Bilkent University

Department of Computer Engineering

CS319 Term Project

Rush Hour Game

Analysis Report

Aldo Tali
Barış Can
Endi Merkuri
Hygerta Imeri
Sıla Inci

Instructor: Eray Tüzün
Assigned TA: Muhammed Çavuşoğlu

Analysis Report
October 21, 2018

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

Contents

Analysis Report

Rush Hour Game

1 Introduction

Rush Hour is a puzzle board game designed by Nobuyuki Yoshigahara in the 1970s, Japan's most celebrated inventor, collector, solver, and communicator of puzzles. Nob commercially 'sold' some of his designs to some game-developing companies and Rush Hour Game was one of them. Today, Rush Hour is a licensed and yet the most successful game under the name of Binary Arts (ThinkFun). Different versions of the game were eventually developed, different features were added, thus making the game more popular and delightful for different generations in a family.

At first, Rush Hour was a wooden sliding block puzzle. The fact that it was turned to a computer game made it even more designable and easier to be extended with new features. Therefore, even though the concept of the game seems to be easy enough to be understood, the game offers a wide range of increasingly-difficult level combinations. Early designs of the game were inspired by the terrible feelings of being caught in a terrible traffic jam. Later on, instead of celebrating frustration because of traffic, the game was redesigned to celebrate the success of finding the way out of it. 'RoyalFlush' group intends to add some features to the design of the game to fulfill this purpose.

Our Rush Hour game is a traffic jam logic game with an optional timer which challenges novices and experts alike. The main aim of our Rush Hour game remains the same with the original game: to arrange the other vehicles by sliding them in such a way that you get your car through the exit of various gridlocks of diverse dimensions which change from one level to another. All you have to do is: slide the blocking cars and trucks in their lanes either vertically or horizontally until you clear the path for the car to escape in order to win some stars. With these stars, you can change the skin of your car or get rid of one car per level and you have to have reached a certain number of stars to be able to get to the harder levels. Furthermore, thinking of the importance of player's motivation, if you play the game daily you can also be gifted some bonuses like stars or hints.

The game features three levels of difficulty: easy, medium and hard, allowing players to progress at their own pace. These levels will have different features regarding their difficulties. For example, harder levels will have stationary cars/blocks.

Among the other features, a player can use the RESET and UNDO button to start the level again or undo some moves in case they get stuck in the traffic. There will also be a SETTINGS button by which the user can change some of the optional features such as sound, themes, timer mode and can see the Scoreboard for the current level. Moreover, the player has the chance to "play-pause", use "hints", "undo", "reset" anytime during the game. To get used to the new features, you can have a look at the provided tutorial any time.

You need to keep in mind one thing while you are playing our game:

There's only one rule- don't get road rage.

2 Overview

Rush Hour by Royal Flush group introduces a simple digital version of the classic game. The game holds on to the idea of a single player mode. The player in this game is not represented by any specific character in the gameplay however he is identified by his progress in the game and his own user dashboard. The player can open Rush Hour – Royal Flush and get the first insights of the game by going to the How-To-Play Screen from his Main Screen. Once the user gets a hold of the gameplay in Rush Hour, he can then personalize his game experience in the Settings section where he is allowed to choose between a timer mode gameplay and a classical version of the play as well as maximize the game experience by several functionalities such as the change background features and the sound volume features.

With the personalized settings all set up the user can now go into the gameplay and choose whether he decides to challenge himself with the 6X6, 8X8 or the 10X10 versions of the games. Within each of these dimensions a level map that shows the progress so far is shown to the user. Initially, he can only play the first levels and the latter ones are unlocked as he chooses to go further in the gameplay. The player now can go into the real user experience when it comes to the gameplay as soon as he chooses one level. Given that he had previously selected that he likes to play in a no timer mode, the player has the right to make all the possible legitimate moves in the game to take the designated car through the exit. Otherwise, the player is constrained to solve the puzzle of the map within the given time period. Nevertheless, no matter the mode, on successful completion of each level Rush Hour rewards the user for his progress by giving access to the next map on the list and as such the player can go on to unlock new levels and increase his rewards status along with the update of his total progress. The game is considered to be complete only when the user has been able to unlock all the levels in all three proposed dimensions. Once the player reaches this point he will be displayed a completeness animation on the screen. In addition, throughout all the game, the progress and the number of levels unlocked so far, along with the number of trials for each level will be shown in the user's personal dashboard which will be represented in terms of pie charts for the best user experience.

3 Functional Requirements

In this section, we discuss the functional requirements of our project.

3.1 PlayGame

The user can enter the play mode of the game. Once the Play button is pressed the user will be directed to the next screen where he will see three boxes occupying the screen corresponding to three variations of the game corresponding to the dimensions of the map: 6x6, 8x8 and 10x10. The user can select any of these boxes to make the choice of the game mode he wants to play next. Once a mode is selected the user will have a list of maps whose configuration he can try to solve with a timer or in pure unlimited time. All the games in this list will be one of the three types of difficulties, easy, medium or hard. The user will have to face restrictions here since he cannot access the medium nor the hard levels without first completing the easy ones.

The main purpose will be to solve the puzzles so that the user unlocks more of them and gets more stars.

3.2 Dashboard

The user can access his visual records, formed based on his gaming patterns. He should be able to use this to change the skin of his car and get informed about different statistics of his gameplay. For example, he can see how many stars he has collected, the completion percentage of all the levels etc.

3.3 Settings

The user can manipulate several features of the game. He can toggle between timer on and off meaning that the game being played is going to be either in a timed mode or with no time restrictions. He should be able to change the themes of the application as a whole, and he should be able to adjust the volume of the sound system according to his wishes.

3.4 Contact Developers

The users are going to be able to contact the developers of the game being the Royal Flush group. In this part, the user should also be able to see credits information, or in other words, who took part in the project and what is the contribution to the project as a whole.

3.5 How-To-Play

The user is going to be able to see some tutorial which will either be in video format, sliding pictures or a pure map with a preset configuration which just shows its moves to the users.

4 Non-functional Requirements

In this section, we discuss the non-functional requirements of our project.

4.1 Usability

Since this is a child game, it will have a simple interface. There will be a tutorial in order to teach the game. Also, when the player is stuck in the game, a hint option that shows them the next move, an undo option and a reset option will help them to solve the puzzle.

The main purpose will be to solve the puzzles so that the user unlocks more of them and gets more stars.

4.2 Reliability

The game will not require an internet connection, and the player's data will be kept in the database. Therefore, there will not be any security issue. The game will be saved automatically after completing each level, but if the player leaves in the middle of the game, the progress will not be saved. Because the game will have an undo option, player's moves will be saved as long as he/she does not leave the game.

4.3 Performance

Since this is a small application, there will not be any performance issues. The game will be optimized.

4.4 Supportability

The game will have a small size and database so that it can be easily transferred into another computer. The game will run in a Windows operating system and require Java(?). Classes and methods will be coded in an organized manner so that future problems can be solved quickly.

4.5 Extendibility

Several car skins and themes will be available in the game. Since the game will have a reward system according to the number of moves made inside the game, the player will get more stars, and spend stars to get more skins or hints. The game will be implemented with an extendible design so that new maps, themes or skins can be added later on the game without changing many code segments in order to use the stars.

5 System Models

5.1 Use-Case Model

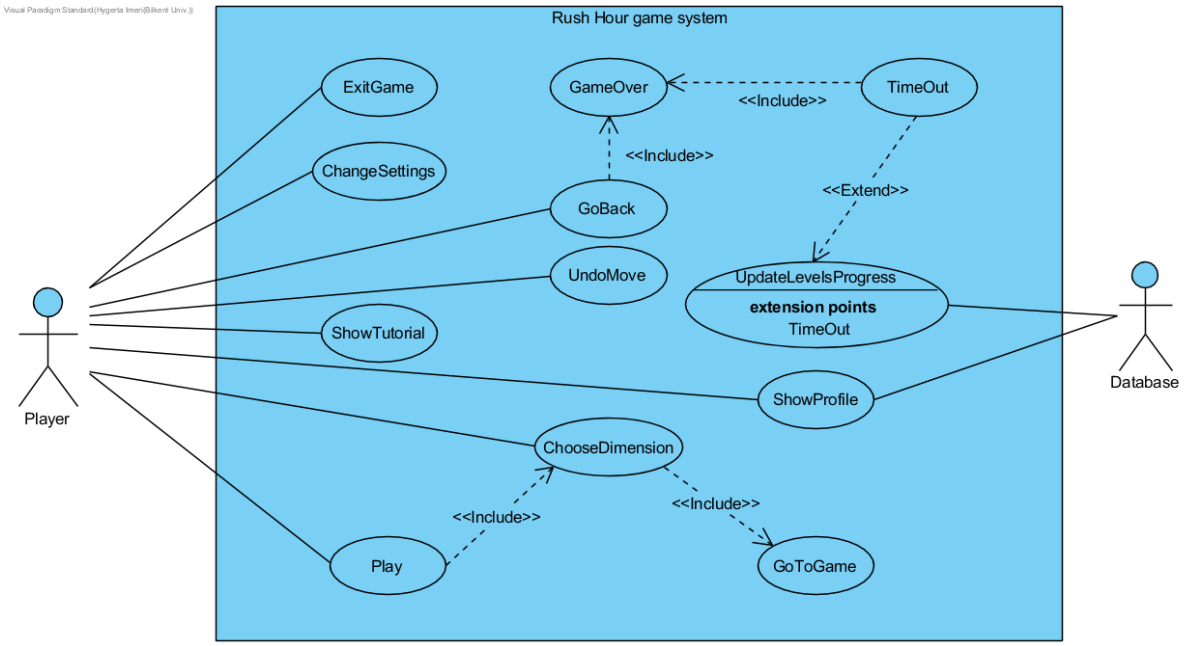


Figure 1: Use Case Diagram

Use case Name:	**ExitGame**
Participating actors:	Player
Stakeholders/Interests:	The Player decides to finish the game.
Flow of events:	1. Player opens the game 2. Player finds himself in a game screen 3. Player presses Close Button 4. Game system closes
Pre-conditions:	Player is in any Screen of the game
Post-conditions:	System closes and it is terminated.
Exit conditions:	System interrupts in between unexpectedly due to external reasons.
Quality requirements:	None

Use case Name:	**ChangeSettings**
Participating actors:	Player
Stakeholders/Interests:	The Player decides to change the settings configuration of his game.
Flow of events:	1. Player chooses Settings button from Main Screen 2. Player turns on the timer mode 3. Player changes sound volume 4. Player presses approve button after changes 5. Configuration is finished.
Pre-conditions:	Player must have pressed the Settings button from Main Screen. Player must have pressed Settings icon in some other screen.
Post-conditions:	Player alternates the options for the sound management. Player checks if he wants to use timer or not.
Exit conditions:	Player presses the close/exit button on the game application. Player presses back button to Main Screen.
Alternative Scenarios:	Player presses back button before approving the changes. Player closes the application before approving the changes. System interrupts due to external factors.
Quality requirements:	None

Use case Name:	**ShowTutorial**
Participating actors:	Player
Stakeholders/Interests:	The Player decides to learn how to play the game. The System opens the application and starts the game.
Flow of events:	1. Player chooses the How To Play button from the Main Screen. 2. Player opens the tutorial for the game 3. Player learns the game play 4. Steps 2-3 might be repeated. 5. Player goes back to Main Screen 6. Player starts playing
Pre-conditions:	Player must have pressed the How to Play button from Main Screen. Player must have configured his game settings.
Post-conditions:	Player sees the tutorial and goes back to Main Screen, Player starts the play or selects the settings.
Exit conditions:	Player presses back button to Main screen. Player presses the close/exit button on the game application.
Alternative Scenarios:	Player closes application System interrupts due to external factors.
Quality requirements:	None

Use case Name:	**Play**
Participating actors:	Player
Flow of events:	1. Player opens the main game window 2. Player chooses to play the game, thus presses ‘’PLAY” button 3. Player is directed to the Choose Dimension screen to continue
Entry condition:	Player has opened the window of the game
Exit condition:	The player is in Choose Dimensions screen and presses ‘GO BACK’ button to return to the main screen OR presses ‘EXIT’ to close the window
Quality requirements:	None

Use case Name:	**ChooseDimension**
Participating actors:	Player
Flow of events:	1. Player presses Play in the main Screen 2. Player enters the ‘ChooseDimension’ screen 3. Player chooses one of the three options for the board size 4. Player is directed to that specific dimension’s levels screen
Entry condition:	The player has already chosen to play.
Exit condition:	The dimension is chosen and player enters the levels Screen of that specific dimension OR the player presses ‘BACK’ and goes back to main screen OR presses ‘EXIT’ to close the game window
Quality requirements:	Waits for the player to choose dimension level to play game

Use case Name:	**GoToGame**
Participating actors:	Player
Flow of events:	1. Player chooses one of the unlocked levels in the Levels screen 2. Player is directed to the gameplay screen
Entry condition:	Player should have chosen to play in a specific dimension and should have unlocked the level he/she wants to play.
Exit condition:	The dimension is chosen and player enters the levels Screen of that specific dimension OR the player presses ‘BACK’ and goes back to main screen OR presses ‘EXIT’ to close the game window
Quality requirements:	None

Use case Name:	**GameOver**
Participating actors:	Player
Stakeholders/Interests:	Player cannot complete the level before time limit, goes back to previous screen or just quits the game.
Flow of events:	1. Player chooses any dimension with any levels. 2. Player opens the timer mode. 3. Player cannot complete the puzzle in given time.
Pre-conditions:	Player should be in the game screen.
Post-conditions:	Player sees the tutorial and goes back to Main Screen, Player starts the play or selects the settings.
Alternative Scenarios:	**1.** Player chooses any dimension with any levels. a. Player quits the game and it resets. b. Player goes back to previous screen using back button and game resets.

Use case Name:	**GoBack**
Participating actors:	Player
Stakeholders/Interests:	Player may want to go back between screens.
Flow of events:	**1.** Player enters any screen from main screen. **2.** Player goes back to previous screen.
Entry condition:	Player clicks on “Back” button in any screen.
Exit condition:	Player returns the previous page.

Use case Name:	**UndoMove**
Participating actors:	Player
Stakeholders/Interests:	Player may want to undo his/her last move.
Flow of events:	**1.** Player chooses any dimension with any levels. **2.** Player makes some moves. **3.** If player is stuck, he/she can undo his/her move and goes back to previous ones.
Entry condition:	Player should be in the game screen.
Exit condition:	Player goes back to his/her last move. (can be repeated)

Use case Name:	**Timeout**
Participating actors:	Player
Flow of events:	1. The player enables timer mode. 2. The player chooses a level and plays the game. 3. The player runs out of time. 4. The player loses the game and gains no stars.
Entry condition:	The player is playing in timer mode.
Exit condition:	The player loses the game.
Quality requirements:	None

Use case Name:	**ShowProfile**
Participating actors:	Player
Flow of events:	1. Player presses the Dashboard button. 2. The player views his profile.
Entry condition:	The player is in the main screen of the game.
Exit condition:	The player is in the Dashboard screen and sees his/her profile.
Quality requirements:	None

Use case Name:	**UpdateLevelsProgress**
Participating actors:	Player
Flow of events:	1. The player solves one of the puzzles. 2. The score for the puzzle in the database is updated.
Entry condition:	The player completes a level.
Exit condition:	The database is updated with the latest scores.
Quality requirements:	None

5.1.1 Scenarios

Scenario #1

Scenario Name:	**Play on timer mode**
Participating actors:	Player
Flow of events:	1. Player wants to play on timer mode. 2. The player presses the settings button. 3. The settings screen appears. 4. The player enables the timer mode. 5. The player chooses a level and continues to play.
Entry condition:	Player is playing the game.
Exit condition:	The game is on timer mode.
Quality requirements:	None

Scenario #2

Scenario Name:	**Change the car's skin**
Participating actors:	Player
Flow of events:	1. The player decides to change the skin of his car. 2. The player goes to the main screen. 3. The player presses the dashboard button. 4. The dashboard screen appears. 5. The player chooses one of the skins. 6. If the player owns the skin it gets applied to his car. 7. If the player does not own the skin he has to buy it. 8. If the player has enough stars to buy the skin he buys it. 9. The player presses the back button and continues to play with his new skin.
Entry condition:	The player is in the game.
Exit condition:	The car has a new skin.
Quality requirements:	The player does not have enough stars to buy a new skin, so he has to get more stars to buy a new skin.

Scenario #3

Scenario Name:	**Exit the game**
Participating actors:	Player
Flow of events:	1. Player is in the main Screen 2. OR looking at his/her 'profile' through Dashboard Screen 3. OR is adjusting Settings 4. OR is choosing level/dimension of the game 5. OR is just playing the game (gameplay Screen) 6. Player presses 'Exit' to close the game window 7. A confirmation question pops up to let the Player know the risks of this action 8. If player agrees Game ends and nothing related to that game is saved/updated in the database 9. Otherwise, Player remains in the game.
Entry condition:	Player is anywhere in the game map
Exit condition:	Player is anywhere in the game environment (every screen provides an exit button).
Quality requirements:	None

Scenario #4

Scenario Name:	**Winning the game**
Participating actors:	Player
Flow of events:	1. The player decides to open Rush Hour. 2. The player decides to create a game and presses Play Game button. 3. The player starts the game with 6X6 dimensions. 4. The player realizes only the first level of the dimension is available and decides to play it. 5. The player plays the game with moving the blocks up and down and moving the car right and left. 6. The player arrives the end. 7. The player completes the level and get three stars. 8. The player quits the game.
Entry condition:	None
Exit condition:	Player has completed the level and the scores are updated.
Quality requirements:	None

Scenario #5

Scenario Name:	**Winning Stars**
Participating actors:	Player
Flow of events:	1. Player starts the game 2. While player plays the game, number of moves is calculated 3. Player must complete the task in the game (in time) 4. The calculated number of moves is compared to the minimum possible number of moves 5. According to the comparison, the user either gets 1, 2 or 3 stars as prize at the end of the game
Entry condition:	1. The player must be in the game 2. The player must have completed the task 3. (Optional) The player must complete task on time
Exit condition:	Total number of stars is updated.
Quality requirements:	None

5.2 Dynamic Models

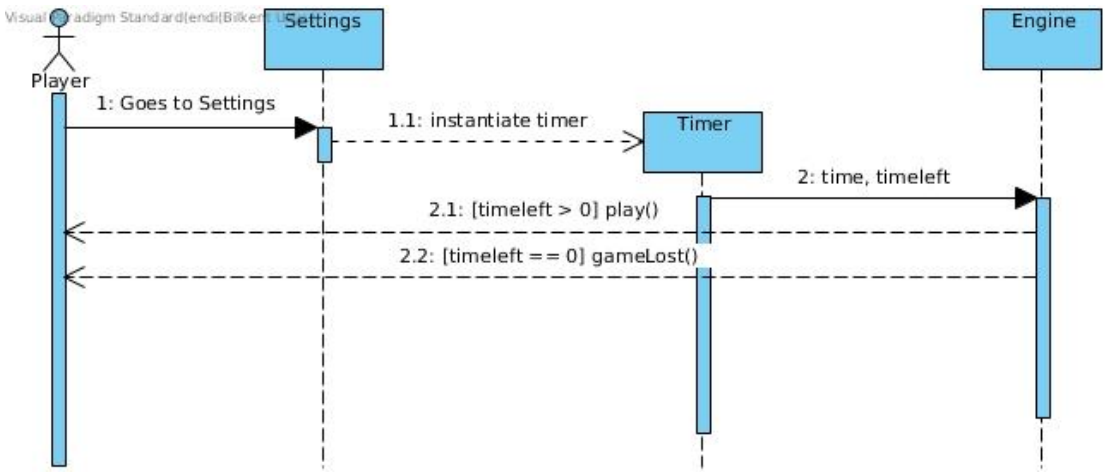


Figure 2: Sequence diagram for scenario #1

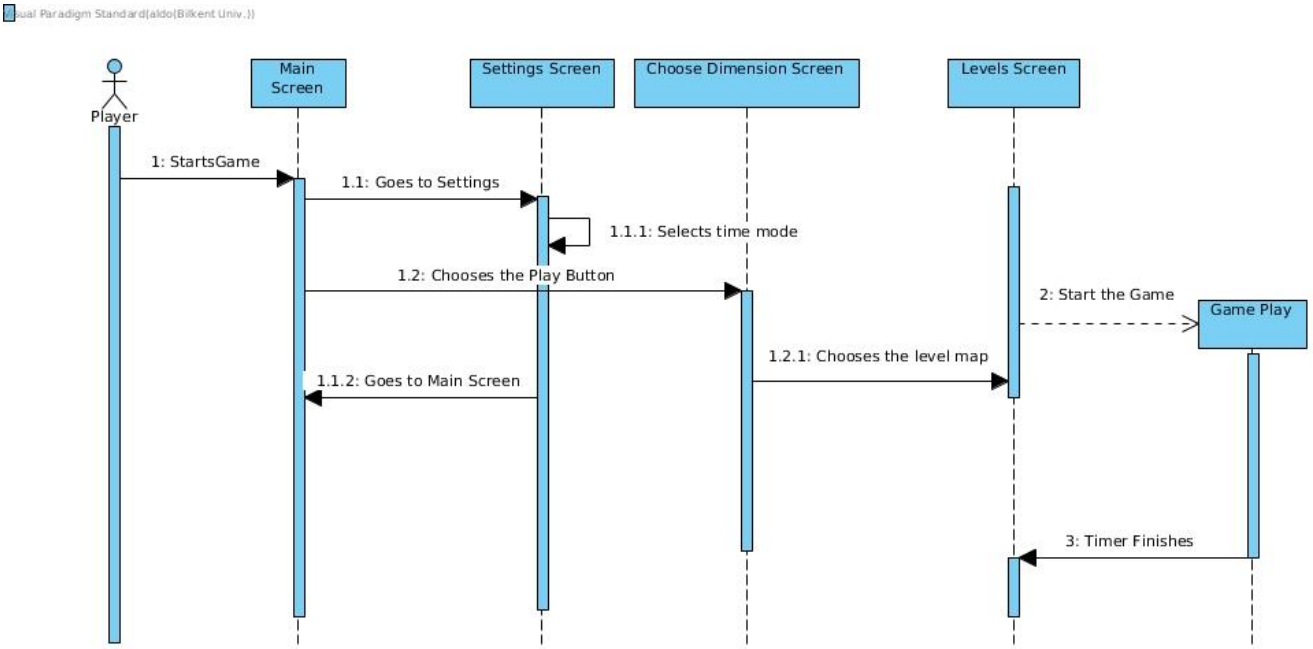


Figure 3: Sequence diagram for scenario #2

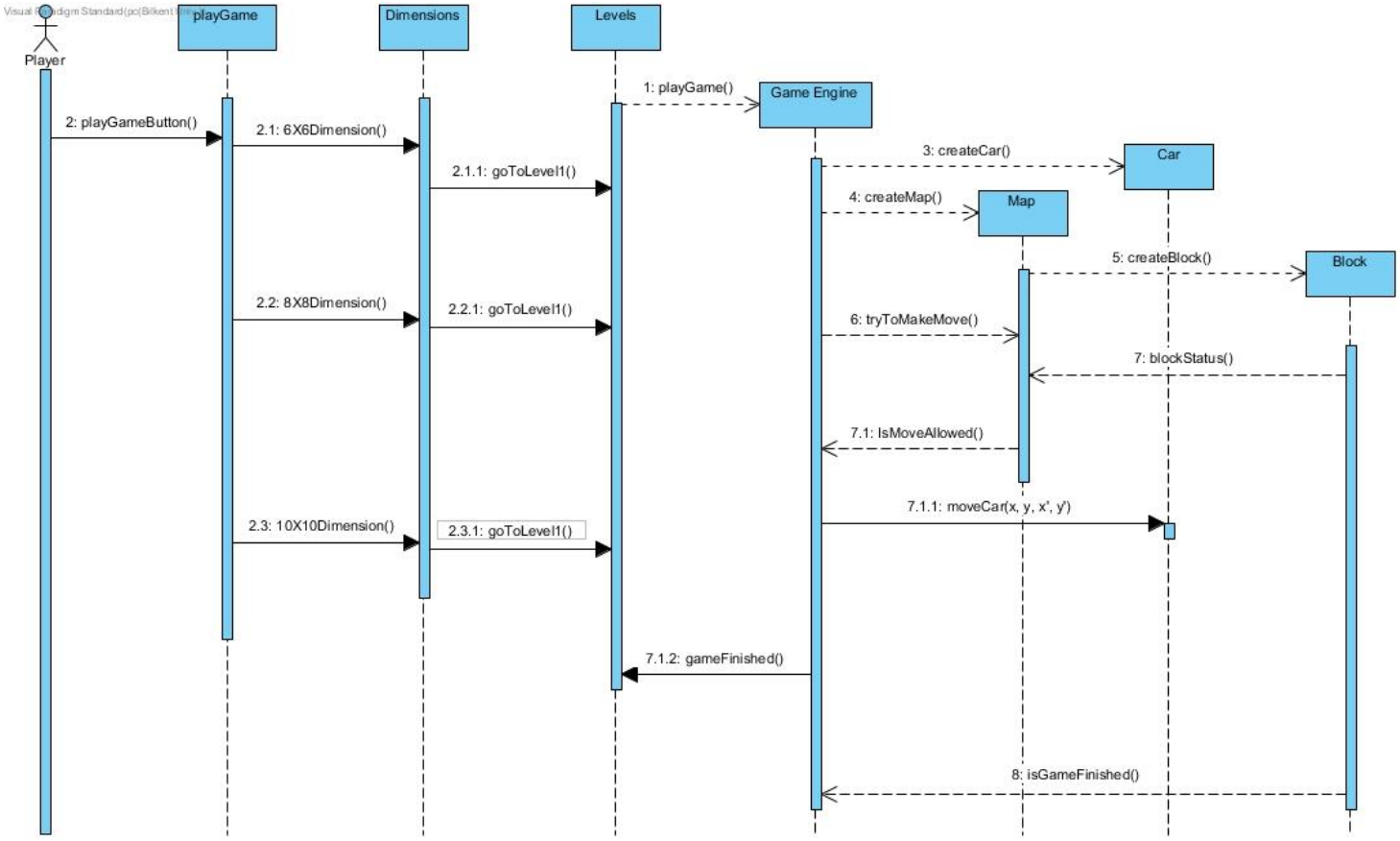


Figure 4: Sequence diagram for Scenario #4

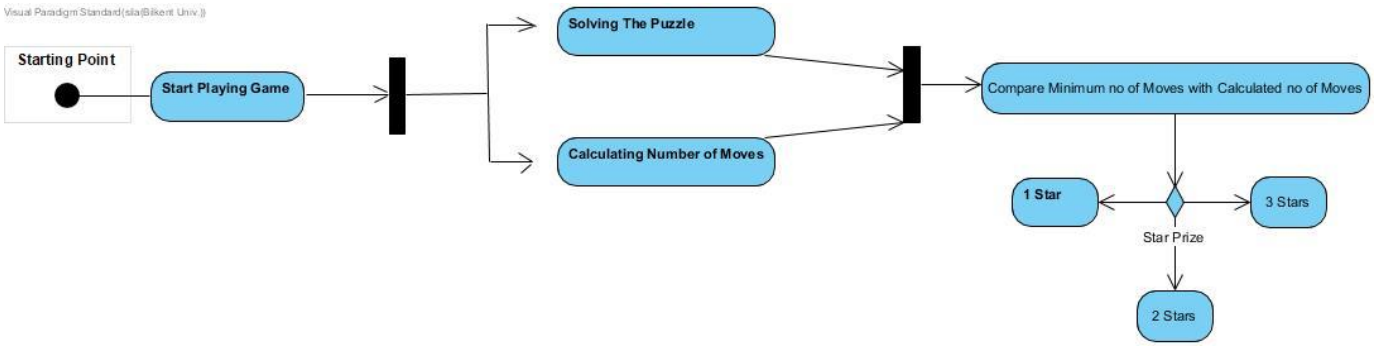
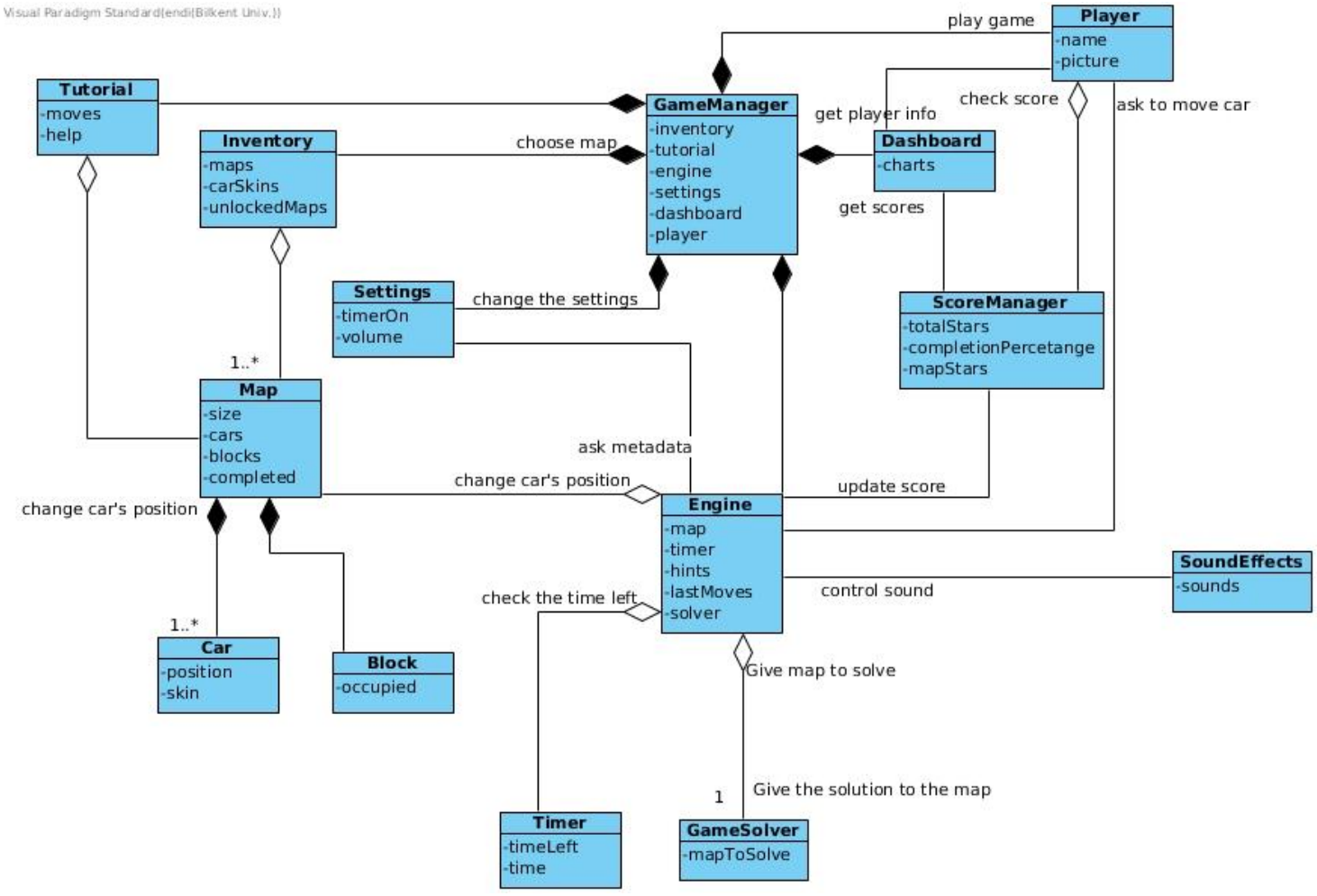


Figure 5: Activity diagram for Scenario #5

5.3 Object and Class Model



5.4 User-Interface (Mockups)

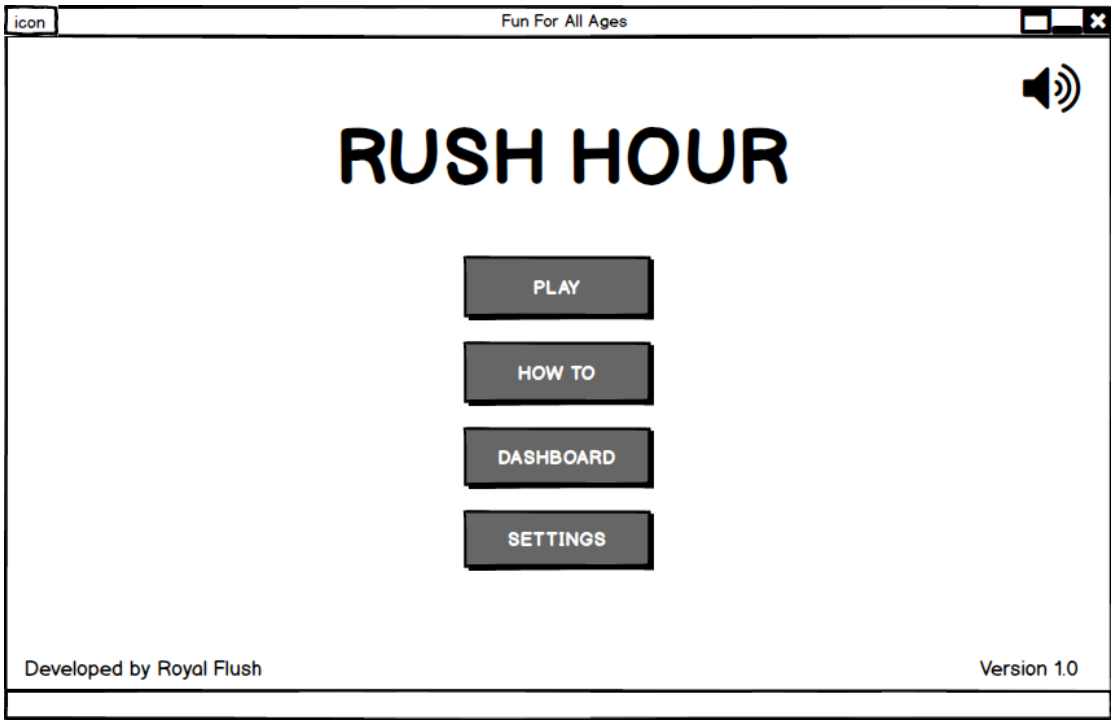


Figure 6: Mockup of the main Screen of the game

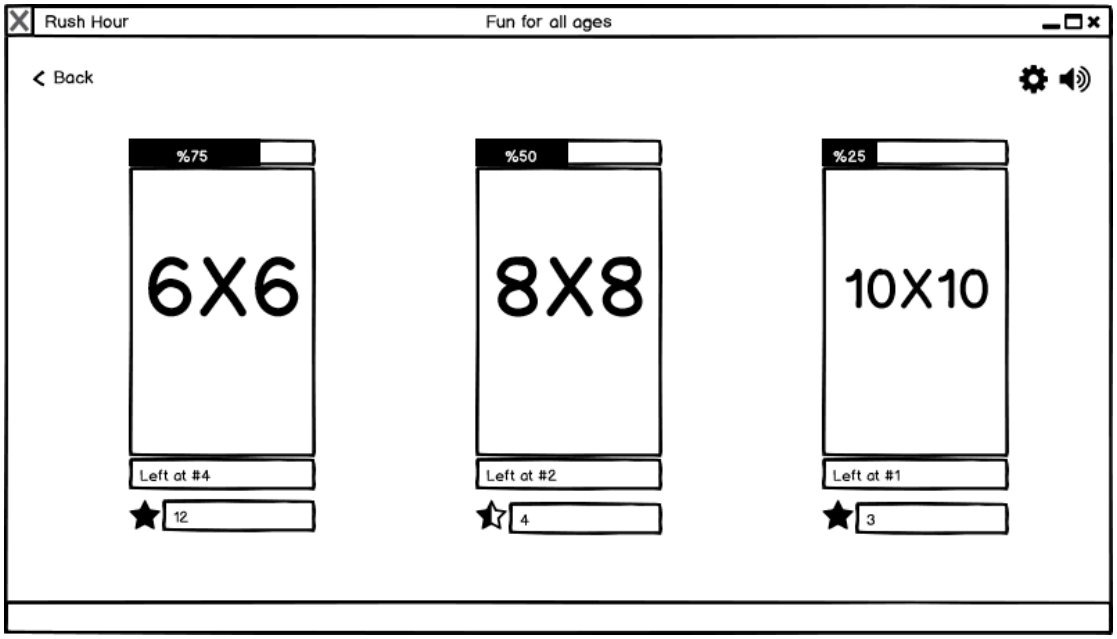


Figure 7: Mockup of the screen where player chooses dimension of the board in the game

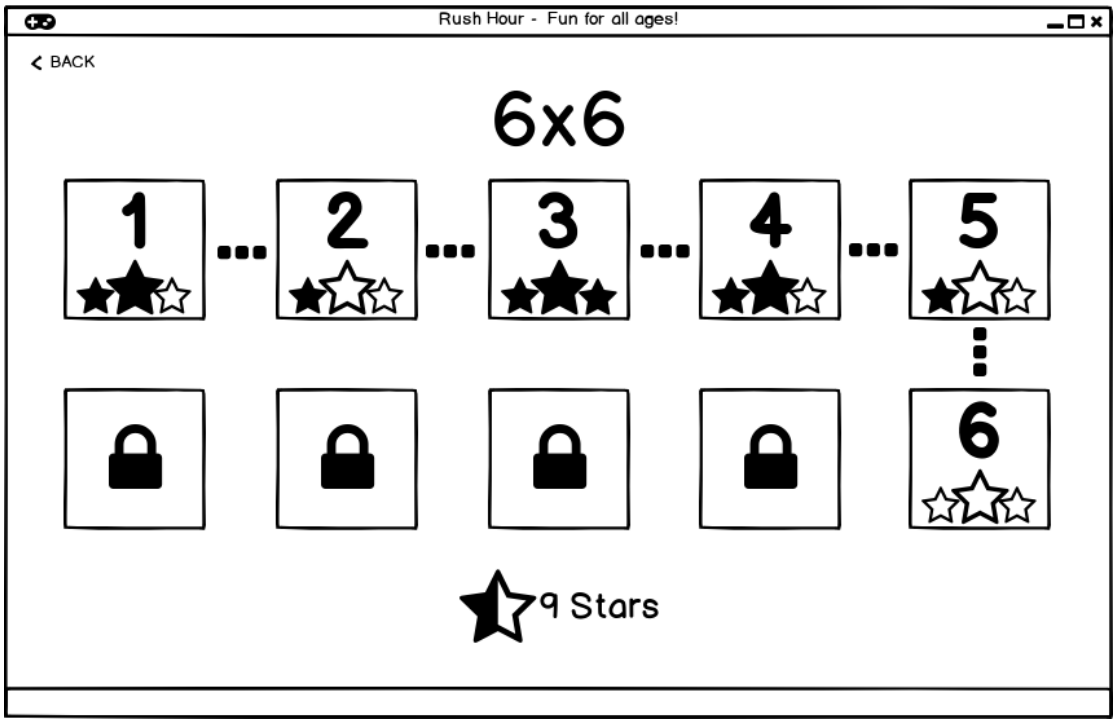


Figure 8: Mockup of the screen of unlocked levels of the chosen dimension

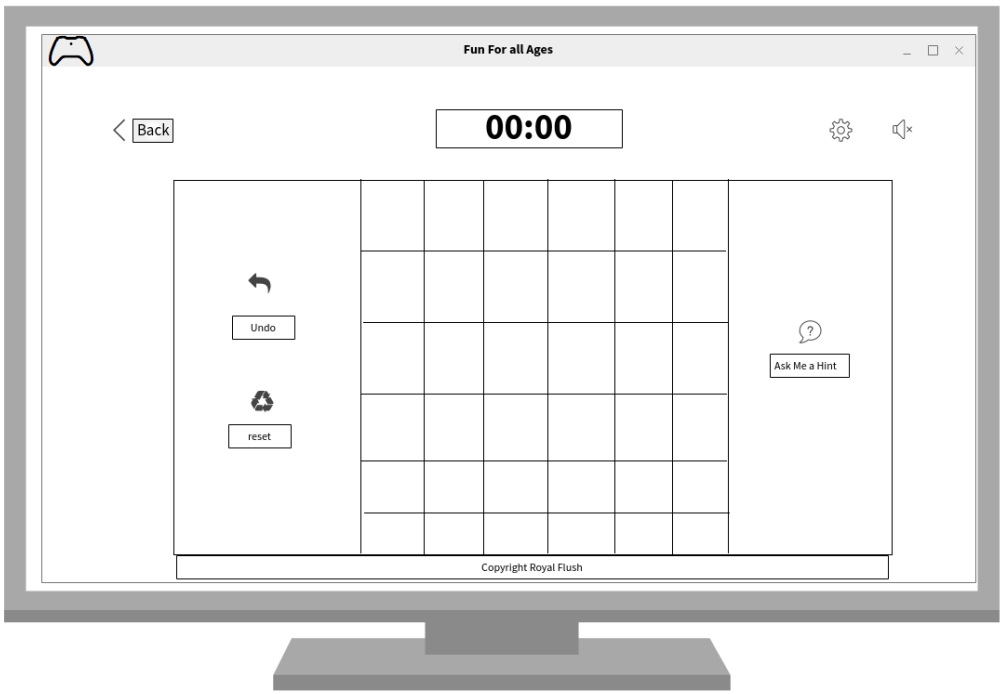


Figure 9: The mockup of the timed game screen

Figure 10: The mockup of the How-To screen

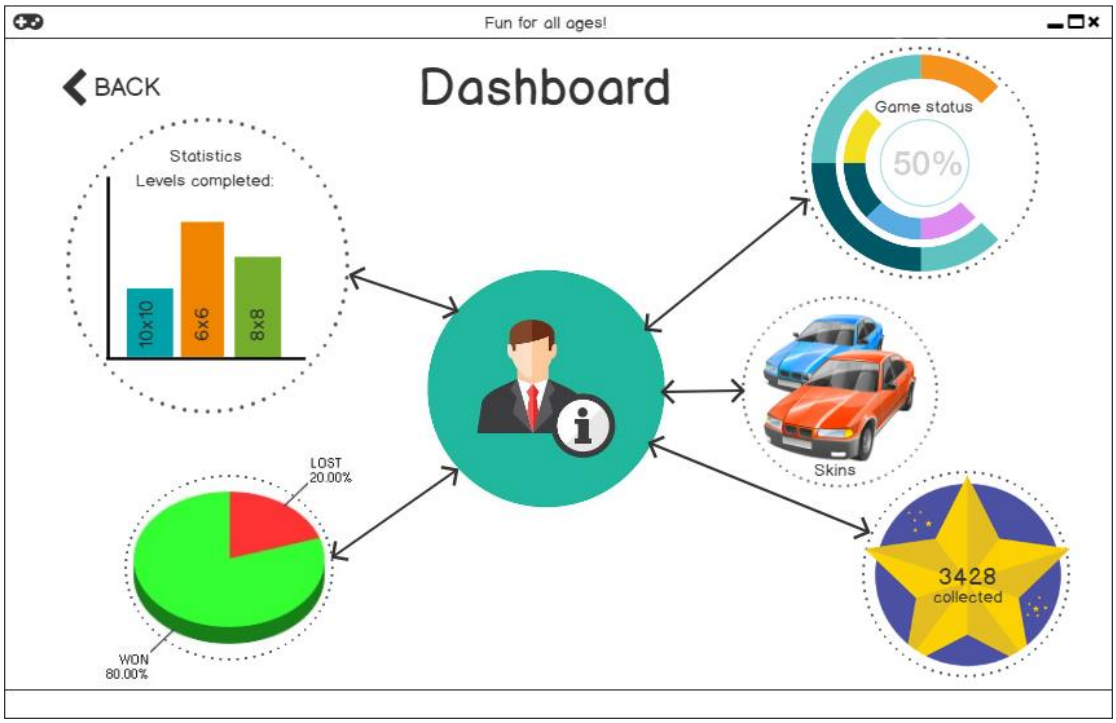


Figure 11: The mockup of the Dashboard screen

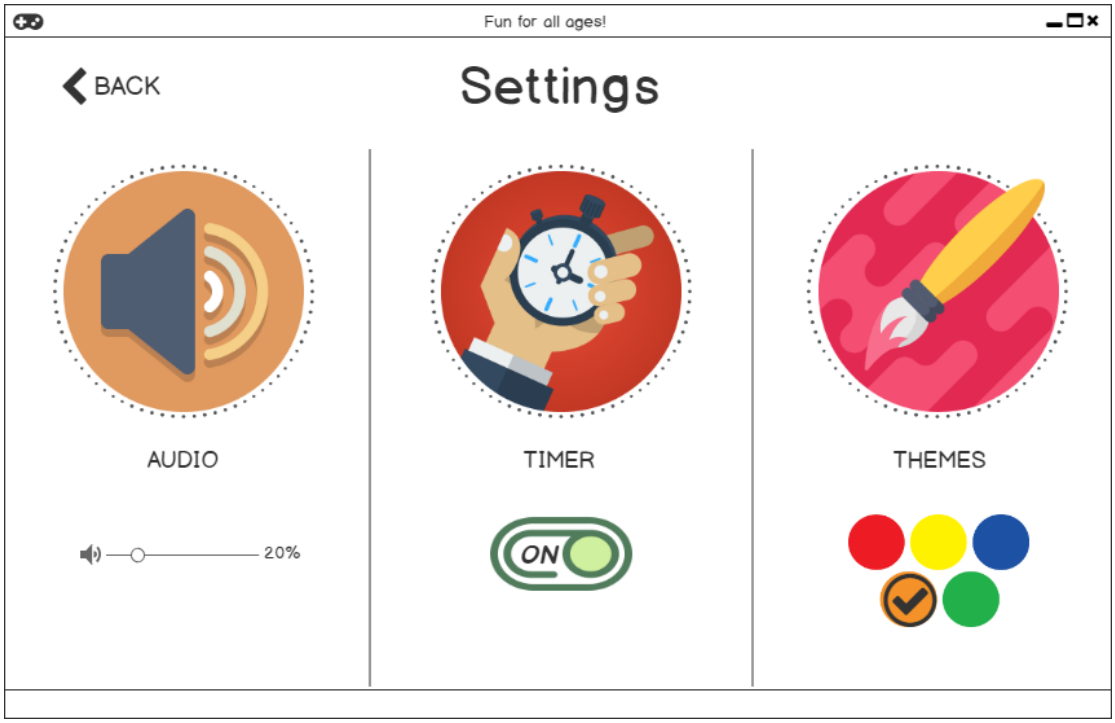


Figure 12: The mockup of the Settings screen

6 References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2] Contreras, M. (2018). The Evolution of ThinkFun's Rush Hour. [online] Info.thinkfun.com. Available at: http://info.thinkfun.com/stem-education/the-evolution-of-thinkfuns-rush-hour [Accessed 21 Oct. 2018].

[4] Us, A., Chatter, R., Educators, F., Blog, O. and Games, O. (2018). Rush Hour - ThinkFun's Play Online Game Library. [online] Thinkfun. Available at: https://www.thinkfun.com/play-online/rush-hour/ [Accessed 21 Oct. 2018].

[5] Crazygames.com. (2018). Rush Hour Online. [online] Available at: https://www.crazygames.com/game/rush-hour-online [Accessed 21 Oct. 2018].

[6] Cs.ulb.ac.be. (2018). Rush Hour Configurations. [online] Available at: http://cs.ulb.ac.be/~fservais/rushhour/index.php?window_size=20&offset=0 [Accessed 21 Oct. 2018].

[7] Cs.sjsu.edu. (2018). [online] Available at: http://www.cs.sjsu.edu/~stamp/cv/papers/rh.pdf [Accessed 21 Oct. 2018].