
# CS319 Object-Oriented Software Engineering AnalysisReport Iteration-2

## RUSH HOUR

### GROUP 3.B

Musab Gelişgen
Kuluhan Binici
Yusuf Dalva
Ahmet Avcı
Melih Ünsal

## 1. Introduction

We, as team 3.B, preferred to implement the game “Rush Hour” as our CS319 term project. *Rush Hour* is a sliding block puzzle invented by Nob Yoshigahara in the 1970s[1]. Normally it is a single player cardboard game in which the purpose of the player is to elude his car from the jammed traffic. However, since we are going to implement it in a digital environment, we decided to enhance the regular design by adding extra features that we believe would make the game more alluring.

Here are some of our ideas of features to the game:

- Single Player Mode
    - Predefined 10 maps for Single Player mode
    - Gain stars in every level by trying to complete the map with as little movements as possible
    - Optimal move count for every map is shown to the users for them to think the best way to solve the puzzle.
    - Earn 3 stars by completing the map in the optimal move count, 2 stars by acceding optimal case by one move and 1 start at the remaining conditions.
    - Star score tracking for every level in Single Player mode for unlocking car skins and themes with the total star numbers.
    - Unlock next levels by completing the previous ones.
    - Ask for help to solve the puzzle from the game by clicking the hint button.
- Multiplayer Mode
    - Power-ups (obtained from wildcards) for multiplayer mode.
    - Shifting map for player’s benefit, moving obstacle cars or moving escape car according to the wildcard power-ups.
    - 2 modes one of which is PvP (Player vs Player) and the other one is PvE (Player vs Environment)
- Sound options
- Different theme options for maps
- Different skin & color options for cars
- Smooth Graphics

To implement the game we are going to use Java and LibGDX for the design/graphics part. The game will be a Desktop Application but thanks to the framework we use, we can transform the game to the android game by a few changes. We are also going to use A* algorithm to find the shortest path from the current map to the winning position of the game. As we implement the game,

[1] “Rush Hour®.” Thinkfun, www.thinkfun.com/products/rush-hour/.

concepts we learned from CS courses, especially the ones in CS 319 are going to be used throughout the project since this is not a 500-line project.

Before we design the model, we played the original Rush Hour game and watched some videos about Rush Hour Shift mentioning about how to play the game and what the rules are and also how the game is designed in real life so that we are sure about how to design the game and make improvements on it.

In the rest of the report, we have clarified our analysis in order to design and implement the project. We have tried to explain the most desirable properties of the game in detail. Then, we have enriched the design of the game by adding extra features in comparison with the actual game.

## 2. Overview

This section contains descriptions of the game we are implementing and its features.

### 2.1 Gameplay

Rush Hour is a 2D game. The game has a single-player mode and a multiplayer mode. Tools included in the game are blocking cars, an escape car which the player controls, a map which all the cars are placed on (in different directions) and play cards which grant the players some special movements (cards are for multiplayer only).[2]

In single-player mode, given a predefined map of cars, the player will try to escape his car from a maze-like structure by moving the blocking cars around. For our version of the game, we plan on rewarding the player with stars based on his/her performance while passing the levels. There will be move count thresholds for the number of stars rewarded and if the player completes a level within less number of moves than a threshold, then he/she will be awarded with adequate number of stars. For instance, if the 3-star threshold for a level is 15 moves and the player solves the puzzle in less moves, he/she will be awarded 3 stars.

In multiplayer mode, two people race their cars to the end of the map by blocking the opponent's way and opening theirs. This mode is played in turns. The first one to get his car out of the platform wins the game. In this game mode, players also draw wildcards which grant them special abilities like moving the platform for their own good or having the ability to move their car by multiple times in a round.

[2] “Rush Hour® Shift.” Thinkfun, www.thinkfun.com/products/rush-hour-shift/.

## 2.2 Player Modes

The game will provide both singleplayer and multiplayer options. In the single-player mode, the player will try to escape his car from the map to pass the current level or to gain more stars than before.

In the multiplayer mode, 2 players will compete against each other to rescue their cars before the other player does.

## 2.3 Cars

There are 2 types of cars in the game: Escape cars - which the player tries to rescue from the map - and blocking cars. Blocking cars come in different sizes such as 2x1 and 3x1 units. Escape car has a 2x1 unit size. All types of cars have different colors for a visual variance.

All the cars on the field can be moved for their direction only, meaning they cannot be moved sideways. These obstacles can be moved by any number of unit blocks as long as their path is not blocked by other cars or the borders of the map.

## 2.4 Maps

For single-player mode, we plan on implementing 10 predesignated square maps which have different star thresholds. In each of these maps, players will try to get their cars out of the maze by moving blocking cars around.

For multiplayer mode, our plan is to have 3 predesignated maps. These will be rectangular maps which are larger than the ones in the single-player mode. Multiplayer maps will consist of 3 pieces, independent of each other. These 3 separate pieces can be moved sideways (with special wildcard abilities explained in section 2.5). These map movements will make the map extend on one side and retract on the other side (i.e. change the rectangular shape of the map and make it a polygon) Player’s cars will start from opposite edges of the game platform. There will be blocking cars on this map as well.

## 2.5 Wild Cards

There will be no cards in the single-player mode. In the multiplayer mode, there will be a pile of wild cards from which each player will draw a single card at the beginning of each turn. The players will not know which card they will receive until they draw the card as the card drawing process is completely at random.

According to the card’s special effect, players will be granted a special movement. These special movements include but are not limited to sliding the platform, sliding the opponent’s car by one layer backward or multiple layers sideways, moving their own car by multiple grids (by multiple layers forward or sideways).

### 2.6 Settings

The player is able to choose the game mode (singleplayer or multiplayer). In the sound settings menu, the player can turn on/off the music, change the music and adjust its volume. The player will be able to choose a theme for the map and skin for his escape car.

## 3. Functional Requirements

In this section of the report, interactions between the system and its environment independent of its implementation will be described .

### 3.1 Additional Requirements

In this iteration of the game, we have eliminated the difficulty levels from the game and added the star system. With this new system, we have reached more logical and balanced way for choosing theme and car skins. Also we have success tracker system with stars for players on specific levels and this brings more competitive atmosphere to the game.

We have added a hint option for user in order to help them to find the optimum path. We are planning to achieve this by using the A* algorithm.

Finally, we have decided to add AI bot for multiplayer mode. Thanks to this feature, without any person opponent, players will be able to play multiplayer game against bot.

### 3.2 Play

- Play Singleplayer Mode: Choose this mode, select the map and start to play game.
- Play Multiplayer Mode: Choose this mode, select map and start to play game.
- Play Against AI Bot: Chose multiplayer mode and then AI Bot mode and play against our reinforcement learning agent.
- Select Map: Before starting to the game, user needs to choose the level.
- Take a Hint: See the next best move by taking help from the game.
- Move a car: Player clicks on a car and moves it by dragging on the map (possible exceptions are explained in non-functional requirements part).
- Draw Wild Card: In multiplayer mode, before making any move in the game, user need to draw a card that gives the specific information and instruction about the next move.
- Shift the map: In the multiplayer mode, if player draws a “shift the map” card, he/she could move a piece of the map (Map is consisted of 3 shiftable pieces).

- Winning starts: After finishing the game, players get starts according to their move number compare to optimum number.

### 3.3 Select Map Theme

- Choose the map theme that specialize the user interface in the game.
- Available themes are determined by users total stars.

### 3.4 Select Car Theme

- Choose the car skin that specialize the cars in the game.
- Available skins are determined by player's total stars.

### 3.5 Adjust Music

Players can adjust the music preferences;

- Change the main music: Players can change the main theme music or close it as they want.
- Change the effect music: Players can change the effects or close them as they want.
- Adjust the level of music: Players can adjust the level of music in the game.

### 3.6 How to Play

Players can get the instructions about how to play game if they want before starting to game.

### 3.7 Credits

Players can see the information about developers and the game.

## 4. Non-functional Requirements

In this part of the report, requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors will be explained.

### 4.1 Additional Requirements

In this iteration of the report, we have increased the performance of game and create more balance between new functionalities and performance with new non-functional requirements. Also, we have increased the supportability of our program with our new design.

- For hint option, finding the optimum path should take less than 5 seconds.
- For the AI bot mode, the time required for bot to move should take less than 5 seconds.

- JAR file will be created.
- Java SE 8

## 4.2 Usability

- Simple User Interface: The buttons are separated from each other (at least 25 pixels) in order to be distinguished easily. Colors are selected accordingly to be distinguished easily by anyone e.g. color blinds. Also, conventions adopted by the user interface.
- Easy-to-use functionalities: All of the functionalities of the game are designed to be done easily thanks to their simplicity and easy-to-follow instructions.
- No need for prior experience: Thanks to the easy-to-use functionalities, everyone can easily adopt the game by playing just one level without needing any tutorial or prior experience.

## 4.3 Reliability

- Input Handling: If there is enough space in the PC RAM (256 MB), all the functional requirements would be handled with in 0.5 seconds.
- Error Handling: In the case of invalid inputs such as car crashes, grid system will not allow the selected car movement and game will stay on its last state.

## 4.4 Performance

- Response Time: It must be at most 1 second.
- FPS: Refresh rate of at least 60 frames per second.
- Availability: The system must be up and operable for the whole run time of the game.

## 4.5 Supportability

- Adaptability: Thanks to our strong object oriented design, the system will always be open to the possible changes.
- Portability: No installation will be required because the execution file is JAR. Thanks to Java SE 8 , our game can be executed even with old systems.

## 5. System models

In this section of the report, design of the system is explained with the required diagrams which includes functional, object and dynamic models.

## 5.1 Use case model

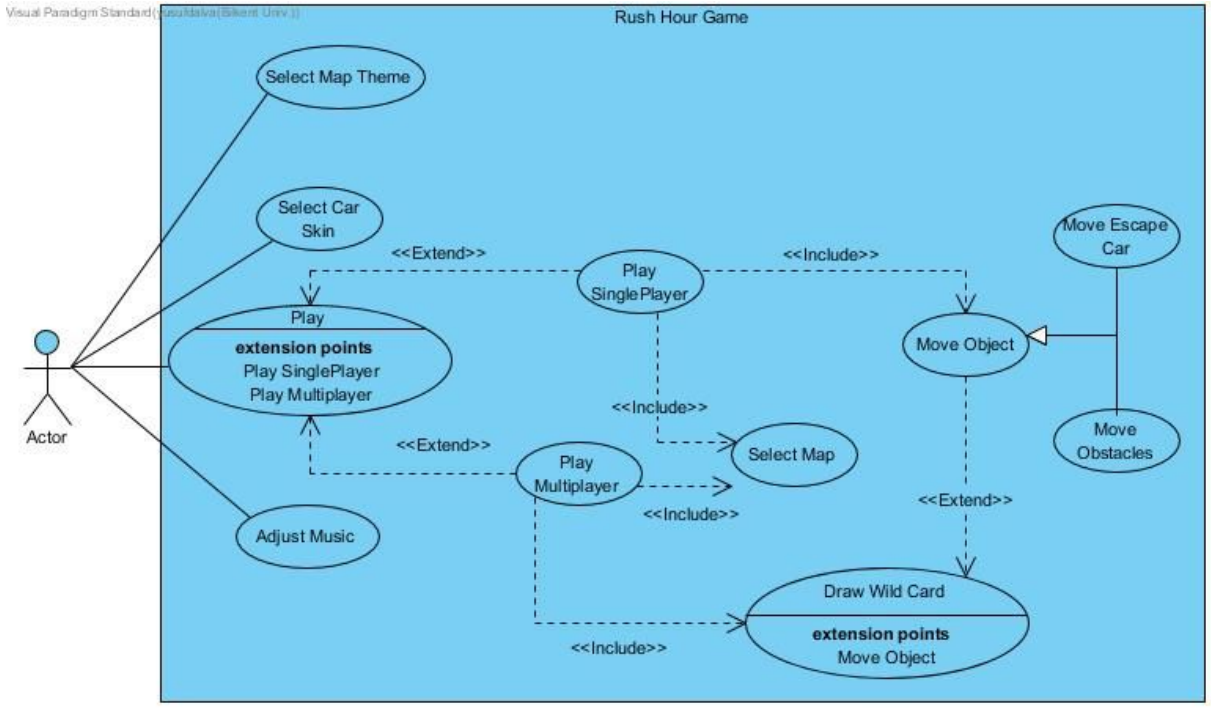


*Figure 1: Use Case Diagram for the game Rush Hour*

Use Case #1:

| **Use Case Name** | Select Map Theme |
|---|---|
| **Participating Actors** | Initiated by Actor |
| **Flow Of Events** | 1. The Actor activates the Select Map Theme option from the Main Menu<br>2. Rush Hour Game responds by showing the available themes to the actor<br>3. The Actor makes a selection among all the themes shown by the Rush Hour Game.<br>4. Rush Hour Game receives the selection that the Actor made. The map selection is saved in the preferences of the game |

| | |
|---|---|
| **Entry Condition** | The Actor is in the Selecting Map Theme Panel in Main Menu |
| **Exit Condition** | - The selection made by the Actor is applied<br>- Rush Hour Game receives the selection made by the Actor |
| **Quality Requirements** | None |

Use Case #2:

| | |
|---|---|
| **Use Case Name** | Select Car Skin |
| **Participating Actors** | Initiated by Actor |
| **Flow Of Events** | 1. The Actor activates the Select Car Skin option from the Main Menu<br>2. Rush Hour Game responds by showing the available car skins to the Actor<br>3. The Actor makes a selection among all the car skins supplied by the Rush Hour Game<br>4. Rush Hour Game receives the selection made by the Actor. The car skin selections (for player one and player two) are saved in the preferences of the game |
| **Entry Condition** | The Actor is in the Selecting Car Skin Panel in Main Menu |
| **Exit Condition** | - The selections made by the Actor are applied<br>- Rush Hour Game receives the selections made by the Actor |
| **Quality Requirements** | None |

Use Case #3:

| | |
|---|---|
| **Use Case Name** | Play |
| **Participating Actors** | Initiated by Actor |

| | |
|---|---|
| **Flow of Events** | 1. The Actor is willing to Play the game<br>2. Rush Hour Game is waiting for the response of the user which is either selecting Single Player Mode or Multiplayer Mode<br>3. The Actor selects a game mode |
| **Entry Condition** | The Actor is in the Main Menu |
| **Exit Condition** | The Actor makes a selection implying the preference of game mode |
| **Quality Requirements** | None |

Use Case #4:

| | |
|---|---|
| **Use Case Name** | Adjust Music |
| **Participating Actors** | Initiated by Actor |
| **Flow Of Events** | 1. The Actor activates the Adjust Music option from the Main Menu<br>2. Rush Hour Game responds by showing the sound and music options to the Actor<br>3. The Actor adjusts the sound and music according to its desire<br>4. Rush Hour Game receives the selections that the Actor made. These sound and music selections are kept in the preferences of Rush Hour Game |
| **Entry Condition** | The Actor is in the Adjust Sound Panel in the Main Menu |
| **Exit Condition** | - The selections that the Actor made are applied from the Panel<br>- Rush Hour Game receives the selection made by the Actor |
| **Quality Requirements** | None |

Use Case #5:

| | |
|---|---|
| **Use Case Name** | Play SinglePlayer |

| | |
|---|---|
| **Participating Actor** | Initiated by Actor |
| **Flow Of Events** | 1. The Actor shows interest in playing the game<br>2. Rush Hour Game waits from the Actor to make a preference for the game mode<br>3. The Actor selects the Single Player option from the Main Menu<br>4. Rush Hour Game initializes the game according to the game mode preference and other preferences specified by the user |
| **Entry Condition** | This use case extends the Play use case. This use case includes “Select Map” and “Move Object” use cases. This use case is initiated just after the Actor makes the preference of game mode as Single Player Mode |
| **Exit Condition** | - The Actor specifies the preference of exiting the game<br>- The level constraints make the Actor to leave the game |
| **Quality Requirements** | None |

Use Case #6:

| | |
|---|---|
| **Use Case Name** | Play Multiplayer |
| **Participating Actor** | Initiated by Actor |
| **Flow Of Events** | 1. The Actor shows interest in playing the game<br>2. Rush Hour Game waits from the Actor to make a preference for the desired game mode<br>3. The Actor selects Multiplayer option from the Main Menu<br>4. Rush Hour Game initializes the game according to the game mode preference and other preferences specified by the user |

| | |
|---|---|
| **Entry Condition** | This use case extends the Play use case. This use case includes “Select Map” and “Draw Wild Card” use cases. This use case is initiated just after the Actor makes the preference of game mode as Multiplayer mode |
| **Exit Condition** | - At the moment that one of the users successfully completes the challenge, game ends<br>- Actor specifically shows the interest on exiting by the exit button |
| **Quality Requirements** | None |

Use Case #7:

| | |
|---|---|
| **Use Case Name** | Move Object |
| **Participating Actor** | Initiated by Actor |
| **Flow Of Events** | 1. Rush Hour Game indicates that the user will make a move<br>2. After that the user will make a specific move by moving the cars to the available spots<br>3. Rush Hour Game recognizes the move and processes it, by taking the move count into consideration. Each Move Object case indicates 1 move in the game<br>4. The turn of the player ends |
| **Entry Condition** | The playing turn is at the player, the game is still not ended |
| **Exit Condition** | The available moving object operation is recognized and processed by the Rush Hour Game. The turn ends. |
| **Quality Requirements** | The move that the user makes is valid according to the game rules. |

Use Case #8:

| **Use Case Name** | Select Map |
|---|---|
| **Participating Actor** | Initiated by Actor |
| **Flow Of Events** | 1. After selecting the game mode (either single player or multiplayer), Rush Hour Game settles up a frame that makes the user to select a puzzle to play<br>2. Among all the possible options, the Actor selects a map to play.<br>3. If the map selection is valid, Rush Hour Game takes the selection into consideration. Else, the Actor will get a warning sound and will have to pick another map<br>4. Rush Hour Game gets the map selection and initiates the level according to the selection |
| **Entry Conditions** | - A game mode is selected by the Actor. |
| **Exit Conditions** | The Actor made a valid selection as map (among the selectable ones) and the Rush Hour Game initiated the level according to the preference of the Actor. |
| **Quality Requirements** | The map preference made by the user must be one of the selectable maps (not restricted with the lock icon) |

Use Case #9:

| **Use Case Name** | Draw Wild Card |
|---|---|
| **Participating Actor** | Initiated by Actor |
| **Flow Of Events** | 1. Actor has the playing turn in the game<br>2. Rush Hour Game gives a random wild card to the Actor which contains the move that the Actor can make in any car. |

| | |
|---|---|
| | 3. According to the wild card, the restrictions of the player is settled again for the user to make a move |
| **Entry Conditions** | - The selected game mode is multiplayer mode<br>- Map selection is made by the Actor<br>- The Actor has the turn to play |
| **Exit Conditions** | The Actor receives the wild card and Rush Hour Game restricts the possible moves that can be done accordingly |
| **Quality Requirements** | None |

Use Case #10:

| | |
|---|---|
| **Use Case Name** | Move Escape Car |
| **Participating Actors** | Inherited from Move Object use case |
| **Flow Of Events** | 1. The turn to play is at the Actor<br>2. The player selects to move the Escape Car<br>3. If the place that the Actor decides to play is compatible with the restrictions, the move is reported to the Rush Hour Game<br>4. If the move is valid Rush Hour Game processes the move |
| **Entry Conditions** | - The playing turn is at the Actor.<br>- The object selected to move is the Escape Car<br>- A game mode and map is selected |
| **Exit Conditions** | - The Actor makes a move with the escape car that is compatible with the restrictions<br>- The move done is processed by the Rush Hour Game |
| **Quality Requirements** | The move that the Actor does is not against the restrictions defined by the game |

Use Case #11:

| **Use Case Name** | Move Obstacles |
|---|---|
| **Participating Actors** | Inherited from Move Object use case |
| **Flow Of Events** | 1. The turn to play is at the Actor<br>2. The Actor selects to move one of the Obstacles<br>3. If the place that Actor decides to play is compatible with the restrictions, the move is reported to the Rush Hour Game<br>4. If the move is valid, Rush Hour Move processes the move |
| **Entry Conditions** | - The playing turn is at the Actor<br>- The object selected to move is the Escape Car<br>- A game mode and map is selected |
| **Exit Conditions** | - The Actor makes a move with an obstacle that is compatible with the restrictions<br>- The move done is processed by the Rush Hour Game |
| **Quality Requirements** | - The move that the Actor does is not against the restrictions defined by the game |

## 5.2 Dynamic models

In this section, some possible scenarios will be explained.

- Play Single Player Mode, Move Cars: Choose the single-player mode and play the game by moving the cars.
- Play Multi-Player Mode, Move Cars: Choose the multiplayer mode and play the game against each other by drawing cards and moving cars.
- Play Multi-Player Mode, Slide Map: Choose the multiplayer mode and play the game against each other by drawing cards and moving map.
- Adjust Settings: Open the game and adjust the game.

### 5.2.1 Sequence Diagrams

Scenario #1:Play Single Player Mode, Move Cars

Actor: Actor

The actor opens the game and chooses the single player mode. Then, the map selection menu comes up and the actor chooses the game map. After making these arrangements, the game starts and the actor plays the game by moving the cars. Unless the game ends, after each drag move performed, current game is going to be updated (by updating car location). As long as the game is not terminated, the actor will make another move by dragging a car object and the game continues.

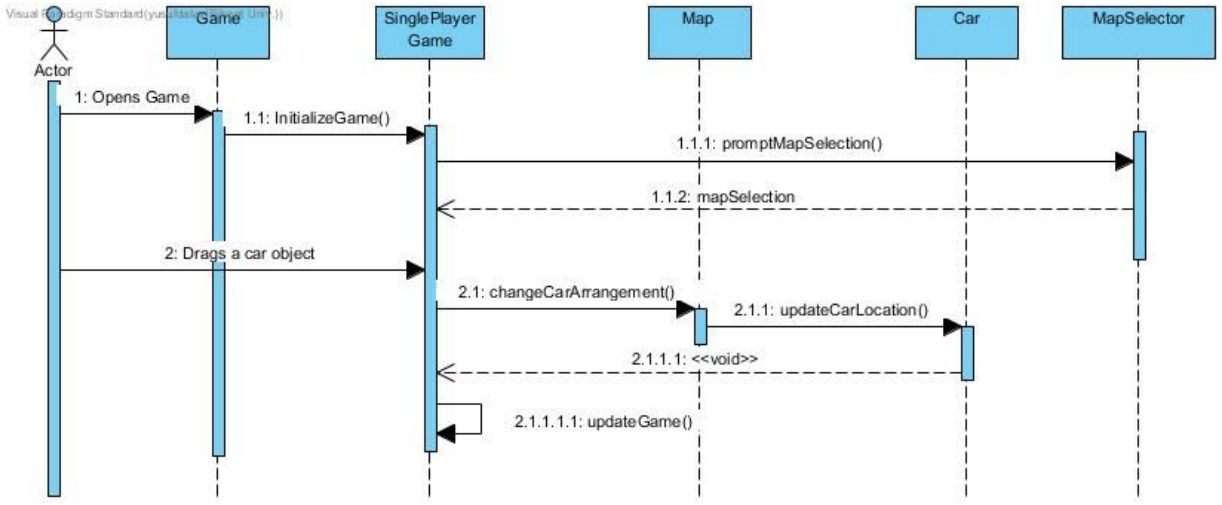


*Figure 2. Sequence Diagram for Scenario #1*

Scenario #2:Play Multiplayer Mode, Move Cars

Actors: Kulu(Actor 1) and Musab/Bot(Actor 2)

Kulu(Actor 1) opens the game and chooses the multiplayer mode and the game initialized. Then, the map selection menu comes up and Kulu chooses the game map. After selecting a map, the game starts and Kulu draws a movement card and move a car according to this card. If the game does not end according to this move, Musab is going to perform a move. In this case, Musab draws a card and move a car according to the card. If the game still continues the game proceeds with

the move of Actor 1. The game is going to be updated after each move since move of the first actor effects the move of the second actor.

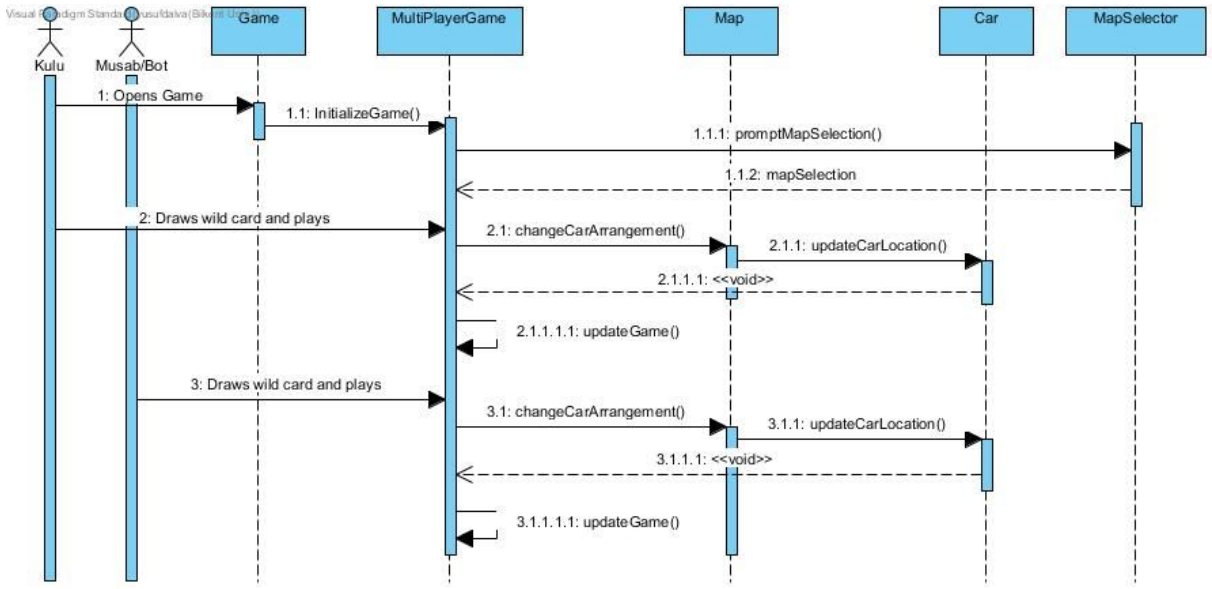


*Figure 3. Sequence Diagram for Scenario #2*

Scenario #3:Play Multiplayer Mode, Slide the Map

Actors: Kulu(Actor 1) and Musab(Actor 2)

Kulu(Actor 1) opens the game and chooses the multiplayer mode. Then, the map selection menu comes up and Kulu chooses the game map. After selecting a map, the game starts and Kulu draws a movement card and slides the one piece of the map according to the card drawn. After Kulu, Musab(Actor 2) draws a card and slides the one piece of the map according to the card drawn. After each move performed the game is going to be updated as the move performed by the first actor effects the move of the second actor.

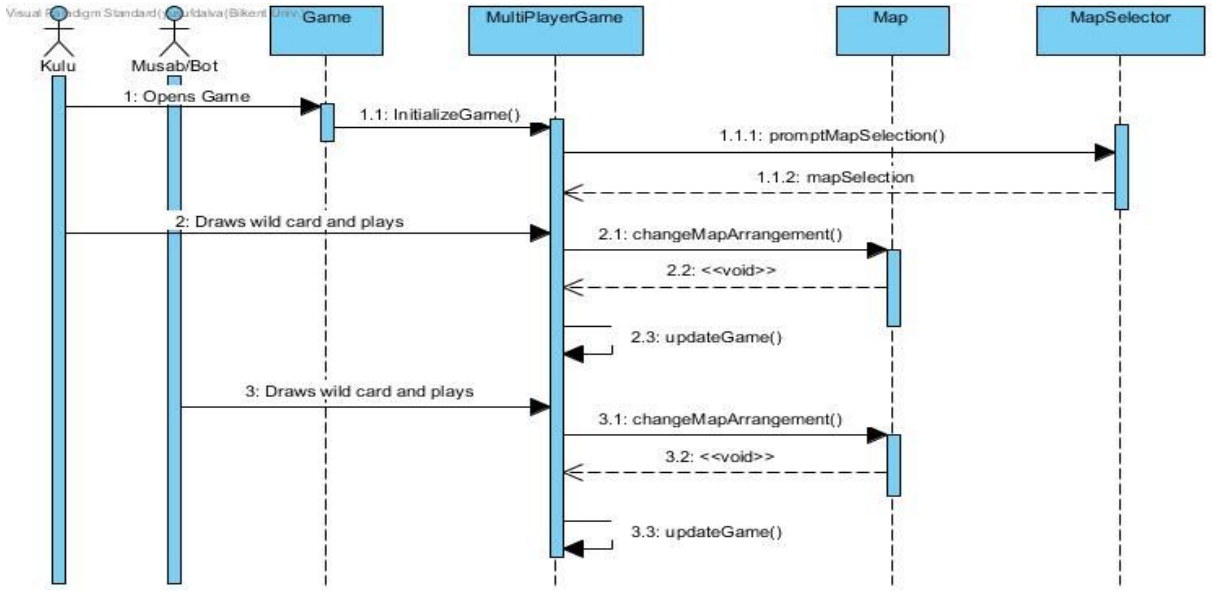


*Figure 4. Sequence Diagram for Scenario #3*

Scenario #4:Adjust Settings

Actors: Kulu

Kulu opens the game and selects the car selection menu. Then, he chooses his car and turns back to the main menu. After this, Kulu selects the theme selection menu and chooses his theme, and turns back to the main menu. Then he selects sound menu and adjusts the sounds. After that, he returns the main menu, gets into Star Scores Menu and he sees the records for the stars obtained for each level.

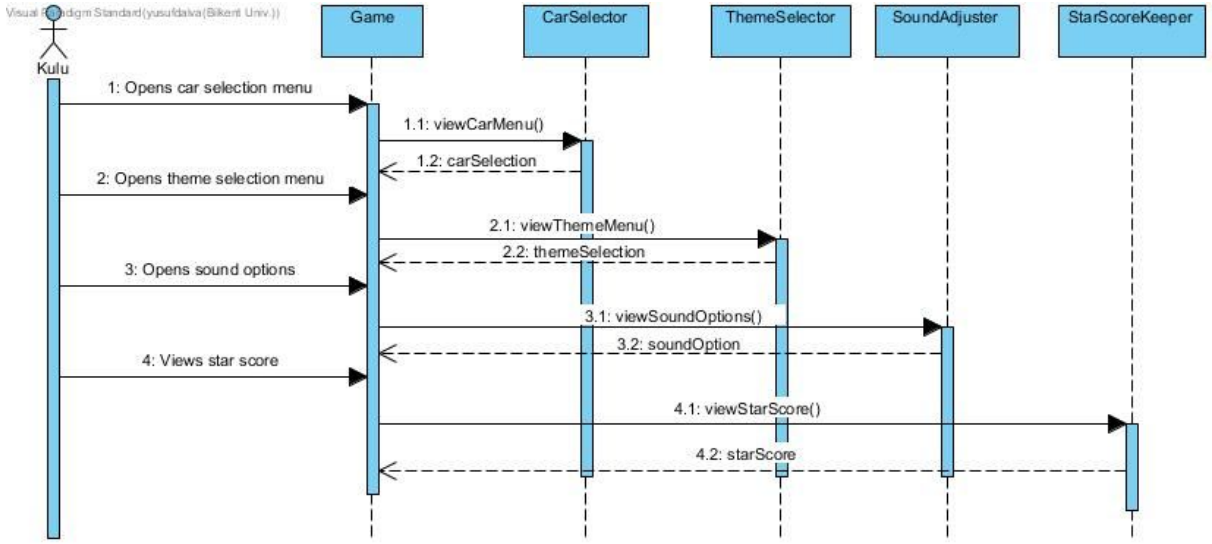


*Figure 5. Sequence Diagram for Scenario #4*

### 5.2.2 Activity Diagram

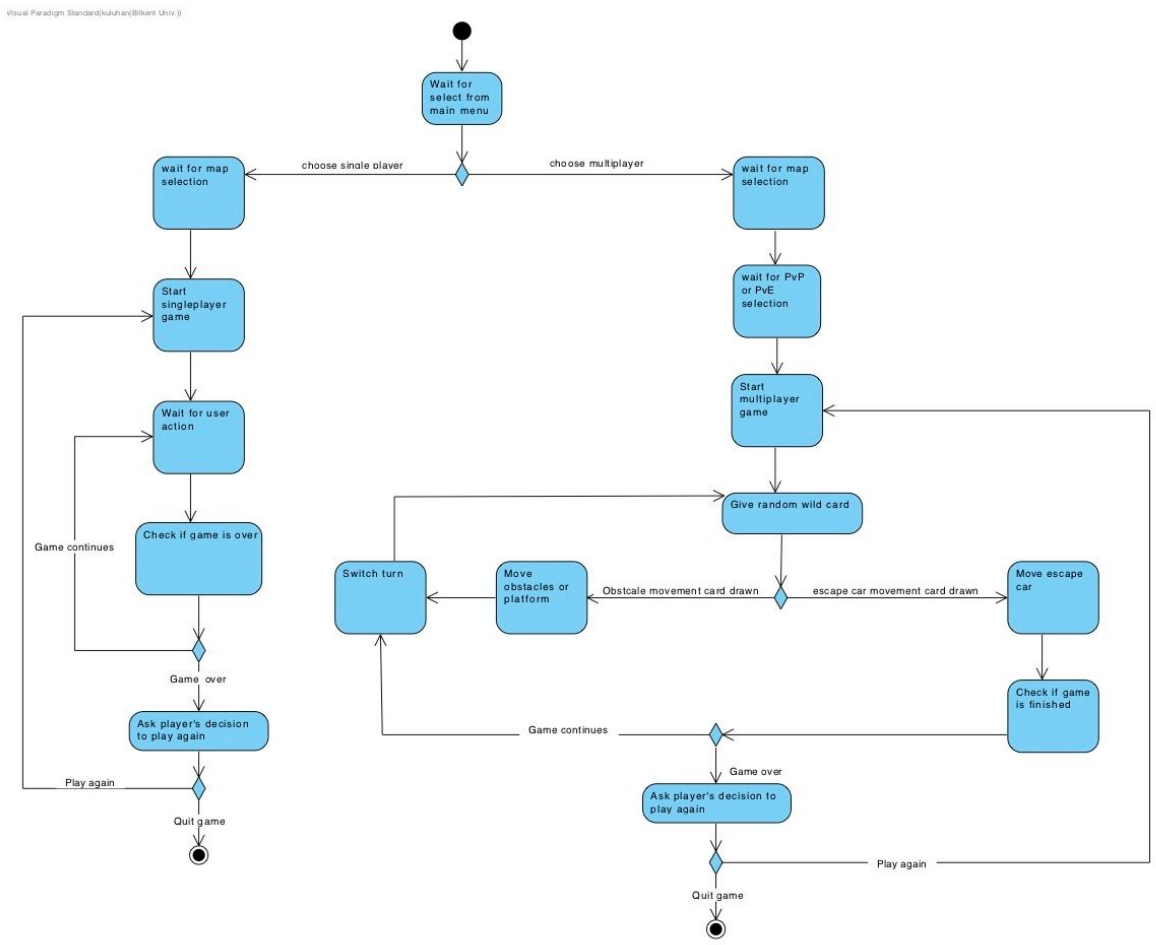


*Figure 6. Activity Diagram for Rush Hour game*

After the game has started, main menu comes up and waits user to select game mode as single or multiplayer. If the users selects single player mode, the game waits for the map selection. Then the game starts and waits for user to move a car. At the end of every turn, the game checks whether the game is over or not. If the game is not over, again it waits for user to move a car and it goes like that until the game is over. When the game is over, the game asks for the user's decision to continue the game or quit.

If the user selects multiplayer mode, game waits for user to choose PvP (Player vs Player) or PvE (Player vs Computer) mode. Then the game waits for the map selection. Then the game starts and gives a random wild card to the players in each turn starting from the first user. If the card is an escape car movement card, the users need to move the escape car. After moving the escape car, the game checks

whether the game is over or not. If the game is not over, the game continues by switching the player turn. However, if the game is over, the game asks for user to start a new game or quit. On the other hand, if the given card is a obstacle movement car, the users need to move one of the obstacle cars or the map and the game continues by switching the player turns. The game does not check whether the game is over or not after obstacle movement car. This is because the game can be over only by reaching the finish by moving the escape car.

### 5.2.3 State Diagrams

The overall state diagram is illustrated in separate diagrams for this part. The aim of this separation of the state diagram is mainly for clarity in representation. The state diagrams mentioned are listed below:

- State diagram for Single Player game (paused, game over or sustained)
- State diagram for Multiplayer game (Player turns, winning the game)
- State diagram for Map (Whether the map is stable or not (for the Map Dragging scenario in multiplayer mode))
- State diagram for Car (States depending on whether the car can move in the desired path or the path is blocked)

#### 5.2.3.1 State diagram for Single Player game

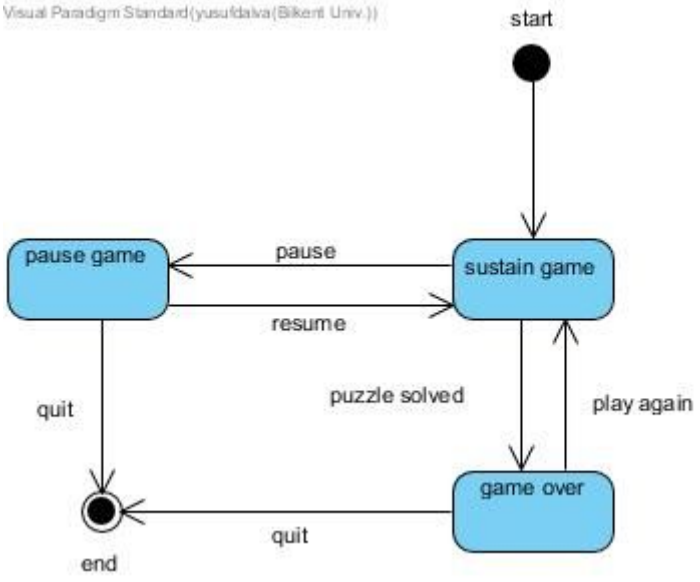


*Figure 7. State Diagram Representing the Single Player mode of the game*

The single player mode of the game has 5 distinct states. After the start of the game (start state), the game is either continuing (sustain game state), paused (pause game state), game ended (game over state) or quitting from the application is performed (end state). As in the beginning the game is going to be continuing, the initial state is “sustain game”. With the moves performed by the user a transition between “sustain game” and “game over” states are enabled. The transition between “pause game” and “sustain game” states are enabled with pause and resuming game actions. As the user performs “quit” action the game ends which directs to the user to the final state which is quitting from the application. The graphical representation is shown in figure 7.

### 5.2.3.2 State Diagram For Multiplayer game

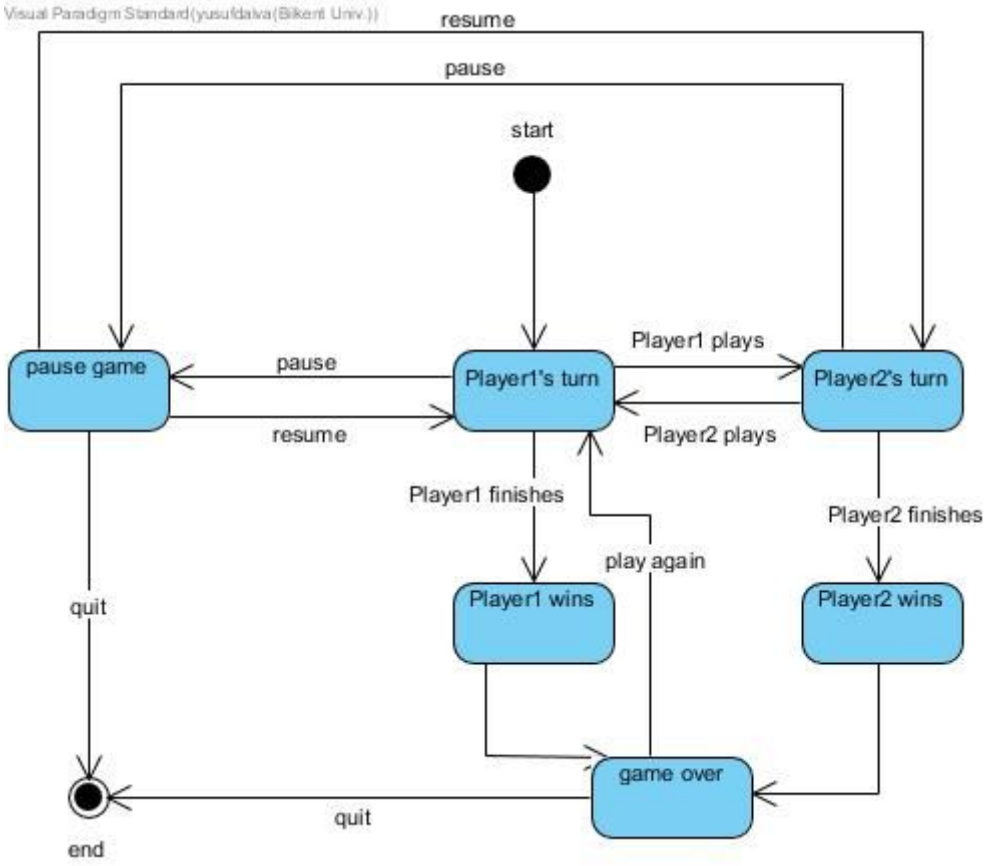


*Figure 8. State diagram Representing the Multiplayer mode of the game*

Different from the Single Player mode of the game, there are two distinct players registered to the game. As the start state, the game starts with the turn of the first player (Player1's turn state). After the turn of Player 1 the game may end or continue depending on the move that the player performed. If the game ends, Player 1 wins the game (Player1 wins state) and the multiplayer game is over (Game Over state). If the game still continues, player 2 is going to play now (Player2's turn state). The game may end according to the action performed by Player 2. If the game ends, Player 2 wins the game (Player2 wins state) and the multiplayer game is over (Game Over state). If the game continues the turn passes to the Player 1 and the game proceeds accordingly. The players can both pause the game and resume the game with "resume" and "pause" action which results with transition with Pause Game state. If the current state is "Pause Game" or "Game Over", the user may "quit" from the game which directs to the end state for the multiplayer game. The graphical illustration is given in figure 8.

#### 5.2.3.3 State Diagram For the Multiplayer Game Map

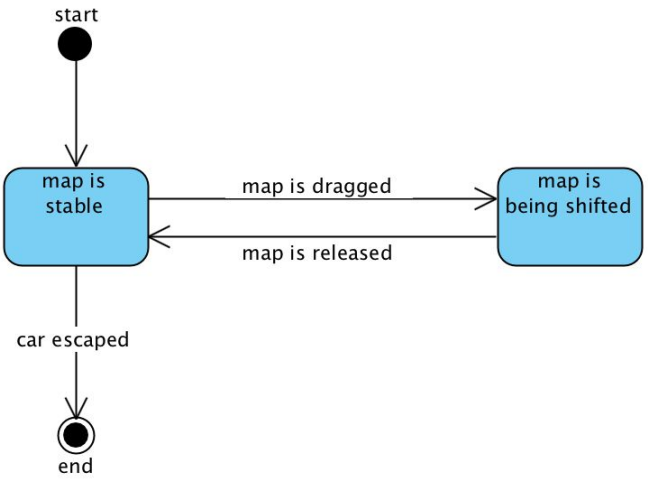


*Figure 9. State diagram for the status of the Game Map*

Even though the game map is stable for the single player game, since the map can be moved in multiplayer mode the stability of the map can be interrupted. In the game initially the game map is stable, so the start state directs to "Map is stable" state. For the multiplayer mode, with certain moves defined in the movement card

drawn (explained in 5.2.1, with Scenario #2 and #3). If the player does a move resulting the map to be dragged, the current state becomes “map is being dragged”. By releasing the Game Map, the map is stabilized again which results with the transition to “map is stable” state. The graphical illustration is given in figure 9.

#### 5.2.3.4 State Diagram For Car(s)

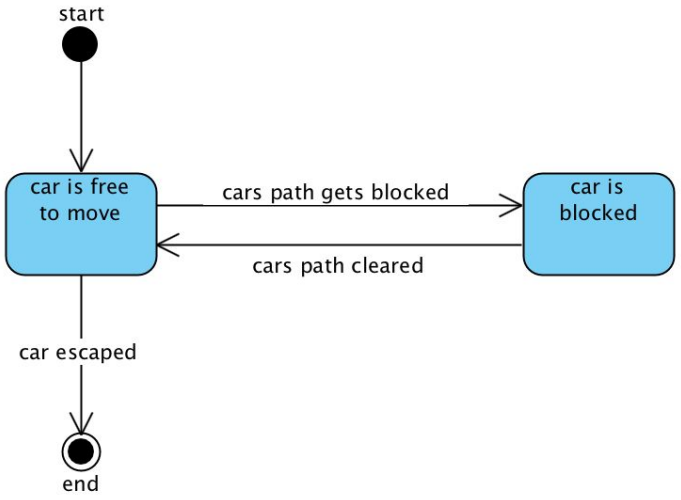


*Figure 10. State diagram showing the moving ability of the car*

Moving the cars is one of the most fundamental moves in the game. In this movement, the movement of the cars are restricted by the Game Map and the other cars existing in the map. As the player attempts to move the car, the attempt may not be successful. Initially (start state) the car is able to move in the direction, as the player makes an attempt to move the car, the movement boundary is checked. After this check if the path is blocked car cannot move (Car Is Blocked state). If the desired path is cleared from obstacles the car can move in that path (Car Is Free To Move state). This boundary check continues in each move consisting movement of a car until car escapes. The “end state” is reached when the escape car successfully escapes from the Game Map.

## 5.3 Object and class model

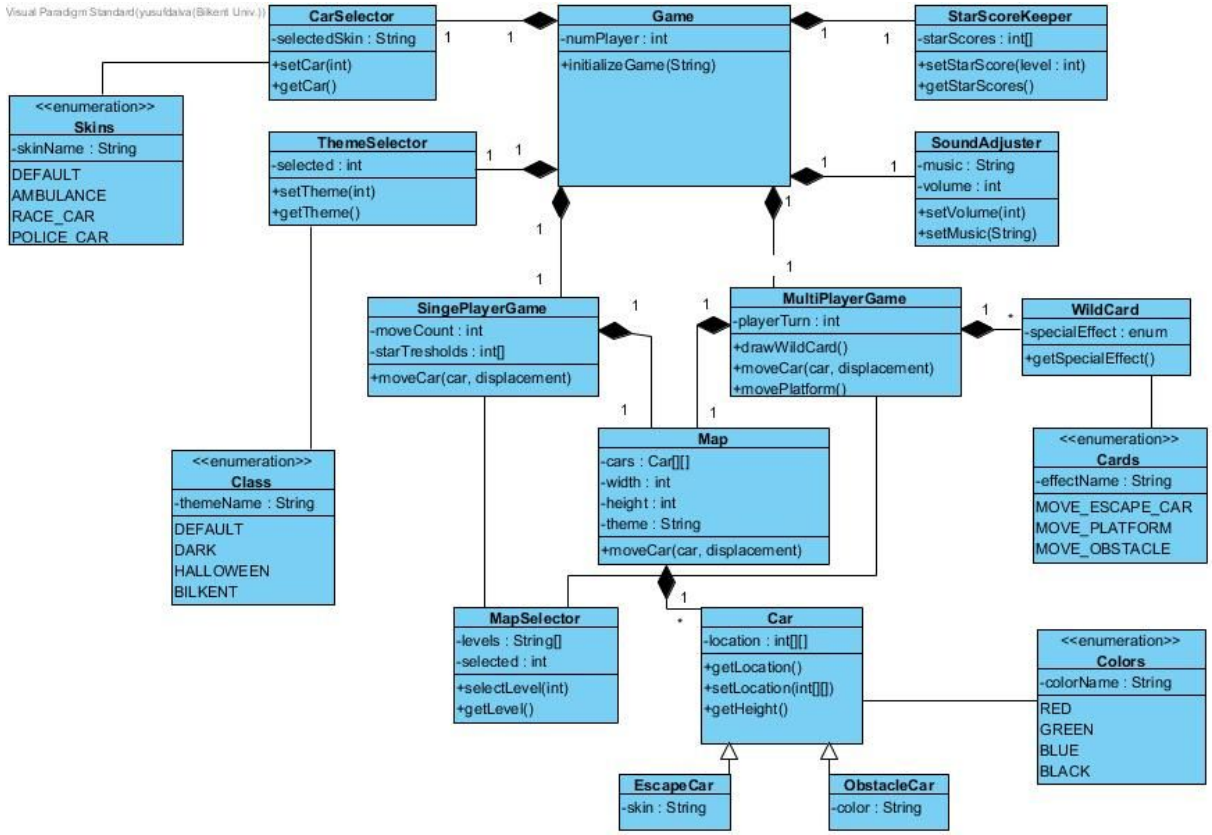


*Figure 11. High-Level Object & Class Model for Rush Hour Game*

The high-level class diagram of the Rush Hour Game is shown in Figure 11. The required methods and instances are illustrated in UML format for the functionality of the game. However this design is not reflecting the implementation of the game.

## 5.4 User interface - navigational paths and screen mock-ups

This section contains the user interface mockups which are going to be nearly identical when the implementation will be finalized. There are 7 mockups contained in this section which are stated in the subsections.

### 5.4.1 Main Menu Mockup


*Figure 12. User Interface Mockup for Main Menu of the game*

This is the main menu page that users are going to see immediately after opening the game. From this page users are able to access other parts of the game

### 5.4.2 Single-Player Mode Map Selection Mockup


*Figure 13. User Interface Mockup for Single Player Map Selection Screen Image*

This is the map selection menu that users are going to see after selecting the single-player mode. The difficulty level of the maps increases according to the number of level. Also levels are going to be unlocked when the previous level is passed.

### 5.4.3 Single-Player Mode Mockup


*Figure 14. User Interface Mockup for Single Player Mode screen Image*

This is an example gameplay from the Single Player Mode. Player is trying to make the car colored in red escape the map. Other than the map where the cars are located, the player can also see the best time for that level, number of moves made, time spent on the level, moves left to finish the level and time left to finish the level.

### 5.4.4 Multiplayer Mode Map Selection Mockup


*Figure 15. User Interface Mockup for Multiplayer Mode Map Selection Screen Image*

This is the map selection menu that users are going to see after selecting the multiplayer mode. All the maps are open in this mode. Players can choose any one of them as they like.

### 5.4.5 Multiplayer Mode Mockup


*Figure 16. User Interface Mockup for Multiplayer Mode Gameplay Screen Image*

This is an example gameplay from the Multiplayer Mode of the game where the escape cars are the yellow and white cars which have the same design pattern. The players are trying to escape the map first. The turn of which player is going to play is shown by the green arrow at the top side of the mockup. The users can play according to the wildcards dealt randomly (one by one). The turn count is shown at the top of the screen. The wildcard given to the player is also visible at the bottom of the screen.

### 5.4.6 Theme Selection Mockup

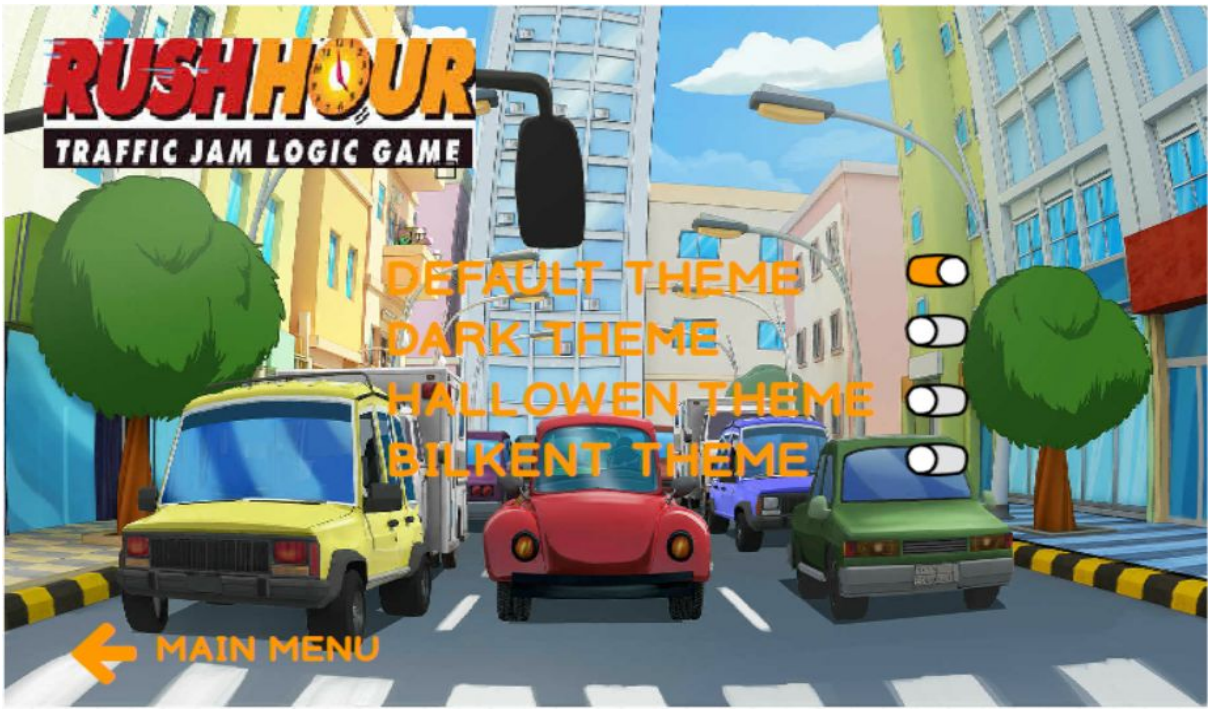


*Figure 17. User Interface Mockup for Theme Selection Screen Image*

This is the theme selection menu. There are different themes for the game in this menu and user can select one of the as they like. The map visuals change according to the theme.

### 5.4.7 Car Selection Mockup


*Figure 18. User Interface Mockup for Car Selection Screen Image*

This is the car selection menu. There are different options for the car skin in the game. They are not all available besides default car. Other cars are going to be opened according to succeeded levels.

## 6. Improvement Summary

First, we made the amendments requested in the feedback. We revised our diagrams according to the changes we made in our requirements (including functional and non-functional) and our design. We added state diagrams of the objects that we consider important for our game for further clarification.

One of the improvements to the game is removal of the difficulty system. We removed it because the new star system we implemented is much logical compared to the difficulty system. In a sense, the star system both aggregates the challenge aspect of the old system and brings other functionalities to the game. Some of this functionalities are unlocking car skins and new map themes depending on the total stars of the player. The other functionality is tracking the player's success rate over many levels and displaying these success rates to the player in a more concise way.

Another improvement to the game is the implementation of the path finder algorithm (A*). With this improvement, the player will get hints about the correct solution of the map if he/she gets stuck. Therefore, the player will not get bored from getting stuck and will keep playing since it get easier with every hint.

Lastly, we added a PvE option for the multiplayer mode. This change will allow the players to play the multiplayer version of the game even when there are no other players around. When this option is selected, an AI powered bot enemy will compete against the human player.

## 7. Glossary & References

“Rush Hour®.” Thinkfun, www.thinkfun.com/products/rush-hour/.
“Rush Hour® Shift.” Thinkfun, www.thinkfun.com/products/rush-hour-shift/.