CS 319 - Object-Oriented Software Engineering

Final Report – II

Rush Hour

Gurup Şurup

Muhammet Said Demir
Ata Coşkun
Zeynep Nur Öztürk
Asuman Aydın
Tarık Emin Kaplan

Supervisor: Eray Tüzün
TA: Muhammed Çavuşoğlu

Table of Contents

1. Implementation Process

We started the implementation phase shortly after the first iteration of design reports. We have implemented our program in Java language in popular IDE, Eclipse; and anyone who would do some changes to the code would commit their code to the GitHub.

At first, we only did the program so that there was only one mode, which is the single player mode, and there wasn't any new and creative mechanics to the game other than facing some obstacles (we proposed some creative mechanisms in the demo by preparing some mock-ups though). That version was the one that we have presented in the first demo; and at that point our there were not many classes so our diagrams looked pretty simple as well. However, after that first demo, we started implementing the multiplayer mode to the game, which was way more complicated both in terms of coding and in terms of game mechanism. Multiplayer mode is played with cards unlike single player mode, as you might remember from our previous reports, but at first we have implemented it so that it was played similar to the single player game mode; meaning that there was no cards and it was simply dragging cars around like in single player mode, the thing that differed from the single player mode was that the movement operations were done so in turns and also the parts of board could also be moved. After that point the coding work slowed down by a lot for a few days due to the work load of other courses; but shortly after we have returned to work on the project, this time we did not go directly into the code but focused more on the design as well, because we could see that things were getting a little complicated. So, we have come up with a more complex, modified class diagram that would be the skeleton of our complete program and we have explained this improved version on our second iteration reports as well. You could refer to those reports for further details and see that our program is in accordance with object-oriented programming.

After that we have implemented those features, we have shown in our second iteration reports, namely, the multiplayer mode would be played with cards and our signature creative feature we have added to the game, which are portals. Additionally, we have also implemented more of the "vanilla" features to the game such as scores, time, changing themes and such.

1.1. System Requirements

The detailed hardware/software requirements are explained in section 2.2 of our second System Design report but briefly put, our program will require JVM (Java Virtual Machine) and Java SDK 8 to run on PC.

1.2. Installation Guide

- Check our GitHub page: https://github.com/AtaCoskunCs/Cs319_GroupSurup_RushHour

To check out the code and play:

- Download the latest ZIP file and extract files of the ZIP file that has been downloaded
- Open the Eclipse and use File/Open Projects from File System
- Navigate it to rushHour (not rushHour/src, cuz we have images and sound files etc.) and click to choose the file.
- Finally, run the RushHour class, it has the main method.

2. Changes and Improvements

2.1. Design Changes

Just by looking over our reports you can notice that lots of changes has been made throughout the life-cycle of this project. The first and definitely the most influential change we have made was actually a decision we have agreed on at the early stages of development, which was to implement the program in Java instead of Unity. We believe this important fundamental change was a good decision overall because even though we might have been able to present a more visually appealing product overall thanks to the Unity's tools, seeing how complex our program actually got over the time now makes us wonder whether if we could have produced a program as mechanically challenging as we have now in Java.

In addition to that fundamental change, as it is mentioned in the previous section, a quick look at the class diagrams we have shown in the first and second design reports shows how much change has been done overall

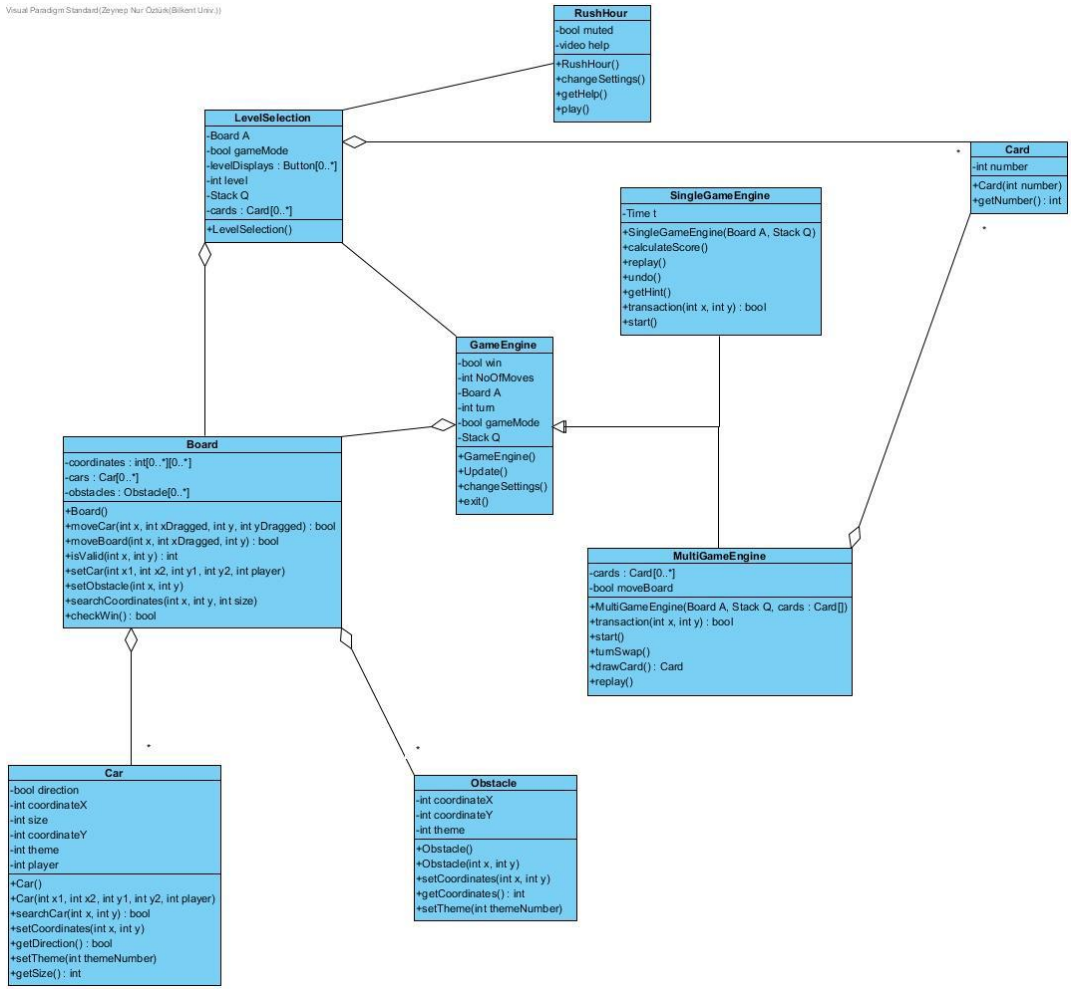


Figure 1: First Iteration Design Report Class Diagram

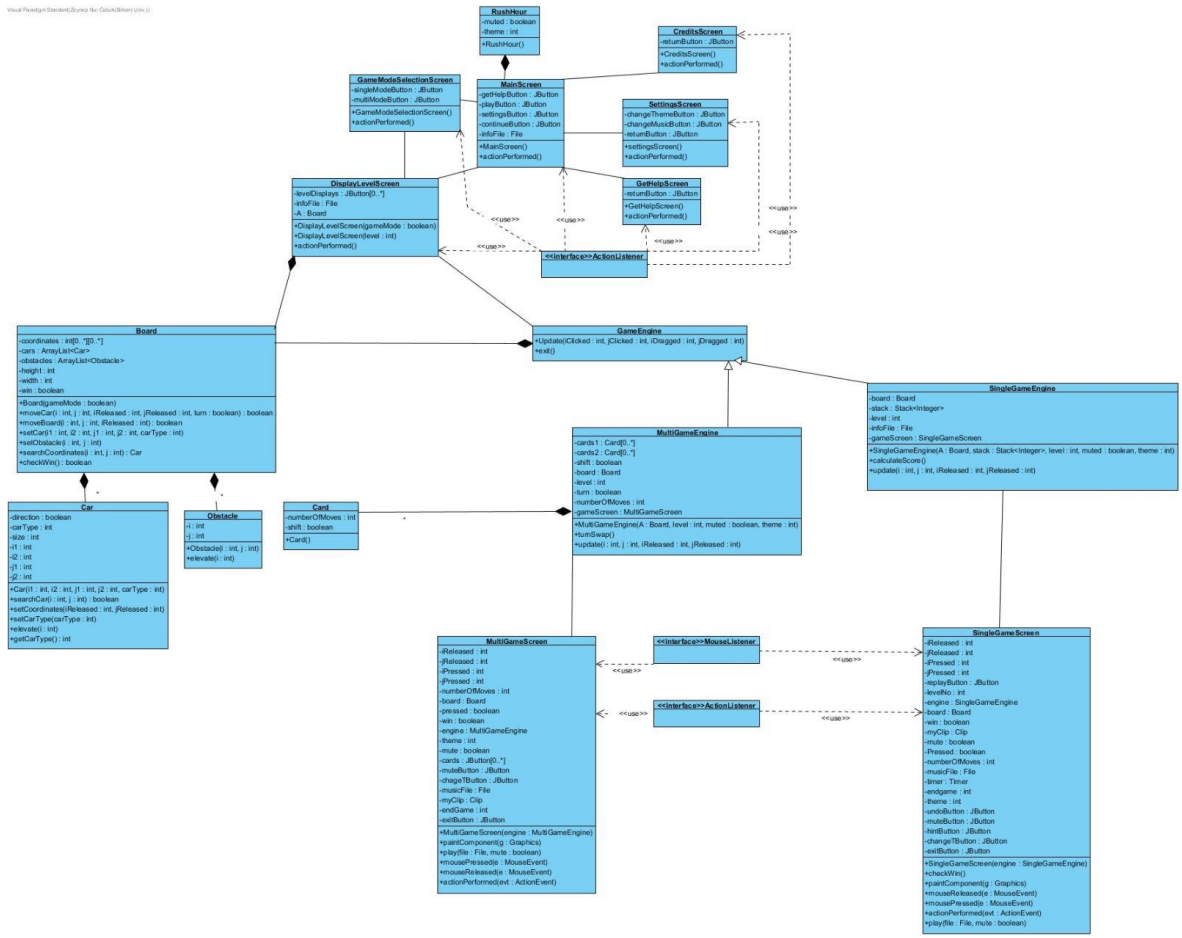


Figure 2: Second Iteration Design Report Class Diagram

The diagrams are displayed as figures to showcase the increased complexity of the overall structure, they are not supposed to be explained in detail here, if you would like to learn more about them, please refer to their respective design reports.

You could also see a very significant change in subsystem decompositions shown in each report, at the first iteration, the diagram given was barely a subsystem decomposition diagram to begin with, but at the second iteration we think we have settled it well

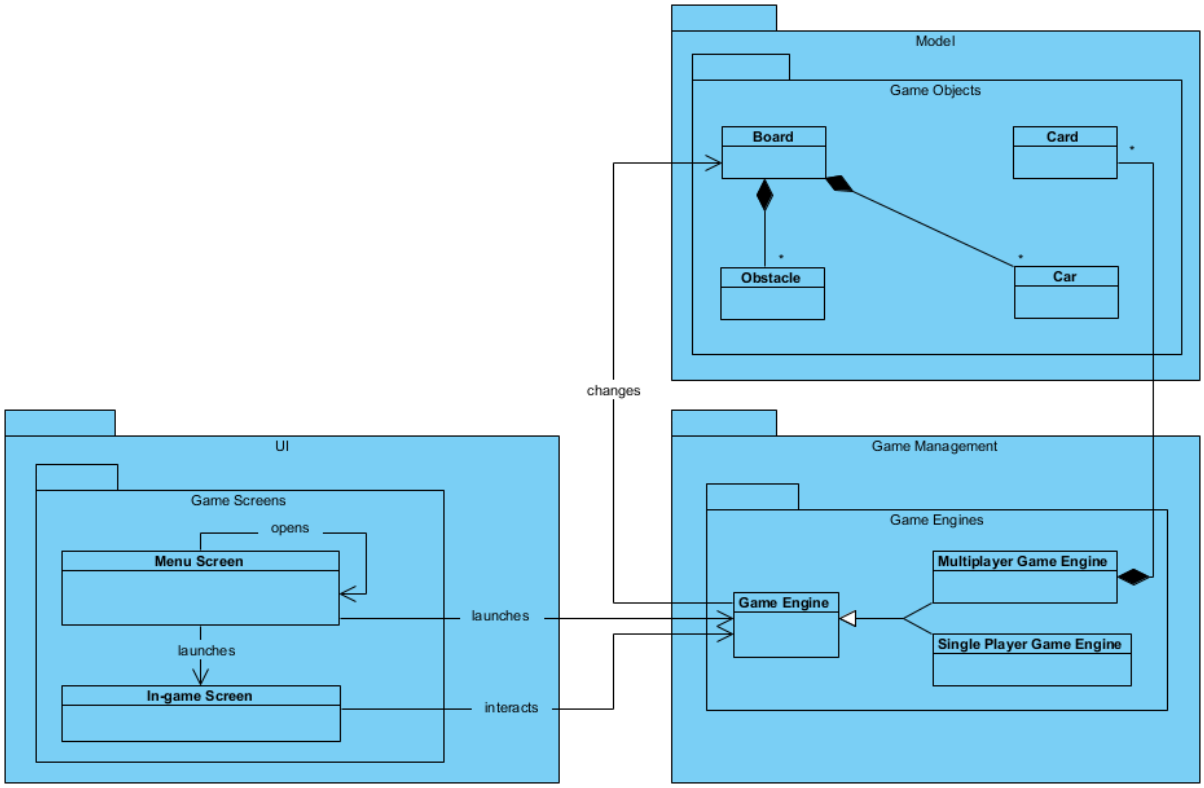
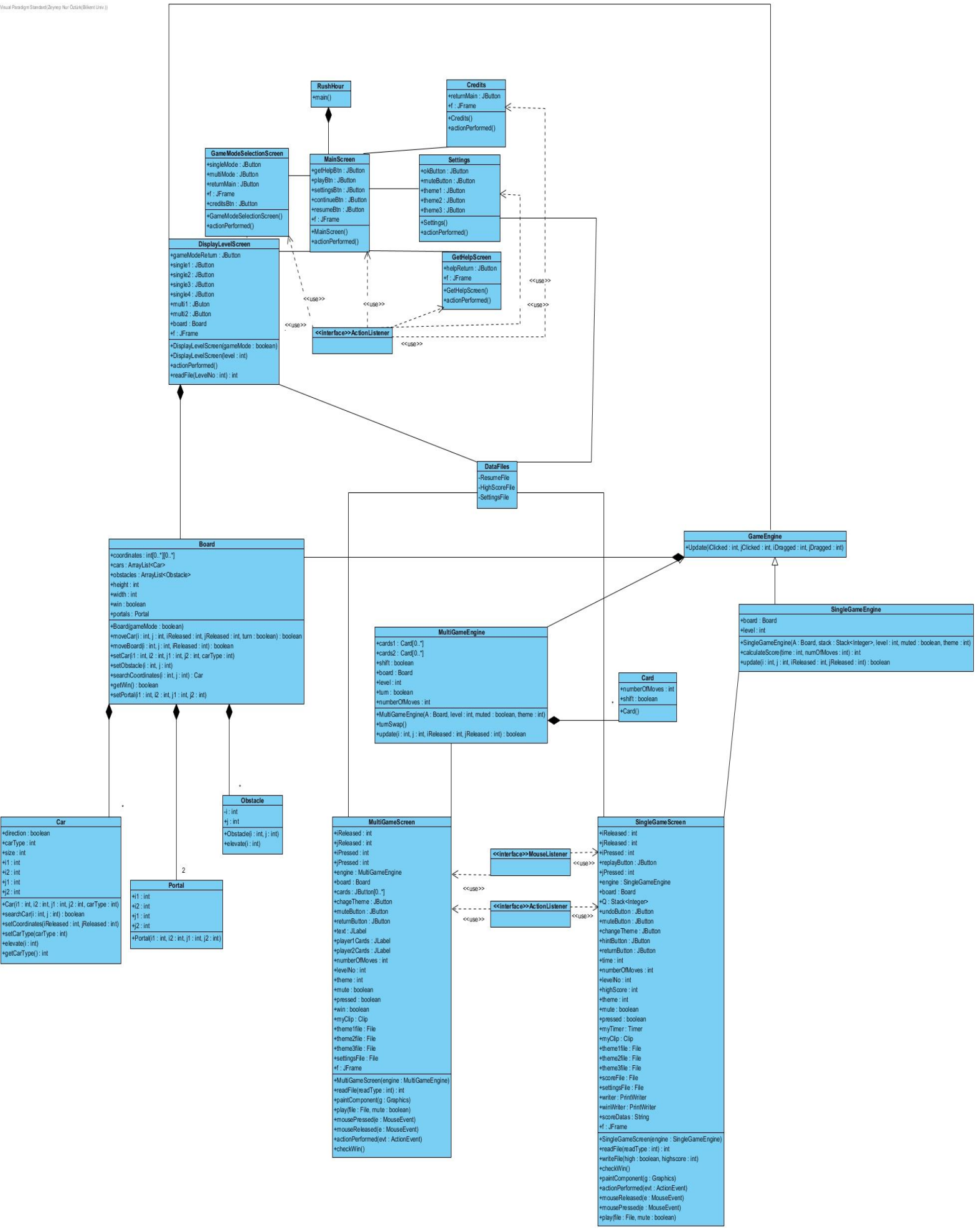
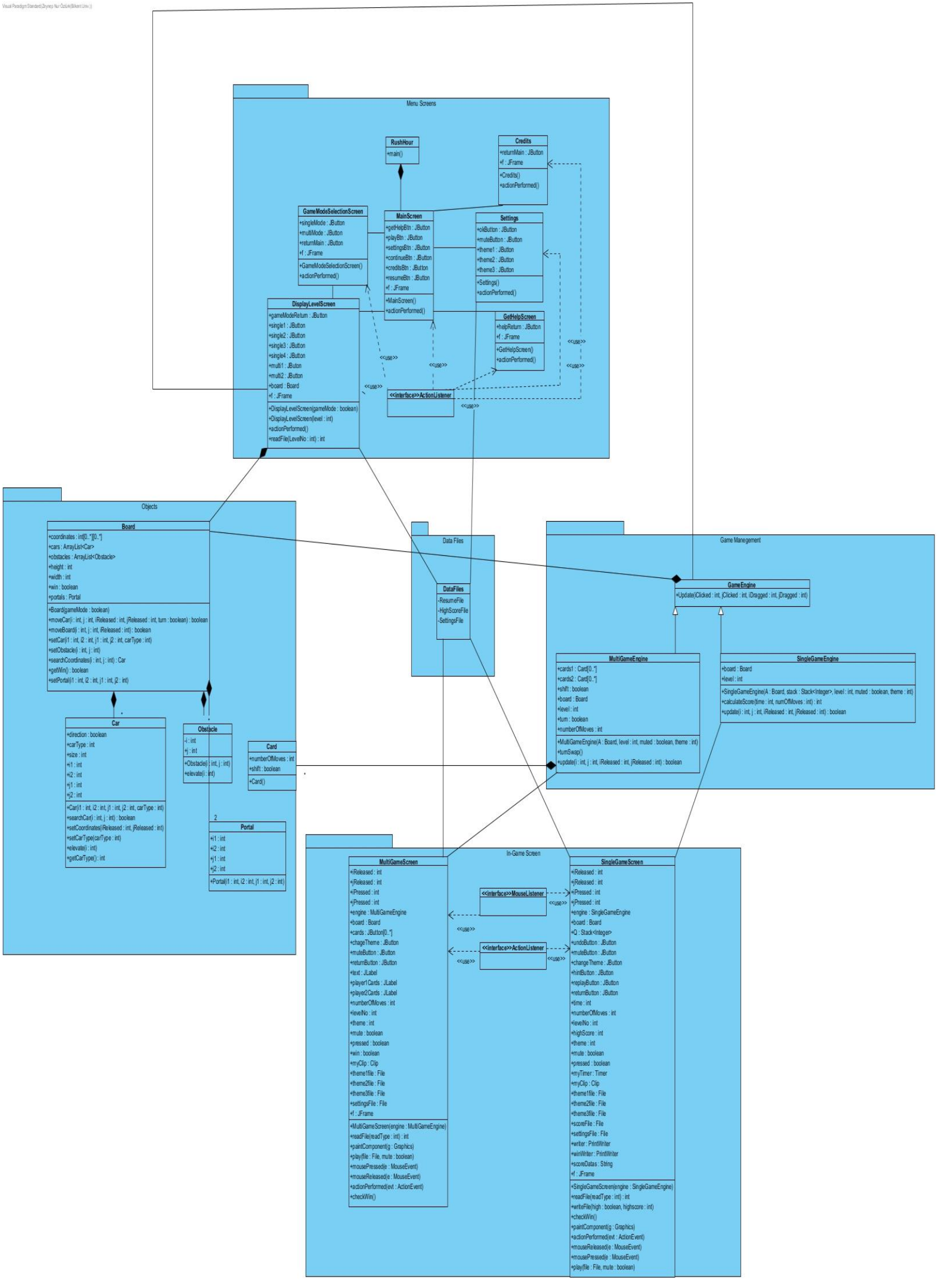


Figure 3: Second iteration subsystem decomposition diagram

Finally, the final class diagram and it's composition





2.2 Game Changes

Of course, these technical improvements also quickly reflected to the game when we implemented it as well, as we mentioned before, the first version of the program we showed in the demo only included single player mode and it wasn't looking very bright:

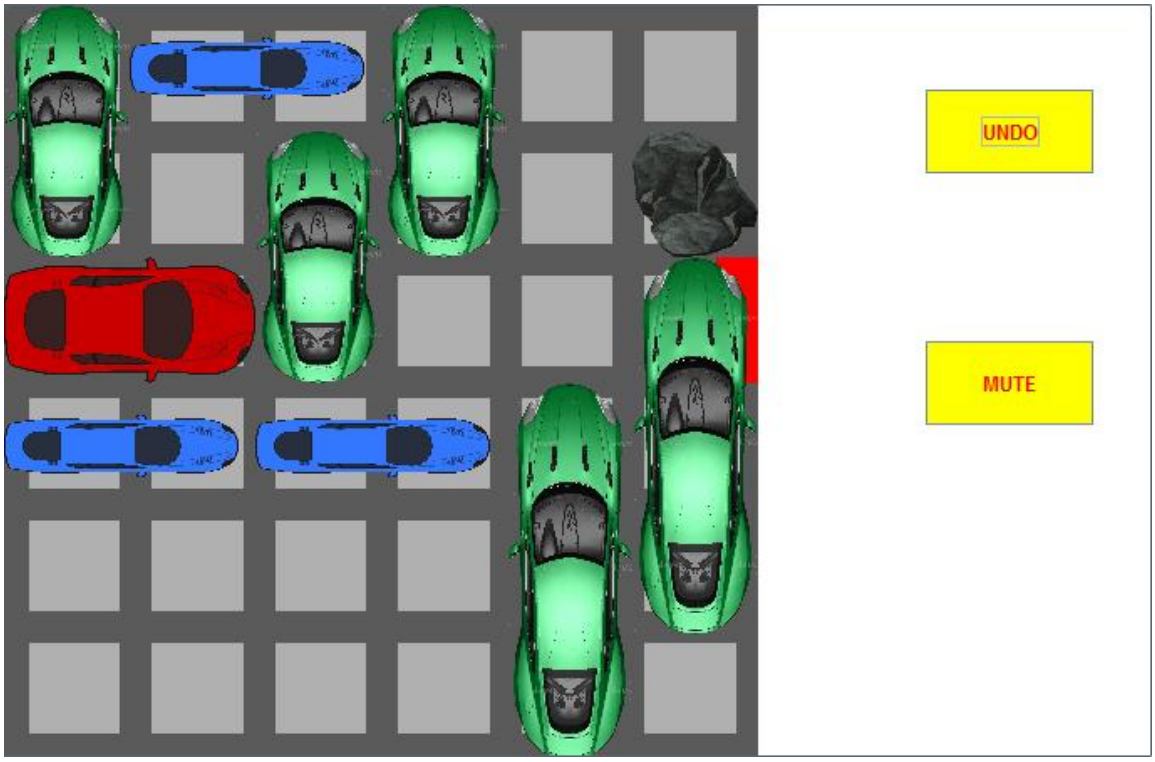


Figure 4: First Demo version of the game - Single player mode

But in our final version it looked more elegant and the features that were promised are implemented. In the figures below, you can see the available features that are implemented in the sidebar.

The ones which are not easy to represent with in-game pictures:

- Replay: Simply restarts the level
- Hint: Does a single correct move needed to be done in order to win the game
- Undo: Undoes the last car movement
- Time: The time spent on this level
- Number of moves: Any car movement increases this by one, undoing decreases it by one
- Mute: Mutes game music (was available in the first demo version)

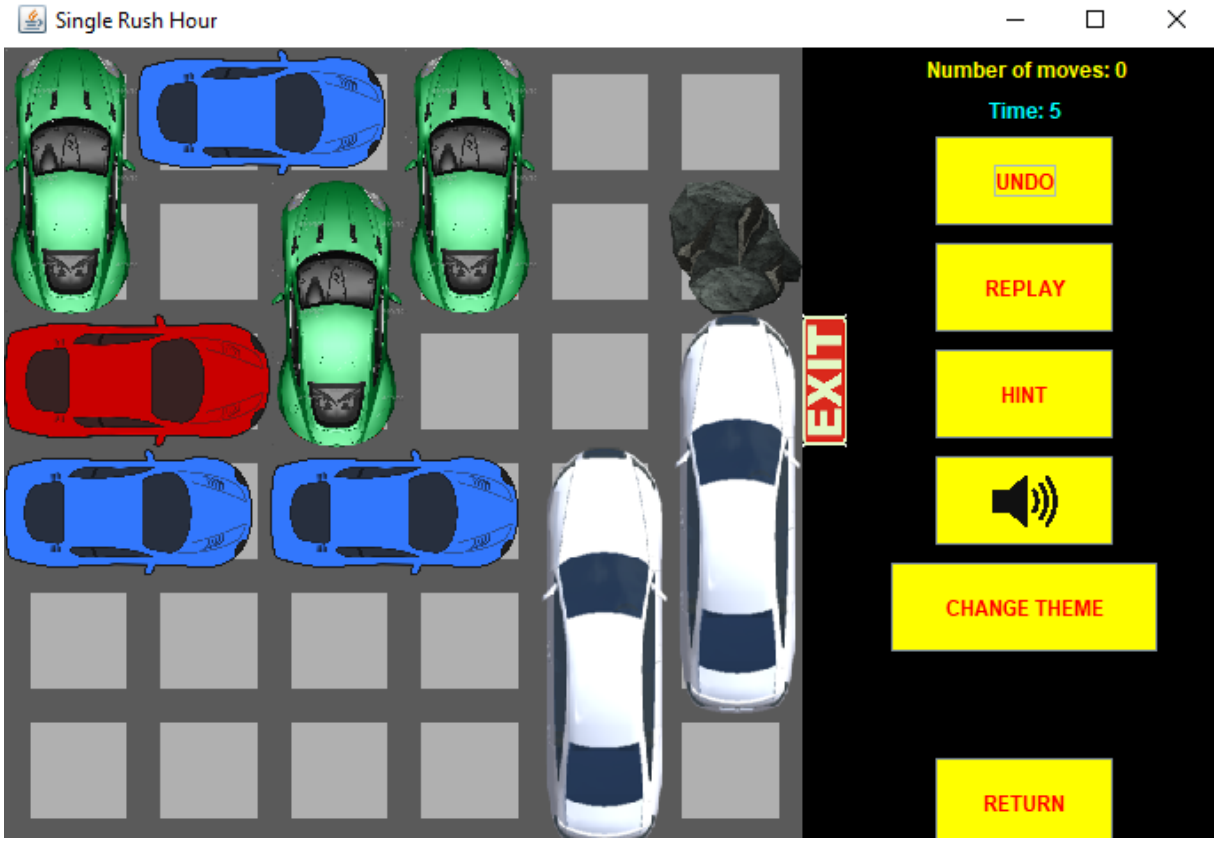


Figure 5: Final version of the game - Single Player Mode - Classic Theme

Figure 6: Final version of the game - Single Player Mode - Space Theme

Multiplayer mode is implemented, along with playing Cards in turns:

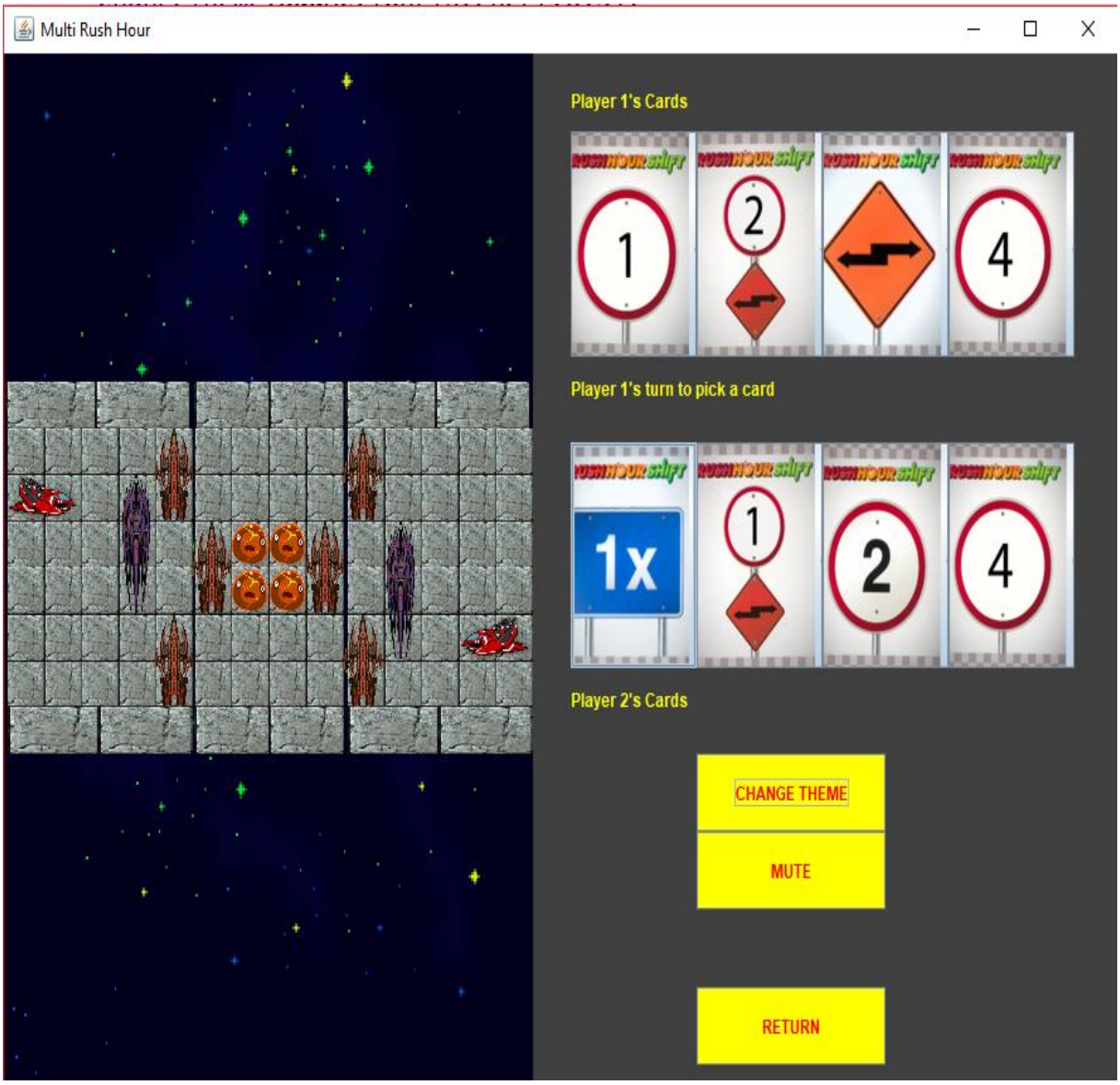


Figure 7: Final version of the game - Multiplayer Mode - Space Theme

And the portals are also implemented:

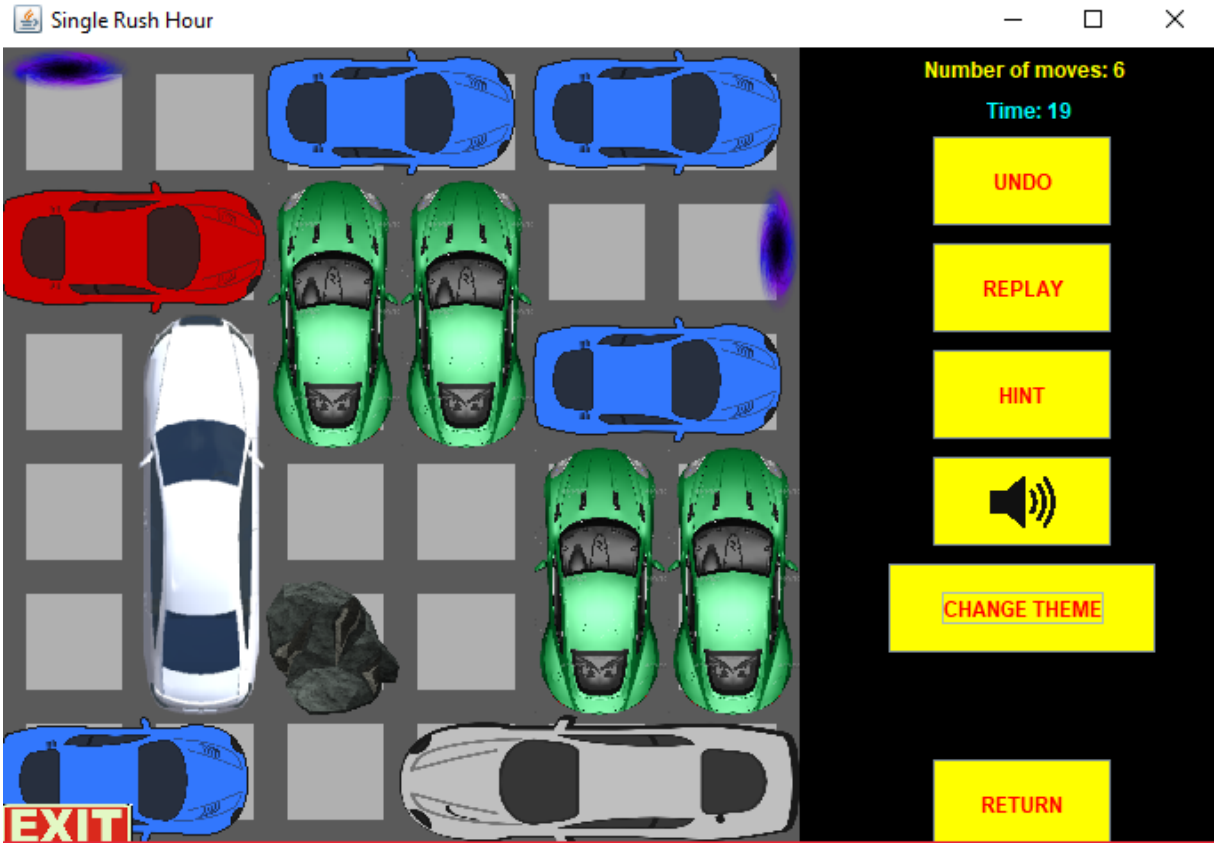


Figure 8: Final version of the game - Before entering portal

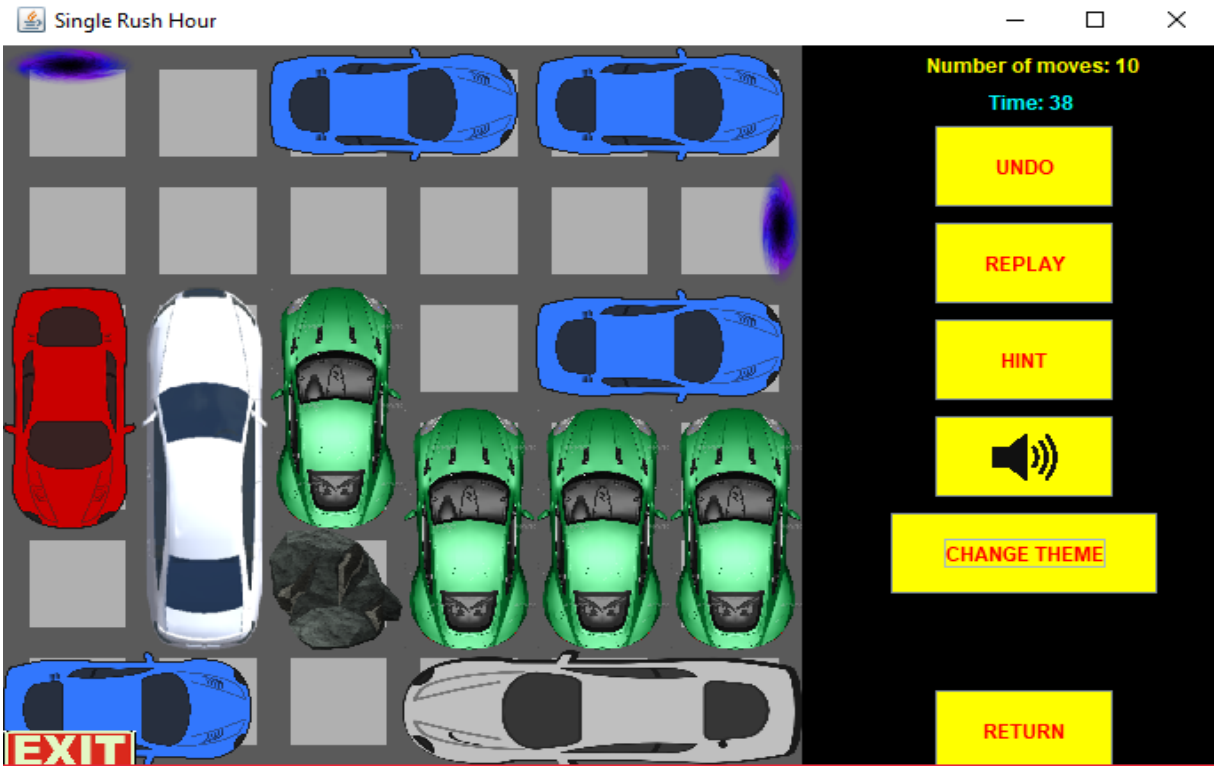
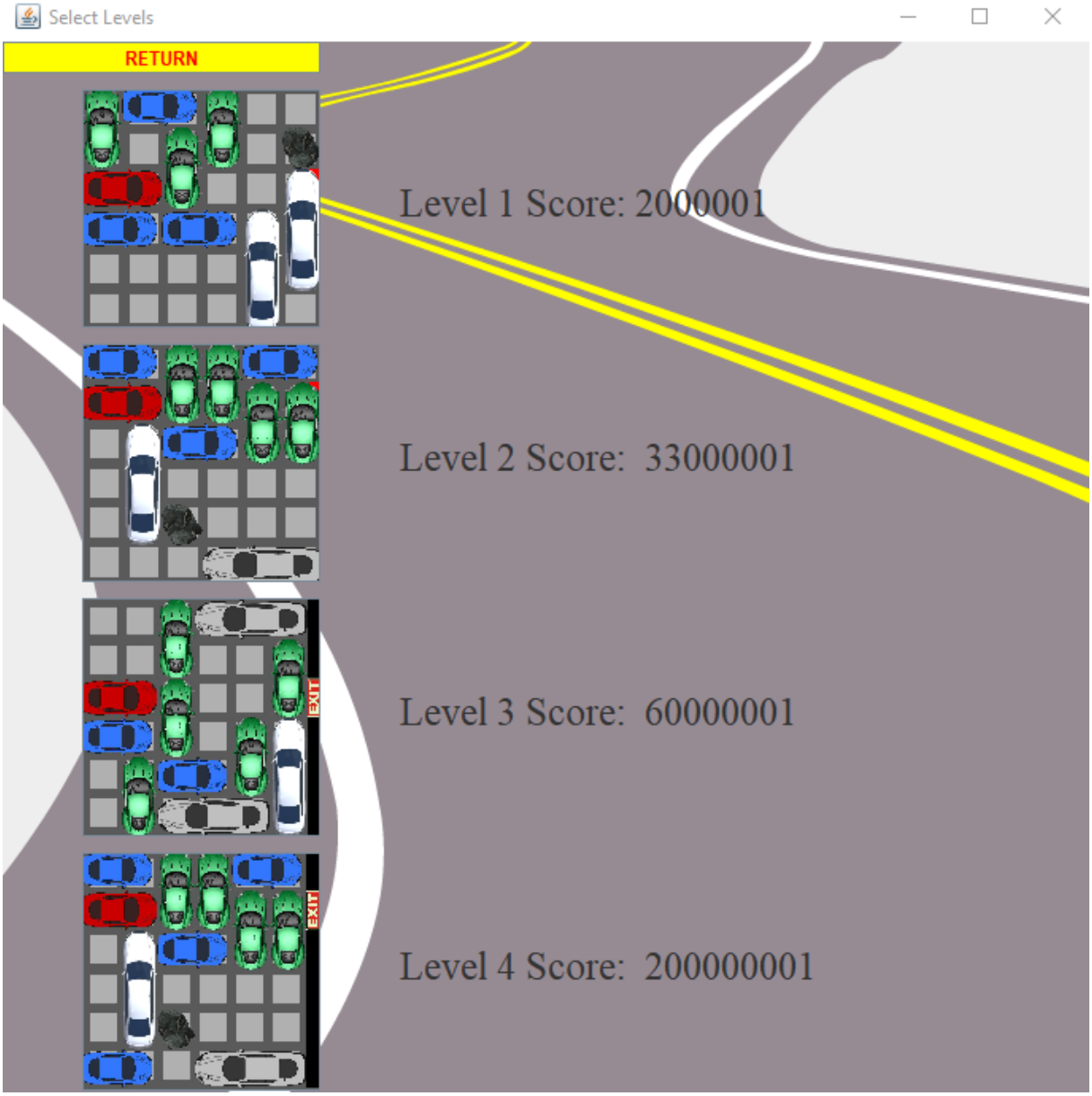


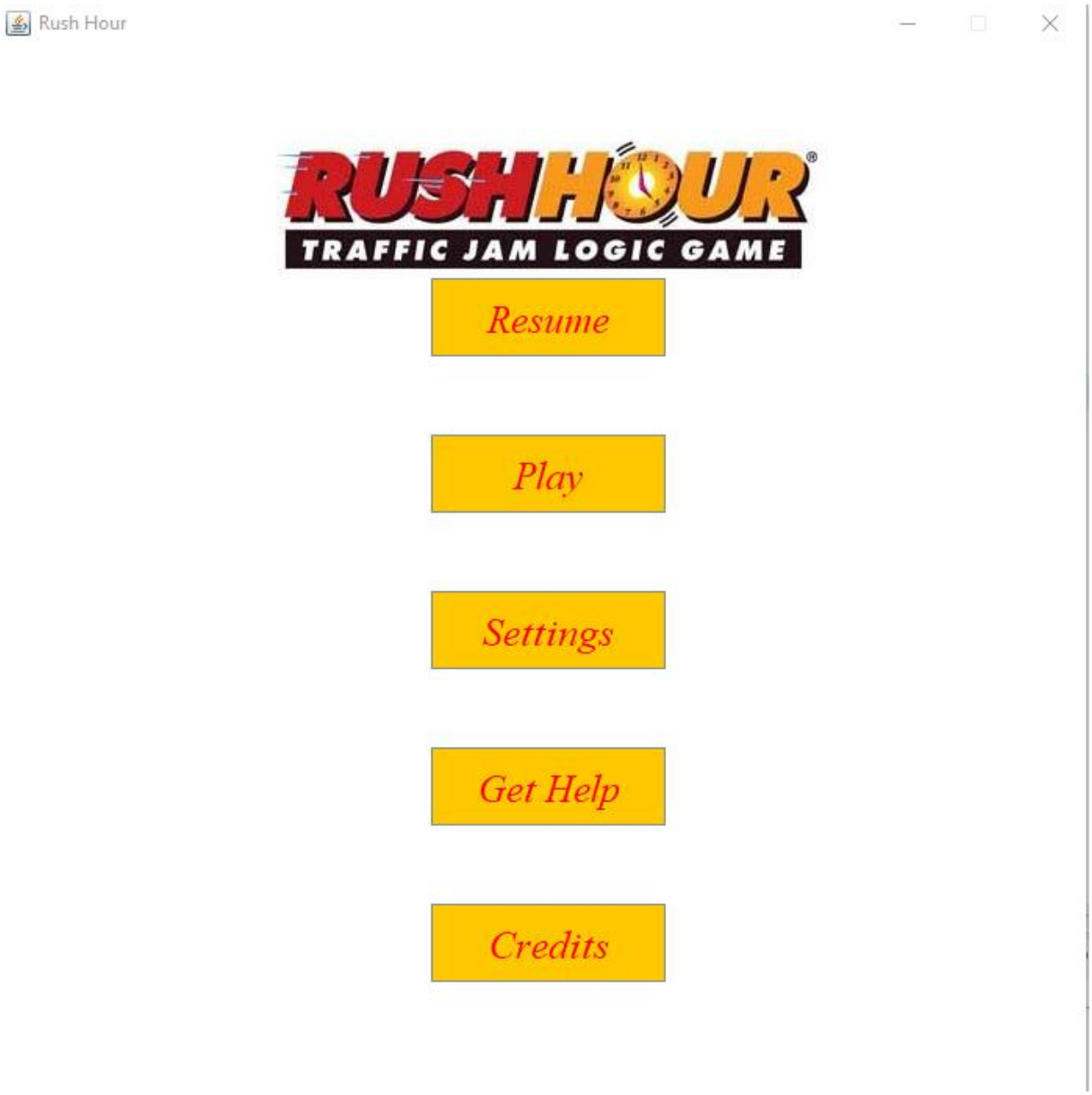
Figure 9: Final version of the game - After entering portal

Also, implementation of high score and resume is done. Highs Score of each level is seeable at displaying levels screen

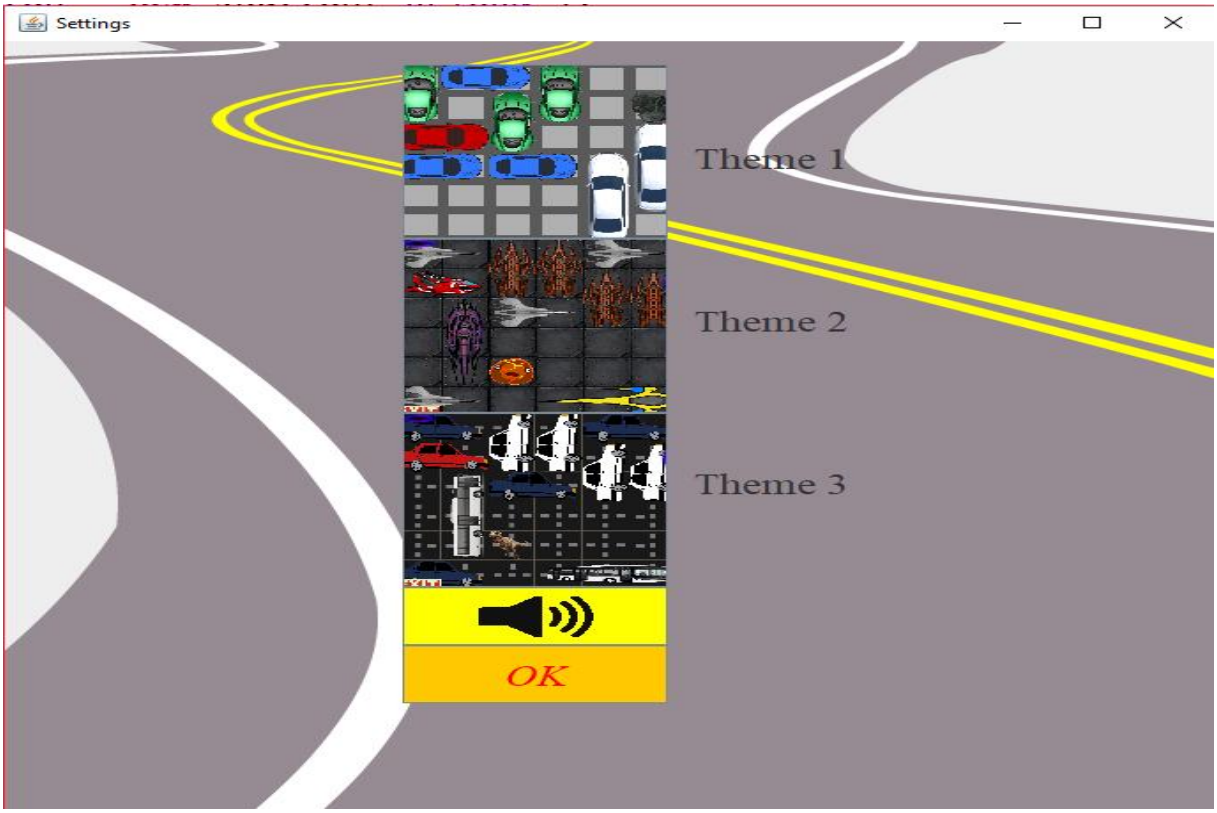


High scores are reached from an info file and rerecorded if your score is bigger than the high score.

Resume button is at the main menu and it directs you to the last played level you were in



Settings at the main menu is also done, it makes you able to choose the initial theme and sound settings of the game:



2.3 Other changes

We have presented in the first demo, in the first iteration reports and the second analysis report, that, as a new creative tool, we would implement “bridges” that a car may use to pass over other cars or obstacles. However, we did not add it to the game simply because there it required a larger map to be useful, since our map is 6x6 and 22x14, it is not very applicable to use 1x4 – 1x5 bridges. But instead of that, we’ve added whole another theme and different music to each theme.

In addition to that, we have also decided to extract portals from the multi-player mode levels and used it only on the single player mode levels (example of this is already given in figures 8 and 9), because when we use the portals, red car can be both vertical and horizontal and that’s why moving board would be unnecessary when the red car is vertical in multi-player mode.

3. User Manual

After following the installation instructions given in section 1.2, you should be able to see the following game screen, which is main menu:

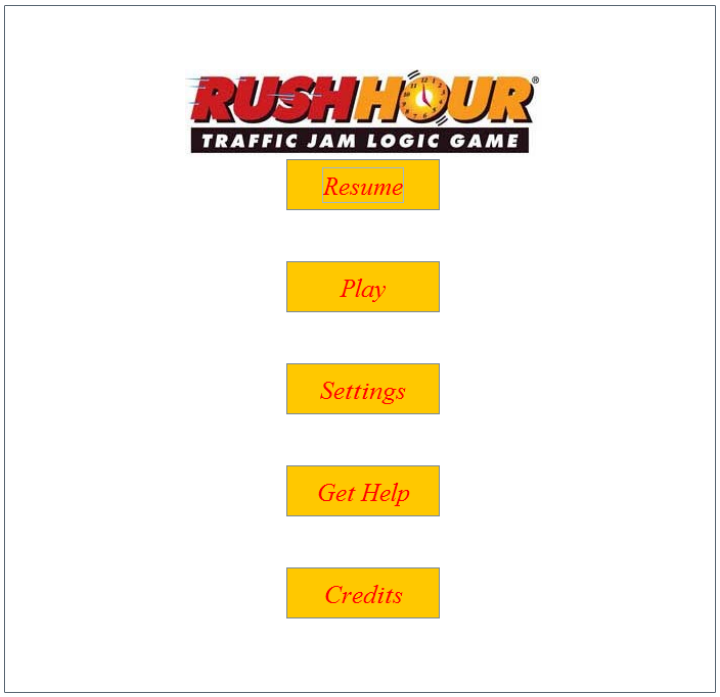


Figure 10: Main Menu Screen

From this screen, you can view the “Get Help” screen that explains the game mechanisms, “Settings” screen to personalize you’re playing experience (changing initial theme, sound options etc.), and “Credits” screen to view the credits. Those screens will not be shown in this report since they are trivial.

The “Resume” option would directly open the level from the last time you played, regardless of mode; but since it’s your first time, you should click on “Play” and as a result you would see this screen:

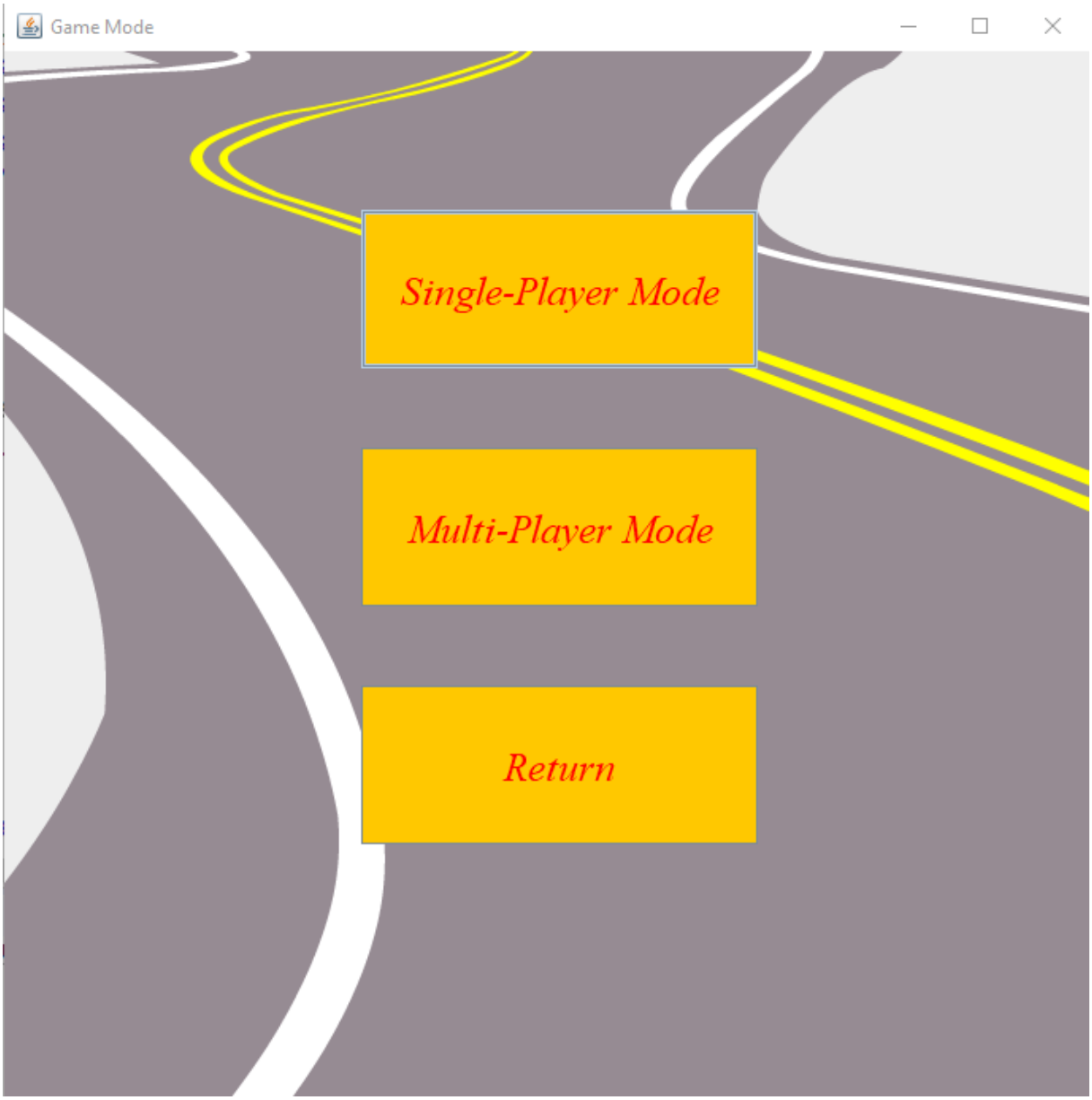


Figure 11: Mode Selection Screen

Depending on your choice, you would see the game modes' respective level selection screens, and would be required to choose a level. Return option will return you to main menu. Below is the example of level selection screen when single player mode has been selected. You can see that all the levels had been played and have corresponding high score data.

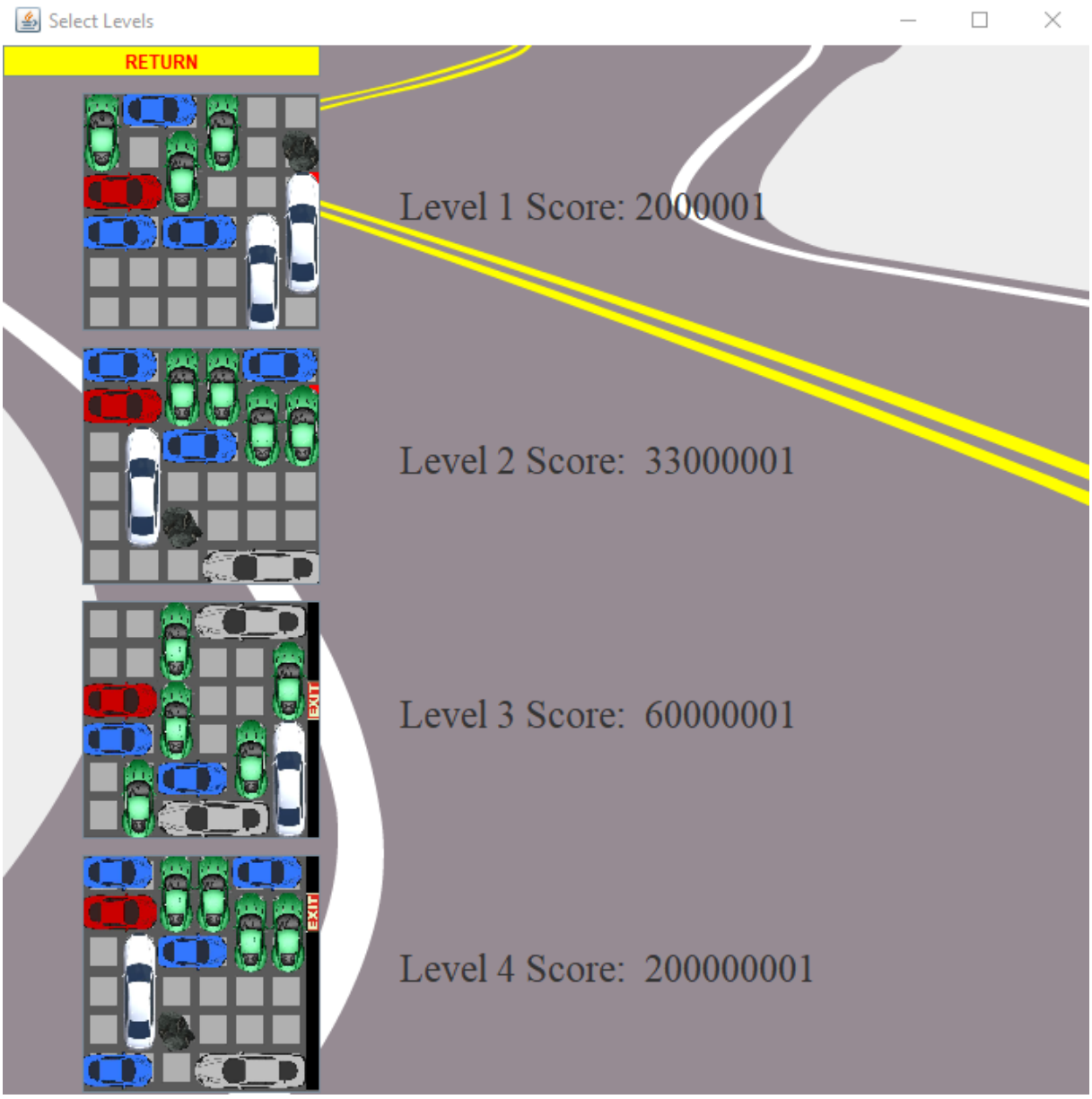
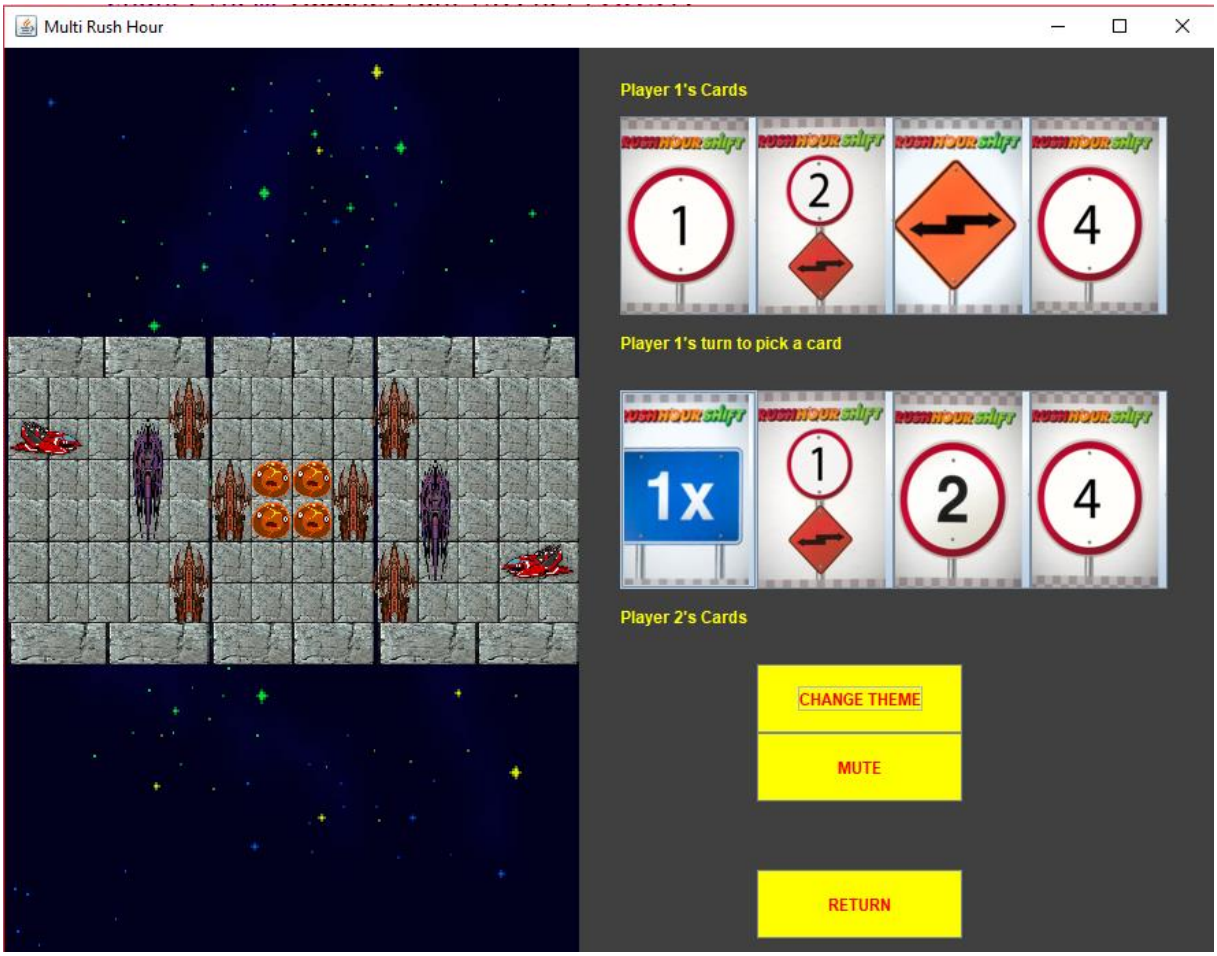


Figure 12: Single Player Mode Level Selection Screen

For the multi-player case, the instructions are given in the text in middle:



Cards are holding the value of number of moves that player can do and shift rights. 1x means player can move a car until there is an object blocking its way.

4. Work Allocation

Muhammet Said Demir:

- Contributed to the base game logic.
- Contributed to screens.
- Contributed to levels.
- Contributed to all Listeners.
- Contributed to reports.
- Contributed to UML diagrams.

Ata Coşkun:

- Contributed to the base game logic.
- Contributed to screens.
- Contributed to all listeners.
- Contributed to File Operations.
- Contributed to reports.
- Contributed to UML diagrams.

Zeynep Nur Öztürk:

- Contributed to the base game logic.
- Contributed to screens.
- Contributed to levels.
- Contributed to all listeners.
- Contributed to reports.
- Contributed to UML diagrams.

Asuman Aydın:

- Contributed to screens.
- Contributed to levels.
- Contributed to reports.
- Contributed to UML diagrams.

Tarık Emin Kaplan:

- Contributed to reports.
- Contributed to UML diagrams.

5. Conclusion

We think overall this project was a good learning experience, it showed us again, similar to the CS102 project, it's not just about sitting down and writing code; design is as important as implementation, if not more. We worked in accordance to the principles of OOP and the end result is, at least to us, pretty good.