BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS319
Object Oriented Software Engineering

Final 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 the 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. All classes are now fully functional.

Game Entities

We are done with all the game objects that are being interacted with the user.

User Interface

On the View side of Rush Hour, we finished all the panels. Additionally, we have a semi-interactive tutorial for the user.

Later Development Process

All of our development goals are achieved.

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. MapSaver and MapExtractor classes are replaced with MapDao, PlayerExtractor and PlayerSaver classes are replaced with PlayerDao that implements the Data access object design pattern.

2. Data saving format has changed from our custom data storing format to JSON, to increase efficiency while working and error handling.

3. Input class implementation now uses KeyBindings instead of a KeyListener due to problems with KeyListener and Panel Focuses. Input class is now more reliable. Additionally, Input class now stores key and mouse pressed in a string array and string hashmap instead of boolean arrays. Because of this Input class can now store the state of the keys and mouse button whether they are pressed, released or clicked.

4. The game objects do not have direct access to associated images anymore. Instead, we use a “ThemeManager” class for loading images and sound files.

5. The tutorial management has been moved to a new panel called “Tutorial Panel” which makes it a lot easier to control the tutorials when compared to using each panel for its own tutorial. This also allowed us to create a semi interactable tutorial.

6. The resources such as images, sounds, levels are extracted as InputStream instances now on rather than File instances in the code in order to make them visible by .jar and .exe files.

7. UIFactory class was added to make the creation of needed JButtons and JLabel follow our standards and to make changes on all of our buttons and labels without modifying multiple classes. This increased the flexibility, reliability, and extendibility of our View classes, which helped us finish the project easier.

8. PowerUpManager was added to make implementing power-ups easy. It allowed us to play animations for power-ups, with unified code for both power-ups.

9. Vehicle controller class was modified to include another control option, Slide. Slide offers user users to control vehicles with only the mouse. Implementation of slide only needed modifications to the vehicle control class because of our modular structure.

10. Data configuration system was added to the program. All modifiable data is saved to a specific location which is the AppData folder of Windows in the user's computer. This system manages these data files and folders. If the data folders haven't been created before, they get set up. Also, DataErrorHandler class was added to handle any kind of missing or damaged data files that are checked by DataConfiguration or DAO classes.

3. Lessons Learned

- We have learned that the most important aspect of both the analysis and the design steps are 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 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 and use abstraction while designing the system, otherwise details can create a lot of difficulty at the start of the design process.
- Using design patterns is really beneficial in maintaining a powerful and easily editable system. It helps us and future developers to work in a 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 maintaining a powerful communication infrastructure because great communication is the key to solving problems and making progress. Each member should be able to attend and express themselves clearly in regularly conducted team meetings.

4. Build Instructions

The build instructions are specific for the Windows operating system and IntelliJ IDEA.

1. Download the latest version of IntelliJ IDEA.
2. Create a project from existing source.
3. Find and select the src folder as the project folder.
4. After the project is opened, press the Project Structure button on the right above of the screen (or Ctrl + Alt + Shift + S).
5. Select Libraries from Project Settings and press ‘+’ button on the left above.
6. Select Java and then the *gson-2.8.5.jar* file which is in the lib folder of the project.
7. Then, accept all the default options and press the Ok button until all the panels disappear.

8. The program is ready to be built and run. Press the Build button (or Ctrl + F9) to build and press the Run button (or Shift + F10) to run the project.

5. Work Allocation

- **Ahmet Ayrancioglu:**
  - Drew or found all visual assets for the game.
  - Created the final demo video for the game with the input of Kaan and Deniz.
  - Worked on all of the reports, the final demo and presentation preparation.
  - Designed and coded the game engine, and the controller base class and the design for the game loop and implemented the singleton pattern for controller classes.
  - Designed and coded the Input class.
  - Designed and coded power-up manager.
  - Designed the relationship between game objects and transforms
  - Designed GameManager, MapController and PlayerManager classes with Kaan and Deniz.
  - Designed and coded the UIFactory.
  - Designed and implemented the LevelButton class to make implementing levels panel much easier.
  - Designed and implemented the first iteration of the vehicle controller which worked with WASD keys with Deniz and Kaan. I later added the ability to control the vehicle with the mouse.
  - Created the Credits panel, Help panel, Main Menu Panel EndOfLevel panel, Create Player popup panel, some parts of Game panel and Settings panel.
  - Added new features such as blacking out some parts of the map and animations for power-ups in the inner game panel.

- **Deniz Dalkilic:**

  - Designed the Theme management system.
  - Designed the Tutorial system with assets drawn by Ahmet.
  - Designed the Sound manager system.
  - Designed the GameManager, MapController, and PlayerManager systems with Kaan and Ahmet.
  - Developed the “draw” methods of vehicles for GamePanel with Ahmet.
  - Developed the page algorithms for different panels with Ahmet.
  - Developed the vehicle control mechanics with Ahmet and Kaan.
  - Developed auto sliding animations and auto finish feature.
  - Worked on map generation/conversion.
  - Worked on the creation of model classes with Ahmet.

    - Worked on map occupation algorithms.
    - Implemented panels like LevelSelection Panel, Tutorial Panel, Change Player Panel.
    - Developed Settings Panel and GamePanel with Ahmet.
    - Worked on the final demo, presentation, and all reports.

- **Kaan Gonc:**
    - Designed and implemented GameManager, VehicleController and MapController classes with Deniz and Ahmet.
    - Designed and implemented Player class, then LevelInformation and Settings classes which are attributes of Player class.
    - After the Player class had been done (without later features), designed and implemented PlayerManager class with the help of Deniz and Ahmet.
    - Provided the connections between PlayerManager and other necessary controllers and necessary panels.
    - After being sure of PlayerManager works correctly, designed Data Access Object design pattern for the Player and Map objects in order to save all progress and preferences of the active player.
    - Implemented the functions of some features such as using powerups, unlocking themes, unlocking levels with their pre and post-conditions.
    - At first, implemented my own parser to extract or save data to the user's file system with text files. Later on, decided to implement JSON data storing format to make working and error handling procedures easier with the suggestion of Ahmet.
    - Designed and implemented DataConfiguration and DataErrorHandler systems.
    - Regularly tested the game to check for bugs, fixed or reported the found ones.
    - Did the parts of reports and presentations which I responsible for.
- **Ali Yümsel:**
    - Designed and implemented the GuiPanelManager.
    - Designed and implemented the initial state of the panels.
    - Implemented the code that draws images to the screen, instead of drawing blocks.
    - Generally worked on the view package of the project.
    - Designed some of the levels.
    - Helped writing the Javadoc.
    - Regularly tested the game to check for bugs, fixed or reported the found ones.
    - Did the parts of reports and presentations which I responsible for.
- **Sina Sahan:**
    - Generally worked on the view package of the project.
    - Designed some of the levels.
    - Wrote most of the Javadoc.

- Regularly tested the game to check for bugs, fixed or reported the found ones.
- Did the parts of reports and presentations which I responsible for.
- Worked on the first design of the User Interface.
- Worked on presentation.
- Worked mostly on the documents.
- Helped to design initial states of panels with Ali.

6. User Guide

RUSH HOUR

USER GUIDE

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 backward out of the way for the player car to move towards the exit.

2. INSTALLATION

2.1. System Requirements

- You need to have the Standard JAVA Runtime Environment installed on your computer. (Version 8)
  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.

2.2. Installing the Game

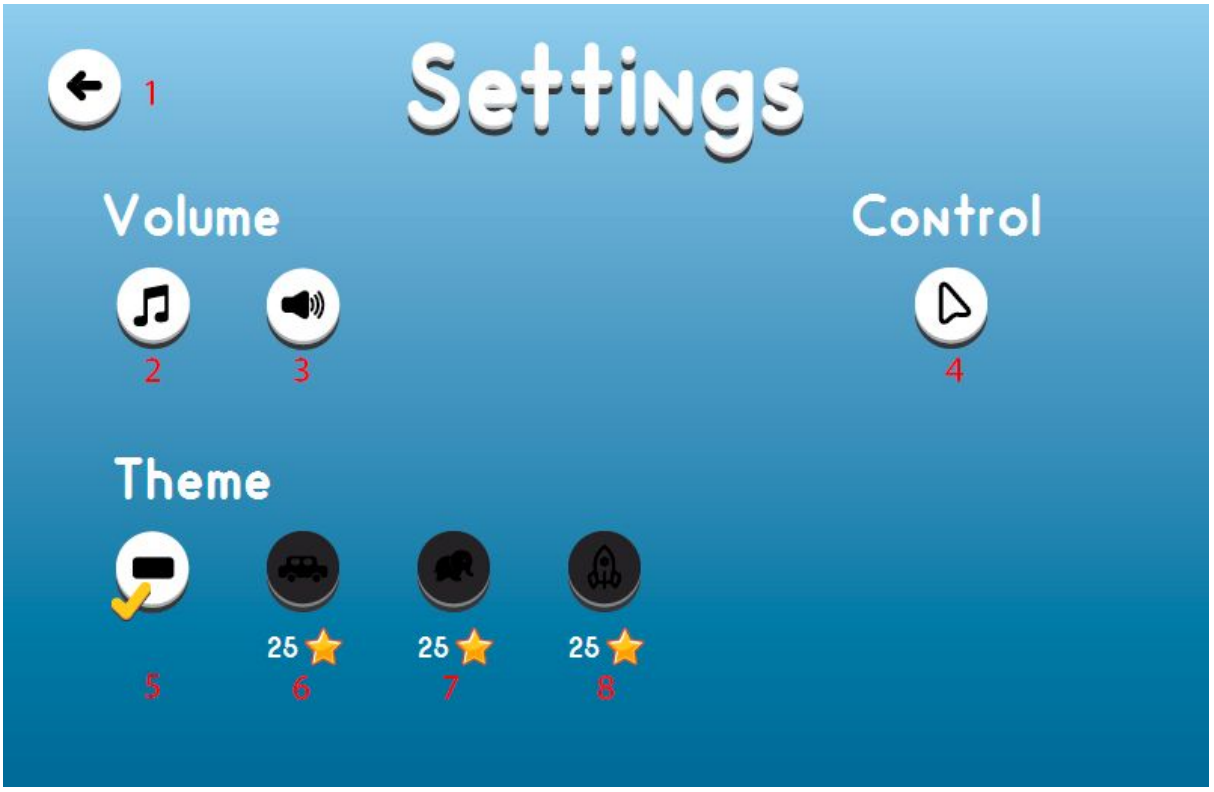
After downloading the RushHour.exe file, just double click on the file and the game will run automatically. There is no installation process.

3. GAME SCREENS

3.1. MAIN MENU

1. **Help Button:** Shows information about how to play the game.
2. **Star Amount:** Shows the number of stars that the user currently has.
3. **Change Player:** Opens the Change Player screen.
4. **Play Game:** Starts the game from the last unlocked level.
5. **Credits:** Opens the Credits screen.
6. **Levels:** Opens the Levels screen.
7. **Settings:** Opens the Settings screen.

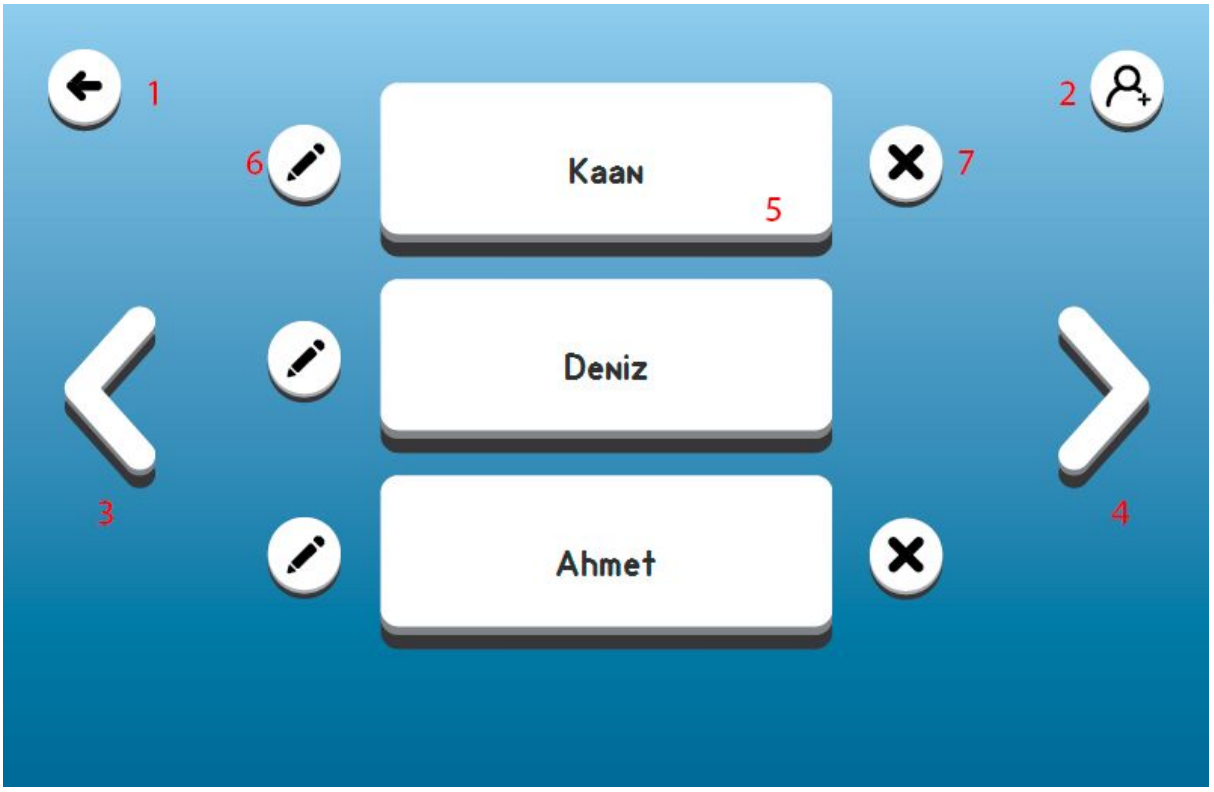
3.2. SETTINGS



1. **Back Button:** Button to go back to previous screen. (Either Main Menu or Game Screen)
2. **Music Button:** Allows player to toggle background music on/off.
3. **SFX Button:** Allows player to toggle sound effects on/off.
4. **Control Preference:** Allows player to switch between slide and keyboard control schemes. In slide control scheme, the player can click on a vehicle and drag the mouse to the direction that he/she wants the vehicle go. In keyboard control scheme, the player can click on a vehicle to select it and use WASD keys to move the vehicle. (W - Up, S - Down, A - Left, D - Right)
5. **Minimalistic Theme:** Toggles the minimalist - block theme. This theme is unlocked from the start.
6. **Classic Theme:** Toggles the classic Rush Hour - Traffic theme. Unlocks when the user has 25 stars.
7. **Safari Theme:** Toggles the Safari theme. Unlocks when the user has 25 stars.
8. **Space Theme:** Toggles the Space theme. Unlocks when the user has 25 stars.

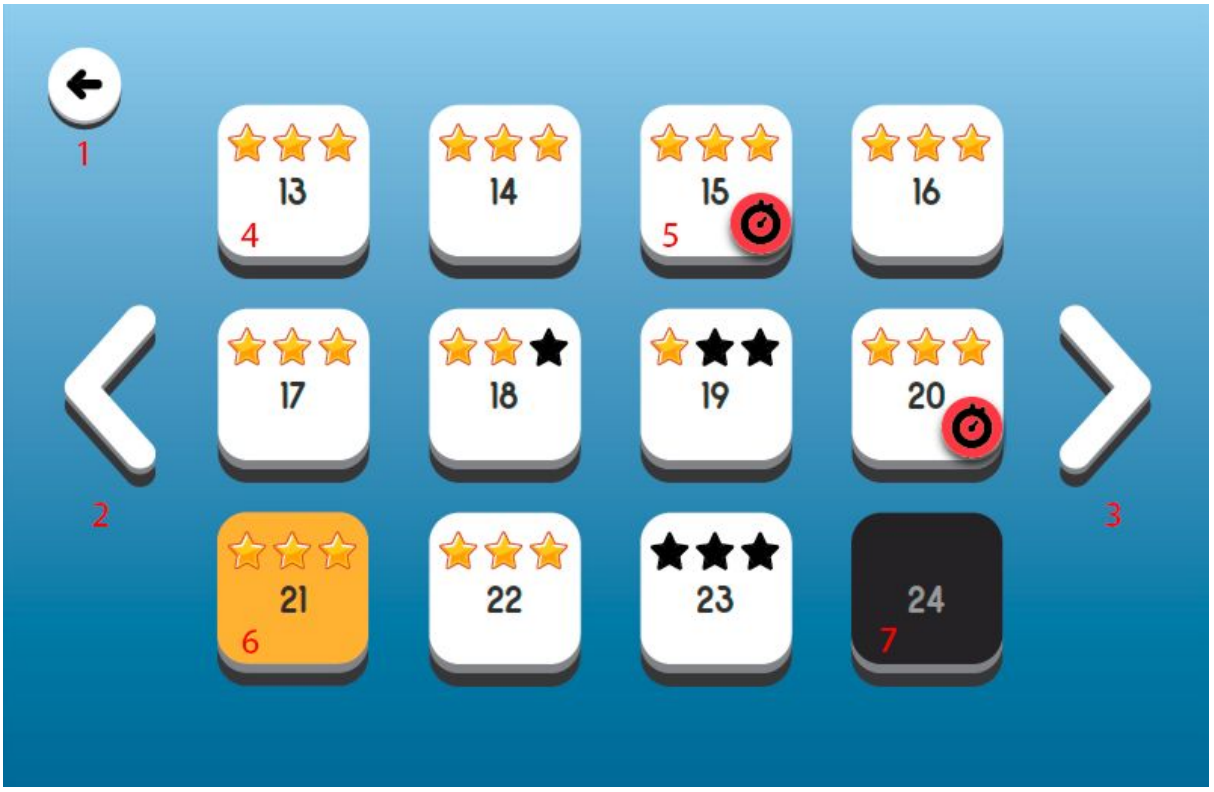
Note: When a new theme is unlocked, the number of required stars for unlocking other themes increase by 25**.**

3.3. CHANGE PLAYER



1. **Back Button:** Button to go back to previous screen. (Main Menu)
2. **Add Player:** Button to add a player. When pressed, A pop-up that asks for the player name will appear.
3. **Previous Page:** Button to go to the previous page.
4. **Next Page:** Button to go to the next page.
5. **Player Button:** Button with the player's name on it. When clicked, the player will be selected, and the game will return to the main menu.
6. **Edit Player:** Button to edit a player's name. When pressed, a pop-up that asks for the new name for the player will appear.
7. **Delete Player:** Button to delete a player. This button will not show up for the currently active player.

3.4. LEVELS

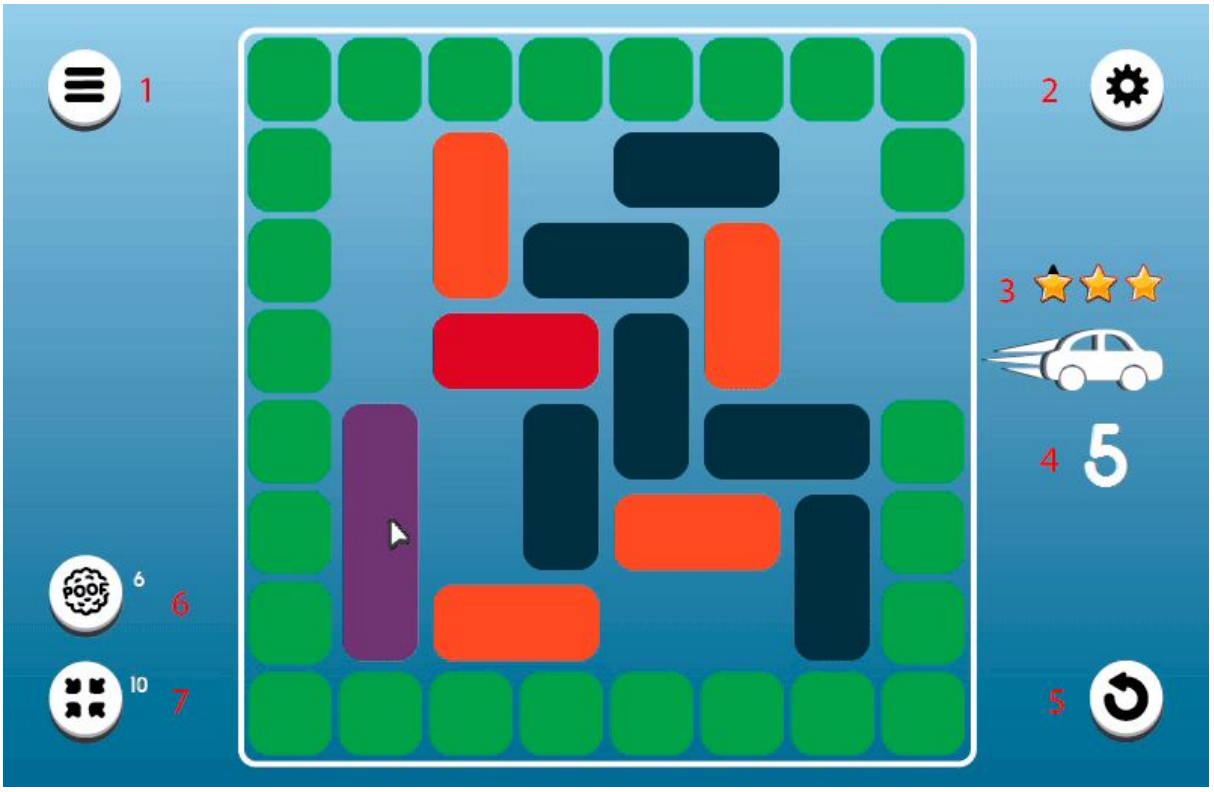


1. **Back Button:** Button to go back to previous screen. (Main Menu)
2. **Previous Page:** Button to go to the previous page.
3. **Next Page:** Button to go to the next page.
4. **Level Button:** Button to play the level. It displays the level number and the amount of stars earned from that level.
5. **Special Level:** The button for a special level will also have a small timer icon at the bottom right corner.
6. **In-Progress Level:** In-progress(Not finished) levels will appear orange.
7. **Locked Level:** Locked levels will appear dark-gray, won't display star amounts, and will not be clickable.

3.5. CREDITS

1. **Back Button:** Button to go back to previous screen. (Main Menu)
2. **Developers:** The names of the developers who created this game.

3.6. GAME SCREEN



1. **Main Menu:** Button to go back to the Main Menu.
2. **Settings:** Button to go to the Settings screen.
3. **Stars:** Number of stars currently. It will drop as the player makes moves.
4. **Number Of Moves:** The number of moves that the player made currently.
5. **Restart:** Button to restart the level. Use with caution because the power-ups that you used will lose effect.
6. **Poof:** Button to use the “Poof” power-up. This power-up will allow you to destroy one of the 1x1 obstacles.
7. **Shrink:** Button to use the “Shrink” power-up. This power-up will allow you to shrink one of the 3x1 trucks into a 2x1 car.

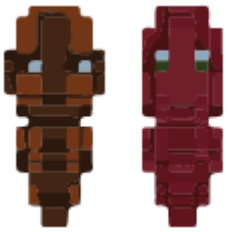
4. GAME OBJECTS

4.1. Car

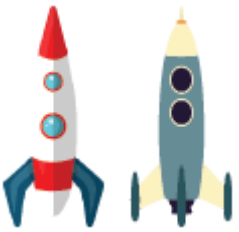
The car is a 2x1 blocking vehicle that the player can slide backwards or forwards to clear the path for the player car.

Minimalist

Classic



Safari

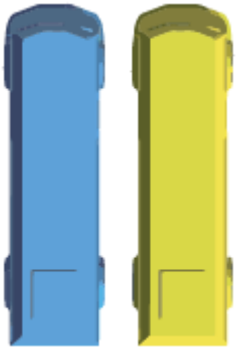


Space

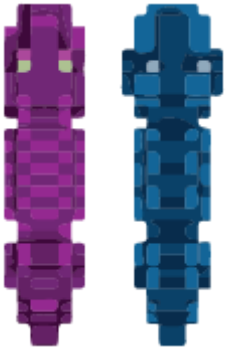
4.2. Truck

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

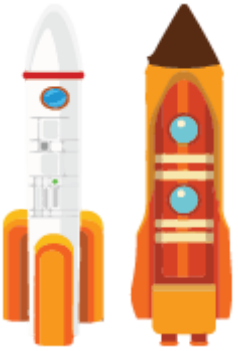
Minimalist



Classic



Safari



Space

4.3. 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.

Minimalist

Classic

Safari

Space

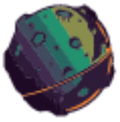
4.4. Obstacle

The obstacle is a 1x1 immovable block that also prevents other vehicles to move through a certain space on the 6x6 map. However the obstacle can be destroyed using the ‘Destroy Obstacle’ power-up.

Minimalist

Classic

Safari



Space

5. HOW TO PLAY

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 or left and right- until the path is clear for the red car to escape. Vehicles can only slide forward and backwards, not sideways. You are allowed to choose your own controlling preference. You can play with the mouse and drag objects to target cells or you can use the keyboard (W, A, S, D keys) to move them.

Bonus Level: Just a regular level with a time challenge. So, try to beat the level before the time ends. You will be rewarded by a couple of power-ups.

Poof Obstacles: If you need help, you can activate a poof power up and it will show you the obstacles that are available for power-up usage. By the help of this power-up, you can change an occupied cell into an empty cell.

Shrink Long Objects: If you need help, you can activate a shrink power up and it will show you the game objects that are available for power-up usage. By the help of this power-up, you can transform a 3x1 object into a 2x1 object.

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