CS 319 - Object-Oriented Software Engineering

System Design Report

Rush Hour

Gurup Şurup

Muhammet Said Demir

Ata Coşkun

Zeynep Nur Öztürk

Asuman Aydın

Tarık Emin Kaplan

Supervisor: Eray Tüzün

TA: **Muhammed Çavuşoğlu**

Contents

1. Introduction

The cars and the challenges of them are some of the passionate of humankind in a sense of speed and control over something. Especially, the car is one of the daily life items that we use for everything mostly. In the game Rush Hour, as we decided to implement as being Group Şurup, this control and car passions can be felt in a minimalistic matter. A car that tries to get out of a parking lot where there are different types of car as being obstacles. The gamer has the control over cars and he/she has to use a logic and strategy to take the specific car out of the challenging level. One of the glimmering specialties of this game is nicely presented and enjoyable interface. There are cars in different types. The car that the user tries to take out from the parking lot is in red color to differentiate from other cars. There will be other types of obstacles to inhibit the car to get out and make the game more interesting in a way of challenging. The game also aims to entertain the user with different themes. In different themes, the car shapes and colors change.

While choosing which game we should implement, this kind of factors was effective and it can be concluded that it was a good decision to do this game because of the characteristics that we mentioned. To give more sight of the specialties of the game; there are some parts we should mention.

First of all, single player and multiplayer mode give an opportunity to choose what the gamer wants. The single-player mode is for entertaining the lonely and self-care times to spend. On the other hand, Multiplayer mode lets the user socialize and bound with the other person by having fun while playing and also enjoying being the winner. Also, in both game modes, the user chooses to play different levels in different themes. Secondly, different themes and having music with it make the game more interesting and easier to focus on the game. However, some people may not find this musical and thematically environment. So, they can easily mute it from the settings in both the main menu and in-game menu.

For the interest of more advanced players, there are 3 specialties of the game. Main two of them are

undo and replay. Another one is hint functionality. Another matter is help functionality of the game. Help is in settings that shows the instructions for the game. Another Help feature is the hint property which gives a little solution in the game under some conditions. Rush Hour is a game for the fun time and developing strategic aspect for anything and everything.

Implementation part of the game is depending on the Java language and the GUI support of the Java. So the User Interface part is all designed by our group

2. Overview

2.1 Game Engine

Rush Hour can be played in both Single Player Mode and Multiplayer Mode. The user will be controlling many cars in the parking lot and try to take the specific car out by using the directions up/down/right/left. Although taking the car out makes the level done, being successful in fewer amounts of time and steps is something considered in the game. There are stars to be filled according to the success of the gamer. For the single player mode, the game has basic instructions and user interface. However, the multiplayer mode has 3 board next to each other hence this makes the game is more enjoyable with the users challenging. There are extra three functionalities to talk about in Game Engine chapter as they are implemented in here.

2.1.1 Undo

First of it is Undo function. It is to take the step back when the user makes a mistake or to see other options to go further. To do this, we keep every step of user made and take the last one back when he/she chooses Undo option. This functionality can be reached from the game screen.

2.1.2 Hint

We can say that one of the most interesting parts of the game is the hint function. The intention to know the solution will make the user play more and have more points to have the right to take the hint in the game. As the user plays better, the hints will come with it.

2.1.3 Replay

To keep the game in basics matters, replay function has fundamental matter. Allowing the user to have this option and giving another chance to make the level better make the game more well-structured.

2.2 Board

The game will be played on the board that includes the cars, the obstacles. Also; levels get difficult within every board. Another thing about the levels is changeable obstacles. For example, if there is one type of obstacle in earlier levels, there would be more than one and more difficult obstacles in later levels because of challenge increases.

2.3 Settings

In game settings and main menu settings, the sound is depending on the player's request. So, this makes the game more compatible with user interface qualities. More importantly, the levels do not have to be in the same shape. The option to change theme is enjoyable for the user. There are different themes such as space, forest or specific locations in our country and in the space.

2.3.1 Sound Settings

Listening to the same music over and over again can be frustrating after a while. So the user may want to mute or unmute the music that plays in the background. Sound Settings allow the user to do this.

2.3.2 Theme Settings

To have a better user interface interaction in the game, there are different themes such

as space and forest. From the main menu, the user can change it by seeing the options and example display of the theme.

2.4 Player

After having the domain study by going and playing the game on the board, we have decided to make the game in both single and multiplayer mode. Because it increases the games interest and the range of the user groups' diversity.

2.4.1 Single Player

Single Player mode is the base of the game. The red car is the target to be taken out from the parking lot that is 6x6 matrices. There are some obstacles in some levels to make the single-player mode more enjoyable.

2.4.2 Multi-Player

Multi-Player Mode is composed of three board. The first board is for the first player and it is movable "up" or "down". The third board is for the second player and it is also movable. This functionality of the multiplayer gives a strategic aspect to the game.

2.5 Cards

This specialty gives some difference and enjoys the ability to the game. The cards include the shift right in amounts. As the board is moving, the user can do only that much shifts in one game.

2.6 Cars

In different shapes and colors, there are many cars on one board. Combinations of setting the cars on the board change by levels. There is one specific car which player will

able to differentiate from others. That car will be the car that player has to take out of the parking lot.

2.7 Levels

Levels are the main object of the game. There are levels for both single and multiplayer modes. Their difficulties vary. It can be chosen after choosing the game mode.

2.8 Obstacles

Obstacles are rocks and barriers in the road of the cars. It cannot be passed by the car so it makes the game more complicated in the sense for a multi-player game.

2.9 Help

Help function includes the game information. The information is basically how to play the game. There are official instructions which include the rules and images of how to move the cars. The idea behind this function is giving further info and pre-screening of the game.

3. Functional Requirements

3.1 Play game

There will be two main game modes, single player and multiplayer modes

3.1.1 Single Player Mode

The whole single player mode can actually be considered as a completely new feature if you consider the Rush Hour board game as a base reference point, which only has multiplayer mode.

In single player mode, the game is played on a 6x6 map, and the objective is to get the red car to the exit by clearing its path by moving the other cars, more often than not, the order of which car you choose to move will matter, because it will affect the path of another car. All the cars including the red car will only be able to move horizontally or vertically, which means the path between the destined exit and the red car is a straight one.

If a level is getting too complicated for the player, they can simply restart the level, or undo their last move. Alternatively, they can get a hint which will show the player what single move needs to be done in order to set the player on the right direction. To get more technical, both the hints and undo uses stack data structure, the correct moves that are needed to be done are pushed on to the stack when level is initialized, whenever player makes a move, that move is pushed onto the stack if it is not the correct move, or popped from stack if it is the correct move. The undo button simply pops the

last move from the stack if the last move done was not correct, or pushes the correct move to the stack if the last move done was correct. The hint button simply shows the move needed to be done to pop the top move on the stack. One problem that arises from using this mechanism is the fact that almost always, there are more than 1 way to solve a level, but it is borderline impossible to create all the possible solution stacks for every level, and technically, this method still works, and there is no restriction on the user to complete the game by using the exact method proposed in the solution stack, meaning a level can still be completed by using alternative ways, so we intend to keep the hint mechanism that way.

3.1.2 Multiplayer Mode

This mode is basically played the same way as the board game, with some additional features.

It is played on a 14x6 map, and the map consists of 3 parts, the left and right parts are both 5x6 of size and they can be moved vertically, which will alter the path of all the cars that are on them or need to pass through them, the middle part is 4x6 of size and it can’t be moved. There are 2 red cars placed on opposing sides (one for each player), both players try to reach their respective exit before the other player, which is placed on the opposing side. To be clearer, one player’s car is placed on the far left and its destined exit is on the right, and the other player’s car is placed on the far right and its destined exit is on the left. The players can’t move the opposing player’s red

car, but all the other cars, and the board, are neutral, meaning they can be moved by either player.

The game mechanism works in a turn-based manner and by using cards, cards have information about the number of moves that can be done, giving a player the right to move a board piece, etc. Both players initially start the game with 4 cards, then one player chooses a card to play and performs whatever operation on the board they want as long as their played card allows, then the other player does the same, after that point, at the start of every turn, each player draws another card. Game goes on until one player reaches its destined exit.

The game is local multiplayer, meaning both players will play on the same device, taking turns. There won't be any internet connection necessary.

3.2 Settings

This screen can be accessed from both the main menu and the in-game menu. On this screen, player will have the option to mute or unmute the game music, and change the theme of the game.

The most important thing to note here is that when the user changes the theme by accessing the settings screen from the in-game menu, the changes will be applied immediately, meaning that the player will be able to see the new theme on the board as soon as they go back to the game screen again, the board will remain in the state they left it in, only the theme will be different.

3.3 How to Play

This screen is accessed from the main mean and its function is to inform the user about how both of those mentioned game modes work, in addition to that, there are also explanations about the additional features which will be explained on following subsections (3.5).

3.4 Credits

This screen is also accessed from the main menu. On this screen, the players will be presented by the hard-working developers who have brought them this game.

3.5 Additional Requirements

There are 3 additional creative features available to use on our version of Rush Hour. Since the bridge require quite a significant amount of board space to be played out effectively, that will be in multiplayer game mode (which has a 14x6 board) but not in single player game mode (which has a 6x6 board).

3.5.1 Obstacles

There will be obstacles initially placed in the game, an obstacle will be of 1x1 size, and won't be moveable directly. Only way the player can avoid obstacles is by moving the board or by using bridges (bridges will be explained in section 3.5.3), since both of those ways are only available on multiplayer mode, there won't be any way to avoid obstacles in single player mode, the player will have to play around it.

3.5.2 Portals

Since all the cars are placed either vertically or horizontally and can only be moved on their initial horizontal/ vertical setting (we will call this h/v setting from this point now on), we have added portals. On portal levels, the entrance to the portal will be on the same h/v setting as the red cars, and the exit of the portal and the exit that the red cars need to reach will be on the same h/v setting as well, but the entrance/exit of the portals will be on a different h/v setting.

Meaning that, if the red cars are initially on a horizontal setting, they will need to go through the entrance of a portal which will teleport them to the exit of that portal, which will place them on a different place on map as well as changing their h/v setting to vertical. And the red cars will have no option other than to go through that portal eventually because their destined exit will also be on a vertical setting.

3.5.3 Bridges

There will be a special card that will enable whoever plays it to place a bridge on the map on any location. The bridge will be of 3x1 size, it will enable a car to pass over it (horizontally) and another to pass under it (vertically). The bridge can be placed on an empty 3x1 space; on top of an obstacle (which is of 1x1 size); or on top of a car (with vertical setting) as long as the car is below the middle portion of the bridge, (the part that allows vertical movement).

3.5.4 High Scores

The player’s performance in single player mode is reflected with points and a 3-star format.

4. Non-functional Requirements

4.1 Extendibility

All the things discussed in section 3.5 - “Additional Requirements” are a good example of how extendable our game is, many more creative ideas can be adjusted and implemented to our game, just like portals and bridges. There is literally no limit for such creative additions, but for now, some ideas that we consider adding to our game in the future are:

- Time attack: A single player game mode where the player has to complete the given puzzle under a specified time limit.
- Custom map design mode: A mode where the player can design and play on a single player or a multiplayer map however, they want.
- Multiplayer with single player maps: Two players will be presented with the same 6x6 map, just like in a single player mode. The first one to complete their puzzle will win. We think a “best of 3” format will be optimal for overall enjoyment. Can be made in an online multiplayer setting with both players playing simultaneously or in a local multiplayer setting where players are supposed to play in a turn-based manner.

4.2 Performance

All of the game operations, whether they are navigation operations between screens, or in-game operations such as moving a car, undoing a move, etc. should take no more than 0.5 seconds to make all the transitions/interactions feel smooth to use.

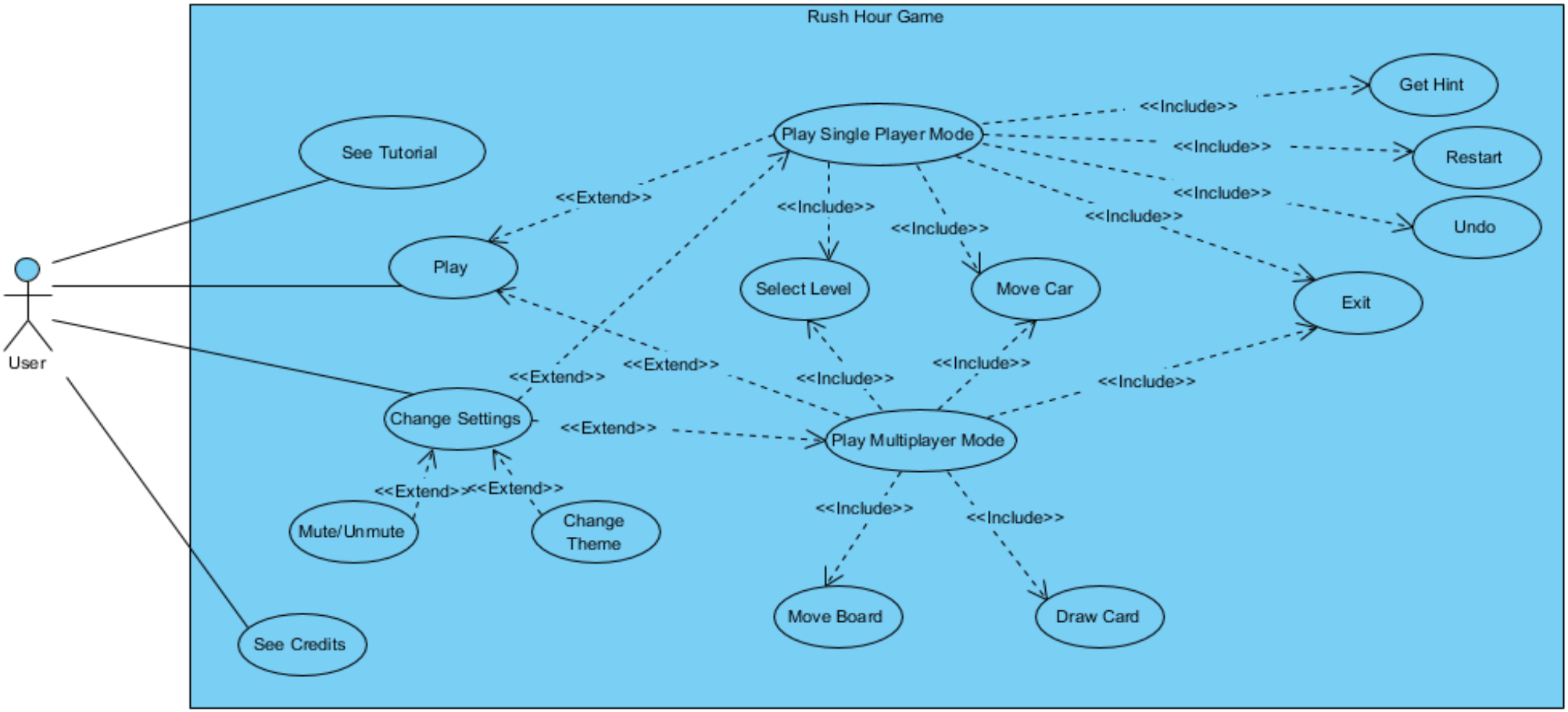
4.3 Usability

The whole game screen buttons, in-game objects, and the window itself should be easily visible and identifiable to avoid creating any sort of confusion. For reference, on a 1920x1080 screen, here are the dimension values of some of the most commonly used game objects / buttons / screens in our game (in pixels):

- The main menu screen is 1012x1012
- Every button in the main menu is 300x100
- Single game mode screen is 700x500
- A 1x1 block on map is 80x80
- Red car (horizontal setting) is 150x75
- In-game “Undo” button is 100x50

5. System Models

5.1 Use Case Model



5.1.1 Use Case Descriptions

Use Case #1

Use Case: Play Single Player Mode

Participating Actor: Player

Stakeholders and Interests:

- Player selects “Play” in the main menu
- System displays mode selection screen
- Player selects “Single Player Mode” in the mode selection screen
- System displays single player levels
- Player chooses a level to play

Pre-conditions:

- Player must be in the level selection screen or must have completed a single player mode level

Post-conditions:

- Player ends up in main menu screen, or goes to next level (plays another single player mode level), or exits the game, depending on their choice.

Entry-conditions:

- Player should choose the single player mode on mode selection screen
- Player should choose a level on level selection screen (single player levels)

Exit conditions:

- Player exits the game by choosing “Exit Game” option from the in-game settings
- Player exits the game by pressing the “X” button of game window
- Player exits the game by choosing “Exit Game” option from the end game pop-up

---

Main Event Flow:

1. Player chooses “Single Player Mode”.
2. System displays single player levels
3. Player chooses a level to play
4. System displays the chosen level
5. Player moves a car
6. System updates the game view depending on player’s movement command
7. Player eventually moves the red car to the exit
8. System finishes the level and displays a pop-up with their points earned and number of moves done, and presents player the options to go to main level, go to next level, or to exit the game.

Alternative Event Flow(s):

1. If player wants to change settings:
    1.1. Player clicks on settings button in the game screen.
    1.2. System displays settings
2. If player wants to exit the game
    2.1. Player can click on exit on button in the game screen
    2.2. Player can click on “X” button of the window
3. If player is stuck during the game
    3.1. Player can choose to undo his last move
    3.2. Player can get a hint

3.3. Player can restart the level.

Use Case #2

Use Case: Play Multiplayer Mode

Participating Actors: 2 Players (on same phone, taking turns)

Stakeholders and Interests:

- Player selects “Play” in the main menu
- System displays mode selection screen
- Player selects “Multiplayer Mode” in the mode selection screen
- System displays multiplayer levels
- Player chooses a level to play

Pre-conditions:

- Player must be in the level selection screen or must have completed a multiplayer mode level

---

Post-conditions:

- Player ends up in main menu screen, or goes to next level (plays another multiplayer mode level), or exits the game, depending on their choice.

Entry-conditions:

- Player should choose the multiplayer mode on mode selection screen
- Player should choose a level on level selection screen (multiplayer levels)

Exit conditions:

- Player exits the game by choosing “Exit Game” option from the in-game settings
- Player exits the game by pressing the “X” button of game window
- Player exits the game by choosing “Exit Game” option from the end game pop-up

Main Event Flow:

1. Player chooses “Multiplayer Mode”.
2. System displays multiplayer levels
3. Player chooses a level to play
4. System displays the chosen level
5. System gives both players 4 initial cards
6. Player chooses a card to play and moves cars or some part of the board.
7. System displays the changed state of the game and gives the other player the turn to play
8. Other player plays (same process in step 6)
9. System displays the changed state again and gives the first player the turn to play again.
10. Player draws another card, and repeats the process on step 6.
11. System displays the changed state again but gives the other player the turn to play.
12. Other player draws another card as well, and repeats the process on step 6.
13. System updates the game view
14. Players keep drawing a card and playing the game in a turn-based manner until eventually one of the players move their red car to their respective exit.
15. System finishes the level and displays a pop-up with who the winner is, and presents the player the options to go to main level, go to next level, or to exit the game.

Alternative Event Flow(s):

1. If player wants to change settings:
    1.1. Player clicks on settings button in the game screen.
    1.2. System displays settings
2. If player wants to exit the game
    2.1. Player can click on exit on button in the game screen
    2.2. Player can click on “X” button of the game window

Use Case #3

Use Case: Change Settings

Participating Actor: Player

Stakeholders and Interests:

- Player selects “Settings” in the main menu or in the in-game screen
- System displays settings screen

Pre-conditions:

- Player must be in main menu screen or in game (any mode)

Post-conditions:

- Player ends up in main menu screen, or in a game screen, depending on where this “Settings” option was called.
- Theme or sound settings are changed if the user has changed them

Entry-conditions:

- Player should choose the “Settings” option in main menu
- Player should choose the “Settings” option in game screen

Exit conditions:

- Player exits the settings screen by pressing “Back” button.
- Player exits the game by pressing the “X” button of game window

---

Main Event Flow:

1. Player chooses “Settings”.
2. System displays the options to change the theme or to mute the in-game sound
3. Player changes mutes/unmutes the game sound if they choose to do so
4. System sets the new sound settings
5. Player changes the game theme they choose to do so
6. System updates the new theme settings based on player’s new theme choice
7. Player presses “Back” button.
8. System displays the screen the “settings” was called from.

Use Case #4

Use Case: See Tutorial

Participating Actor: Player

Stakeholders and Interests:

- Player selects “How to Play” in the main menu
- System displays the tutorial screen

Pre-conditions:

- Player must be in the main menu

Post-conditions:

- Player ends up in main menu, presumably having learned more about the game

Entry-conditions:

- Player should choose the “How to Play” option in main menu

Exit conditions:

- Player exits the tutorial screen by pressing “Back” button.
- Player exits the game by pressing the “X” button of game window

Main Event Flow:

1. Player chooses “How to Play”
2. System displays the rules on how to play the game
3. Player presses “Back” button.

4. System goes back to main menu screen

Use Case #4

Use Case: See Credits

Participating Actor: Player

Stakeholders and Interests:

- Player selects “Credits” in the main menu
- System displays the credits screen

Pre-conditions:

- Player must be in the main menu

Post-conditions:

- Player ends up in main menu

Entry-conditions:

- Player should choose the “Credits” option in main menu

Exit conditions:

- Player exits the tutorial screen by pressing “Back” button.
- Player exits the game by pressing the “X” button of game window

Main Event Flow:

1. Player chooses “Credits”
2. System displays the people who worked on the development of this game.
3. Player presses “Back” button.
4. System goes back to main menu screen.

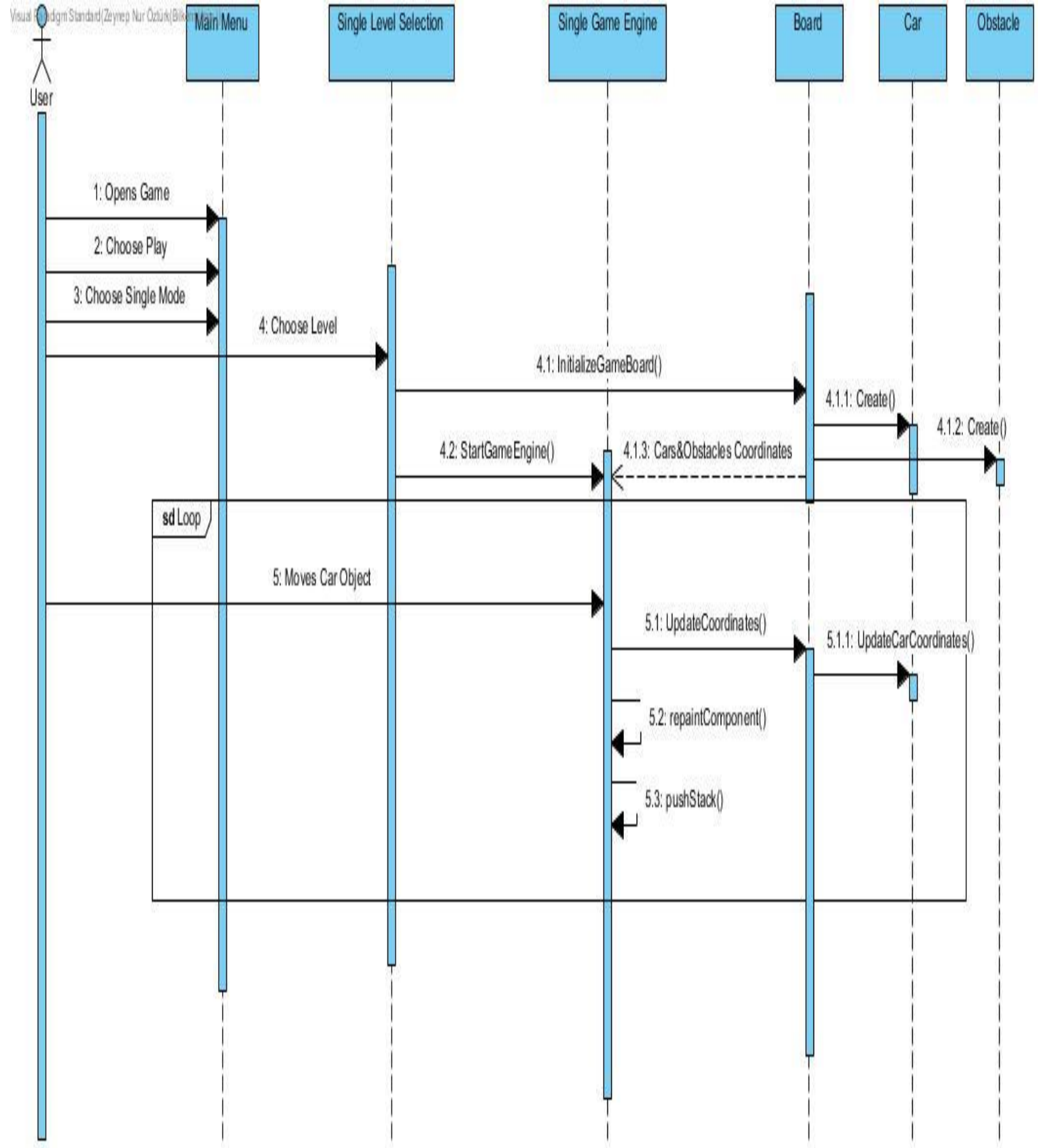
5.2 Dynamic models

5.2.1 Sequence Diagrams

5.2.1.1 Play Single Game Mode

Scenario: Staring a new game in Single Game

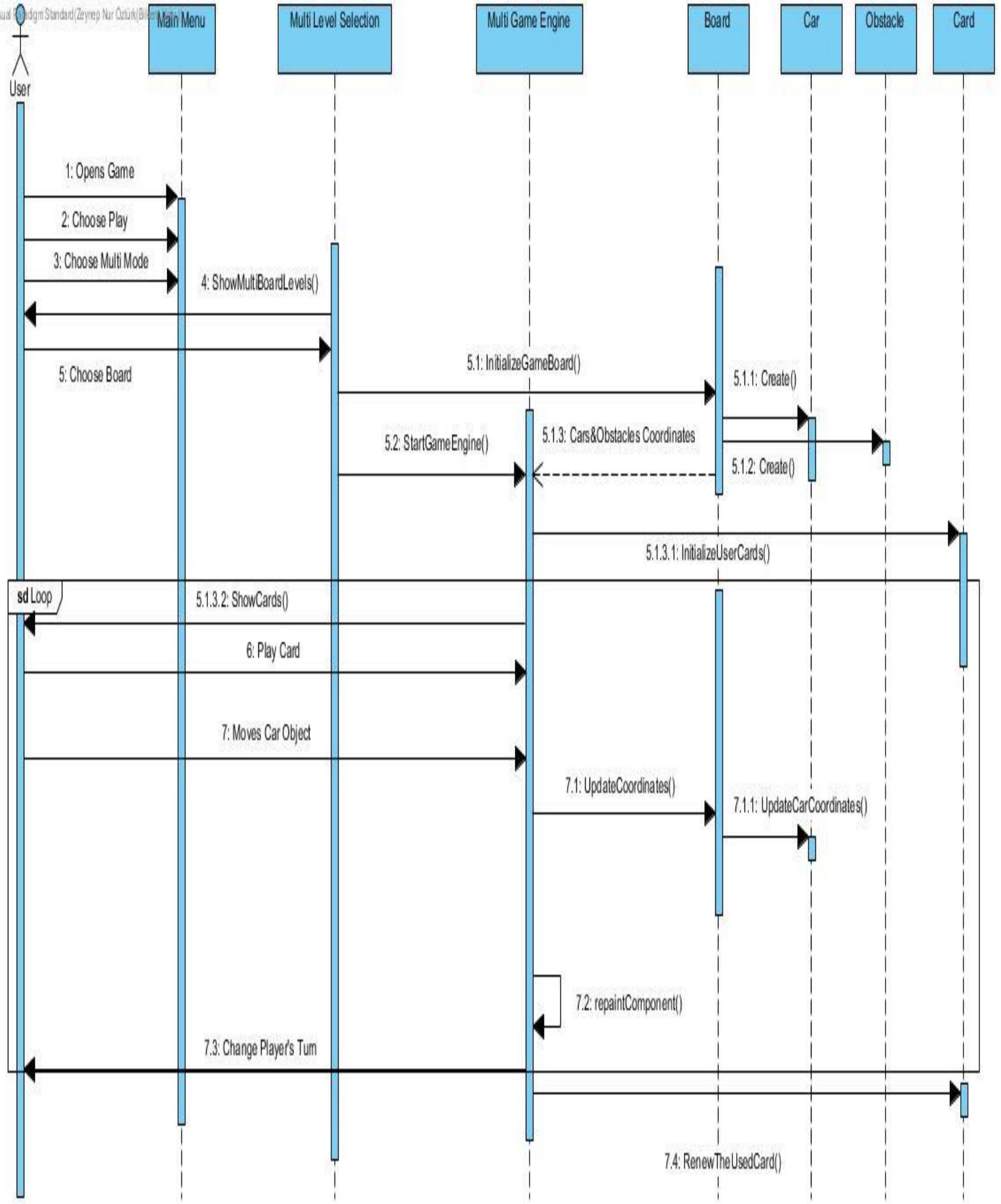
The user opens the game, the main menu is shown to the user. He or she chooses the play option. The single and the multiplayer mode buttons are shown to the user. After User chooses the mode Single Level Selection shows the possible levels that the user wants to play. Single Level Selection initializes the game board. Board class place the obstacles and cars on the board. Then board sends that information Single game engine with that information single game engine starts the game. When the user moves the car single game engine updates the coordinates of the cars and obstacles and repaints the screen based on movements. These movements pushing the stack.


User
Main Menu
Single Level Selection
Single Game Engine
Board
Car
Obstacle
1: Opens Game
2: Choose Play
3: Choose Single Mode
4: Choose Level
4.1: InitializeGameBoard()
4.1.1: Create()
4.1.2: Create()
4.2: StartGameEngine()
4.1.3: Cars&Obstacles Coordinates
sd Loop
5: Moves Car Object
5.1: UpdateCoordinates()
5.1.1: UpdateCarCoordinates()
5.2: repaintComponent()
5.3: pushStack()

5.2.1.2 Play Multi Game Mode

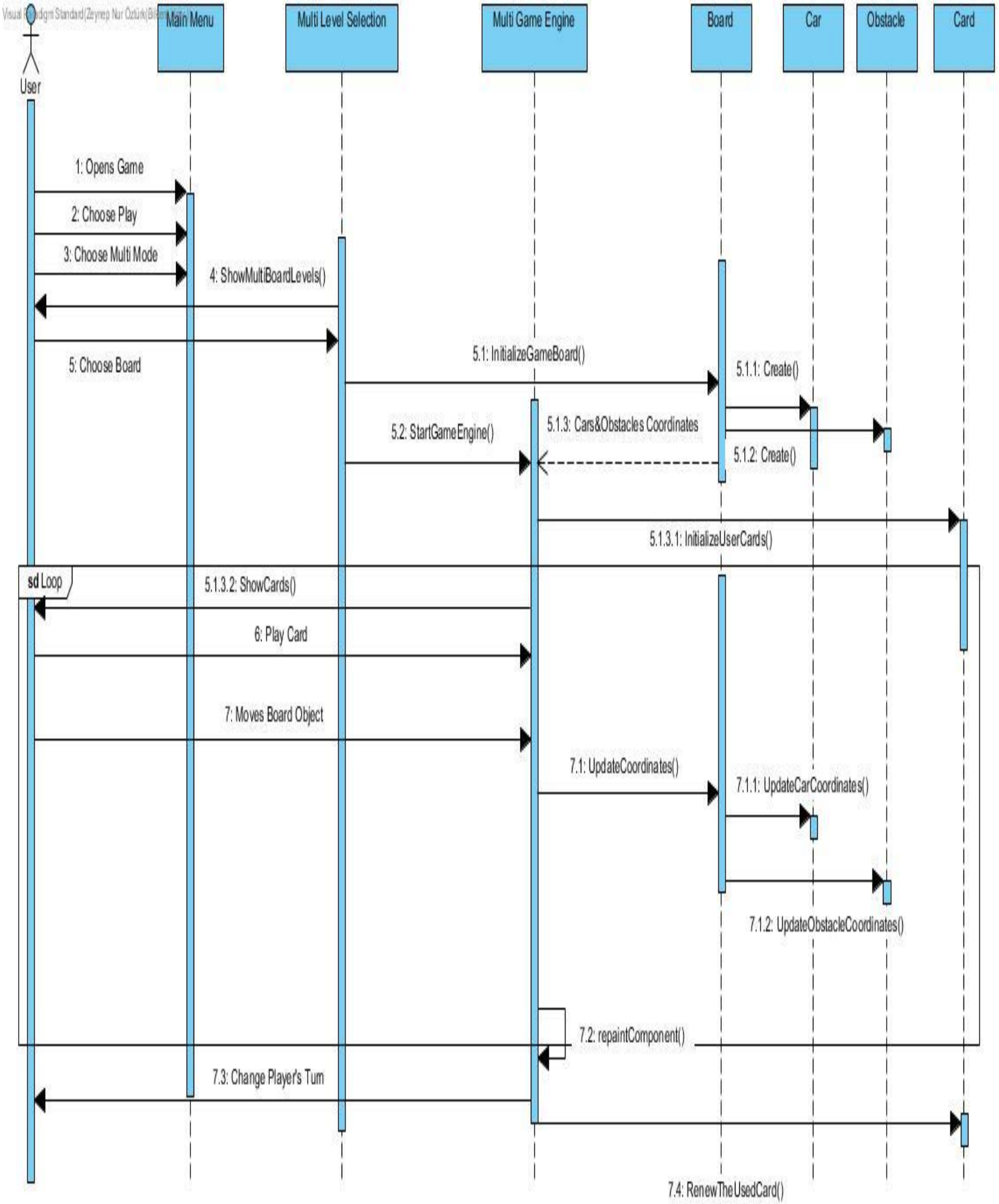
Scenario: Staring a new game in Multi Game. Usage of move car option.

The user opens the game, the main menu is shown to the user. He or she chooses the play option. The single and the multiplayer mode buttons are shown to the user. After User chooses the mode Multi-Level Selection shows the possible boards that user wants to play. Multi-Level Selection initializes the game board. Board class place the obstacles and cars on the board. Then board sends that information. Multi-game engine with that information multi-game engine starts the game. Multi-game engine class initializes the user's cards. Cards are shown to the user by the multi-game engine and the user plays the cards that he or she want to use. Based on the number of movement information in the card user moves the cars. When the user moves the car multi-game engine updates the coordinates of the cars and repaints the screen based on movements.


Visual Paradigm Standard(Zeynep Nur Öztürk(Bilkent))
User
Main Menu
Multi Level Selection
Multi Game Engine
Board
Car
Obstacle
Card
1: Opens Game
2: Choose Play
3: Choose Multi Mode
4: ShowMultiBoardLevels()
5: Choose Board
5.1: InitializeGameBoard()
5.1.1: Create()
5.1.3: Cars&Obstacles Coordinates
5.2: StartGameEngine()
5.1.2: Create()
5.1.3.1: InitializeUserCards()
sd Loop
5.1.3.2: ShowCards()
6: Play Card
7: Moves Car Object
7.1: UpdateCoordinates()
7.1.1: UpdateCarCoordinates()
7.2: repaintComponent()
7.3: Change Player's Turn
7.4: RenewTheUsedCard()

Scenario: Staring a new game in Multi Game with move board option

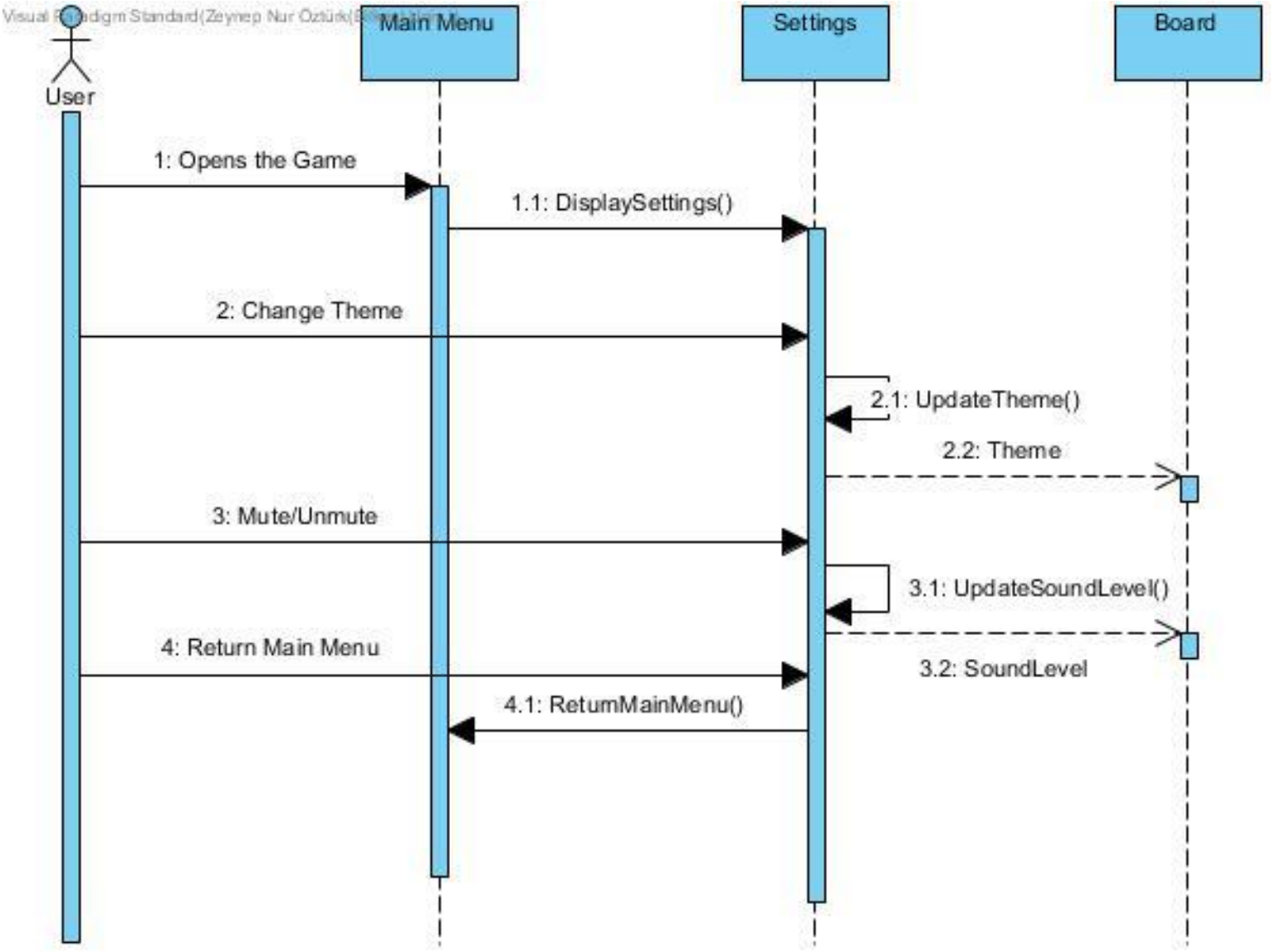
The user opens the game, the main menu is shown to the user. He or she chooses the play option. The single and the multiplayer mode buttons are shown to the user. After User chooses the mode Multi- Level Selection shows the possible boards that user wants to play. Multi-Level Selection initializes the game board. Board class place the obstacles and cars on the board. Then board sends that information Multi-game engine with that information multi-game engine starts the game. Multi-game engine class initializes the user's cards. Cards are shown to the user by the multi-game engine and the user plays the cards that he or she want to use. Based on the shift movement in the card user moves the board. When the user moves the board multi-game engine updates the coordinates of the cars and the obstacles and repaints the screen based on movements.


Visual Paradigm Standard(Zeynep Nur Öztürk(Bilkent))
User
Main Menu
Multi Level Selection
Multi Game Engine
Board
Car
Obstacle
Card
1: Opens Game
2: Choose Play
3: Choose Multi Mode
4: ShowMultiBoardLevels()
5: Choose Board
5.1: InitializeGameBoard()
5.1.1: Create()
5.1.3: Cars&Obstacles Coordinates
5.2: StartGameEngine()
5.1.2: Create()
5.1.3.1: InitializeUserCards()
sd Loop
5.1.3.2: ShowCards()
6: Play Card
7: Moves Board Object
7.1: UpdateCoordinates()
7.1.1: UpdateCarCoordinates()
7.1.2: UpdateObstacleCoordinates()
7.2: repaintComponent()
7.3: Change Player's Turn
7.4: RenewTheUsedCard()

5.1.2.3 Change the Settings

Scenario: User enters the settings screen from the Main Menu

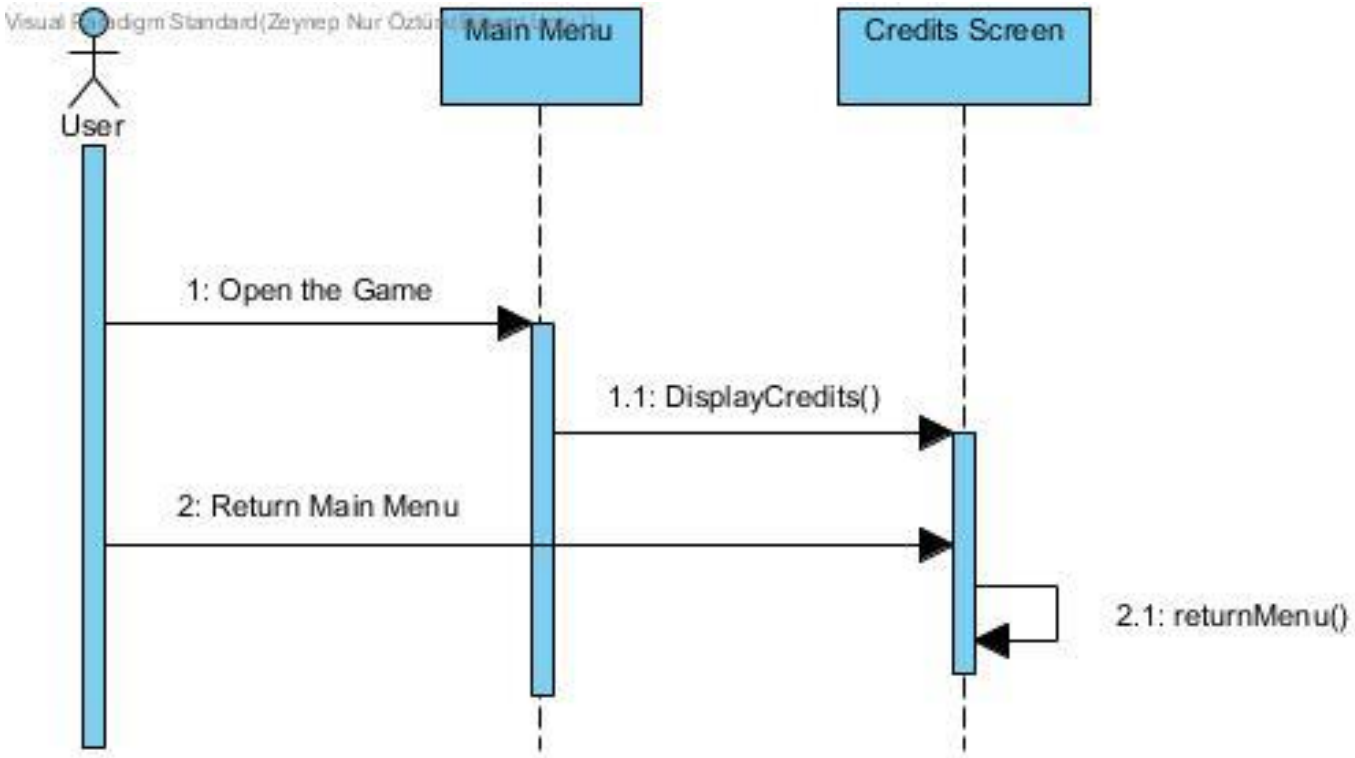
The user wants to change the settings for his/her preferences, therefore; enters the main menu and chooses the settings option. If the user doesn't want to do any changes, he/she can return to Main Menu. Otherwise, the user can change the theme. Chosen theme information sent to the Board class and theme is updated. Also, the user can change the volume r simply mute or unmute the sound. Settings sent the updated volume to Board. If the user doesn't want to do any further changes, he/she can return to Main Menu.



5.2.1.4 Main Menu Options

Scenario: The user enters the Credits and Get Help Option

The user opens the game and wants to see the credits. He/she enters the game and chooses the credits option. After the user sees the credits, he/she can return to the main menu. The Get help Option has exactly the same sequence. Therefore, we didn't duplicate it.

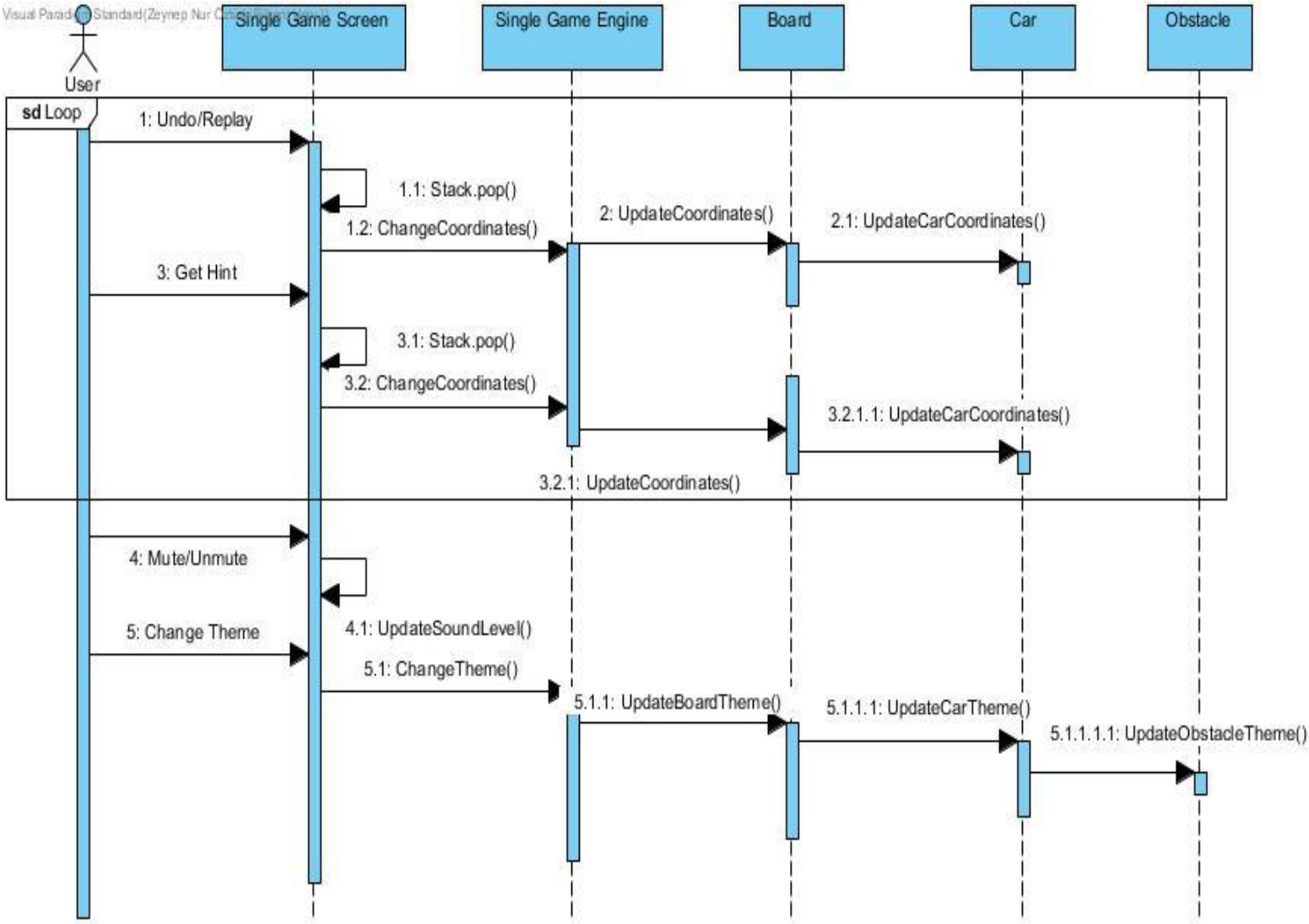


5.2.1.5 In Game Settings

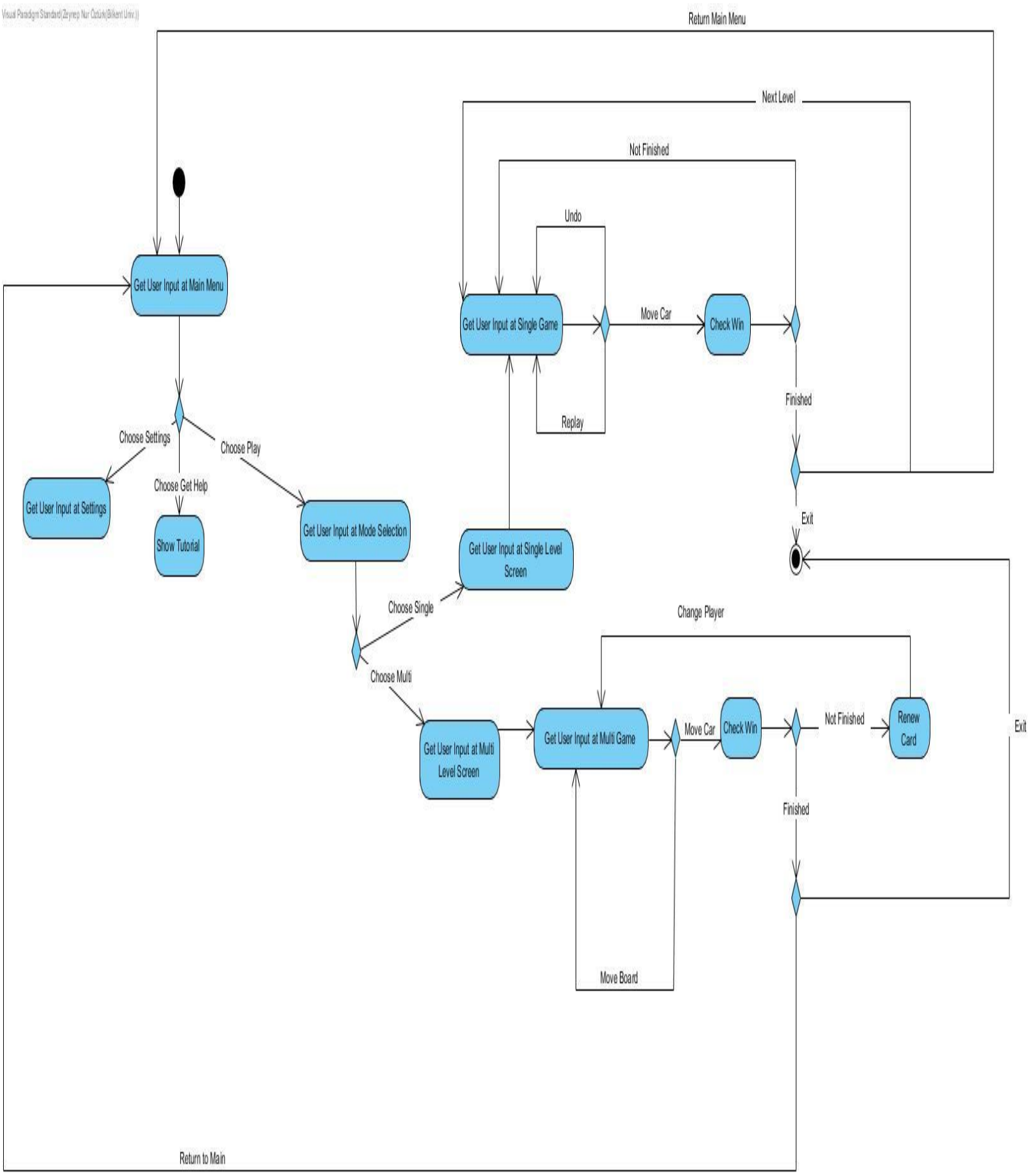
Scenario: The user clicks the undo, hint and mute option

While the user playing the game, he/she can click undo button. This button will pop a movement in the stack. Then the movement just before the last one will occur. Also, the same strategy will be used during the hint option hit will turn back the user's movements and at last will show the solution steps by popping the stack. Mute Option will mute the music's volume by user's click. Clicking it

again will unmute the sound. Also, the user can change the theme. The single game screen takes the user's option and changes the board's, car's and obstacle's theme. Players can reach to in game settings as mute and change theme in multiplayer version as well



5.2.2 Activity Diagram

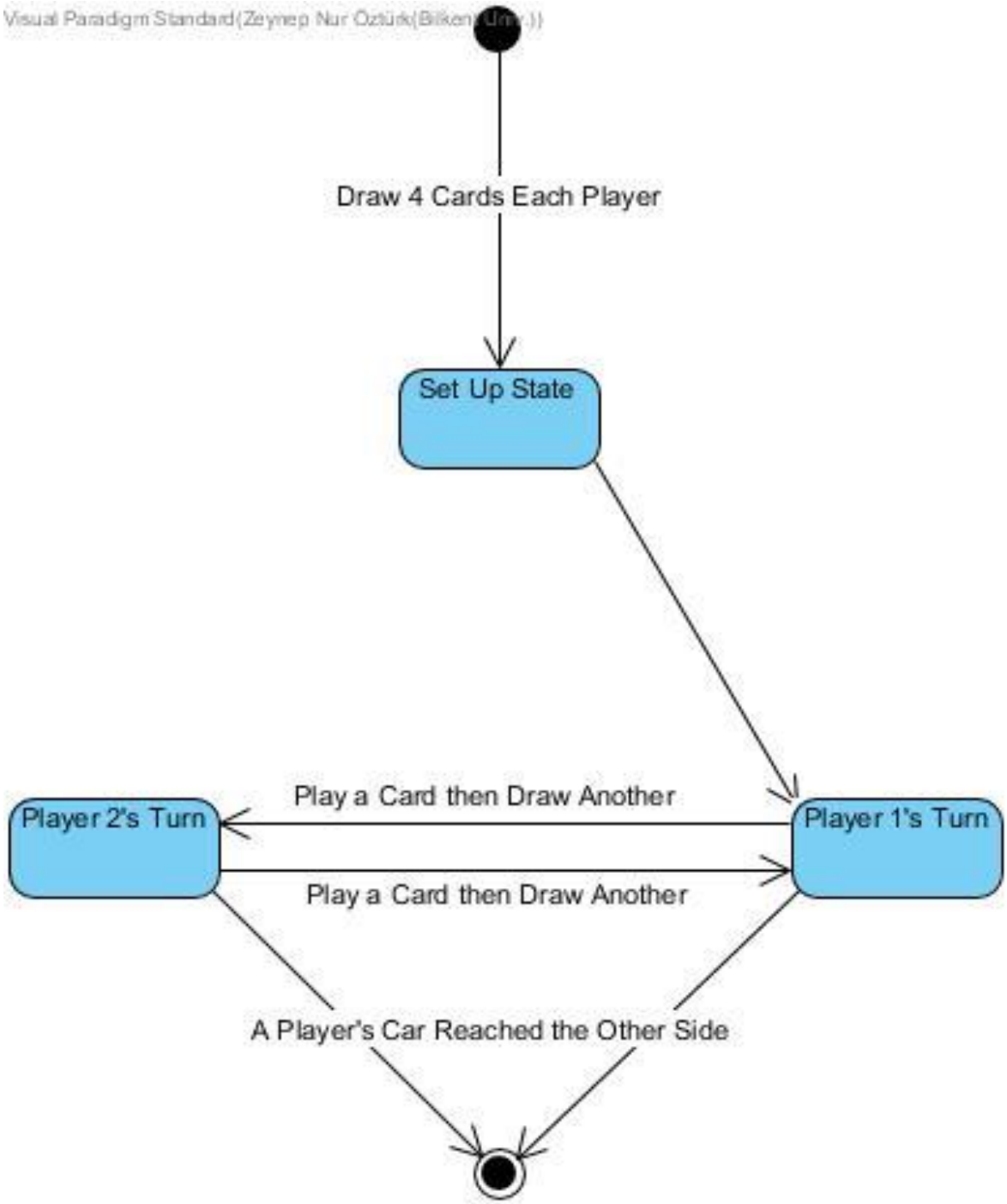


Activity diagram demonstrate the steps according to user input at Main Menu. There are different orientations:

- Showing setting: The user wants to change the settings. Then user input differentiates to two either sound changing or theme changing. When the user chooses the sound settings, input adjustment will be given to rush hour object and output will be either muted or unmuted. When the user chooses the theme change setting, the flow of different themes will be taken from the stored data. While the player is choosing, he/she can also see sample display and decide. When the input theme is entered, all game will be set up.
- Show tutorial: There is tutorial to show according to this input. There are pictures that includes everything about the game.
- Select play: The player chooses to play. Then there are two options. The player can either choose Single Player Mode or Multi-Player Mode. When he/she chooses the single mode, the levels selection screen opens, and the user's input gives the level number that he/she wants to play. Then the player starts to play and depending on the directions of cars, the number of steps is kept and the time will increase while playing. The user can replay the game or undo the steps that he/she made. In every step the winning conditions will be check, if the user reaches the target place the game will be finished, if not the game will continue. When the exit is seen, the score will be displayed and according to score, the stars will be given. He/she can continue with the next level or simply returns to the main menu. When he/she chooses the multiplayer mode, the levels selection screen opens, and the user's input gives the level number that he/she wants to play. At the beginning each player has 4 cards. Each turn the players will be given a new card that determines their number of action limit and shifting limit. It will be given randomly. They will be able

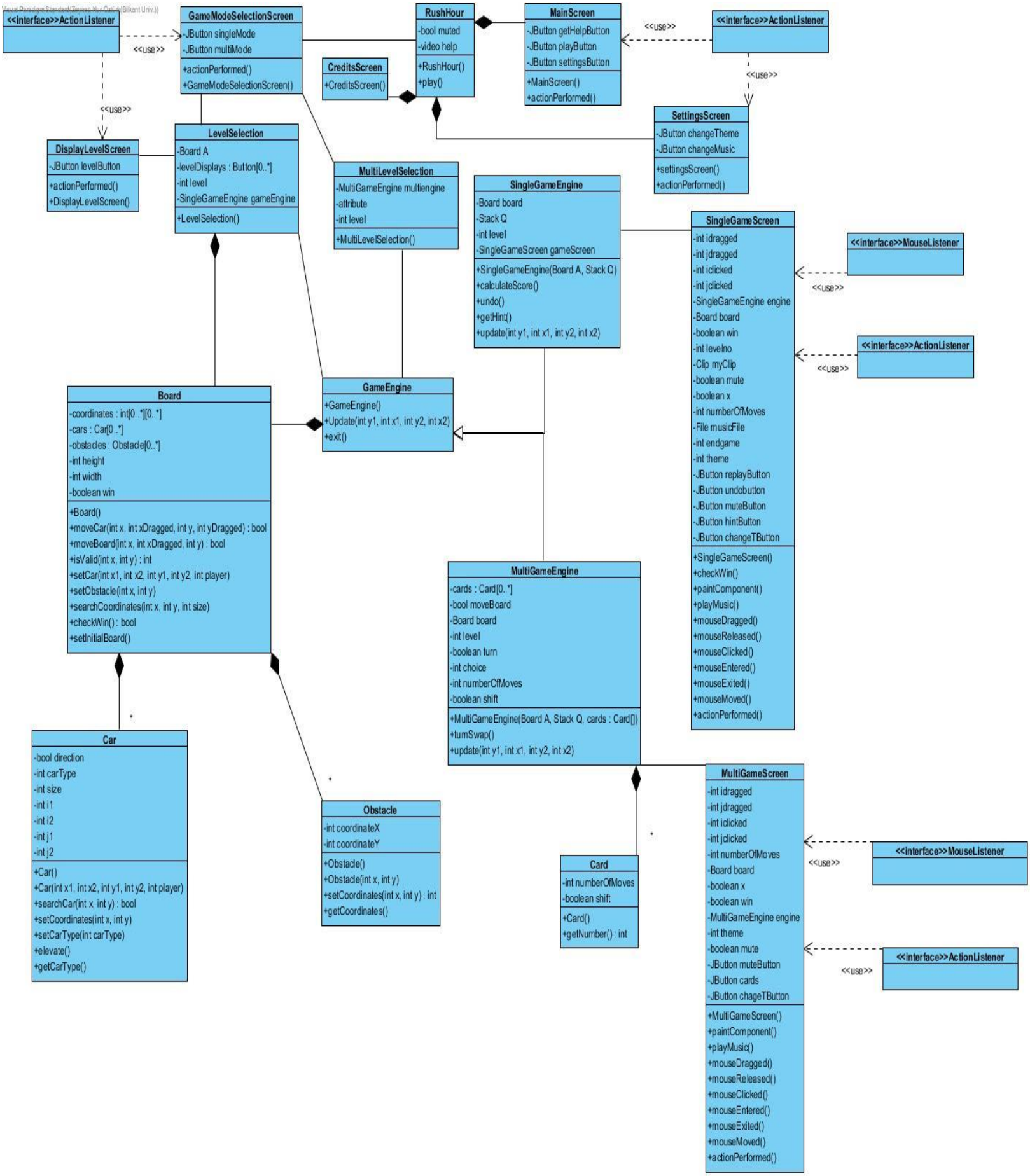
to move the board as well. And the rest is the same with the single-player mode.

5.2.3 State Diagram



State diagram for multiplayer game states. Each player starts with 4 cards initial at their hands. The game always starts with the first player. Players each turn play a card and get the number of moves they can perform and if they can move board. After their actions are completed, they automatically get another card and it will be other player's turn, if they didn't win. If they win, the game ends.

5.3 Object and class mode



Rush Hour Class: This class starts the game with the main menu.

Board Class: Board class will be the base system the game is played on. Cars and obstacles hosted in this class. Game engines use valid positions or not. It has array-based implementations. All movements and changes are handled in the board. There are 2 types of a board: one is the single game board (6x6) the other multi-game board (22x14).

Car Class: This class represents players' cars and other cars. All cars will have a coordinate on board. All cars have car type indicator.

Obstacle Class: Obstacles will have a coordinate on the board. Players will not be able to move these obstacles.

Card Class: Cards includes information about actions in the multiplayer game. Cards consist of moves of cars, obstacles or board. For example, one card may include 2 moves up for cars or 3 moves left for the obstacle or 2 moves change of boards. Each round player pick the one of the 4 cards in their hand and play that card to move.

Hint Class: This class keeps the moves of players in queue. When the player wants to help, this queue gives some hint to the player.

GameEngine Class: GameEngine class is an abstract class has update method which forces all kind of gameEngine to implement this method. Our game is based on updating on game board so

update method is the fundamental method for the game. There are 2 kinds of game engine one of them is SingleGameEngine, the other is MultiGameEngine.

SingleGameEngine Class: SingleGameEngine class is responsible for the single game mode. It has a single game board type(6x6). This class manages the change in the game. Game data is controlled in this class. SingleGameEngine class updates the game data and sends new data to SingleGameScreen.

SingleGameScreen Class: This class is the painter of the game. It takes data from SingleGameEngine, after that paint the new board states accordingly. This class has a single game engine. Every movement single game engine updates the board. This class gets the board and paint it. In addition, this class plays music, change the theme, has the undo, replay buttons.

MultiGameEngine Class: MultiGameEngine class is responsible for multi-game mode. It has a multi-game board(22x14). It checks player turns and update them. It also manages some important game rules for example if player1 is playing, he cannot touch player2's car. Each round it gives the random cards to the user.

MultiGameScreen Class: MultiGameScreen class is the painter of multi game. After all, movements update the screen according to multi-game engine's new data. Painting is based on board. It gets boards coordinates and cars array and obstacle array and paint screen. This class has cars as buttons. It also has change theme, mute buttons.

GameModeSelectionScreen Class: It is the screen allows players to choose the game mode. These 2 buttons represent the game modes which are the single game and multi-game.

DisplayLevelScreen Class: It allows players to choose which level they want to play. The board pictures are JButtons and players choose the level by clicking these buttons.

LevelSelection Class: Level selection class is an embedded database for the single player game. There is an empty board in the beginning. It gets the level choice data from DisplayLevelScreen, after that it initializes the board by using setCar, setObstacle methods. At the end of the initialize it calls the SingleGameEngine and initialized board goes to the single game engine.

MultiLevelSelection Class: This class keeps datasets for the multi-game board. It initializes multi-game board according to level knowledge.

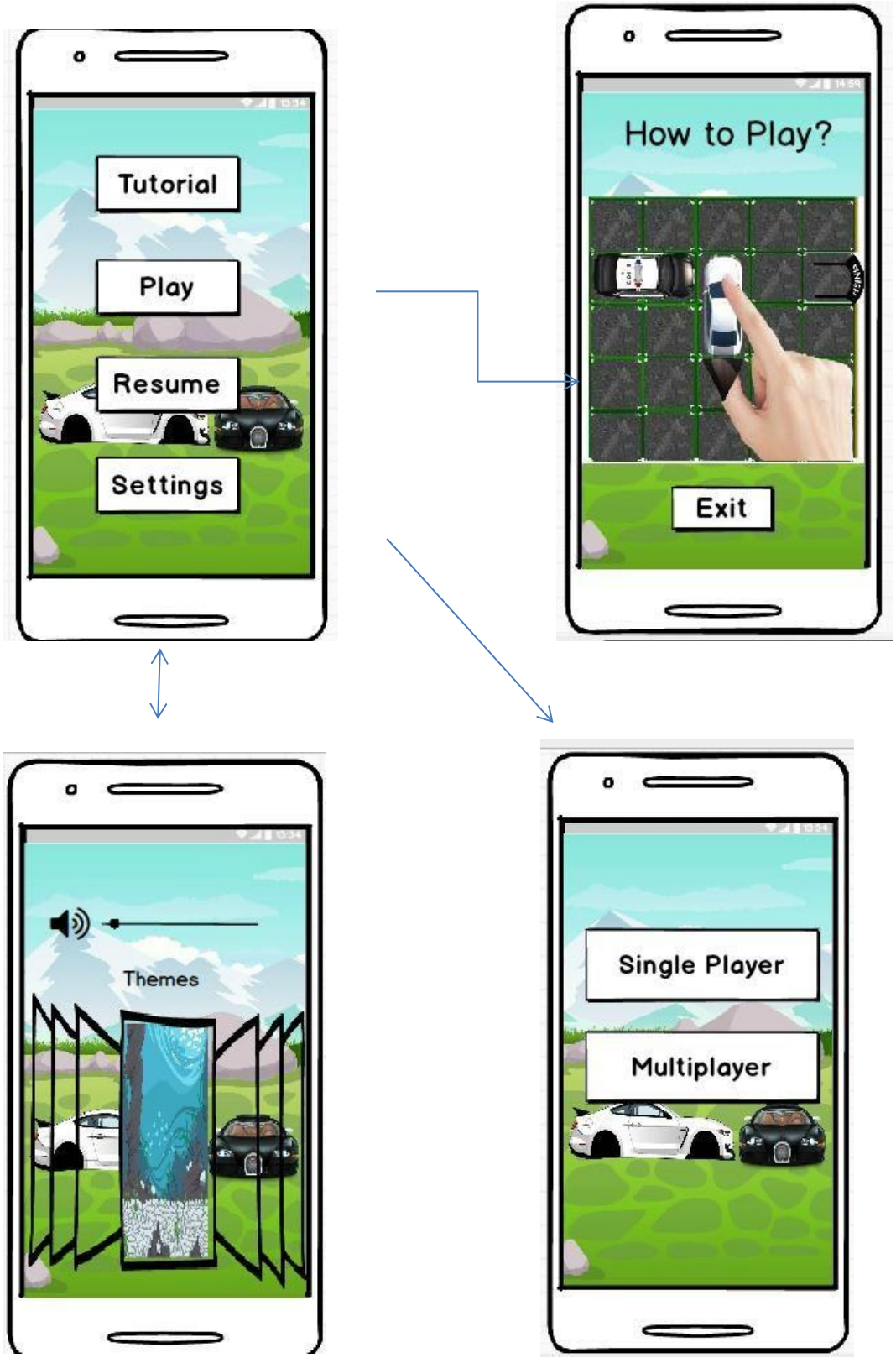
MainScreen Class: Main screen class is a frame and shows the options to players. It has a play, settings, tutorial, credits buttons to enables user these actions.

Credits Class: This shows the people who have contributes to the game

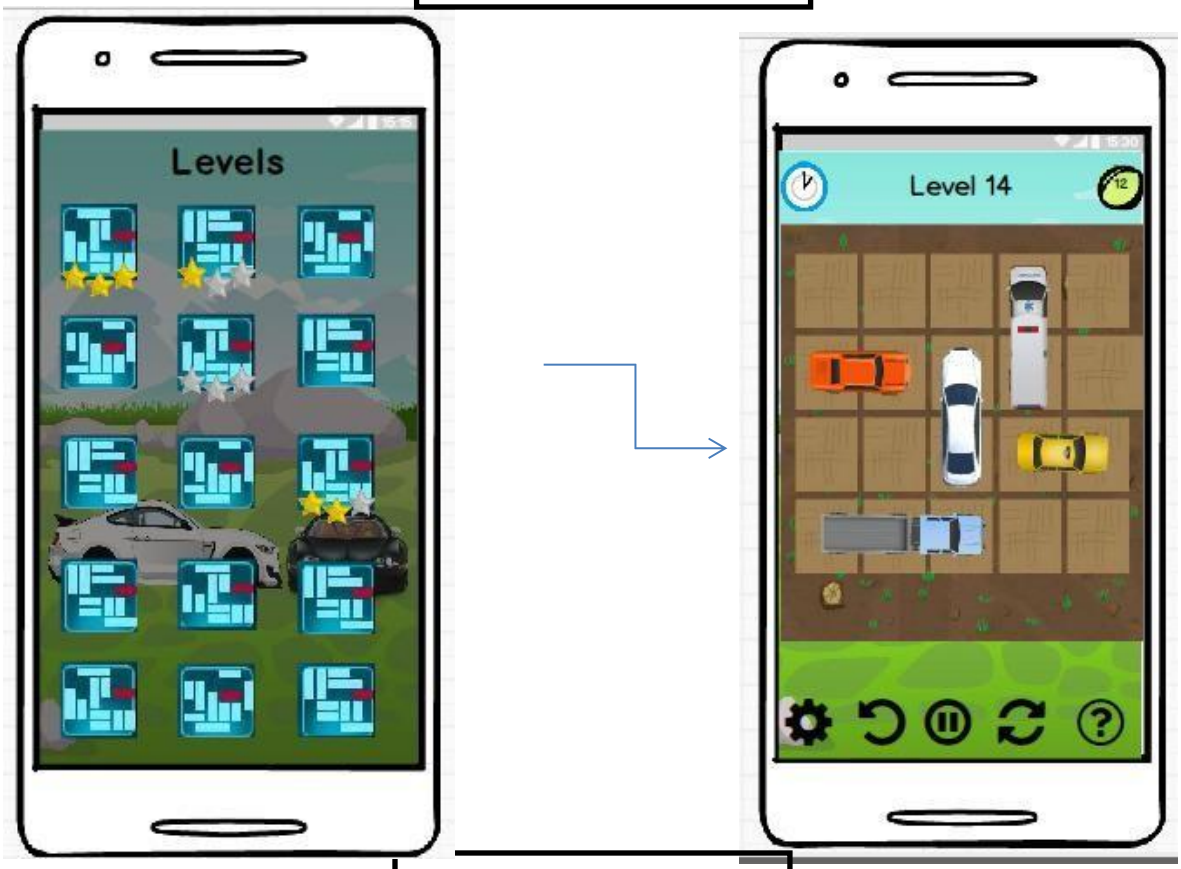
SettingsScreen Class: Settings screen class shows the themes in the game and allows players to determine the theme. It also offers players to music options

5.4 User interface – navigational paths and screen mockups and screenshots

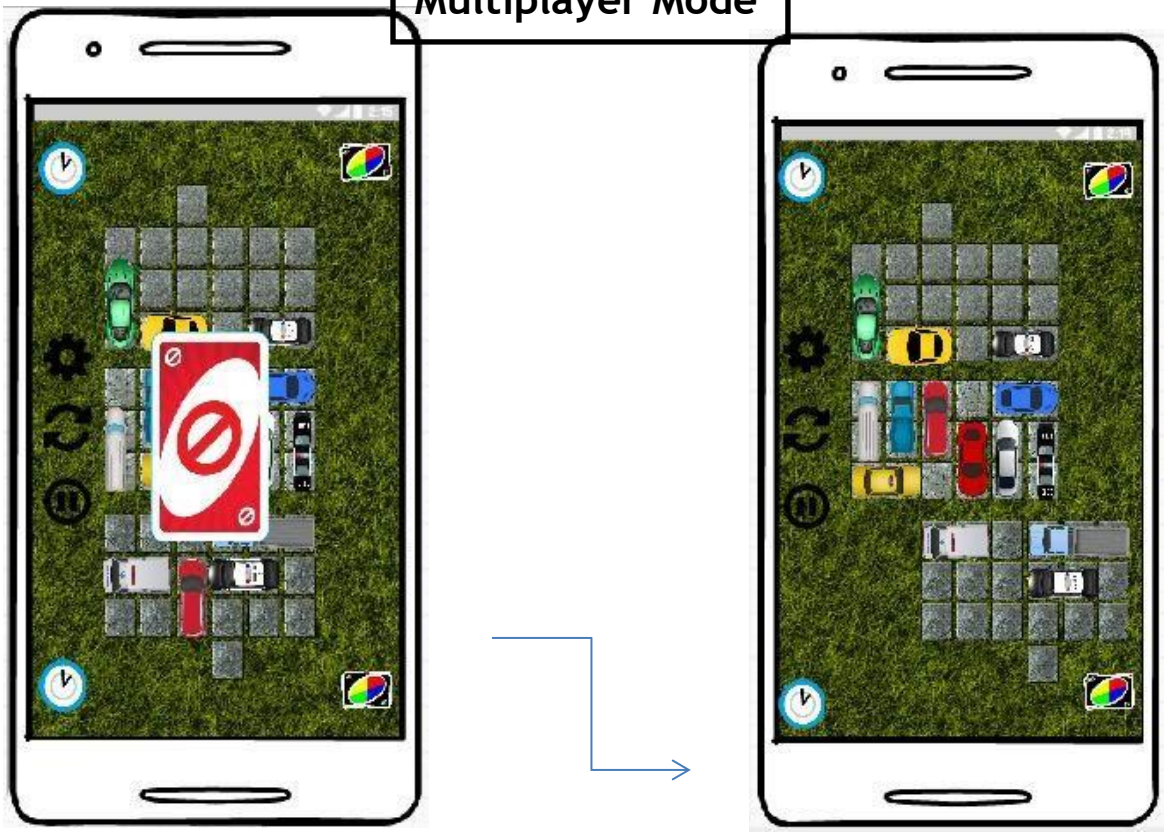
These pictures were from our mockups. It includes the things we've done so far and the things we haven't implemented yet.



Single Player Mode

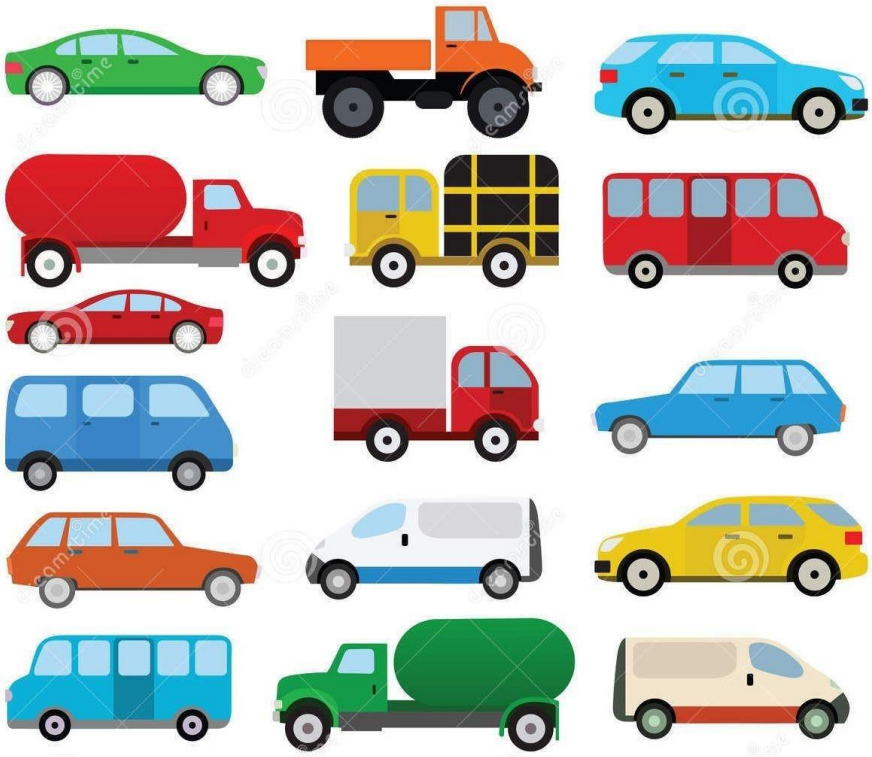


Multiplayer Mode

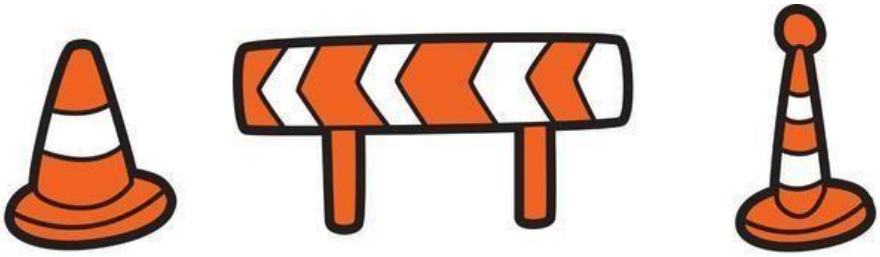


Theme Example: Space

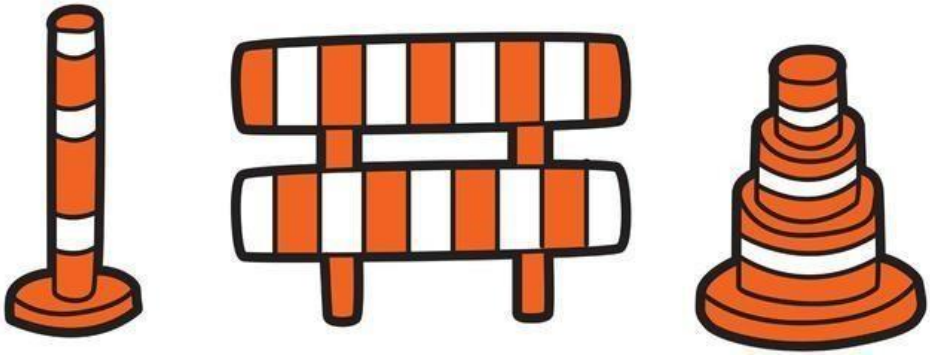
Theme Example: Forest



Car examples

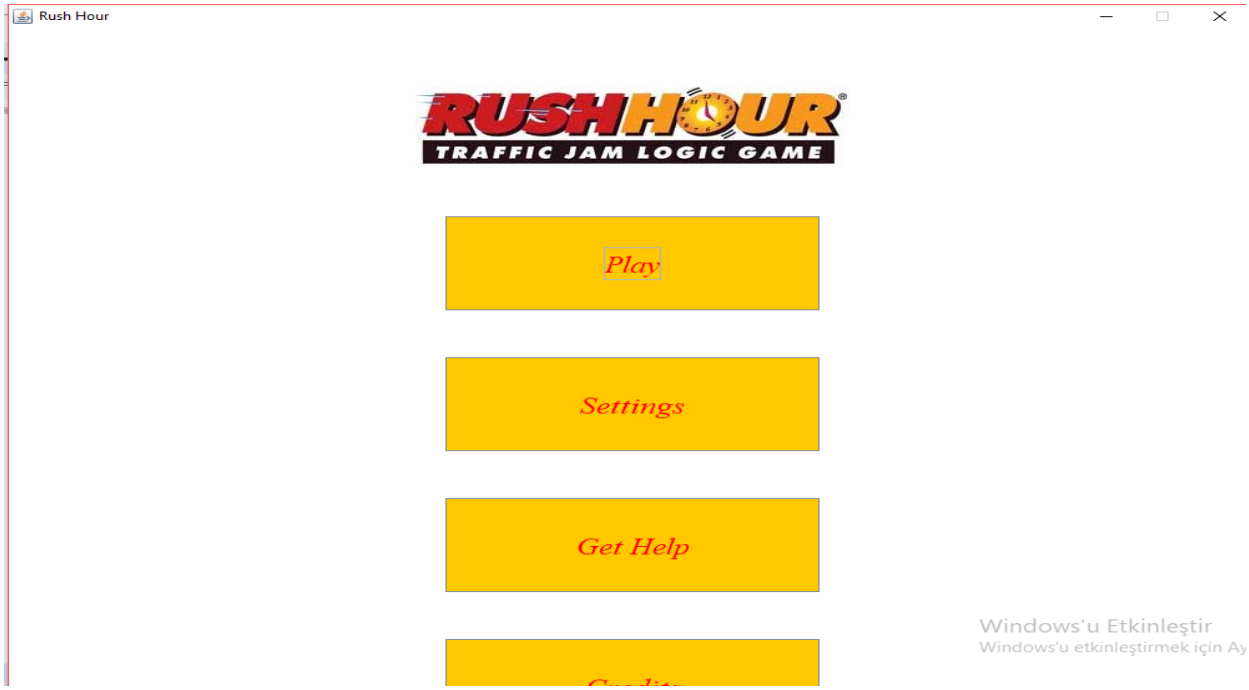


Obstacle Examples

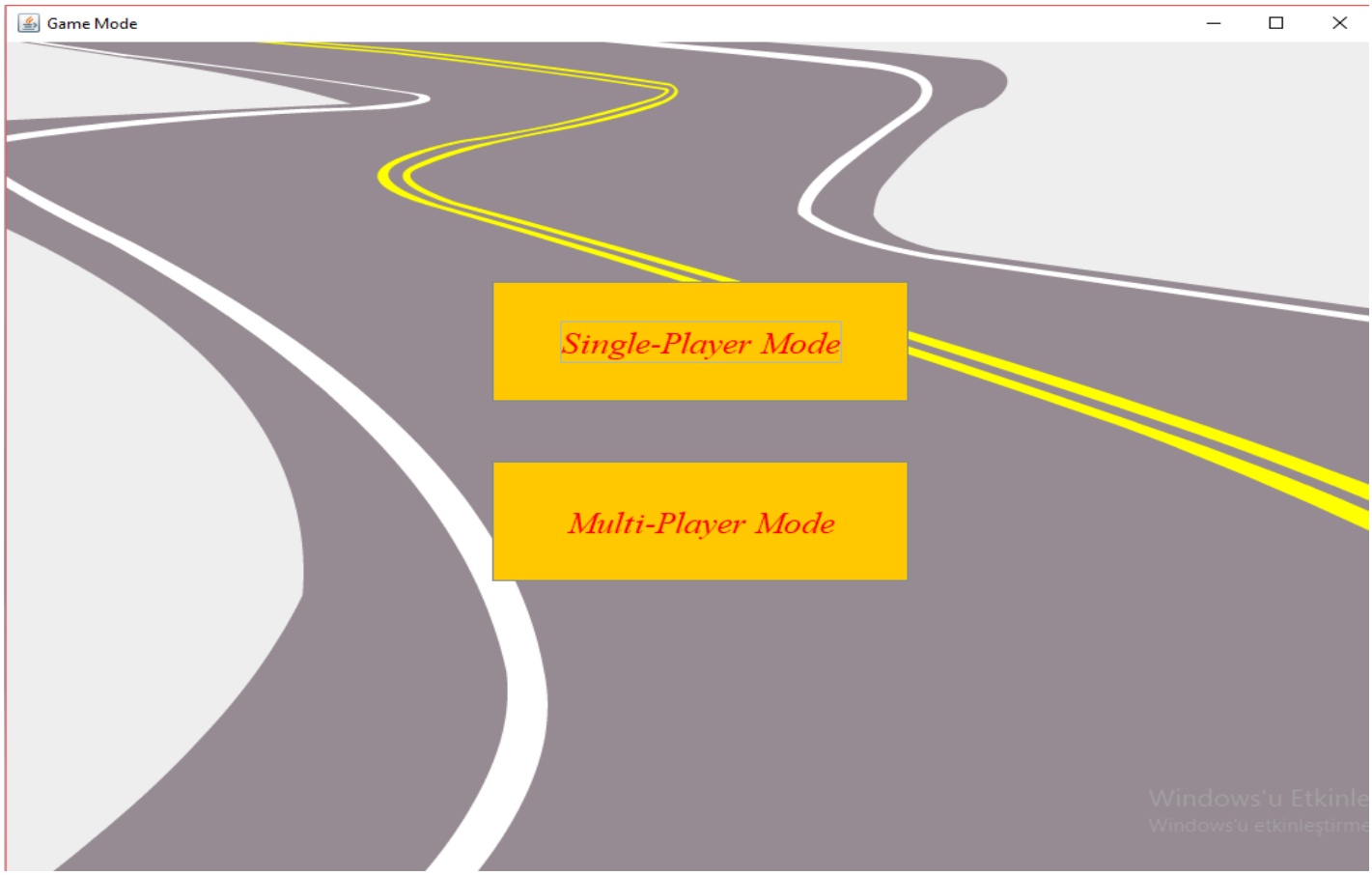


And these are the screenshots from our game:

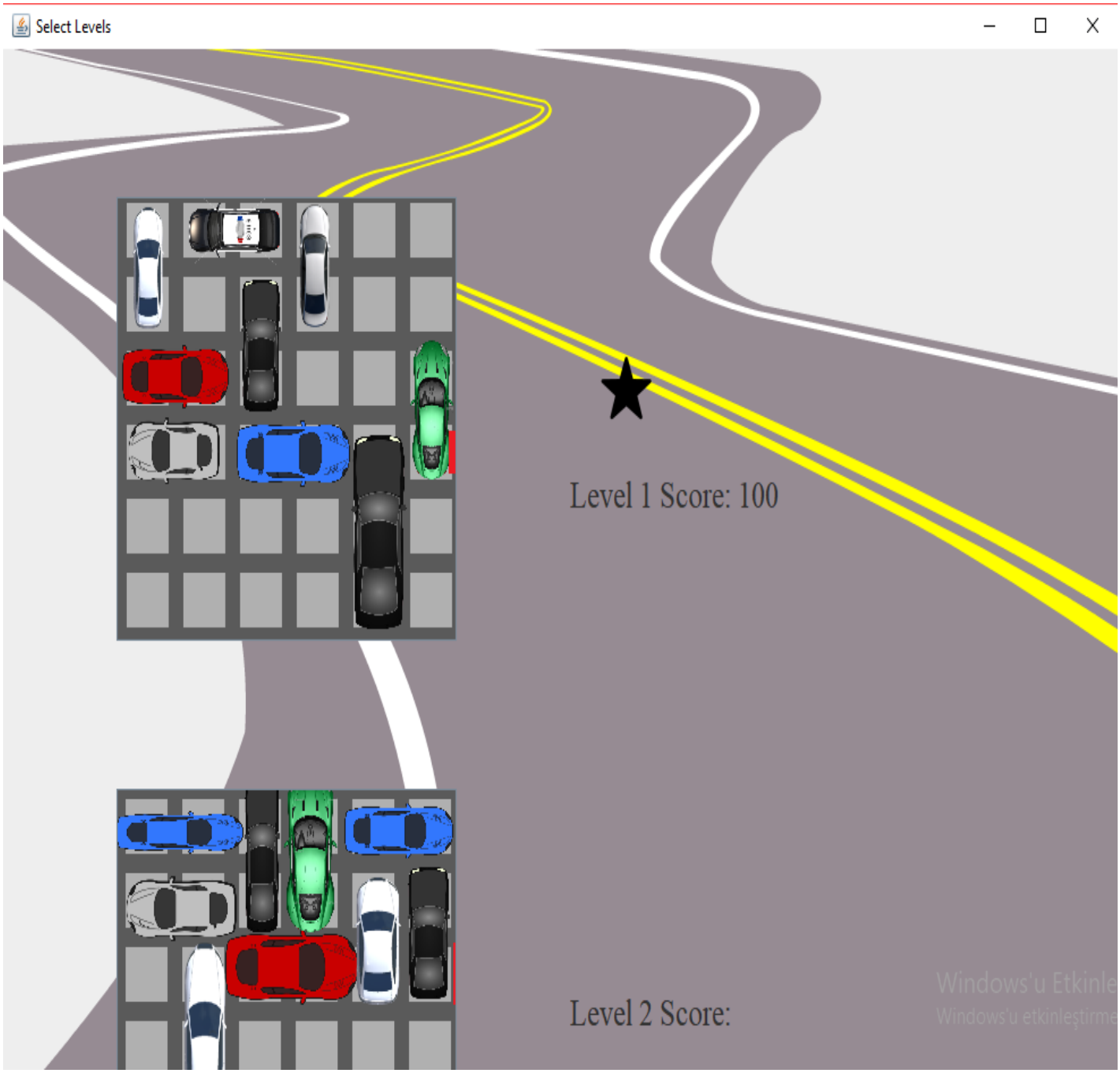
Main Menu:



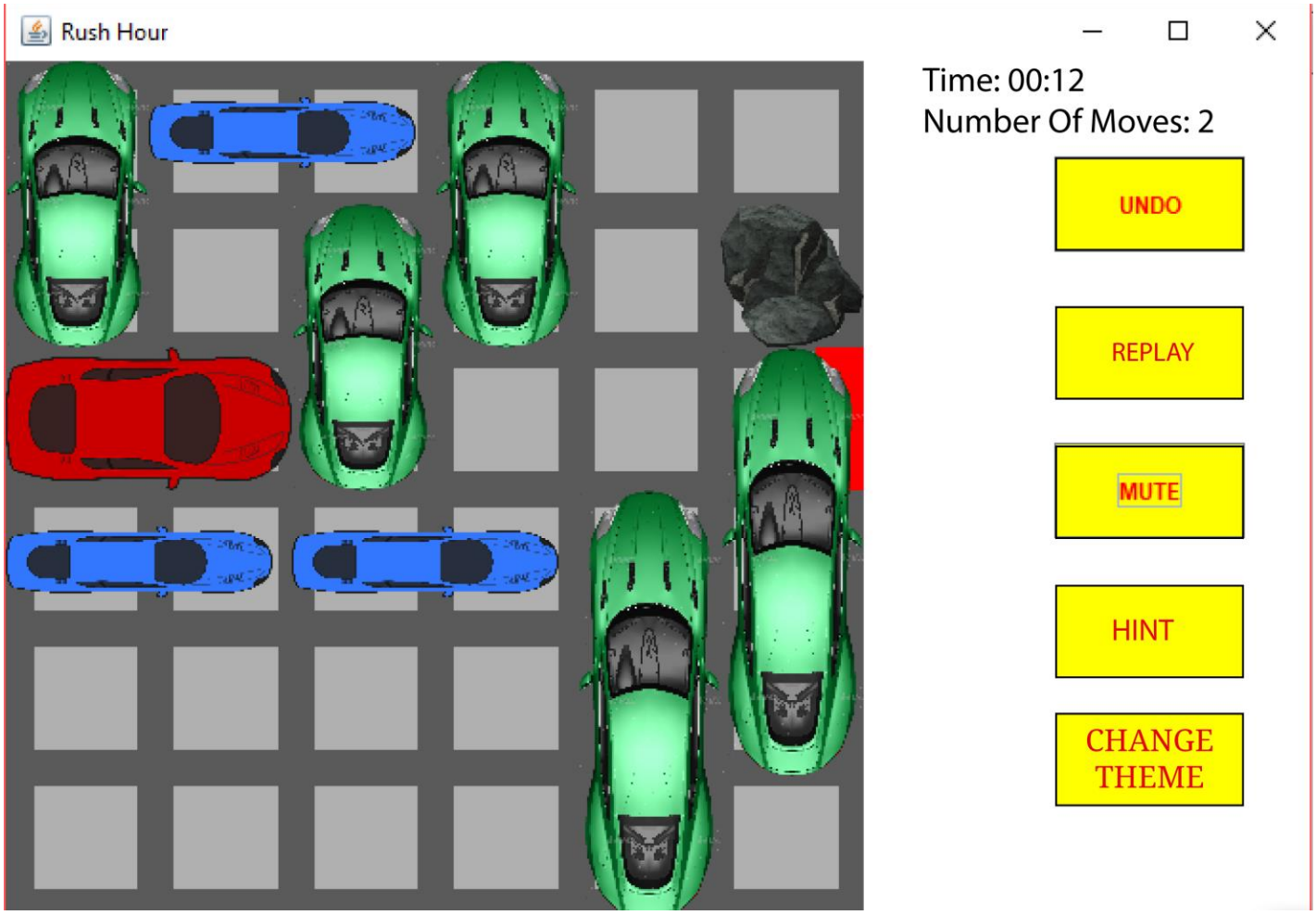
Game Mode Decision Part



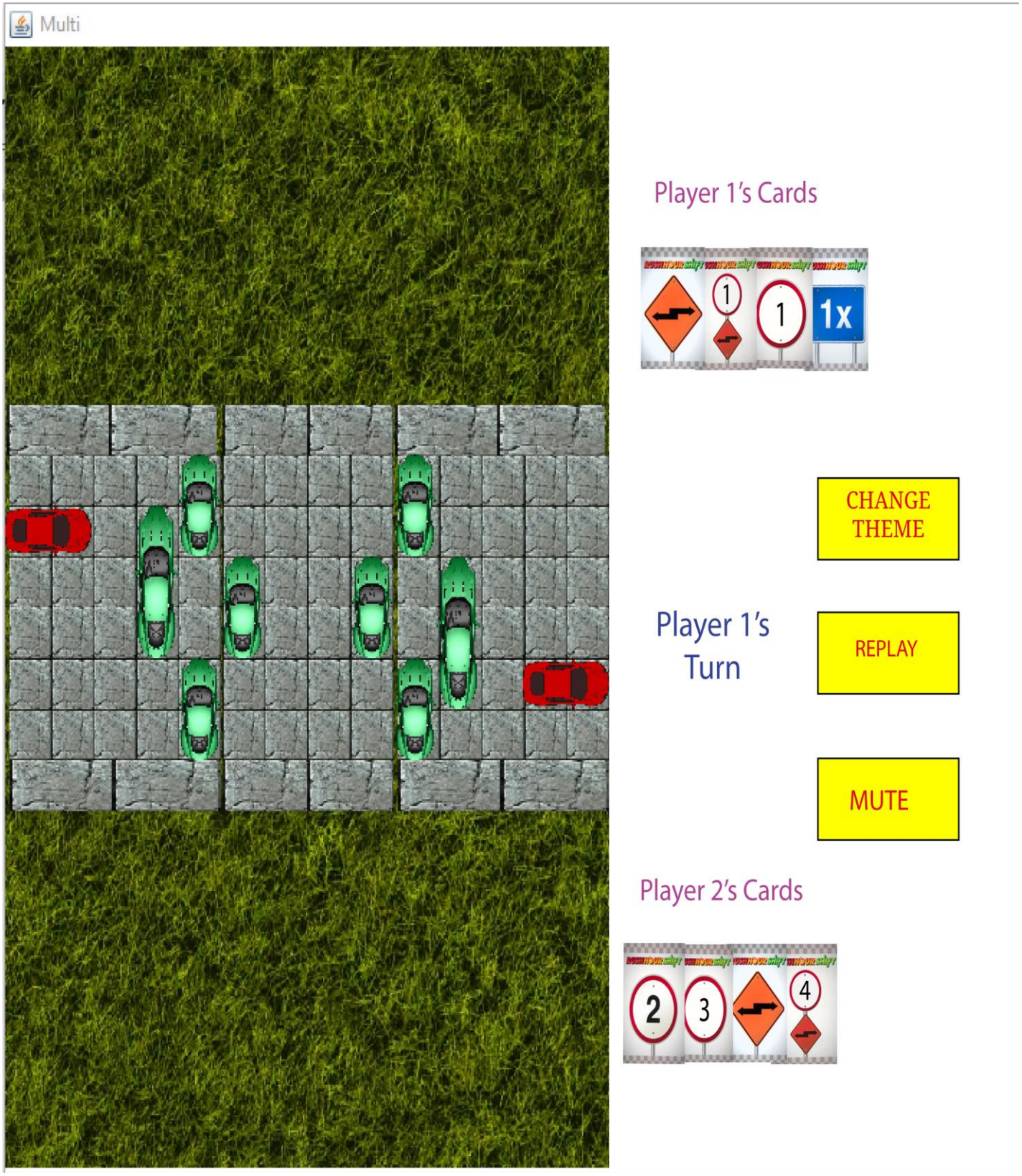
Level Selection Part:



Single Player Part:

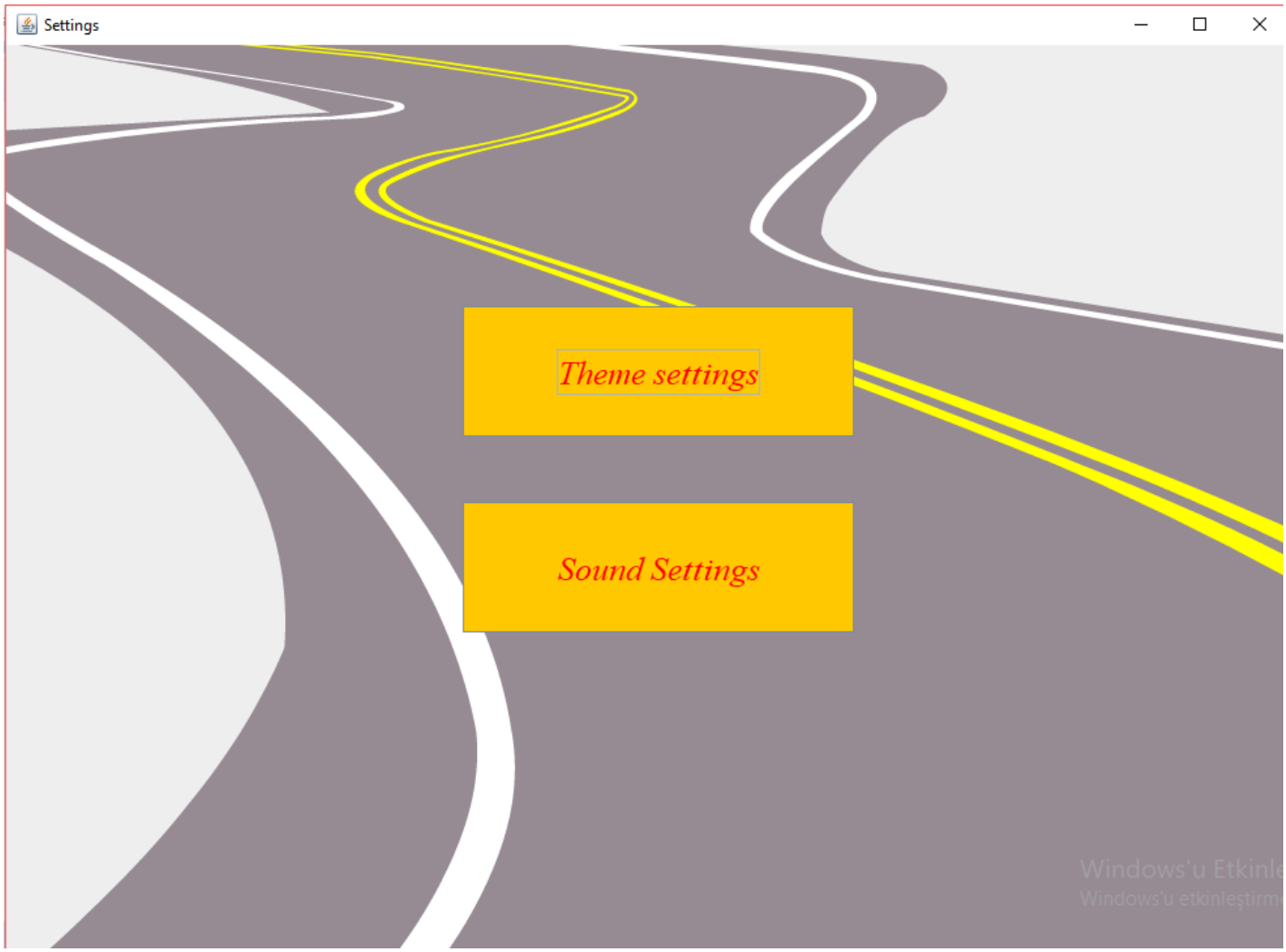


Multi Player Part:

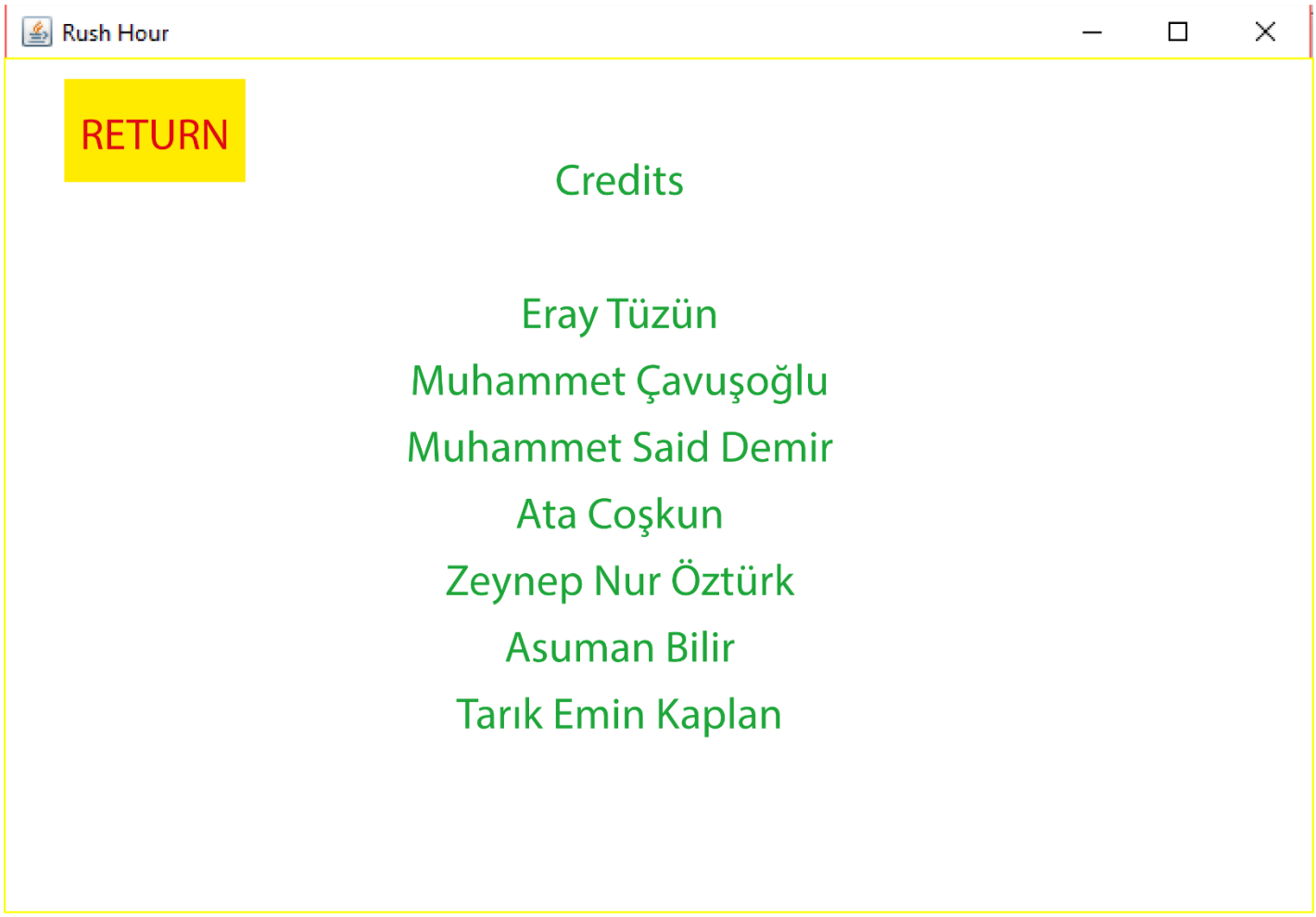


How to Play Part:

Settings Part:



Credits Part:



6. Conclusion

As a result, the Rush Hour is a fun game which can be played both as single player or multi player. In this report, we've talked about how we've implemented it and we can say that we've implemented %90 of the game. But just like our instructor (Eray Tüzün) told us, remaining %10 is ought to take more time than the %90 we've already implemented. Again, in this report we've showed our diagrams to indicate how we understand and designed the game. For more information, you can always check our GitHub page, where whenever one of us makes a change we update it, or you can mail us for arranging a meeting to further see our implementation or just want the code as a zip file. Or you can mail us if you're having a hard time seeing the diagrams, we've tried to show it as clear as possible but this is the best it gets, you can have hard time seeing the little texts in there, in such case you can mail us to get .vpp files.

7. Glossary & References

https://www.thinkfun.com/products/rush-hour-shift/

https://www.thinkfun.com/products/rush-hour/