BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS319
Object Oriented Software Engineering

Requirement Analysis Report
Project "Rush Hour"

Group 2.C Not So Oriented

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

1. Introduction

Implementation of Rush Hour was divided into several segments as our architectural style was Model View Controller (MVC). Each group member had different tasks under these three subsystems and because of the flexibility that MVC has provided us, we had the chance to work at the same time on the implementation. Therefore, it did not take a long time for us to start combining different parts of the game. The most problematic task was the connection between Controller and View subsystems. However, we have achieved an efficient way to maintain such a connection by making some design changes throughout the implementation process of first iteration.

1.1. Current Status of the Project

Game Algorithm

In Rush Hour's implementation process, our priority was to develop the Model and Controller subsystems first as they would be constructing the core game mechanism which will be manipulating the game data through the user's interaction with the View subsystem. Therefore, we need to say that we have done significantly more work on the backhand development of Rush Hour than the parts that will interact with the user. Our Controller classes are almost ready to use, although we have some minor modifications and additions to do. We have implemented the core features that are necessary to play a normal level. Therefore, currently a user can play all available levels as well as restarting a desired level.

Game Entities

We are done with all the game objects that are being interacted with the user. The only additions left are the LevelInformation and Player classes which handle the data for different players.

User Interface

On the View side of Rush Hour, we have primitive panels that the user can interact and access to different parts of the game. Currently, the game panel is displaying a level with different colored blocks which represent the different types of vehicles. Additionally our sound system is almost ready to use which basically needs a few minor modification. However, we should do more work on improving the graphical design of the view subsystems in the later process.

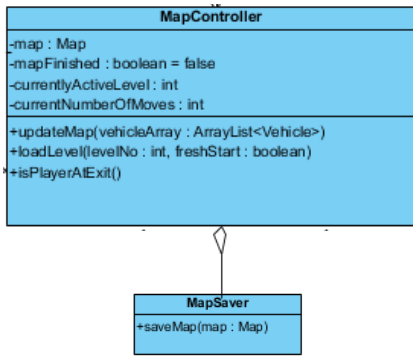
Later Development Process

We are going to start the implementation of special features that we planned to add to the original Rush Hour experience such as advanced levels that have a time and move limit. Additionally, we will start the implementation of the interchangeable theme structure for the user interface. Themes can be changed by earning stars in the game so another important implementation step for us is to construct the Reward system. Finally, our latest task is to make the game available for more than one user in the same device. Therefore, we need to develop the progress saving system for different users.

2. Design changes

We had to make some design changes throughout the implementation process so that we can find the easiest and the most efficient way to implement our core gameplay mechanism.

1. MapController and MapSaver classes are added so that the model Map class is no longer handling any operations that is done on the map which lead a direct connection between the Model and the View subsystems. (It should be done by a controller so that the coupling between the subsystems are decreased)



2. All controller classes are now extending a Manager class which has a “start” and an “update” method in order to be able to identify them with similar tasks which is to update the game data in GameEngine class. (Composite Design Pattern)

3. During the implementation process we have realized that creating objects of some controller classes more than once was not a good approach. Therefore to increase the maintainability of the game, we decided to apply the Singleton

Design Pattern to our Manager and Controller classes which reside in Controller package. Our reasoning was, because there should only be one manager or a controller that is actively controllering an aspect of the game.

4. In the very beginning of the implementation stage we were planning to handle the inputs with ActionListeners that are used in every controller class, however in the later process, to increase simplicity and writability we decided to make an Input Class which is being accessed by our Controller classes and return boolean values for the keys being pressed.

3. Lessons learnt

- We have learned that the most important aspect of both the analysis and the design steps is sticking to the prepared plan. Each group member must stick to the analysis and design steps while implementing the game and should work synchronously and finish their work on time so that there will be enough time to discuss about the problems that should be solved in order to move on with the next tasks.
- In terms of the implementation, we learned that the design should be made in the most efficient way to adopt possible changes. Because if it is not, it takes too much time and effort to add new features and to fix the encountered problems. Additionally, we realized that developers should ignore the details at the first stage which can create a lot of difficulty without maintaining a working system.
- Using design patterns is really beneficial in maintaining a powerful and easily editable system. It helps us and future developers to work in well defined development environment and also helps the software to have a well suited architecture that aids to further modifications.
- Another important aspect is the communication between the team members. It is really crucial to maintain a powerful communication infrastructure because great communication is the key for solving problems and making progress. Each member should be able to attend and express themselves clearly in regularly conducted team meetings.

4. User's Guide

4.1. 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". The player has to move vehicles on the map forward or backwards out of the way for the player car to the exit.

4.2. Installation

4.2.1. System Requirements

- You need to have the Standard JAVA Runtime Environment installed on your computer.
  Download Link:
  https://www.oracle.com/technetwork/java/javase/install-windows-64-142952.html
- Pentium 3 CPU or higher.
- 128 MB of Ram or higher.
- Screen resolution: 800 * 600.

4.2.2. Installing the Game

Unzip the RushHour.zip to your desired directory on your computer and execute the RushHour.bat to execute the game.

4.3. Playing the Game

The Objective: Slide the player car through the exit to freedom.

Set Up: Either open the last unlocked level from the main menu, or select a level to play from the level menu.

To Play: Slide the blocking cars and trucks in their lanes -up and down, left and right- until the path is clear for the red car to escape. Vehicles can only slide forward and backwards, not sideways.

If You're Stuck: Just restart the level and start over.

4.4. Game Menus

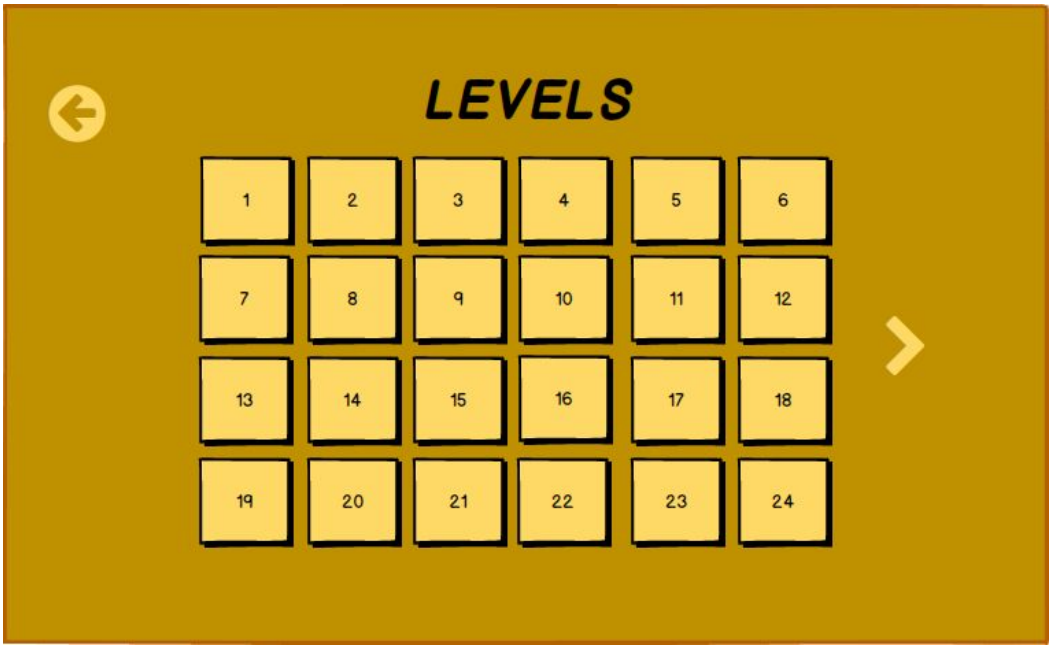
The game opens with the main menu which gives you the main options like: settings, change player, play last level and see all levels.

4.4.1. Playing the Last Unlocked Level

In order to play the last unlocked level, press the “Play” button in the main menu. This will open the map for the last unlocked level for the selected player and start the game.

4.4.2. Playing a Specific Level

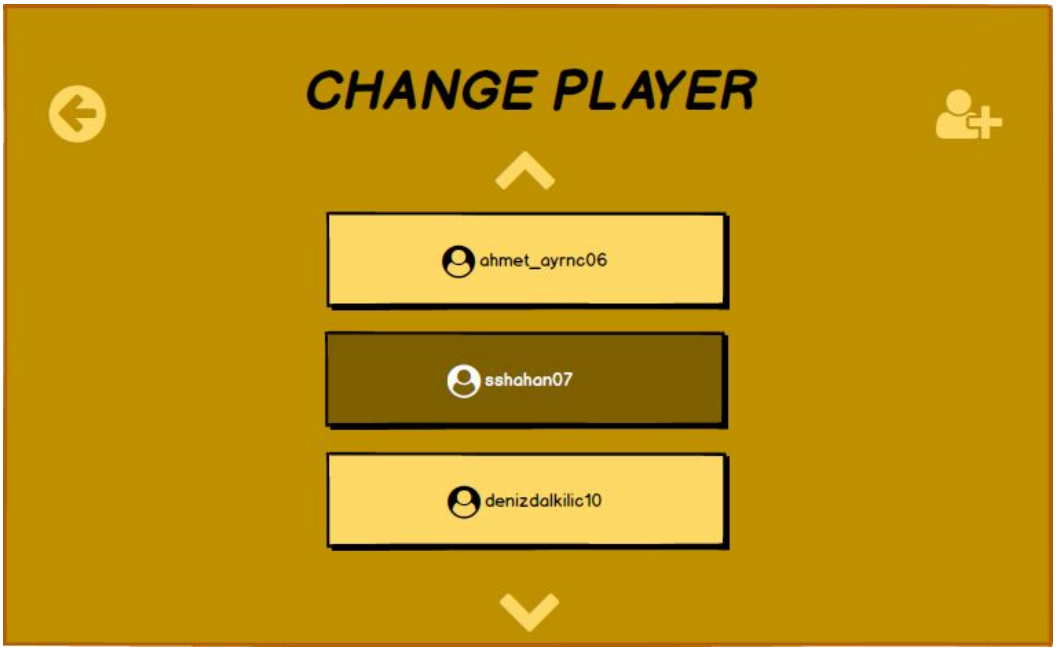
In order to play a specific level, press the “Levels” button in the main menu. After pressing the “Levels” button, Levels Menu will open.



In the Levels Menu each level is represented by its level number. You can press any level, which is not locked, to open it and play.
To go back to the Main Menu, press the “Back” button.

4.4.3. Change Player

In order to change, add, delete a player, press the “Change Player” button in the Main Menu. After Pressing the “Change Player” button, Change Player Menu will open.



In the Change Player Menu, each player will be shown in a list. To Change the currently active player, simply click on the name of the wanted player. To add a new Player, press the “Add Player” icon.
To go back to the Main Menu, press the “Back” button.

4.4.4. Change Settings

In order to change the setting for the currently active player, press the “Settings” button in the Main Menu. After pressing the “Settings” button, Settings menu will open.

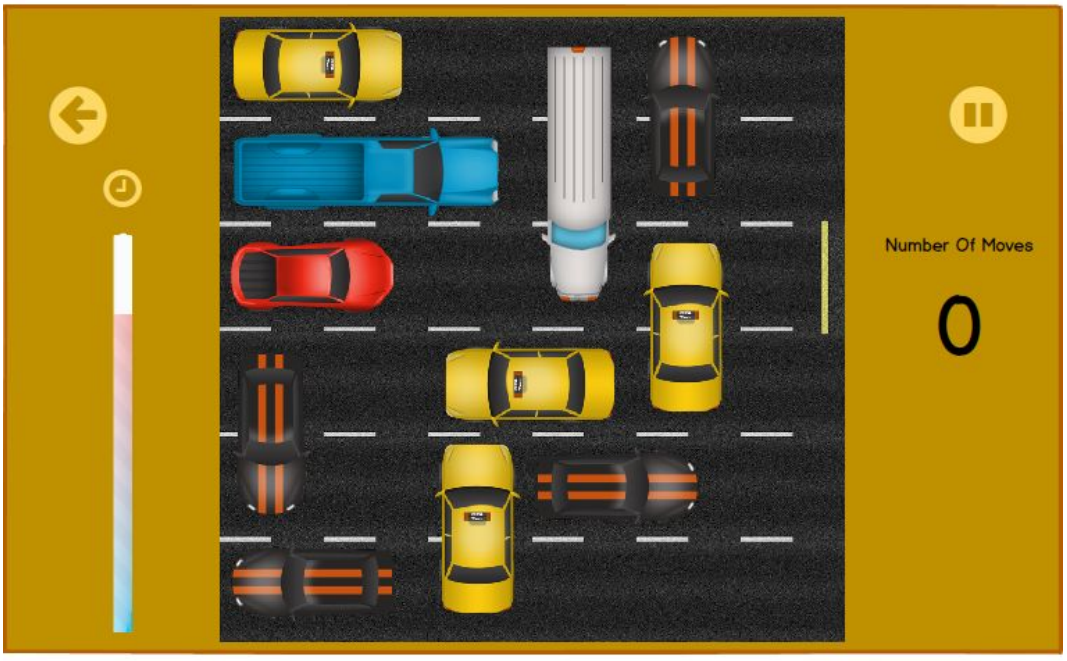
To change the volume of the sounds inside the game, press the appropriate volume icon for your desired volume settings.
To change the theme for the game, press the button with the name of the wanted theme.
To go back to the Main Menu, press the “Back” button.

4.4.5. Get Help

In order to see instructions for the game, press the “Information” icon in the Main Menu. After pressing the “Information” icon, How to Play menu will open with visual and textual instruction on how to play the game.

4.5. Game Screen

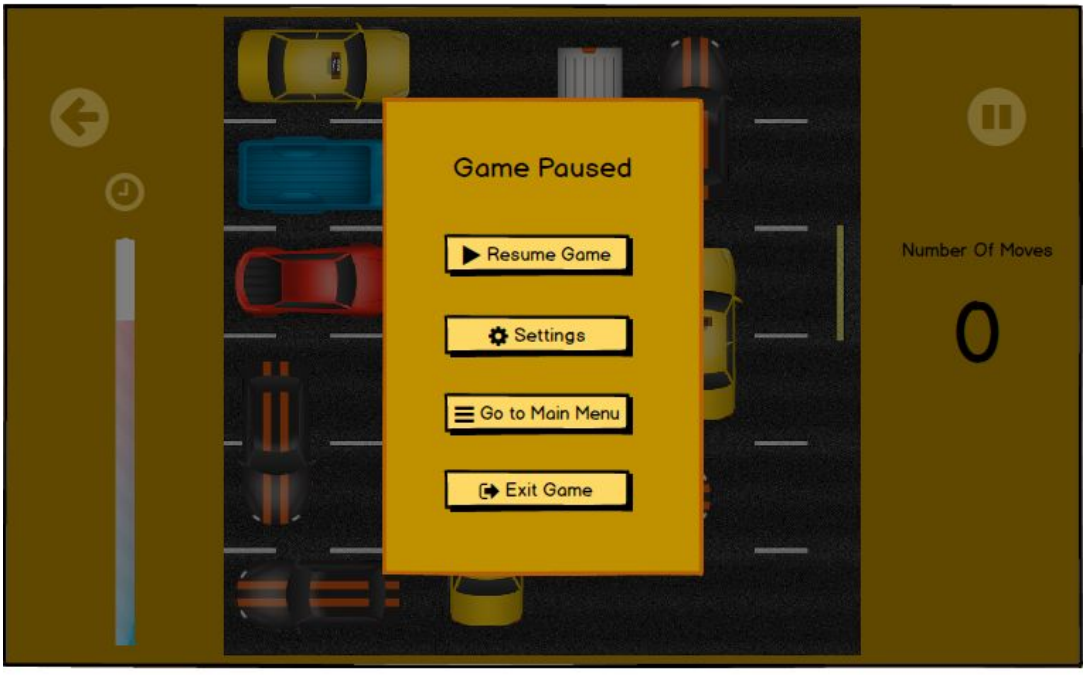
When the level starts, you will see the game screen, which shows the map an the vehicles.



You can click on any vehicle to select the vehicle. Selected vehicle will be highlighted. By pressing the arrow keys or the WASD keys, you can move the selected vehicle forwards or backwards.
In special levels where there is time restriction, The timer bar on the left will decrease with time and you have to move the player car to the exit before the timer expires.

4.5.1. Pausing the Game

To pause the game, press the “Pause” icon on the game screen. Upon pressing the “Pause” icon, the game will be paused and the pause screen will open.



In the pause menu, you can, resume the game, open the setting, go back to the main menu or exit the game by pressing the respective buttons.

4.6. Game Elements

4.6.1. Car

The car is a 2x1 blocking vehicle that the player can slide backwards or forwards to clear the path for the player car.

4.6.2. Truck

The truck is a 3x1 blocking vehicle that the player can slide backwards or forwards to clear the path for the player car.

4.6.3. Player Car

The player car is a special car that the player can slide backwards or forwards. In order for the players to finish the level, they have to move this player car to the exit.