Bilkent University

Department of Computer Engineering

CS319 Term Project

Rush Hour Game

Final Report

Aldo Tali

Barış Can

Endi Merkuri

Hygerta Imeri

Sıla İnci

Instructor: Eray Tüzün

Assigned TA: Muhammed Çavuşoğlu

Final Report

December 23, 2018

This report is submitted (softcopy) to GitHub repository in partial fulfilment of the requirements of the Term Design Project of CS319 course.

Contents

Final Report

1. Introduction

In this report, the implementation process of our game will be discussed. The process was completed in two Iteration phases, the first of which was about the whole process of development (Requirements, Design, Implementation), whereas the second Iteration was mostly focused on the enhancements made after the first Iteration.

1.1. Implementation Process

The implementation process consisted of some internal milestones being the distribution of the workload to the group members, the definition of the responsibilities for each respective subtask assigned to the members and reporting of progress status for the work that was assigned for the first and second iteration. The latter was done in order to be able to dynamically assign tasks to more members in the case where they were running off the projected milestone and to be able to merge the work done into a single deliverable. We distributed the work into three main divisions of roles: UI and Models and Controllers. The models and controllers were initially structured and laid out by a team of two(Hygerta and Sıla) whereas the UI was assigned to a team of three (Aldo, Barış and Endi). Then, after the second Iteration and during the implementation phase, there were changes on the class assignments between group members and collaborative work for the denser classes that required more time than the others, meaning that the team of two worked on UI and the team of three worked on Game Logic as well. The design, the graphics, and the needed controller functionalities for the merging of the subtasks into one project were made by the whole team leading to a single functional deliverable. Our game was implemented on different operating systems (Ubuntu, Windows) and on different IDEs for Java such as Eclipse[1], IntelliJ [2] and a simple Vi editor.

Our meetings were taken on intervals as frequent as twice per week. This methodology of work was chosen to get better insights on the status progress and the project milestones. Apart from that, we kept a constructive discussion and project planning on various platforms which enabled us to keep one another aligned on parts where the subtasks could potentially be misinterpreted and diverge from the proposed design of the project. The platforms we used

were the following: for the documents and the files, we have used Google Drive and GitHub whereas for the milestone progress checking, subtask and role assignment we made use of Meistertask, an online task management tool. All this was separated into “intermediate works” being the files that were in progress and the “final works” being the ones ready to be merged. The intermediate works were uploaded to Drive in case we needed anything from each other’s parts. Especially on the last phases of iteration 1, when the graphics of the game were being developed, we depended on each others’ work since for example, a member would provide an icon that he/she created or edited to another teammate who would customize it and implement it on the project depending on the formats and dimensions. Github was used for the final version of files as it easily synchronized our work by pulling and pushing source codes and image folders. It was the main platform that helped in the merging of the project and providing for source code support between team members. Lastly, our environment of communication has been Whatsapp through a designated group conversation that helped with asking questions about ambiguities either on design or structure as well as keeping the group members informed and on track. We have also used Google Hangouts to further discuss the implementation of the extra features introduced in the second iteration. The rest of the communication was straightforward, meaning meetings were held when all the members were present.

2. Design Changes

This section will represent the updated diagrams.

2.1. Changes

We did not make many changes in our design but we completed our features and added new features in the game like shrink car or colorblind option. In the first iteration, maps or skins were hard-coded and in this one we are taking them from .csv and text files.

2.2. Improvements

In the second iteration of the design report, we decided to make several changes and improved several functions in our game.

- Functions that were mentioned in the design report were implemented. The proposed “change exit” feature was removed. The added features include:
    - a “blow-up car” button that effaces one car which is chosen randomly from all of the existent cars in the map (except player’s car)
    - a “rotate car” button that rotates one car (NOT the player car) clockwise or counterclockwise (this feature is inactive whenever no rotation is possible in the map)
    - a “shrink” button that makes a car(chosen randomly from all of the existent cars in the map (except player’s car)) one block-size smaller,
    - a “timer button” that starts the timer in timer mode.
- Added hints option which was mentioned in both first iteration of final report and second iteration of the design report. Hints are optimized meaning they lead the user to complete the game in a minimum number of moves.
- Implemented DashboardPane, DashboardData, CreditsPane.
- Implemented Tutorial class which consists of a set of screenshots of a solved model game.
- Implemented StarManager class, winning the game now increases stars and using a game functionality such as hint, blowCar, shrinkCar decreases stars.
- Changed some of the icons, images, buttons, labels, background colors.
- Added a new colorblind theme to the theme set.
- Added more levels, and locked the consecutive levels.
- Implemented DataStorage class.
- Created map configurations for all of the three dimensions and added new car skins.
- Prepared graphical representations for the data displayed in the dashboard screen.
- Decided on the sound effects that will be used during the game, when the game is lost/won.

- Buttons in the PlayGame screen were categorized, added a label below them to indicate their function.
- Added animations to the buttons.
- Added a pop-up screen after winning or losing the game. The screen consists of a gif, gained stars for that level, the total number of moves made during the level, one button to exit to levels' screen and total time spent (if the player is on timer mode).
- Added an inner Move class to the engine that facilitates the movements of the cars in the game maps.
- Introduced a clear division, unlike in the iteration 1, between our DataStorage, Engine, Window Manager and GameManager classes.

3. Lessons Learned

The lessons we learned can be divided into two main groups: technical knowledge and other social and entrepreneurial skills.

Technological Knowledge and Academic Writing Skills:

We had not used JavaFX before for such big projects, but we thought we could improve the UI with JavaFX more so we learned how JavaFX worked and what its classes were (such as Stage, Application or MouseEvent). Furthermore, we learned how JavaFX uses a Cascade Style Sheet to enhance the graphical aspect of the application (the game in our case). We learned how to choose the most appropriate design pattern, how to use the MVC design pattern as the most appropriate architectural style for this project. The whole process of development was a learn-and-apply experience, we learned how to write scenarios, transform those scenarios to activity diagrams and then how to use course material to write UML diagrams such as sequence, state, activity, use case, and class diagram utilizing tools like Visual Paradigm. We also learned how important it is for all of the team members to write clear code. We learned that in big projects like this one (the first of this kind that we worked on), the code might be misunderstood if it is not well-commented or clearly written. While we had some report writing experience from a previous project we worked on as part of the CS102 course, this experience taught us to prepare the kind of more advanced reports better.

Soft Skills:

Regarding the soft skills, we were able to realize that keeping the group work structured and layered correctly means that the load balancing and the milestone provision for the deliverables become easier and it is a benefit on the long run. Furthermore, we learned how to hold formal meetings, how to document those meetings through writing meeting logs and how to get every bullet point of the agenda covered efficiently in a well-organized meeting. We learned how to separate the work to a group of people and how to help other group members complete their parts so that the work flows nicely and everything is ready on time. We learned how to deal with strict deadlines which made us develop our organizational skills even more. Moreover, we learned how a future engineer has to interact with the user and how to use strategies to get their support on the app (software/game) being developed. Before this course, we were only asked to do assignments that were solely coding or projects that would not require us to do any of the work we have done for this course.

4. User's Guide

In this section we will discuss our system requirements, installation process and how to use our game via screenshots.

4.1. System Requirements

Rush Hour can be played on all platforms. It will require JVM (Java Virtual Machine). Other than the case when user's machine uses JDK 11, this game will not require any external packages, but if the player uses JDK 11, JavaFX library will be needed since JDK 11 does not include JavaFX by default. Any computer which has these will be able to install and run the game.

4.2. Installation

Prerequisites:

- Oracle JDK
- Source Code

Note:

- The user must be able to run javac and jar from the command prompt or terminal.

Build Instructions:

Along with the source code of the project, a script called “build.bat” and a makefile is provided. In order to build the JAR file, the user must simply run build.bat on Windows and run “make jar” in Linux. In order to open the game, the user must run java -jar RushHour.jar from the terminal in Linux, but he or she can simply double-click on the JAR file to open it in Windows.

4.3. How to Use

4.3.1. How to Start

Fig. 1 Main Screen

This is the first screen of the game. Players can choose to play the game by pressing “PLAY”, go to the tutorial to learn how the game is played and how it is won by pressing “HOW-TO”, go to the “DASHBOARD” to see some charts (number of levels played for each dimension, ratio of won-lost levels, car skins, stars collected and levels completed), go to “SETTINGS” if they would like to adjust the volume of sounds, choose the timer mode or just to change the theme of the game and go to “CREDITS” to see the members of the developer team. The player can mute background sound by pressing the sound button at the right top corner of the screen.

4.3.2. Choosing Dimensions

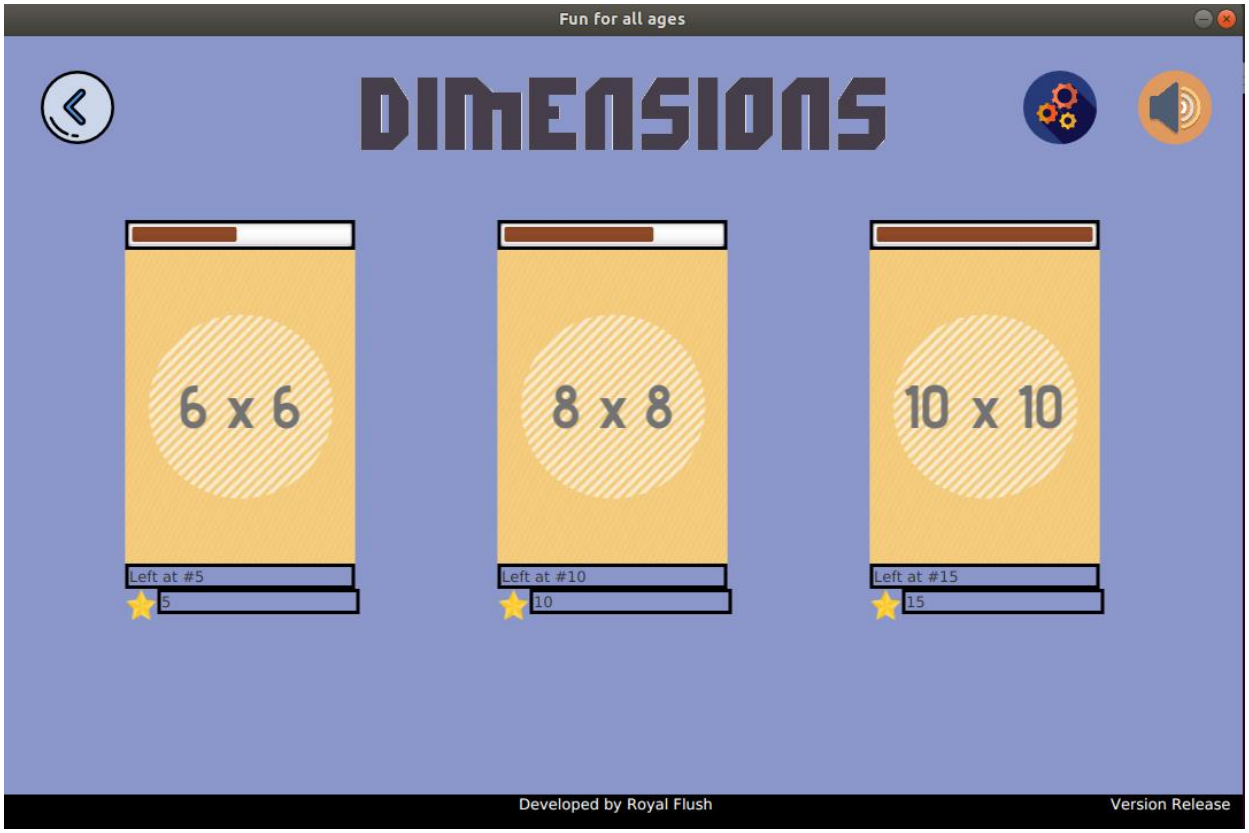

Fig. 2 Dimensions Screen

Players will encounter three different dimension choices for the game: 6X6, 8X8 and 10X10. They can also see their progression for each dimension and earned total stars for that particular dimension through the dimensions screen. The screen also shows where the player left off when they last played the game. The right top corner also includes two buttons which are for Settings and Sound button. Setting button sends the player to the settings screen and Sound button mutes/unmutes the sounds if the player desires it.

4.3.3. Choosing Levels

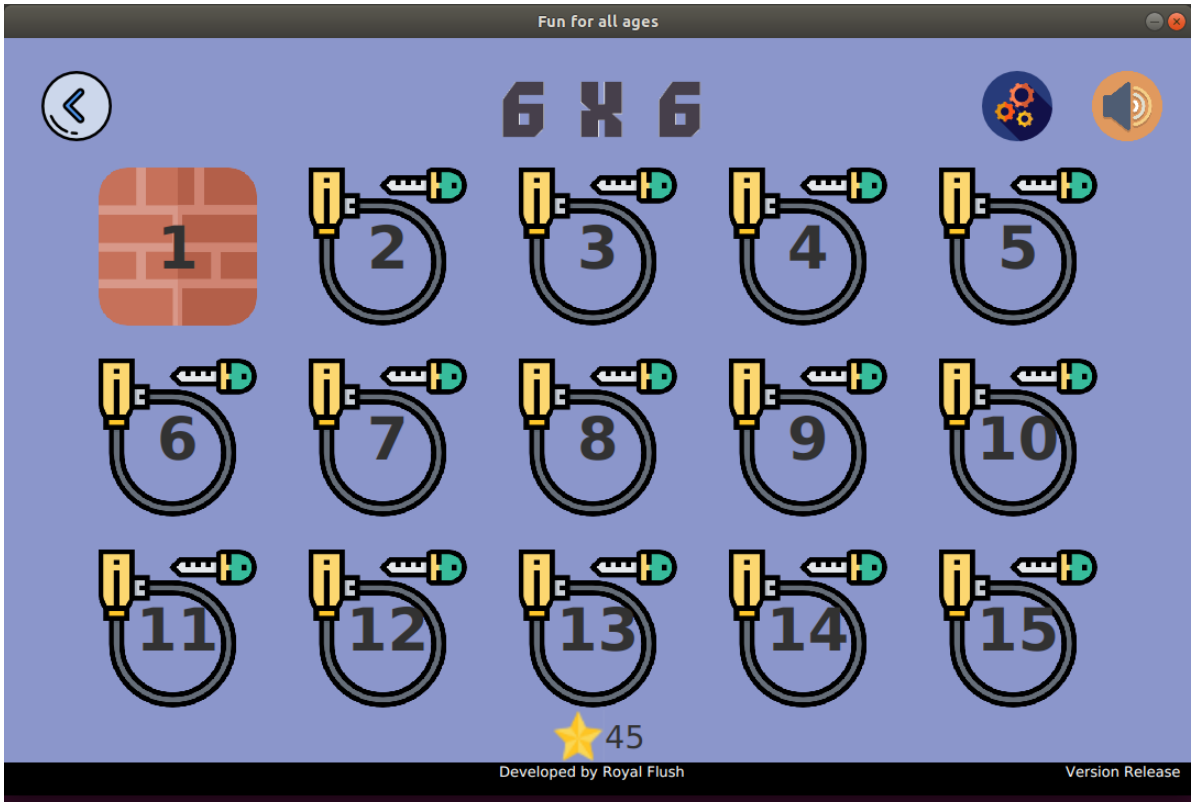

Fig. 3 Levels Screen

Fig. 4 Levels Screen after playing a game and unlocking another level

After choosing a dimension from the play screen, the player will be directed to Levels' screen. The player is be able to see the number stars they have gained for that level on each level button along with the total stars gained in that dimension which is displayed at the bottom of the "Levels" screen. After choosing the level, players will go straight to the "Play Screen". Some levels will be locked by default and will be unlocked after winning the previous level. The right top corner also includes two buttons: Settings button to take the player straight to the settings screen and Sound button to mute the sounds if the player desires.

4.3.4. Play Game

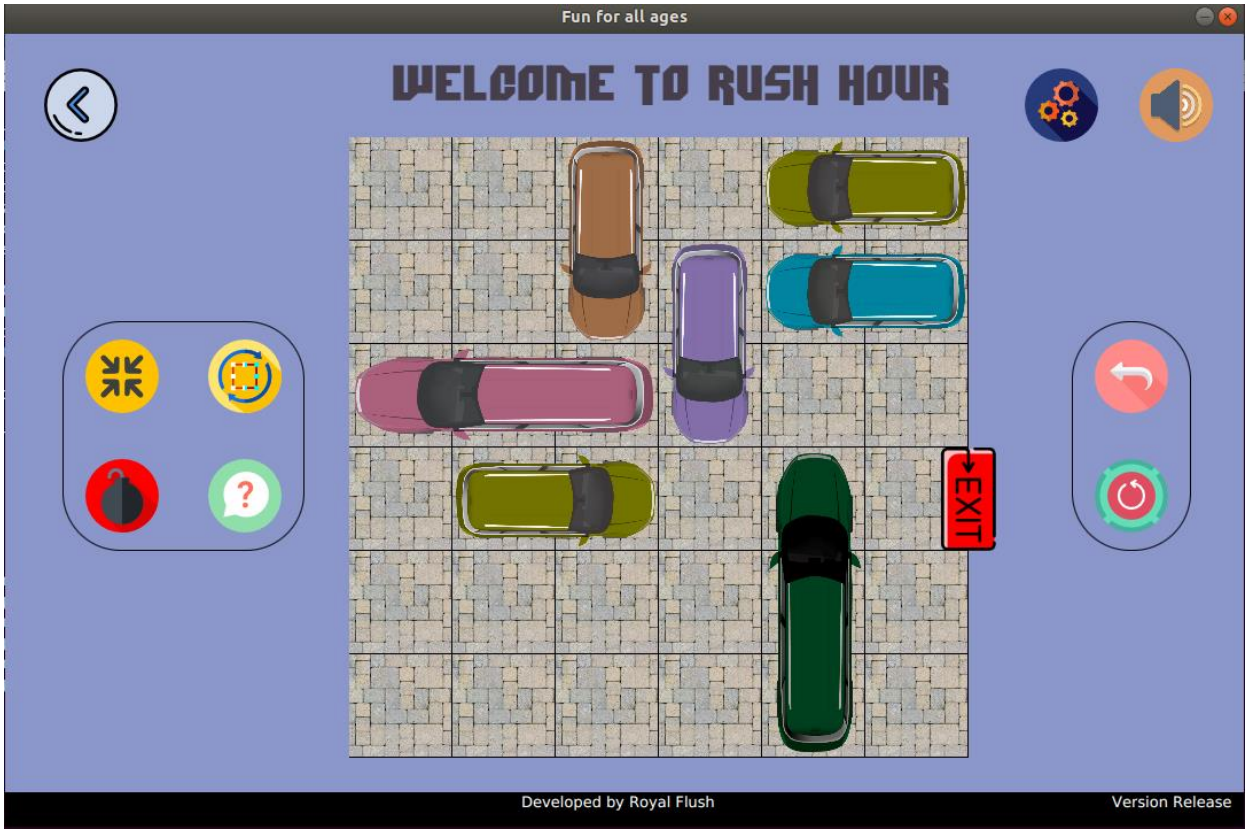

Fig. 5 Gameplay Screen for 6x6 dimension

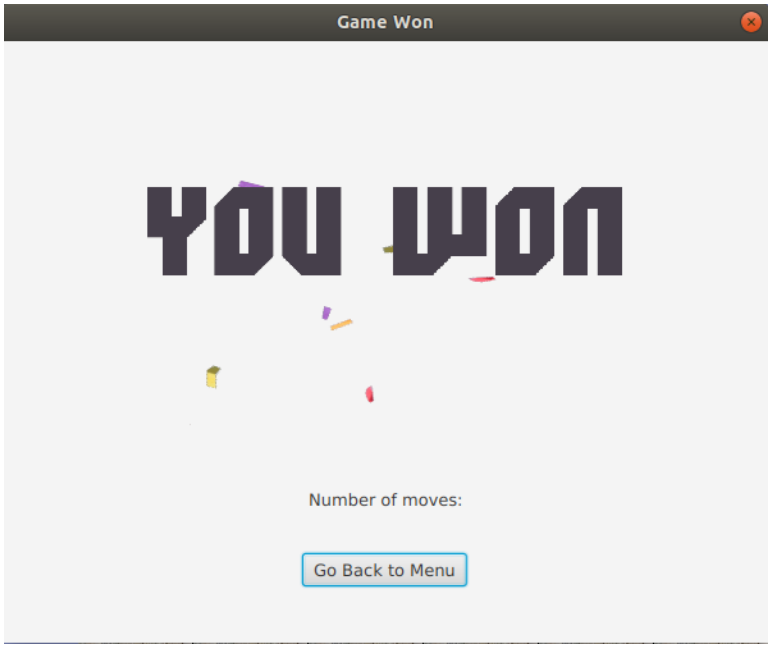

Fig. 6 Pop-up is shown when you win the level

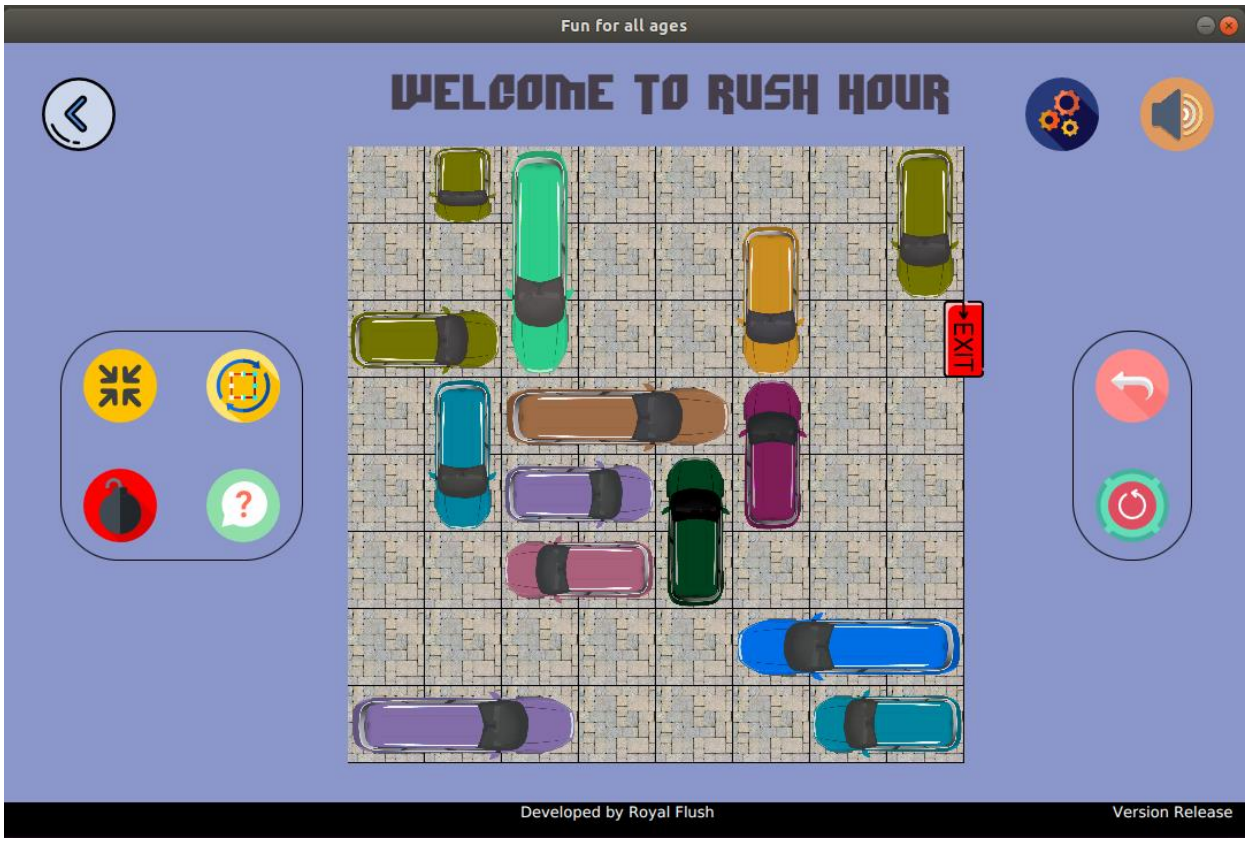

Fig. 7 Play Screen for 8x8 dimension

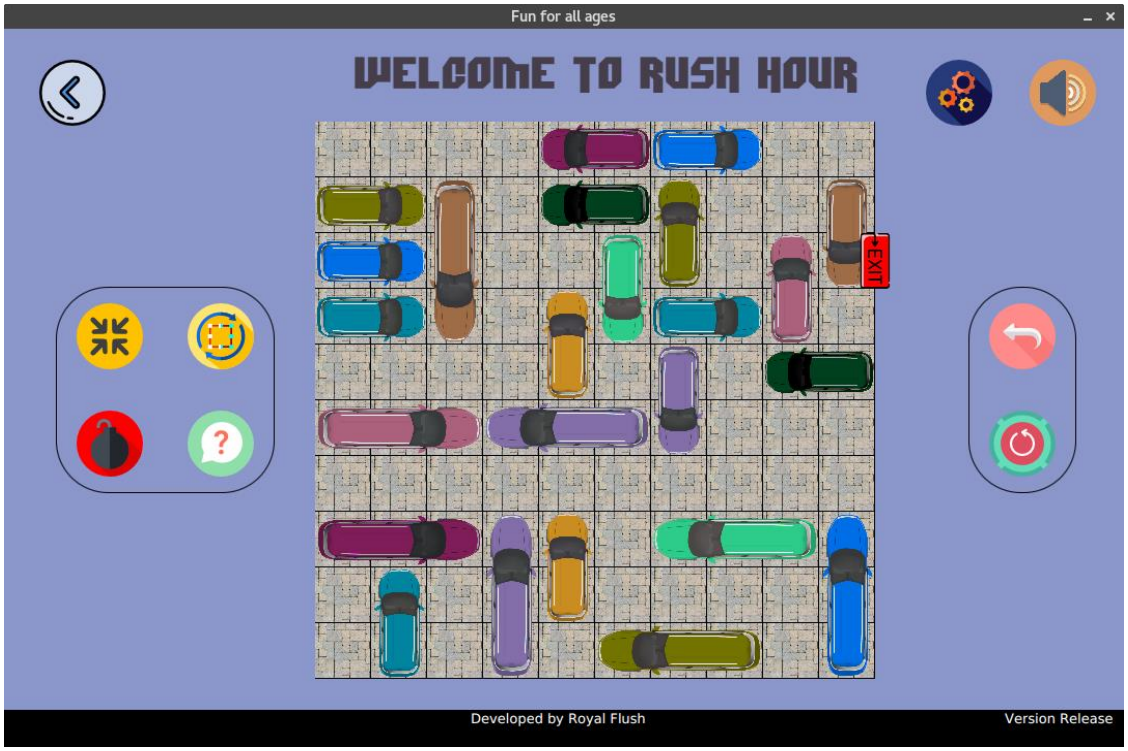

Fig. 8 Play Screen for 10x10 dimension

Fig. 9 Gameplay Screen when the Timer Mode is on

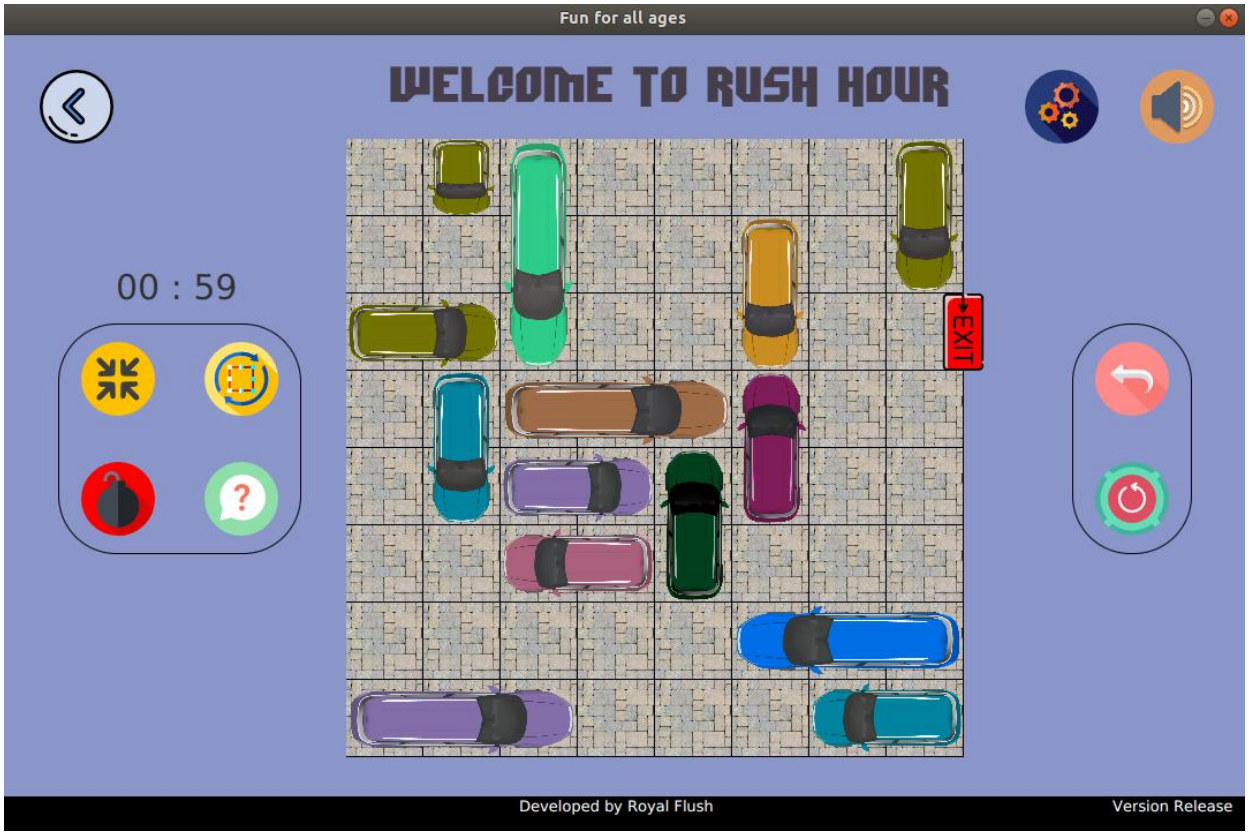

Fig. 10 Gameplay Screen when the timer is running

The player screen contains several buttons and the map configurations (the game to be played). Cars can be moved by dragging and dropping the mouse. Players can use: the BACK button to go back to levels screen, SETTINGS button to go to settings to set the timer mode On or Off, SOUND button to mute game sounds, UNDO button to undo their moves and RESET button to reset the game screen. Also, there will be four helpful buttons if the player is stuck. Players can choose to have one car shrunk (SHRINK CAR button), to have one car rotated (ROTATE CAR button), to have one car disappeared (BLOW UP button) and to get some hints (HINT button). These four buttons will require some stars to be enabled. There will be a different number of cars for each level. After players come to end block, the game will end, stars will be calculated and dimensions, levels screen, and the dashboard will be updated according to earned stars. After finishing the game, there will be a pop-up screen that shows a gif, number of moves, number of gained stars, total time if timer mode is on and a button to go back the levels screen. When timer mode is on, and the gameplay screen is shown,the player needs to press the square to start the timer and then the map configuration is shown for the game to start.

4.3.5. Settings

Fig. 11 Settings Screen

This is the settings screen. Players can change the volume of the sound effects, turn on or off the timer mode and change the background colour of the screen. There will be five color option, blue, green, lilac, pink and yellow. There will also be a colorblind option which changes icons and images.

4.3.6. Tutorial

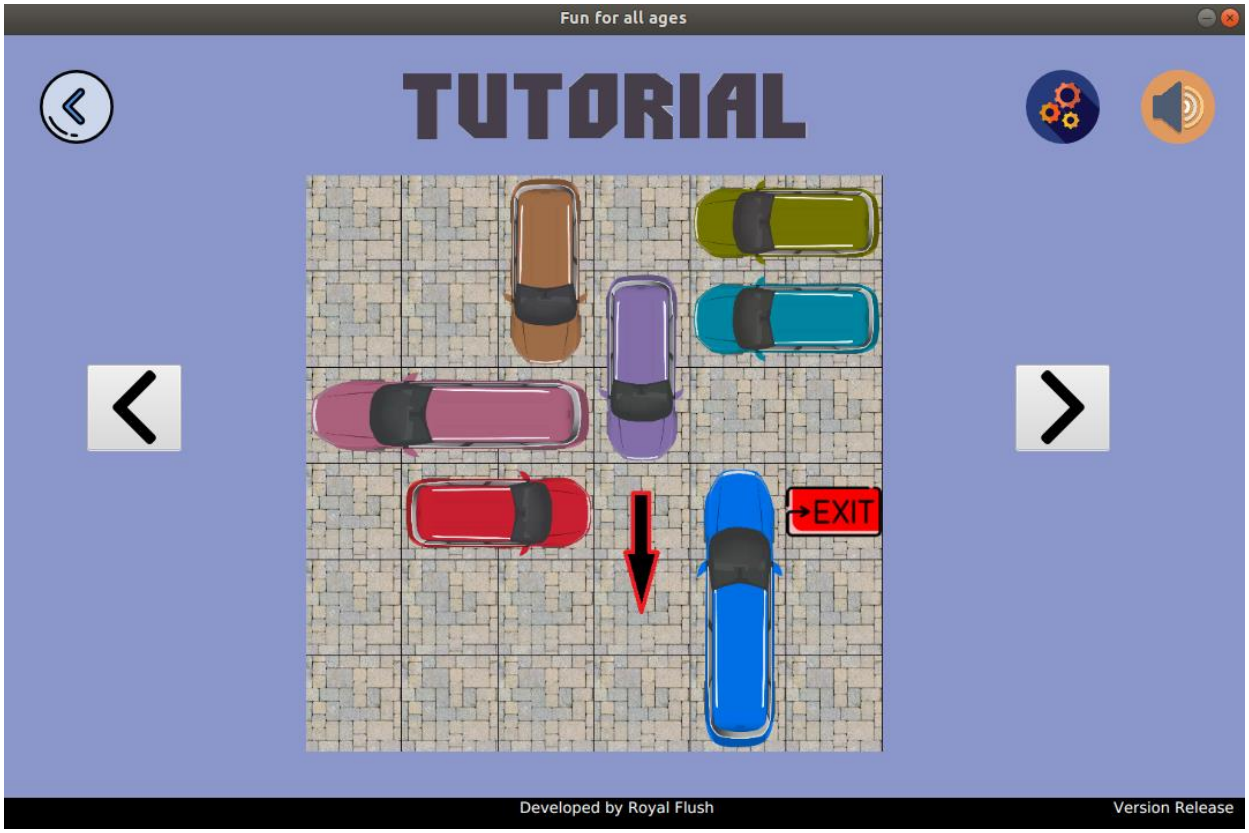

Fig.12 Tutorial Screen

This is the “How-To” screen. Players can see some explanatory screenshots of the game and these screenshots are represented in the form of a slideshow which can be played using the left and right arrows (as shown above). The right top corner also includes the Settings and Sound buttons to go to settings or mute game sounds if the player desires.

4.3.7. Dashboard

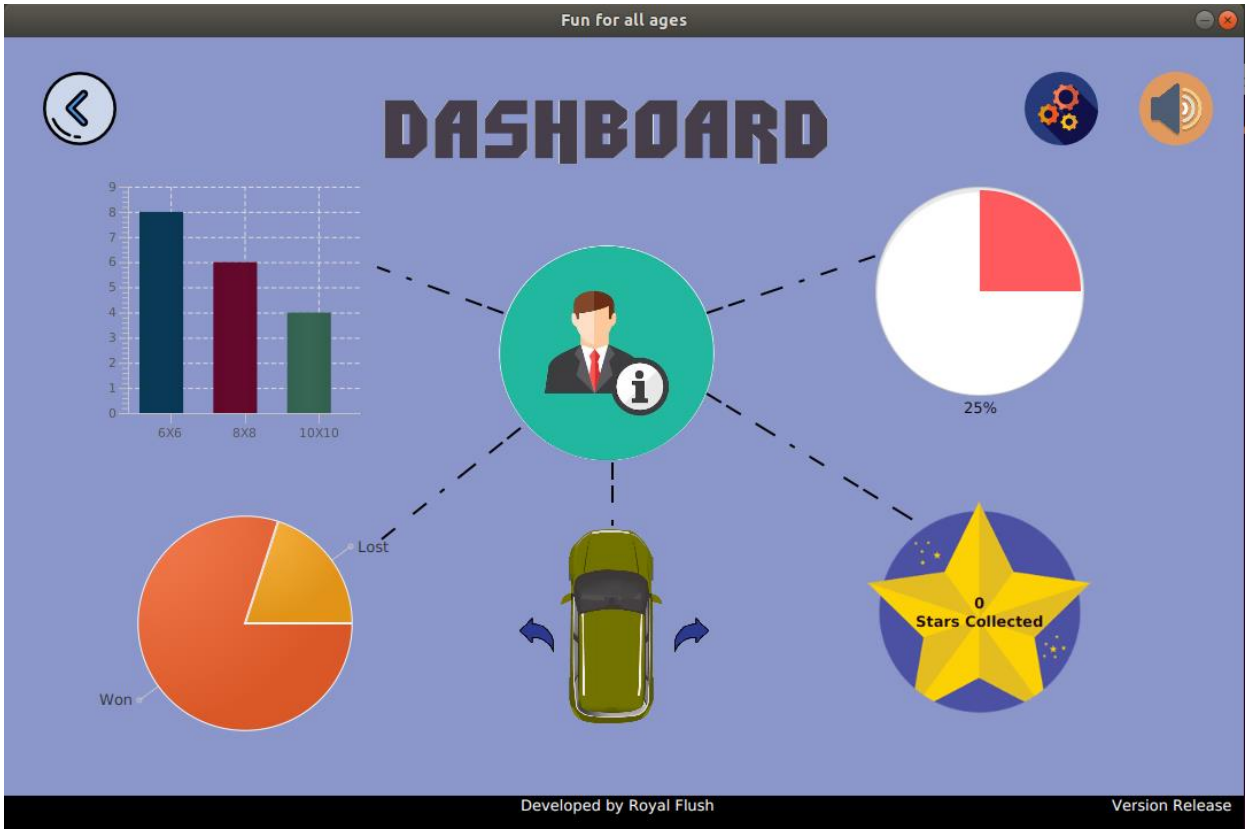

Fig. 13 Dashboard Screen

The dashboard will include several statistics a pie chart that shows win and loss ratio, a bar chart that shows unlocked levels for each dimension, a progress indicator that shows the completion of the game and a label that shows the total number of stars. In addition, players can buy some skins by using their stars and change their skins through the dashboard. Players can also change their profile image which is located in center of the dashboard. The right top corner also includes the Settings and Sound buttons to go to settings or mute game sounds if the player desires.

4.3.8. Credits

Fig. 14 Credits Screen

In this screen, the user is able to see and learn who developed this game. This screen also has the “MUTE” and “SETTINGS” button just as all screens.

5. Work Allocation

According to the report requirements, this section was supposed to be approximately one page but we believe that each of the group members did a great amount of work which was logged weekly and in a detailed manner in our GitHub repository.

All group members

Worked collaboratively on writing reports and deciding on most of the main parts (functionalities) of the game, as well as reviewed the reports according to the given feedback after the first iteration.

Aldo Tali

- Prepared MockUp (‘How-To’ screen and ‘Game/Play’ screen), Wrote the overview
- Wrote Functional Requirements and Low-level design part.
- Fixed the timer mode sequence diagram from the first iteration of the Analysis Report.
- Wrote why we chose a File System instead of a database for the storage structure.

- Designed the UI (Screen) classes which inherits main, dashboard, tutorial screens etc. with the UI manager and summarized their work on UI classes and further worked on the UI and prepared game screenshots which will be used in the final report
- Focused on the new features as well on the parts of the game play that have bugs and that will be changed on the final version of the game.
- Was assigned the generation of new possible features that would be represented into the the game design. All these were inserted into the “Checklist” logging file created the previous week.
- Updated the class diagram to include the newly proposed features both in functionality and the in the UI design. Wrote the missing textual descriptions of the attributes for each class.
- Decided on the package names and divisions on the class diagram.
- Fixed small bugs present on the first demo of the project. They also have to implement the newly added features of the game along with the Game Timer and the Star Manager for the scoring system.
- Worked on the Implementation of the game Functionalities and Game Logics for the final iteration along with Endi.
- Took Part in the Implementation of Map, Block, DataStorage, Play Game and GameManager.

Barış Can

- Did MockUp (‘Play’ screen), Write Nonfunctional Requirements
- Design the Data Storage class which holds the charts, maps, sound effects, tutorial and skins.
- Wrote the introduction on the Iteration 1 Analysis Report.
- Did Play Game sequence diagram, to make minor fixes that would reflect the intended changes on the game play.
- Worked on the explanations of the diagrams and the figures introduced in the first iteration to match with the newly introduced terminology of the second iteration.
- Updated the class diagram to include the newly proposed features both in functionality and the in the UI design. Wrote the missing textual descriptions of the attributes for each class.
- Worked on the Game Solver class with Sila and Hygerta. Determined the logistics of the algorithm that will solve any given map input in the game and that can display hint accordingly with other people. They also have to take case of the Sound Effects and the missing UI for each of the features that need to be implemented.
- Coded MainPane, Dimensions, How_To, Dashboard, DashboardData, Settings in UI.

Endi Merkuri

- Design of the Object and class model in Visual Paradigm and created the mock up for the Levels' screen.
- Copy use case descriptions and scenarios to the same table format
- Took the part of object diagram and part of sequence diagram.
- Was assigned with making the change on the object model to reflect the added attributes and methods for this report along with some higher level managing structures of the game.
- Had to work on the notations of the class diagram to match the dependency relation introduced on the component diagram by Hygerta.
- Dealt with the Access Matrix building for the Rush Hour game design.
- Designed the UI (Screen) classes which inherits main, dashboard, tutorial screens etc. with the UI manager.
- Focused on the new features as well on the parts of the game play that have bugs and that were changed on the final version of the game.
- Decided on the package names and divisions on the class diagram.
- Was assigned the fixing of the small bugs present on the first demo of the project.
- Helped in the implementation of the Blow Up Car, Shrink Car, and part of the Undo moves features.
- Took part in the implementation of the GameSolver class and the Hint option in the game.

Hygerta Imeri

- Prepared MockUps ('Dashboard' screen and 'Settings' screen)
- Merge and format the report and Checked scenarios and the use cases of the first Analysis Report.
- Worked on improving the component diagram to make use of the notations taught in the last week of classes and to cut down the no longer valid components (due to changes in the project design in this phase).
- Drew the Use case diagram and UML design the engine and the game solver classes with the addition of timer and score manager Reviewed the engine, timer and starManager class and completed the UML diagram for these classes also made component diagrams and write Subsystem Services part.
- Prepare presentation slides with Sila and Worked on visuals of the game

- Set up the logistics on Google Drive Docs for the team members to edit a “Checklist” Document that would be updated each time a missing part from the Design Report of the Iteration 1 would come across. This was kept as an internal logging structure of the “To do-s” in the team.
- Checked if all the requirements have been met. Gave the to be added feature design for the system. Reviewed the system design and system decomposition.
- Worked on some initial Game Solver class and algorithm (with Sila and Baris). Determined some of the logistics of the algorithm that will solve any given map input in the game and that can display hint accordingly. They also have to take case of the Sound Effects and the missing UI for each of the features that need to be implemented.
- Did the credits screen and overall graphic design with Sila.

Sıla İnci

- Did the MockUp for ‘MainPage’ screen.
- Was assigned to do the explanation of the activity diagram and part of the game solver class.
- Drew the Use case diagram.
- Was assigned the provision of the textual parts that were missing for the diagrams and figures in the first iteration of the report. She also had to update the use case diagram to have the proper changes that reflected the second iteration design of the game.
- Designed the engine and the game solver classes with the addition of timer and score manager with others.
- Reviewed the engine, timer and StarManager class and completed the UML diagram for these classes.
- Made component diagrams and wrote the Subsystem Services part in Design Reports.
- Prepare presentation slides with Hygerta.
- Worked on visuals of the game with Hygerta.
- Gave the to be added feature design for the system in design and final reports. Reviewed the system design and system decomposition.
- Were assigned the generation of new possible features that would be represented into the the game design.
- Worked on the Game Solver class with Hygerta and Baris. They have to determine the logistics of the algorithm that will solve any given map input in the game and that can display hint accordingly. They also have to take case of the Sound Effects and the missing UI for each of the features that need to be implemented.
- Did the credits and overall graphic design with Hygerta.
- Worked and helped Baris and Aldo with the development of StarManager.

6. References

[1] http://www.eclipse.org/

[2] https://www.jetbrains.com/idea/download/#section=windows