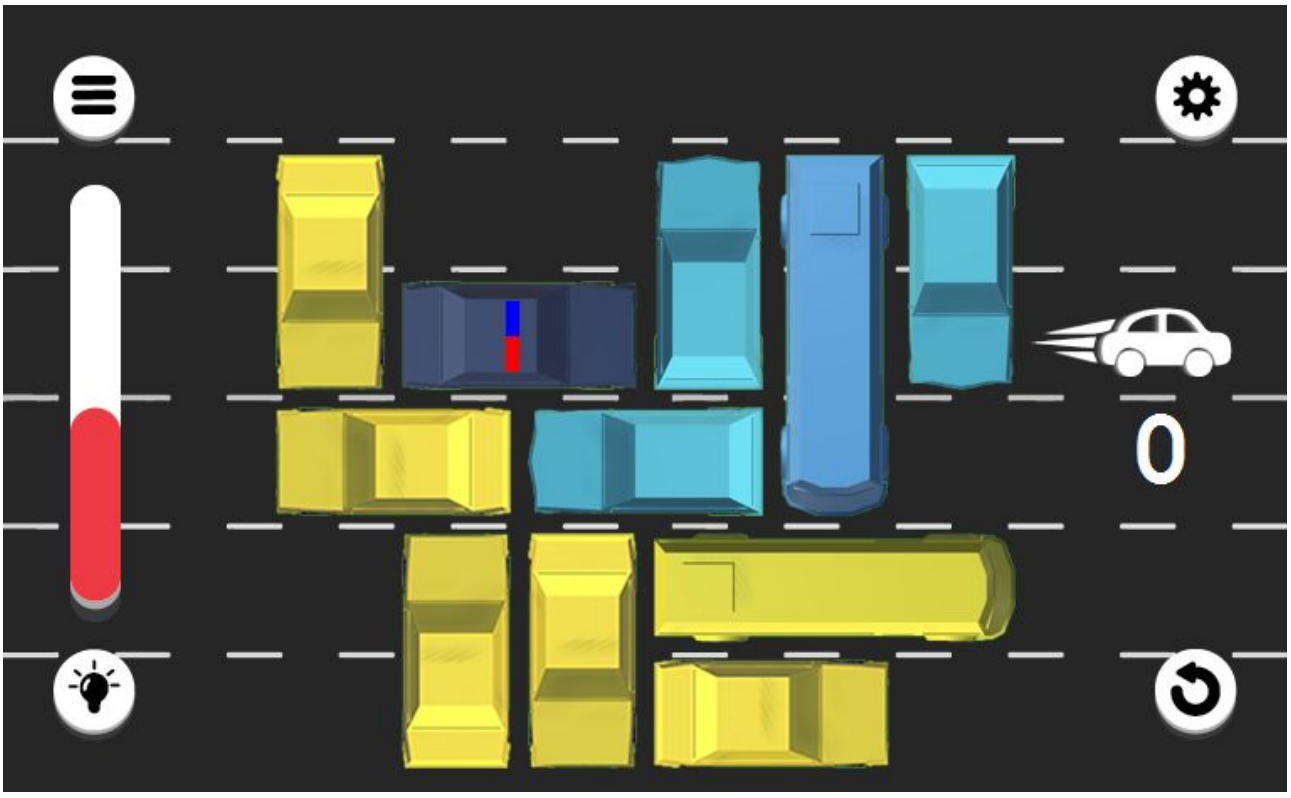


Figure 17 - Gameplay Screen of a Special Level

The special levels offer a gameplay which has special player vehicles that can be a police car, an ambulance or a fire-truck . In addition, the player has a specific time limit and player can observe his/her number of past moves in order to understand his/her award condition. The player should finish the level in specific time limit to succeed.

6. Improvement Summary

1. Mockup Designs are updated to have a simpler and modern look and feel.
2. New functional requirements are added such as the restart level function and the power-ups to ease the gaming experience of the player.
3. The new themes are added such as minimalistic theme.
4. Diagrams are updated such as Use Case and Activity diagram, also new diagrams (two State diagrams) are added.

.

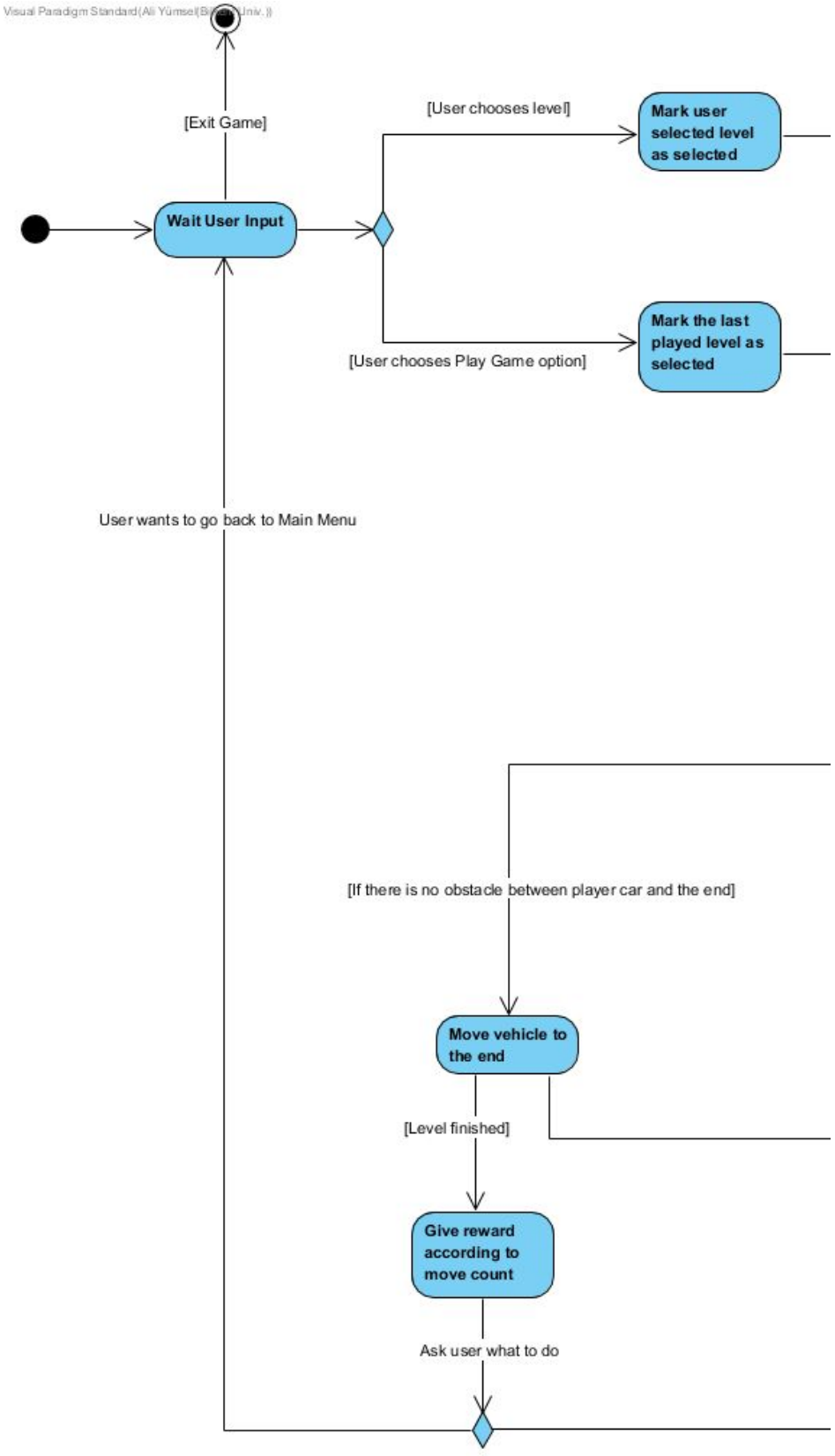
7. References

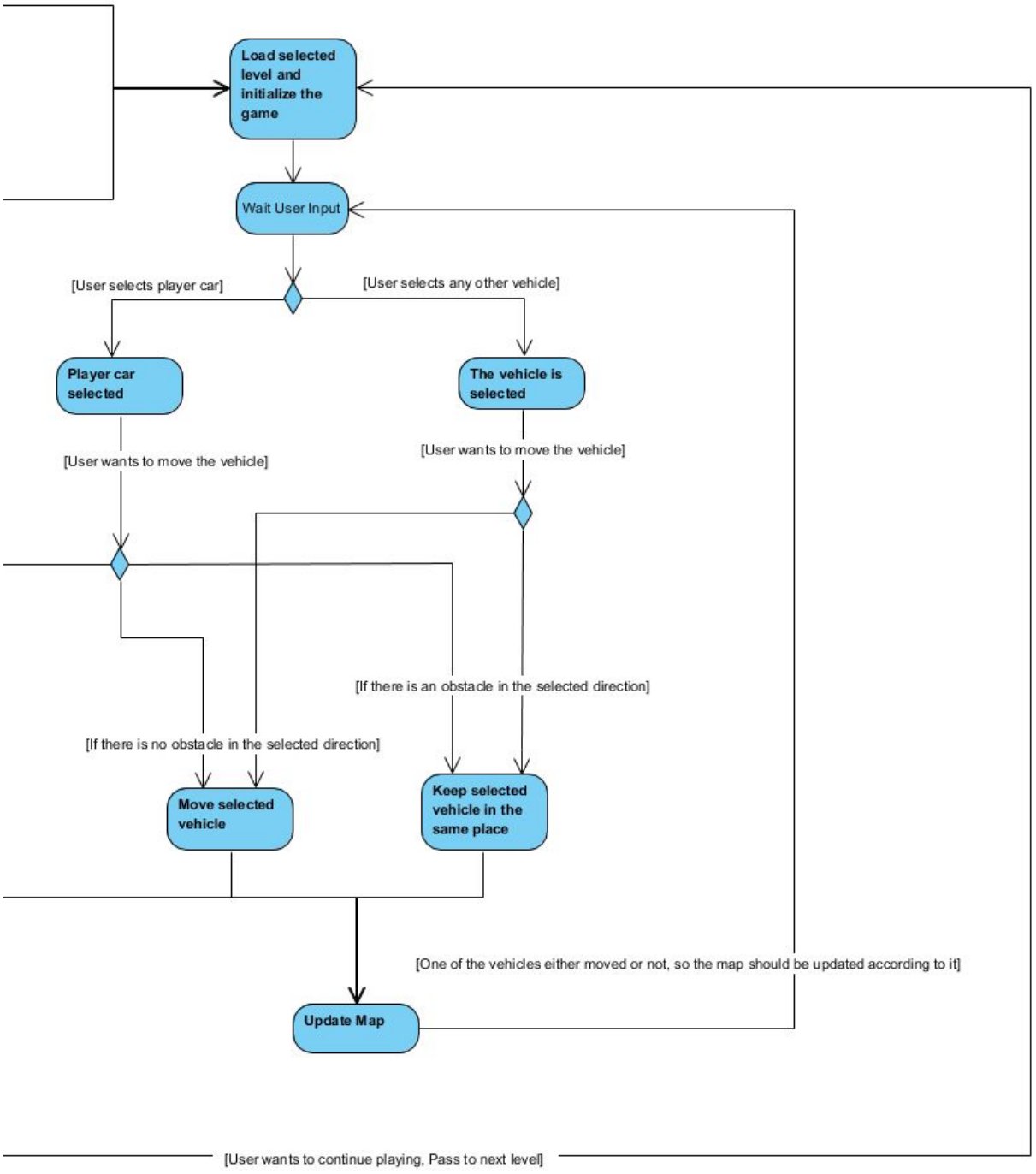
"Rush Hour." Thinkfun. November 24, 2018. https://www.thinkfun.com/products/rush-hour/

ThinkFun. "How To Play: Rush Hour - by ThinkFun". Filmed[February, 2012]. Youtube video, 1:36. Posted[February,2012]. https://www.youtube.com/watch?v=HI0rlp7tiZ0

8. Appendix

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


Visual Paradigm Standard(Ali Yümsel(Bi
[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
Wait User Input
[User selects player car]
[User selects any other vehicle]
Player car selected
The vehicle is selected
[User wants to move the vehicle]
[User wants to move the vehicle]
[If there is an obstacle in the selected direction]
[If there is no obstacle in the selected direction]
Move selected vehicle
Keep selected vehicle in the same place
[One of the vehicles either moved or not, so the map should be updated according to it]
Update Map
[User wants to continue playing, Pass to next level]