
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

November 18, 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

### 1.1. Implementation Process

The implementation part of our 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 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 layed out by a team of two( Hygerta and Sıla) whereas the UI was assigned to a team of three (Aldo, Barış and Endi). 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] or a simple Vi editor.

Our meetings were taken on intervals as frequent as once in 3 days. This methodology of work was chosen to better get 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. These came about to the several changes we had to introduce from our initially proposed game. The platforms we used were the following: for the documents and the files, we have used Drive by Google and GitHub whereas for the milestone progress checking, subtask and role assignment we made use of Meistertask. All this was separated into “intermediate works” being the ones that were in progress and the final versions the ones to be merged. The intermediate works were uploaded to Drive if 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 represented ease in 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 in between teams and 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.

# 2. Design Changes

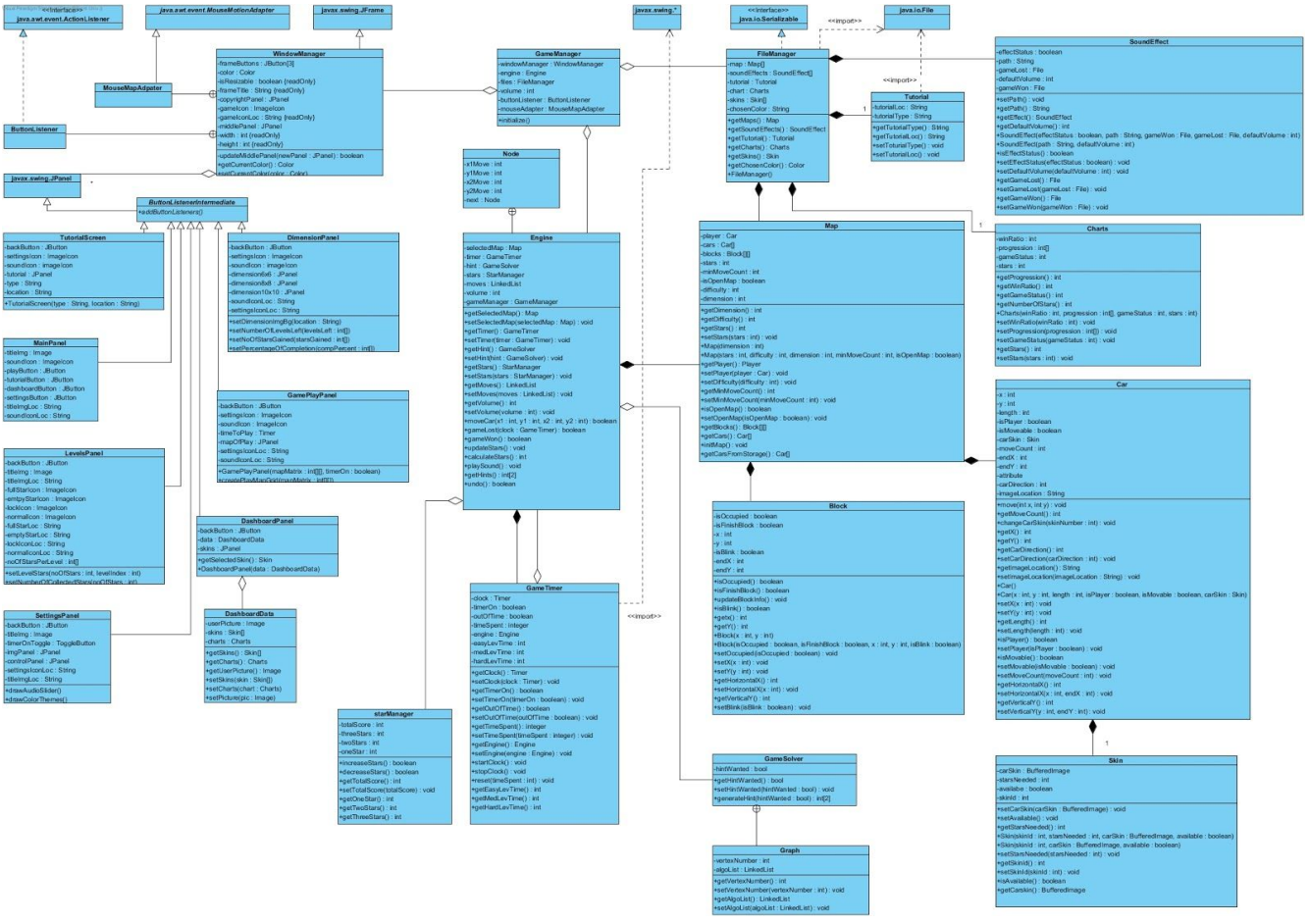


*Fig. 1 Updated Class Diagram*

## 2.1. Changes

During our implementation process, we decided to make some changes in some of the classes regarding some names (of the used parameters and methods). Earlier we wanted to use Java Swing but when we started the implementation, we decided to use JavaFX architecture.

Furthermore, we removed some of the methods we didn't really use in our implementation, or that we replaced with some methods found in already build Java classes or interfaces. We also added some layouts in UI like HBox or Vbox. The diagram below represents the updated class diagram.

Engine

- Added: a gameManager instance
- Added inner class: Node

GameTimer

- Added: getEasyLevTime(), getMedLevTime(), getHardLevTime()

Map

- Changed: OPTIMAL_MOVE_COUNT to minMoveCount
- Added: difficulty, dimension, getPlayer(), setPlayer(), setDifficulty(), getMinMoveCount(), setMinMoveCount(), isOpenMap, setOpenMap(), getBlocks(), getCars(), initMap, getCarsFromStorage
- Removed level: int, removed getLevel, getMapInfo, addCar, isMapFinished, starCount

StarManager

- Added: getOneStar(), getTwoStars(), getThreeStars()

Car

- Added: endX, endY, carDirection, getCarDirection(), setCarDirection(), imageLocation, getImageLocation(), setImageLocation(), two constructors, setX(), setY(), getLength(), setLength(), isPlayer, setPlayer(), isMoveable, setMoveable(), setMoveCount(), getHorizantalX(), setHorizantalX(), getHorizantalY(), setHorizantalY(), getVerticalX(), setVerticalX(), getVerticalY(), setVerticalY()

Block

- Changed: getBlockInfo() to isOccupied(), isFinished to isFinishBlock, getIsBlinked() to isBlinked()
- Added: endX, endY, two constructors, setOccupied(), setX(), setY(), getHorizantalX(), getVerticalY(), setBlinked()

Skin

- Changed getSkin() to getCarSkin(), checkAvailable() to isAvailable()
- Added: skinID, two constructors, setStarsNeeded(), getSkinID(), setSkinID(), setCarSkin(), setAvailable()

Charts

- Added: One constructor, setWinRatio(), setStars(), getStars(), setGameStatus()

Tutorial

- Imported file class

SoundEffect

- Changed: url to path
- Added: gameWon, gameLost, two constructors, getPath(), setPath(), getGameLost(), setGameLost(), getGameWon(), setGameWon()
- Remove soundEffect, setSoundOn(), setSoundOff()

FileManager

- Changed: Color chosenColor to String chosenColor
- Added: One constructor

User Interface (general)

- All panels become panes, listeners changed according to JavaFX, added some css codes

Main Screen

- Changed: titleImage to titleLabel

Dimensions Screen

- Added: some vBoxes and hBoxes, progression bar

Levels Screen

- Changed: titleImage to titleLabel
- Added: some vBoxes and hBoxes

Settings Screen

- Added: some buttons and vBoxes

## 2.2. Improvements

We were not able to finish the implementation of all the methods that we intended to implement during the design phase, but we have provided for a functional game basic layout of the gameplay that will be extended in our implementation for the second iteration. One of our main goals for the second iteration is to make some improvements for the game mechanics and play screen buttons along with the implementation of the GameSolver class which will 'predict' moves and suggest the user the next move that he could potentially make to solve the puzzle. The access to this suggestion is provided by the hint button on Play Screen. In order to achieve the latter and to be able to provide for the functionality of the undo button we will need to keep track of the player's moves as a linked list. For the first iteration, we have only one map, but we are planning to extend this to more maps with different difficulty sets for the second iteration. Currently, the game mechanics do not support a timer mode which needs to be provided as well. User Interface is almost completed but we might make some changes regarding the colours or icons used in the visual part of the project.

*For future implementation:*

- Implement getHints() method in Engine class
- Implement reset(), startClock(), stopClock(), timeSpent() in GameTimer
- Implement GameSolver using Dijkstra's Shortest Path algorithm
- Implement the Dashboard screen.
- Add sound files in order to play in a win/lose scenario.

# 3. Lessons Learnt

We have 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' are classes such as Stage, Application or MouseEvent. We learned how to use the MVC framework for these kinds of projects, how to interact with the user and how to control the actual game. Before this

course, we were only asked to do assignments that were solely coding or projects that would not required us to do any of the work we have done for this course. However, in this project, we learned how to write basic UML diagrams like sequence, state, activity, use case and class diagram, how to write a scenario, apply those scenarios to the diagrams and code. We also learned how to hold a formal meeting and separate the work in a group of people. While we had some report experience from CS102, this experience taught us to prepare thee kind of reports better. We are still in the first iteration but we also learned how to write code clearly since this project is bigger than the projects that we have ever done before. Regarding the soft skills, we were able to realise that keeping the group work structured and layered correctly means that the load balancing and the milestone provision for the deliverables becomes easier and it is a benefit on the long run.

# 4. User's Guide

## 4.1. System Requirements

Rush Hour can be played on all platforms. It will require JVM (Java Virtual Machine) and Java SDK. Any computer which has these will be able to install and run the game.

## 4.2. How to Use

### 4.2.1. How to Start

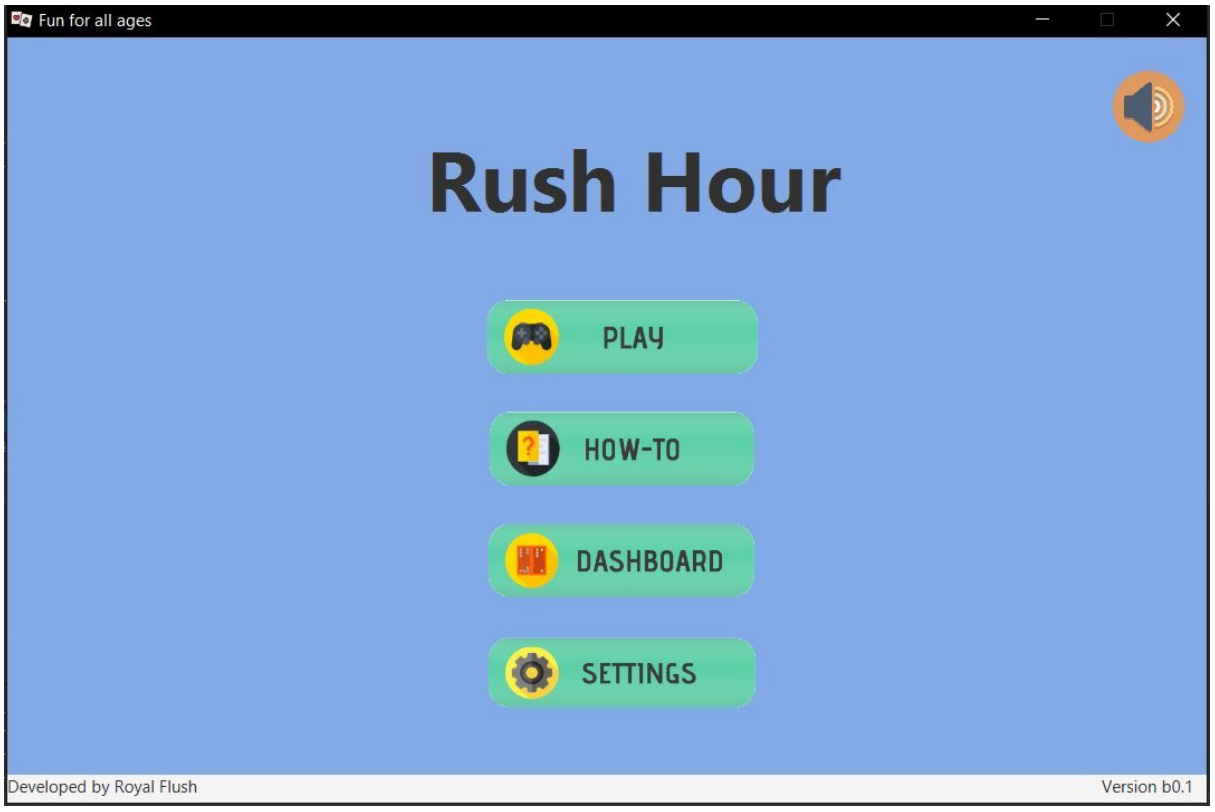


*Fig. 2 Main Screen*

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 the features of the game by pressing “HOW TO”, go to the “DASHBOARD” to see some charts and change car skins and 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. The player can mute all sounds by pressing the sound button at the right top corner of the screen.

### 4.2.2. Choosing Dimensions/Levels

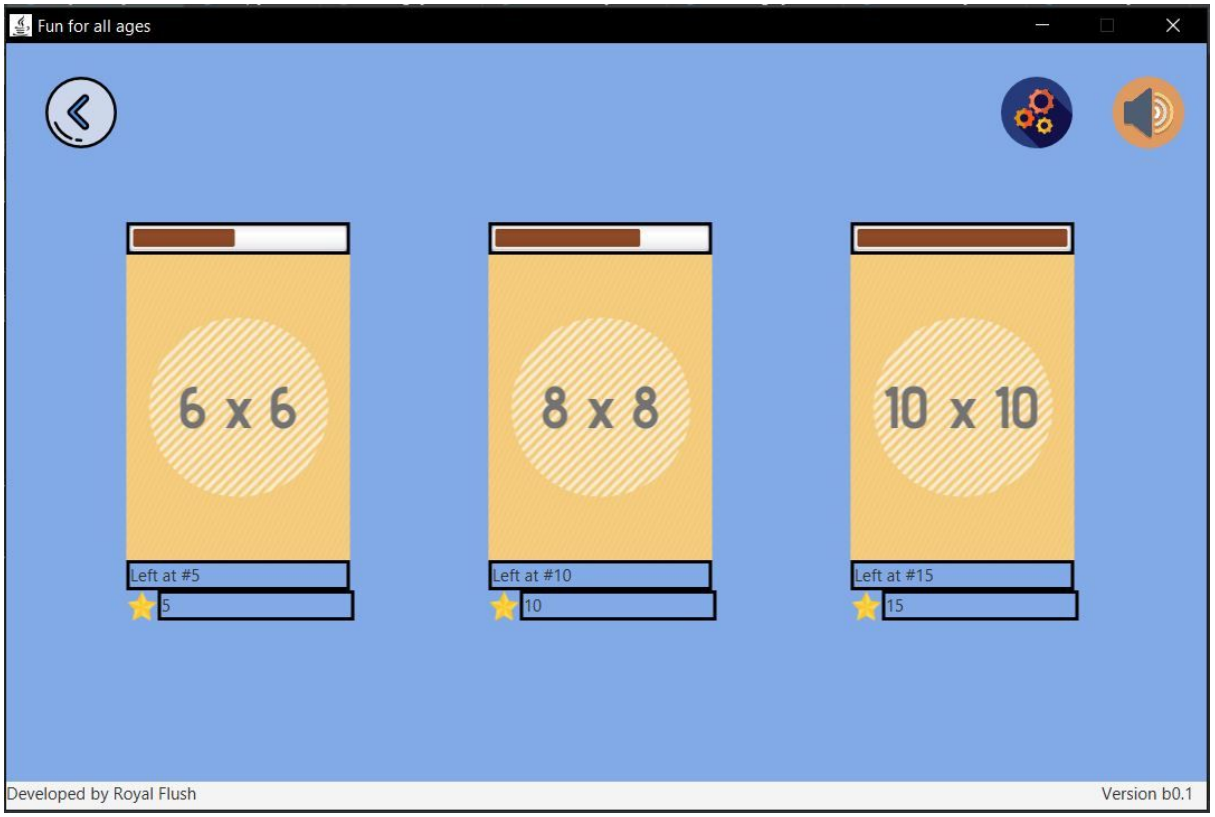


*Fig. 3 Dimensions  Screen*

Players will encounter with 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 going straight to the settings and to mute the sounds if the player desires it.


*Fig. 4 Levels Screen*

After choosing a dimension from the play screen, player will also choose a level. There will be X levels (to be decided on the second iteration) for each dimension and their difficulties will be different. The player will also be able to see their stars that they have gained for that level on each level button along with the total stars gained in that dimension at the bottom of the "Levels" screen. After choosing the level, players will go straight to the "Play Screen".

### 4.2.3. Play Game


*Fig. 5 Game Play Screen*

The player screen, There will be several buttons on the play screen. Players can go back to levels screen, go to settings to open the timer, mute the sound, undo their moves, get some hints and reset the game screen. 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 dashboard will be updated according to earned stars. Cars can be moved by dragging the mouse.

### 4.2.4. Settings

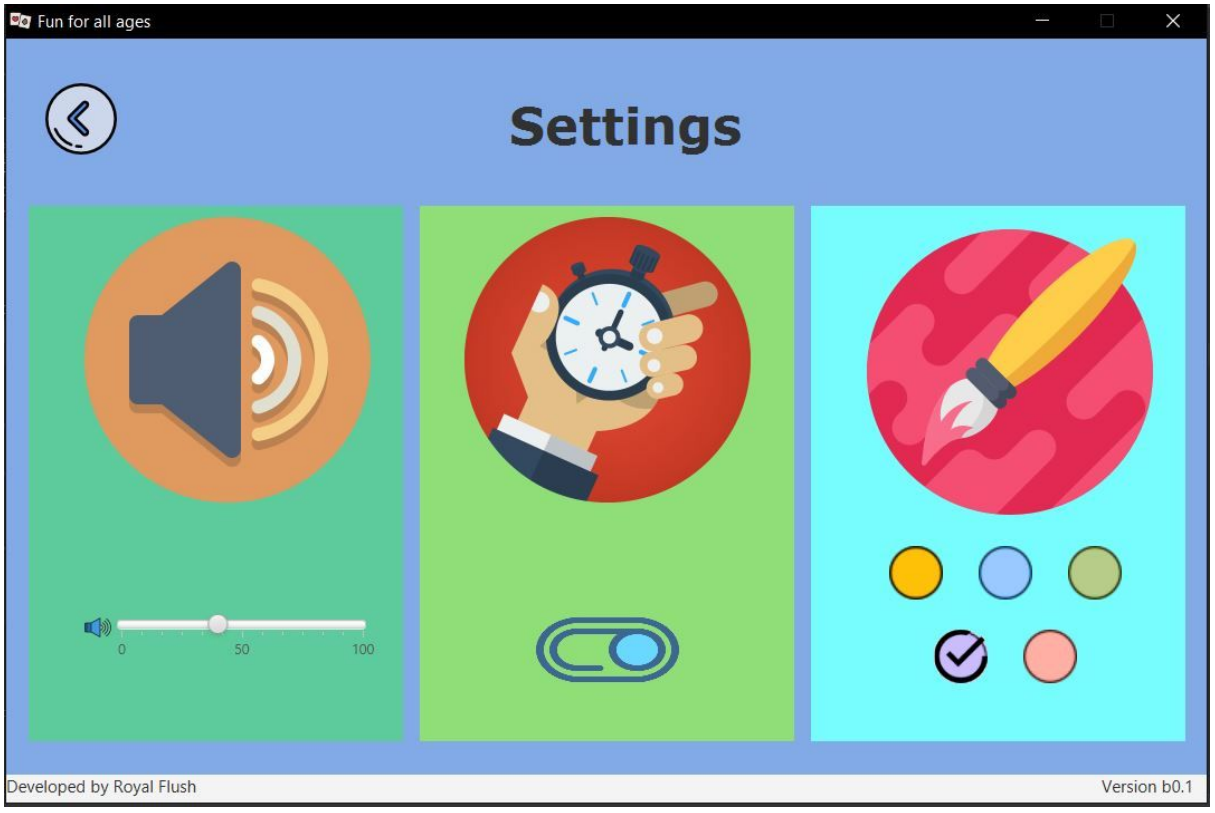


*Fig. 6 Settings Screen*

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.

### 4.2.5. Tutorial

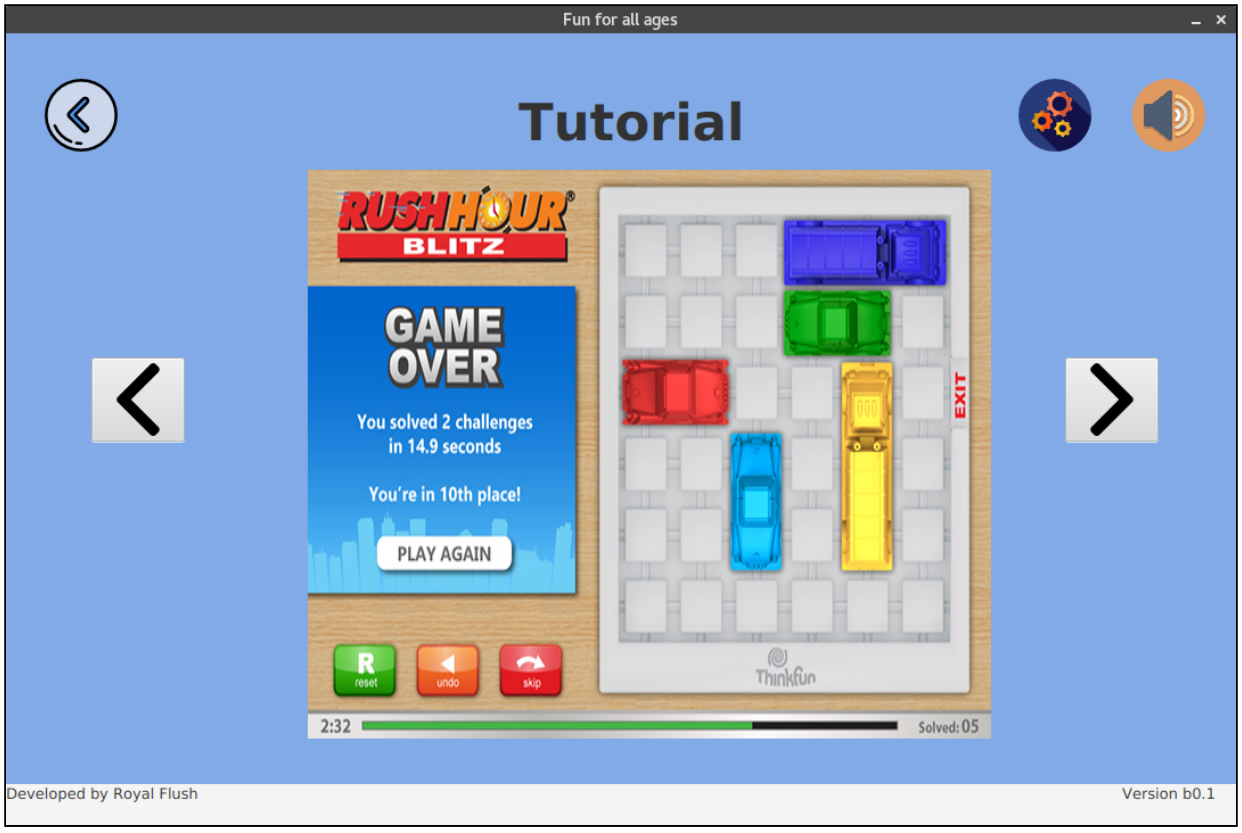


*Fig. 7 Tutorial Screen*

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).

## 5. References


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

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