Analysis Report

CS 319-1

Group 1A, Team Java++

Project: Road Block

Members:

Denizhan Soydaş, 21502231

Deniz Ufuk Düzgün, 21402164

Kaan Atakan Öztürk, 21302164

Cavid Gayıblı, 21500636

Selim Can Gülsever, 21601033

Serkan Delil, 21501289

Contents Table

1. Introduction

Road Block is a game that we are planning to develop. The main objectives and game play of it are as follows: There is a thief in the game that is trying to escape from the police. The user's main objective is to block the paths which the thief may use in order to get away. The game is played on a map that offers 36(6x6) squares. This map offers various spaces in which the user can place differently shaped blocks and the puzzle in it is to place all the blocks where they have to be. These blocks are offered to the user at a seperate place from the game board and they can be chosen and placed in a random order as long as they are in the places where they have to be in the end. User is also able to, and usually should, rotate these blocks in order to find the solution for that level. The game consists of different levels which increase in difficulty after each one is completed.

Additionally, we are planning to develop the traditional game a few steps further and make some adjustments to it. For instance, since it is a single player game, we wanted to add some self competition and we will be using time limitation for some levels. Again for enhancing the same feeling , we will get involved with score tracking. The player will obtain scores based on different parameters such as time, count of moves etc. Also we will add one element called "park" in order to make some levels a bit harder. Detailed information about this feature can be found in Overview section.

Our aim is to implement this game as a desktop application in Java programming language while abiding by the principles of Object Oriented Programming. This report offers game overview, our in-game objects and structural information of the application. In this report it is also possible to find functional, non-functional requirements along with the class, use case, state and activity diagrams.

2. Overview

2.1. Gameplay

The game is played using only the mouse. Using the mouse, the user is both able to rotate and move the blocks to the places on the map. Simple left click operations will be sufficient to choose the desired rotation of blocks and place them to the desired spots.

2.2. Map

The game is played on a 2D 6x6 map on which involves the components of the game. The puzzle of our game is the different combinational layouts of the game components(one by one introduced from 2.3 to 3) on the map and it is the main environment on which the gameplay is conducted.

2.3. The Thief

The thief is the villain of the game and user's main objective is to prevent the thief from escaping the map by manipulating and using different Police Car Blocks.

2.4. Police Car Blocks

These components are the main pieces of the game which are in user's control to place in the empty areas on the map in order to solve that level's puzzle. The user is able to(and usually must) rotate these blocks clockwise or counter-clockwise to fit them to the right places.

2.5. Buildings

These components are placed in different orders on the map and they are intact components, meaning the user is not able to move or manipulate them whatsoever. Both the Police Cars and The Thief is not able to move through these blocks. In other words, they are used to borderline the playing area of that specific level.

2.6. Park

This is one of the additions that we have added to the traditional game. It serves as a easy getaway path to the thief and is planned to make certain levels more difficult for the user.

2.7. The Clock

The other additional component we will be adding to the traditional game is "The Clock". The Clock serves as a self competition component for the user to earn score points accordingly. Meaning: The faster the user completes the level the more points he/she will earn.

2.8. The Move Counter

For each level of the game, we will be implementing a move counter. This move counter is designed just like "The clock" to enhance the competitive spirit throughout the game. The less moves the user makes the more score points he/she will gain.

3. Functional requirements

3.1. Play Game

3.1.1. Select Difficulty

The game will be consist of 3 difficulty levels : Easy, Medium, Hard. Each of these difficulty levels offer 3 levels to the user : Level 1, Level 2, Level 3.

3.2. Settings

The user will be able set their preferences for the in-game effects voice level and music volume.

3.3. About Us

This section will allow user to view information about the developers that participated in creation of the game and also about the references that were used to obtain useful information during the development of the game.

3.4. Help

Through this option, it is possible to learn about the objective of the game and how to play it.

3.5. Exit Game

Using this option, the user is able to exit and close the game.

4. Non-functional Requirements

4.1. Game Performance

Since our game does not require high PC specifications, any average computer with a Windows operating system will be enough to run our game on high frames per second(FPS).

4.2. User Friendly

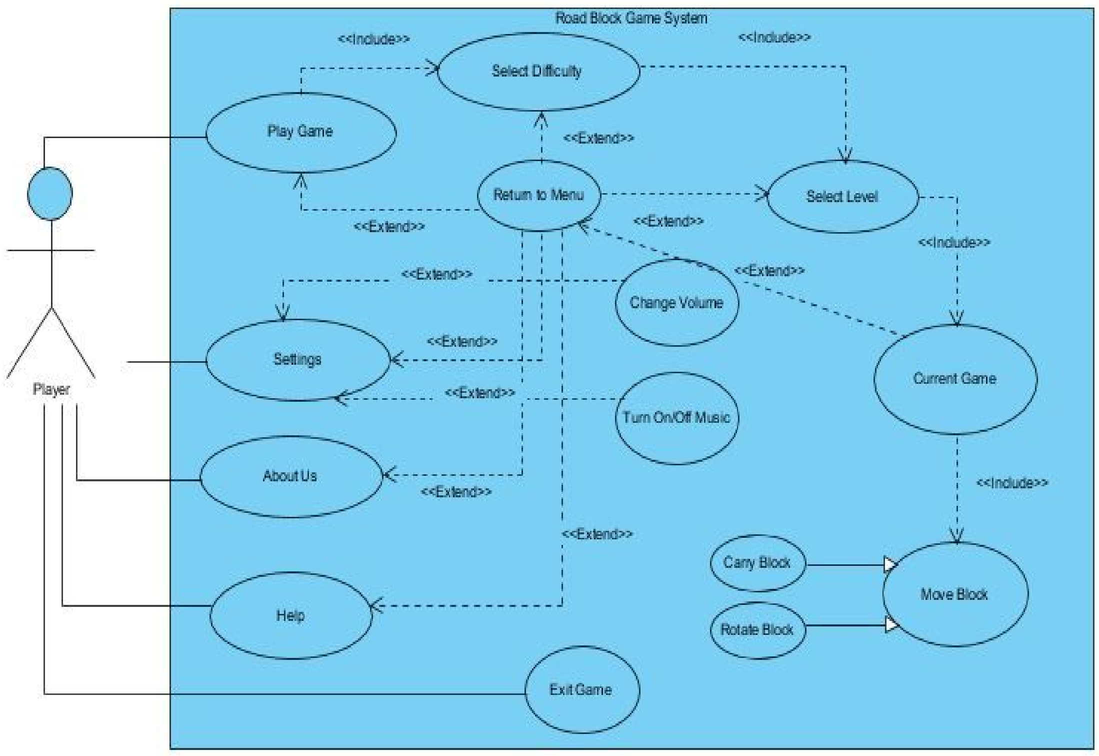
Our game's traditional version is recommended for 7 years or older old users. Therefore our application's features will be as simple as possible to understand and use. We will have children users along with the adults to test our game in order to see whether they can comprehend the UI’s difficulty or not.

4.3. Response Time

The game should be able to respond to the user input almost instantly. Therefore, our images have to update as in the instant that user provides changes. By this way, smoother gameplay will be obtained.

5. System models

5.1. Use case model



Use Case #1

Use case name: PlayGame

Participating actor/s: Player

Entry condition: Player should open the main menu by clicking the “PLAY” button.

Exit condition:

1. Player has to pass the current level OR,

2. Player has to exit the game by clicking the exit button.

Main Flow of Events:

1. Player clicks the "Play Game" Button

2. System comes up with selection page that asks if player wants to select the difficulty or return back to menu.

3. Player plays the game and returns to main menu at the time player wants.

Alternative Flow of Event:

1. When the player wins and passes the current level, the player’s score will be shown as stars (out of 5).

2. Player can exit the game any time.

Use Case #2

Use case name: Settings

Participating actor/s: Player

Entry condition: Player should open the settings by clicking the “Settings” button.

Exit condition:

1. Player has to click the “Return Menu” button.

Main Flow of Events:

1. Player clicks the "Settings" Button

2. System comes up with selection page that asks if player wants to change volume, turn on/off the music or return back to menu.

3. After player setting up his/her preferences he/she can return back to menu.

Use Case #3

Use case name: About Us

Participating actors: Player

Entry condition: Player has to open main Menu and has to select “About us ” button

Exit condition

1.Player has to click Return Menu option to exit the About Us Case

Main Flow of Events:

1.Player clicks the "About Us" Button

2.System comes up with the promoting the group name and names of the group members

3.Player returns to the main menu of the game.

Use Case #4

Use case name: ExitGame

Participating actors: Player

Entry condition: Player is on main menu.

Exit condition: Program is closed.

Main Flow of Events:

1.Player clicks "Exit" button from main menu.

2.Program is closed.

Use Case #5

Use case name: Help

Participating actors: Player

Entry condition: Player firstly opens the game and goes to Help Button.

Exit condition: Player returns to main menu.

Main Flow of Events:

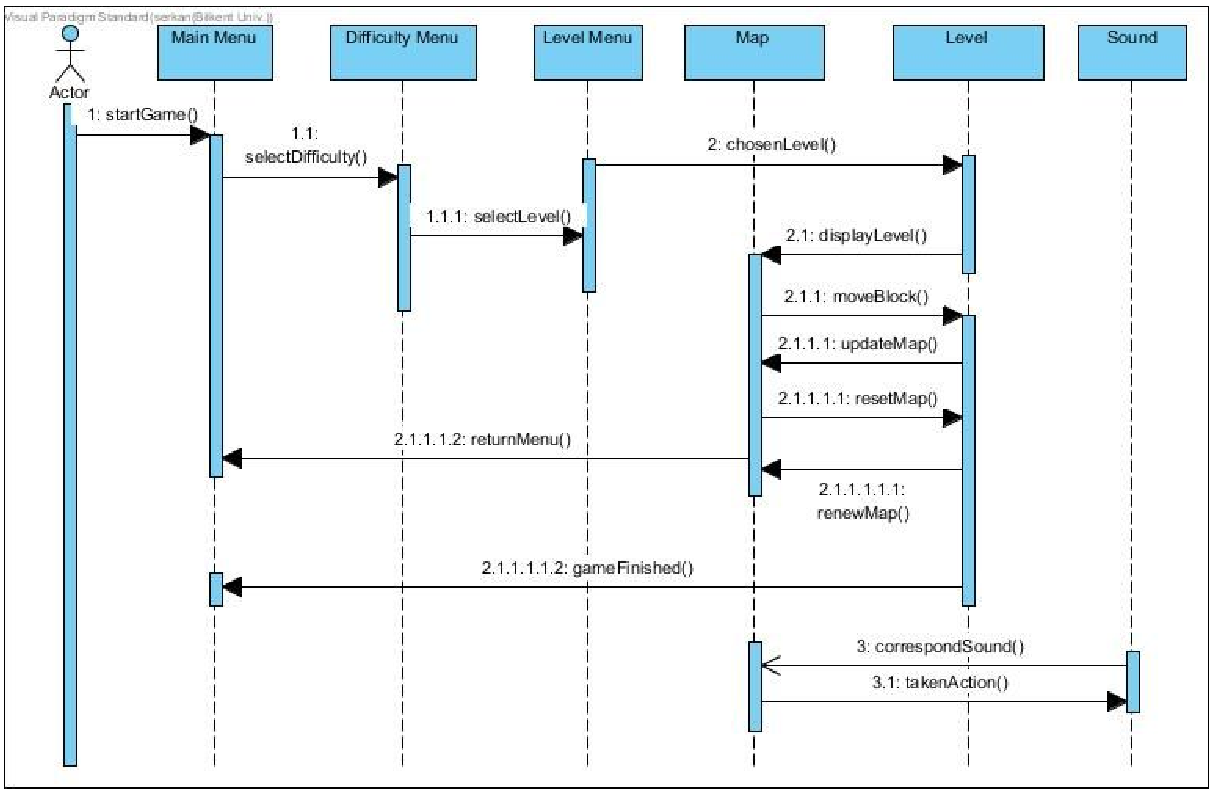
1.“Help” button is chosen by Player.

2.Player gets informed about the instructions of the game.

3.Player returns to the Option menu.

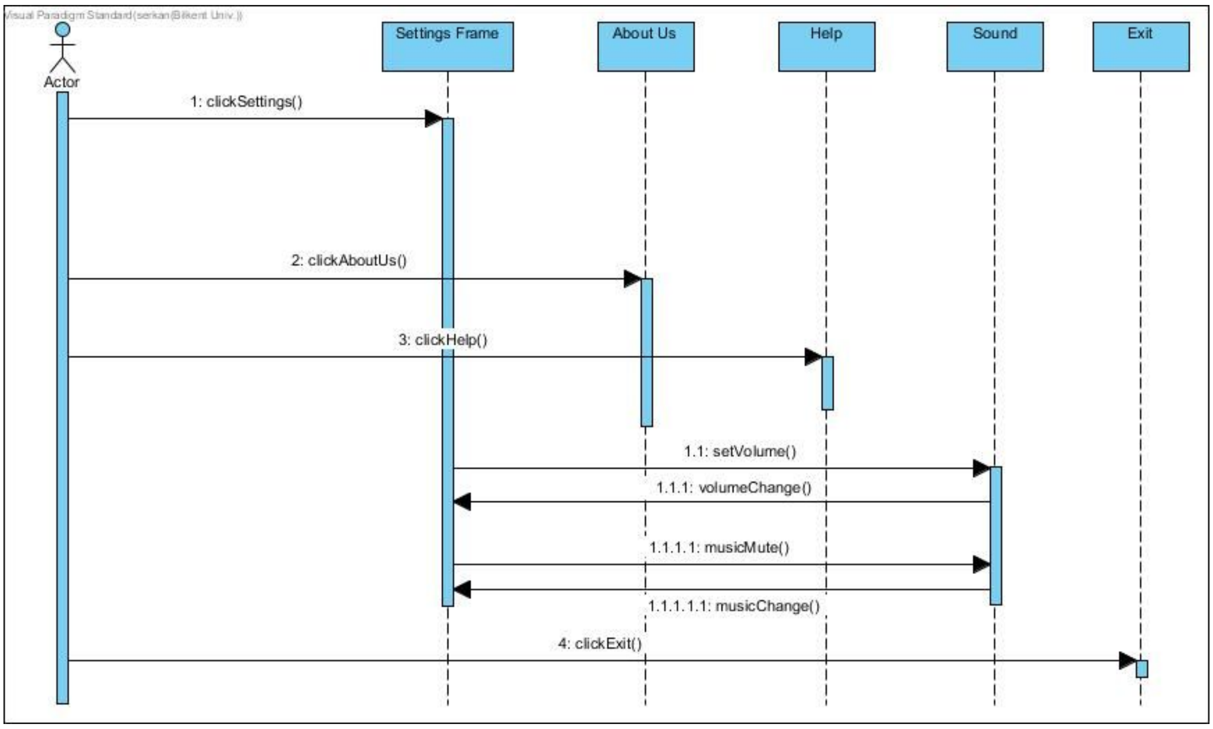
5.2. Dynamic models

5.2.1. Sequence Diagram



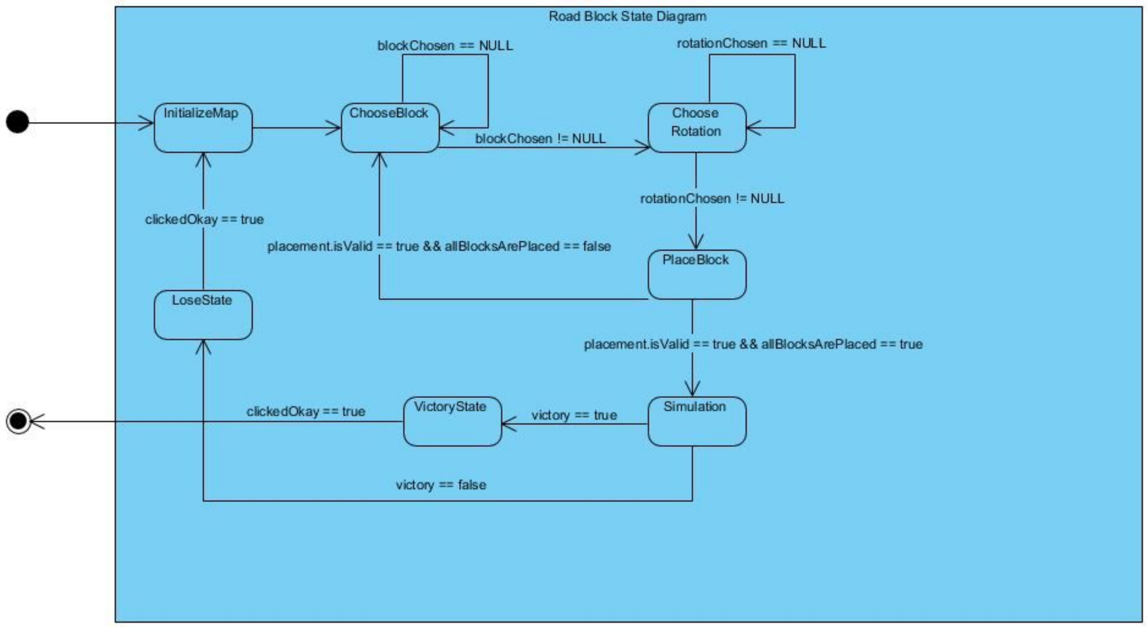
In this diagram, basically how the application works through classes and how they interact between them. First, actors interact with main menu when they start the game. Then, they select the difficulty of the game, after they determine which difficulty they will play, they need to choose level in that difficulty, and this decides how the interface will be created, namely level and map. All these processes are controlled by Level and Map. Moreover, it will interact with the other classes when it is needed.

Level first creates the map and entities in terms of input information, difficulty and level that user want to play, lastly loads the sounds. Manager recursively takes the inputs from mouse and according to inputs it will interact with entities. After putting entities, map will be updated and when all empty spaces is filled by entities, the game is finished and information will be sent to screen.



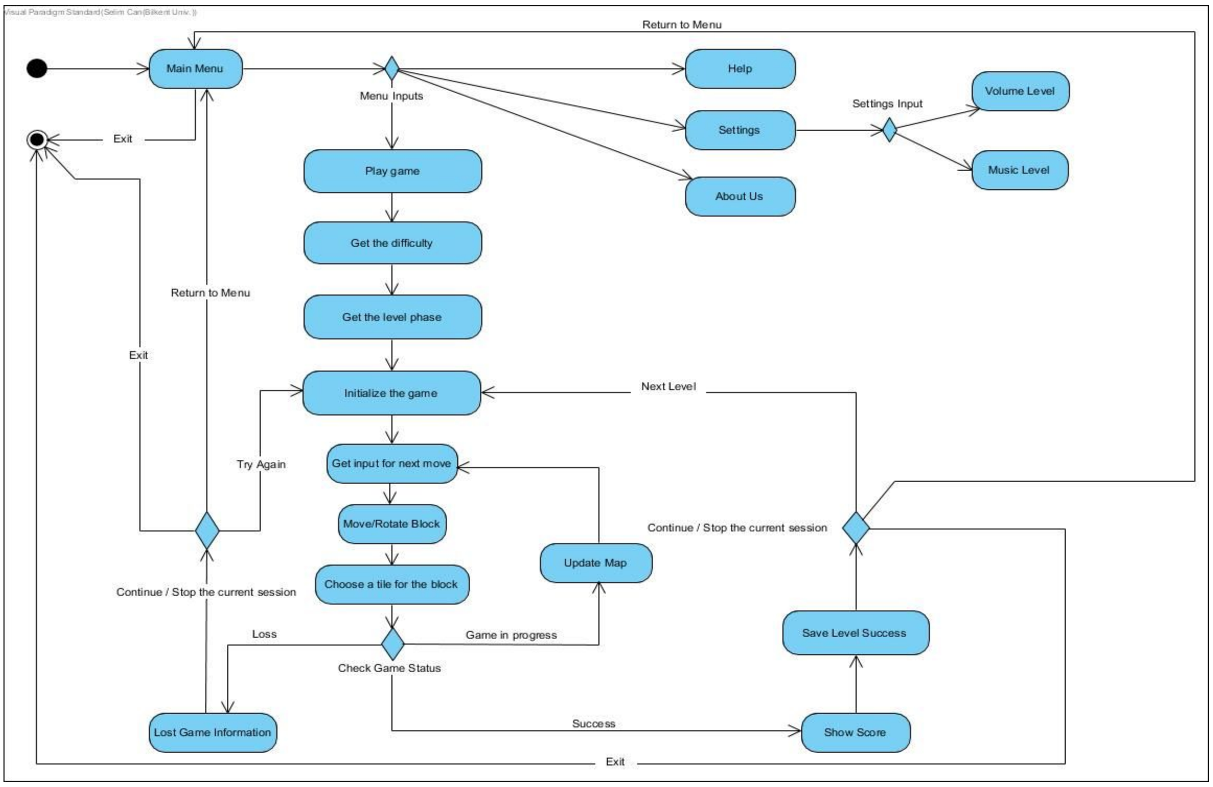
This diagram shows that how settings interacts with the classes. Users go to the setting from menu. What they can do in setting menu, is shown in diagram. They can change volume of attempts in the game, turn off /on background music . Furthermore, players can see informations about creaters of the game and learn how to play with the help of help screen. Lastly, user can only quit the game by clicking exit button because there is no another option to quit in other screens.

5.2.2. Statechart Diagram



In our game, when user enters the game through GUI, the system initializes the map. Then, there is a choose block statement that is for choosing which block to place. Then, when the user chooses the block that will be placed, there is another statement for rotating the block that is chosen. When the user chooses both the block and rotation, the system places the blocks. Unless all blocks are placed, the system return to choose block state and expects for a new block placement. If everything goes well, such that all blocks are placed, it goes to an simulation step. In this step, the program tries thief to escape. If thief escapes, user loses and the program returns to the first lose state, and then initialization of map state. If escape has nowhere to escape and user wins, then the program goes to a victory state then the final sta

5.2.3. Activity Diagram



The activity diagram represents the movements, inputs and the game management when interacting with the user. It is a proper way to introduce to the user how to play or how to use the application if the person has no information about the program.

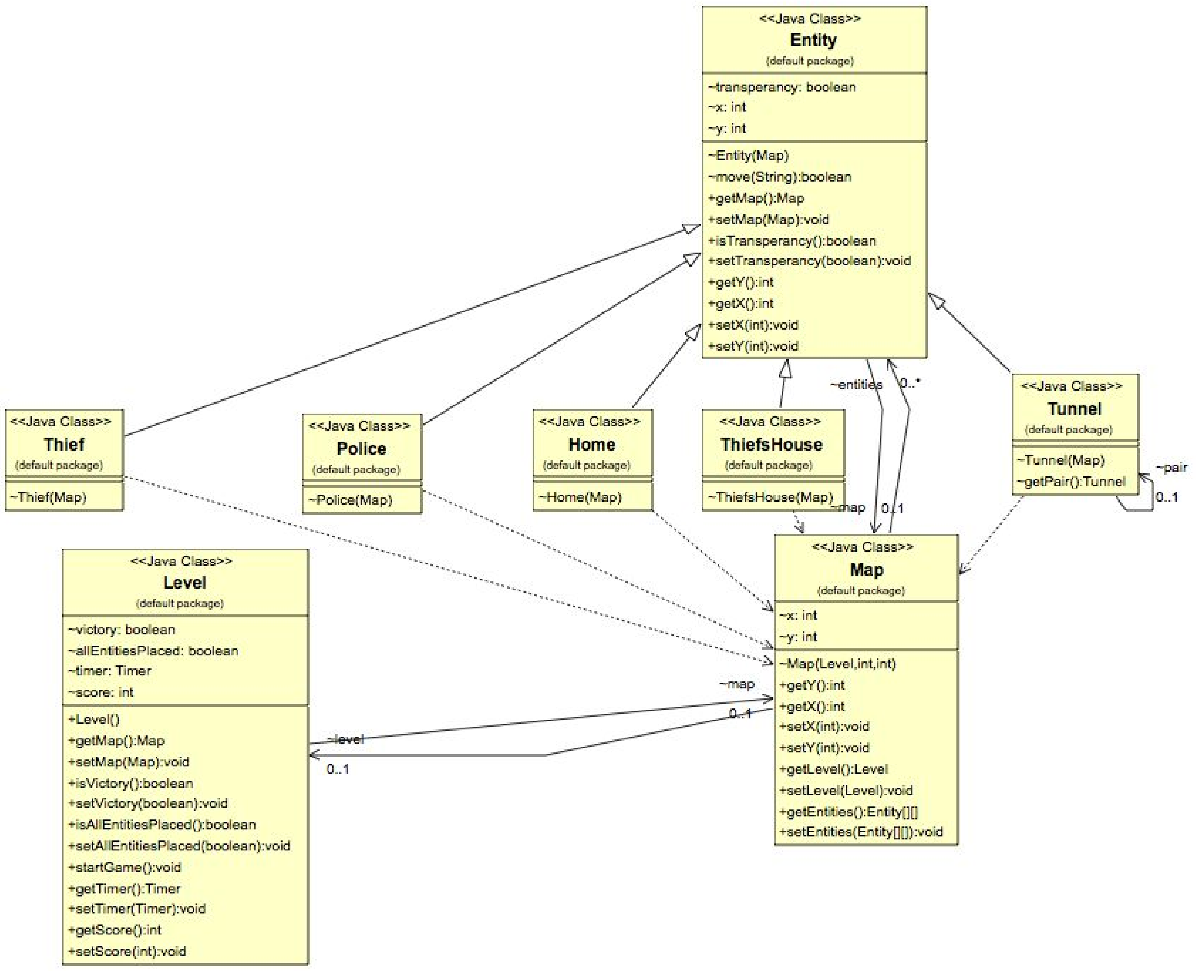
The initial start of the application is the opening of the program and a little after, there is a menu which is to start the game or in order to help the user, there are information and settings about the game itself. If the user wants to start a level, then program waits for an input from the user to take an action.

When the input needed to start the game is taken from the user, there are certain decisions which are mandatory to be chosen such as difficulty of the game and current level phase of the exact difficulty. When the requirements are satisfied, the user can start a new level.

During the level, the game itself, player has no restrictions such as move counts or health, but a clock that is independent from the player’s moves. The blocks can be placed or taken from the location they are found, or they can be rotated to block the available roads. Every active movement of the player, meaning that if the person places a block other than taking or replacing it, will be checked by the game management to decide whether the game is finished. If it is not, the block that is chosen will be placed to the desired location if the location is available.

When it comes to the end of the game, there are two ways of exit: one wins or loses. Lost can occur when the time given to the player ends. In this case, the player can try again the exact level or return to menu to choose another difficulty or level.One of the exits in the game is in the loss menu when a player loses the current game. If the player wins, by placing the correct blocks, then the player can continue to the next level or return to the menu to change the current game if there is need for more challenging levels. If the person doesn’t want to continue to the game, another exit in the game is in this menu.

5.3. Object and class model



Classes:

- Level : Its instance hold the current the current level, a map, and scores etc.
- Map : A map holds each 2D world object and their place on the 2D World.
- Entity : Objects of 2D world.
    - Thief : A 2D object that represents a thief of real world.
    - Police : A 2D object that represents a police of real world.
    - Home : A 2D object that represents a home of real world.

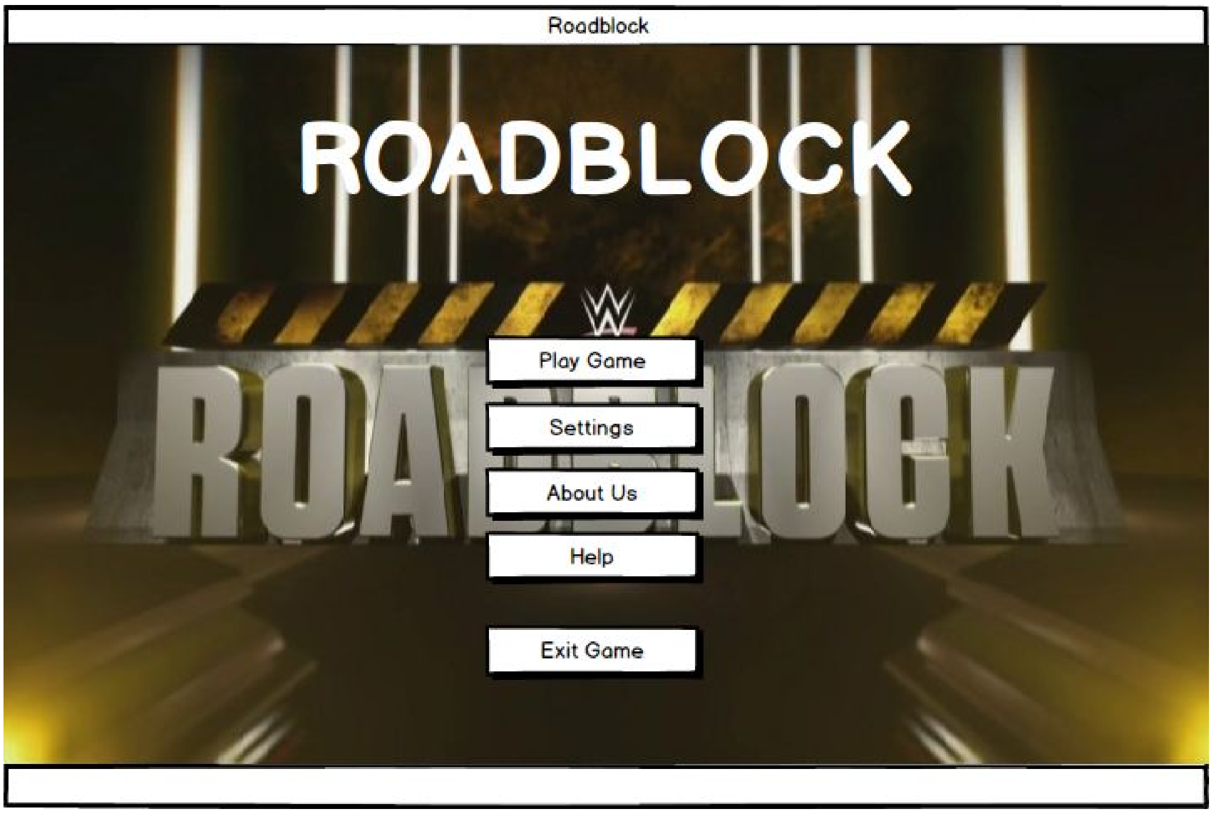
- ThiefsHouse : A 2D object that represents a house that belongs to a thief in the real world.
- Tunnel :A 2D object that represents a tunnel of real world.

We are going to implement our game through the Level Class. When the game starts, the system initializes the level and associated other objects in it.

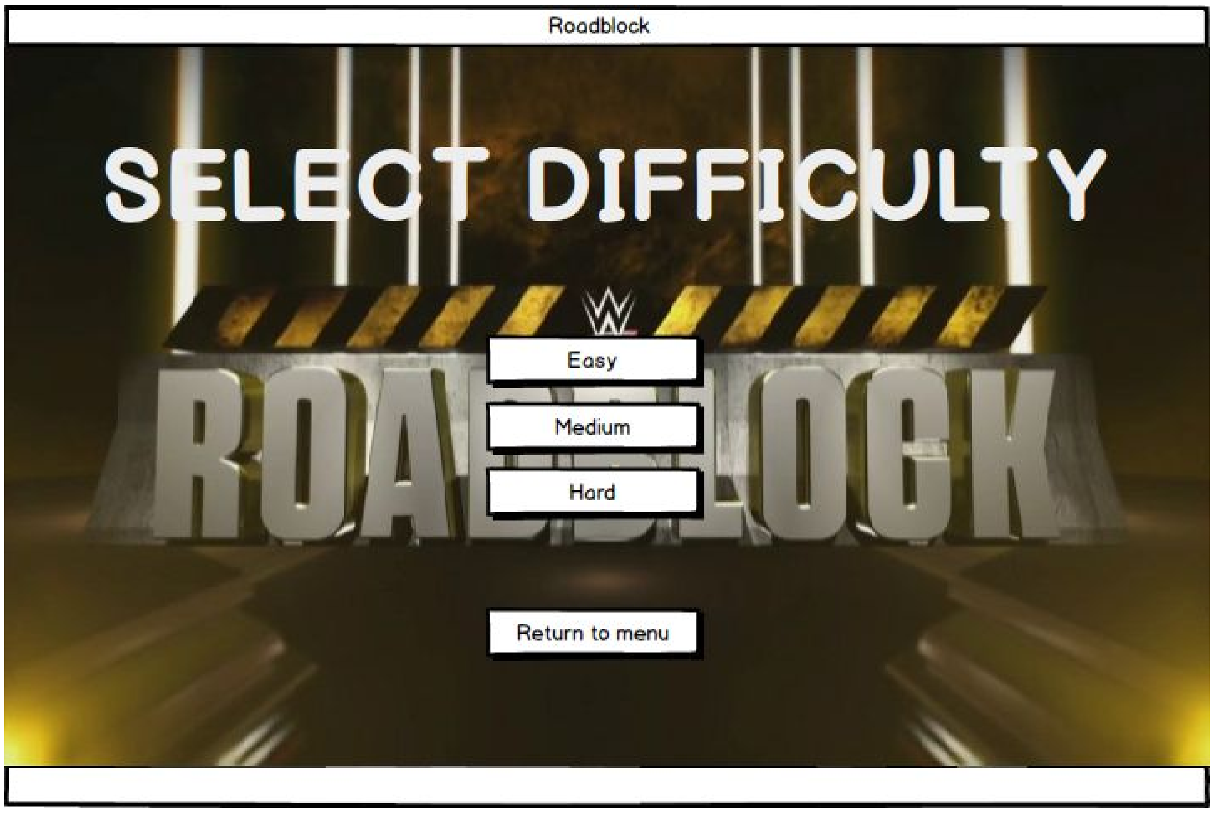
Every Object Class that occupies a space in this 2D World is a subclass of the Class Entity. They may be transparent or not. Transparent object such as Tunnel, lets the Thief pass. Others does not.

In each level, there will be a [x][y] Map that holds every entitiy object.

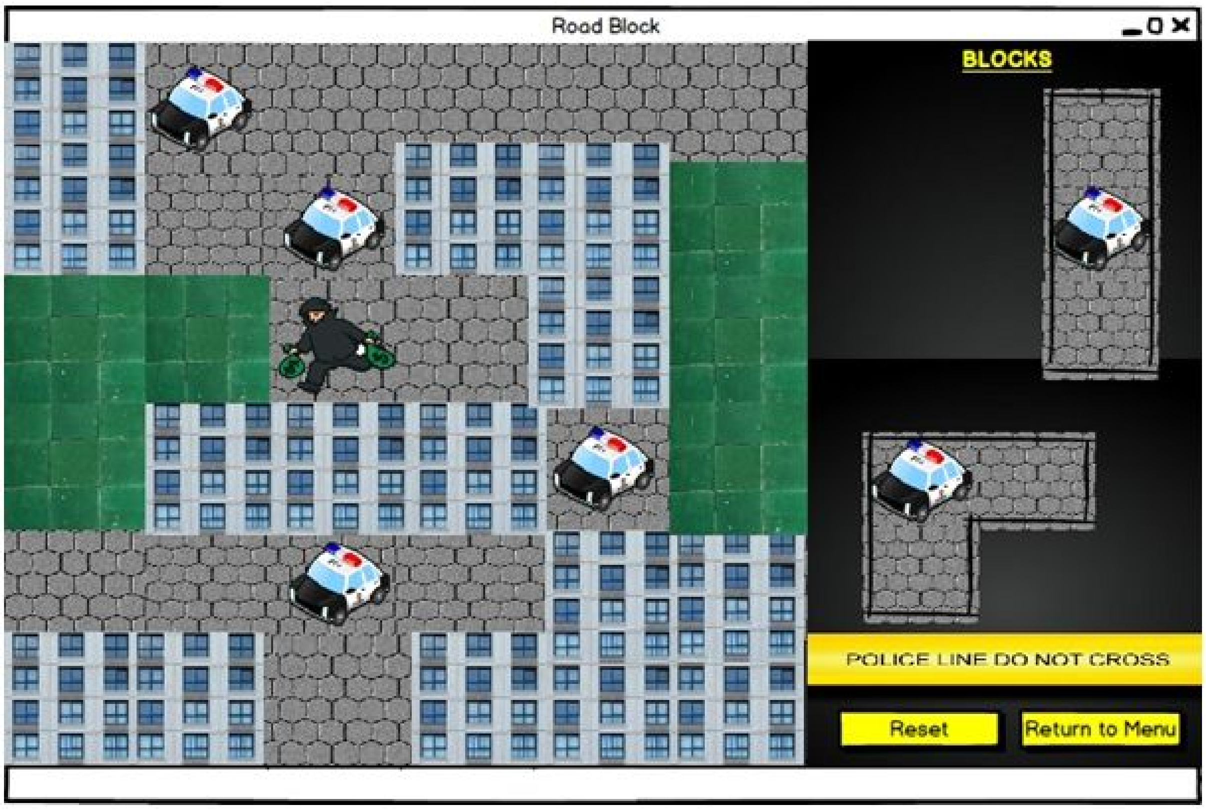
6. User interface - navigational paths and screen mock-ups



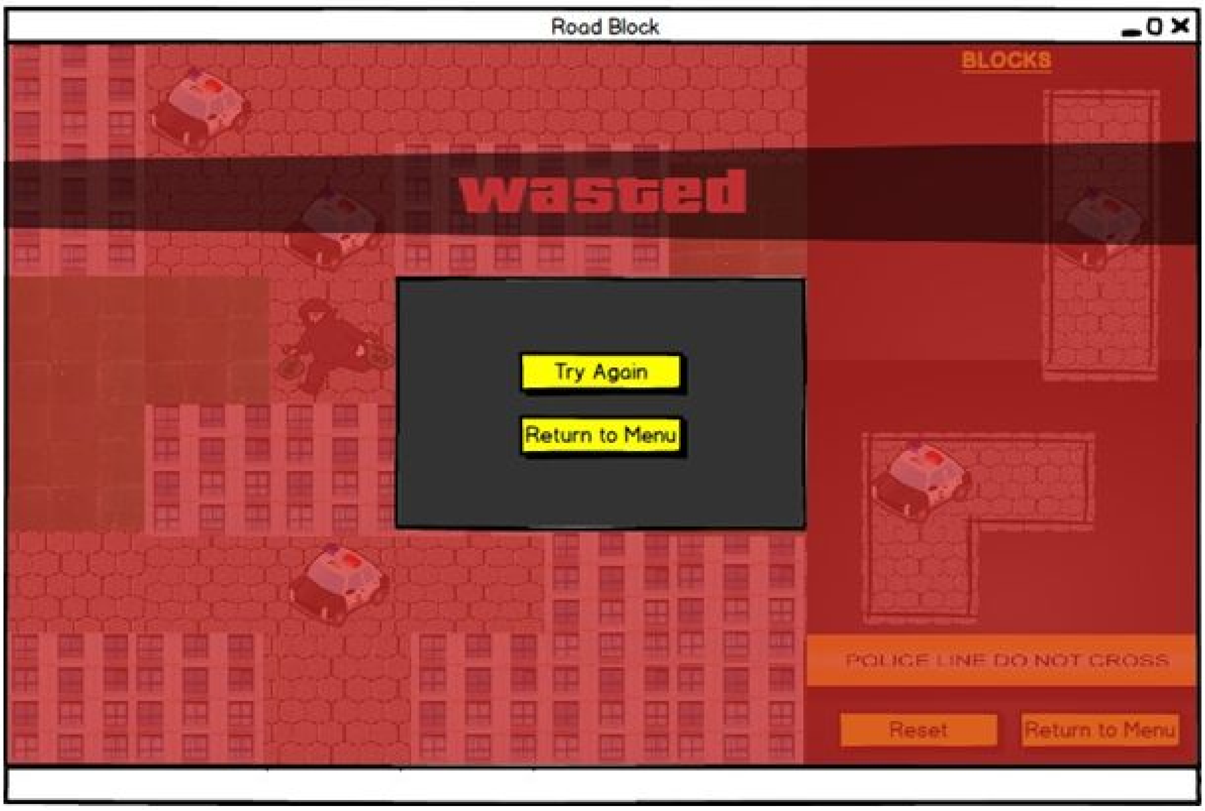
This is the entrance interface of the RoadBlock game. Using this interface it is possible for user to start a new game, set their setting preferences, learn about us,developers of the game, and get some useful information about how to play the game. This interface also offers the user to exit and close the game.



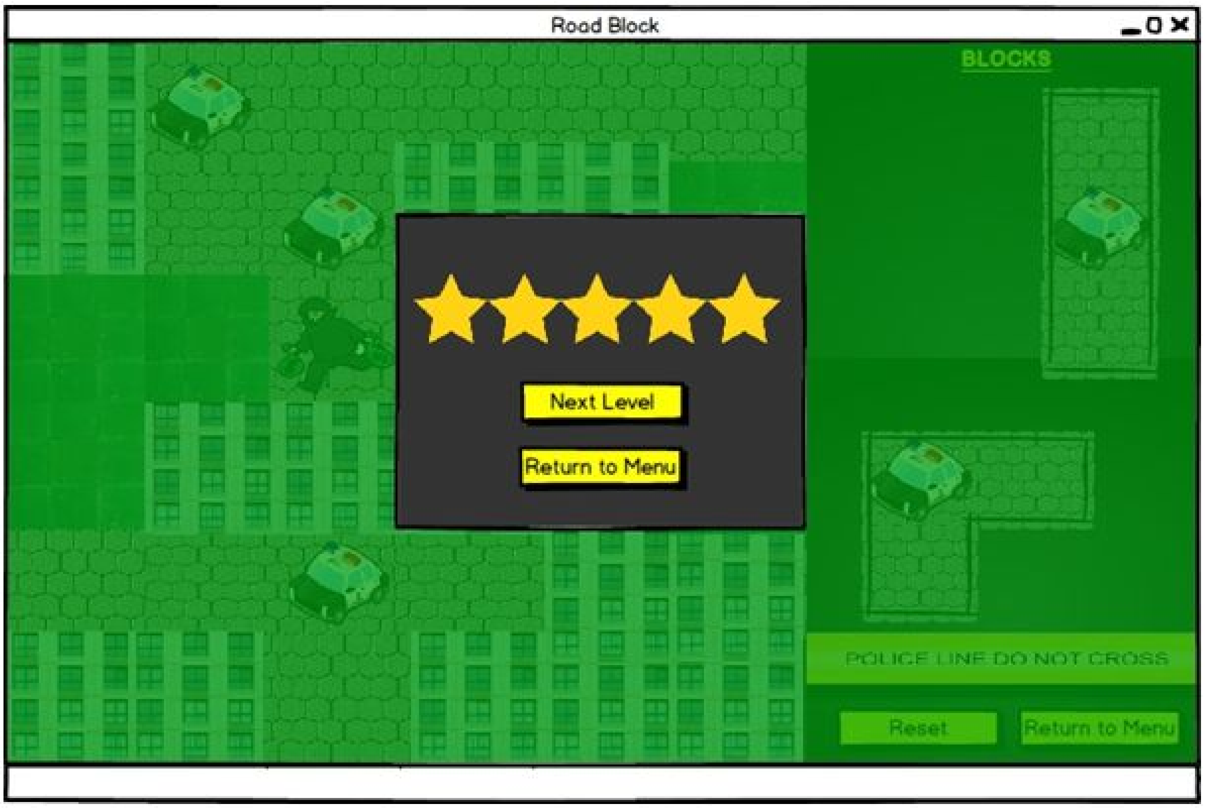
This page allows the user to choose a desired difficulty level of the game. It also has the option to return the previous menu.



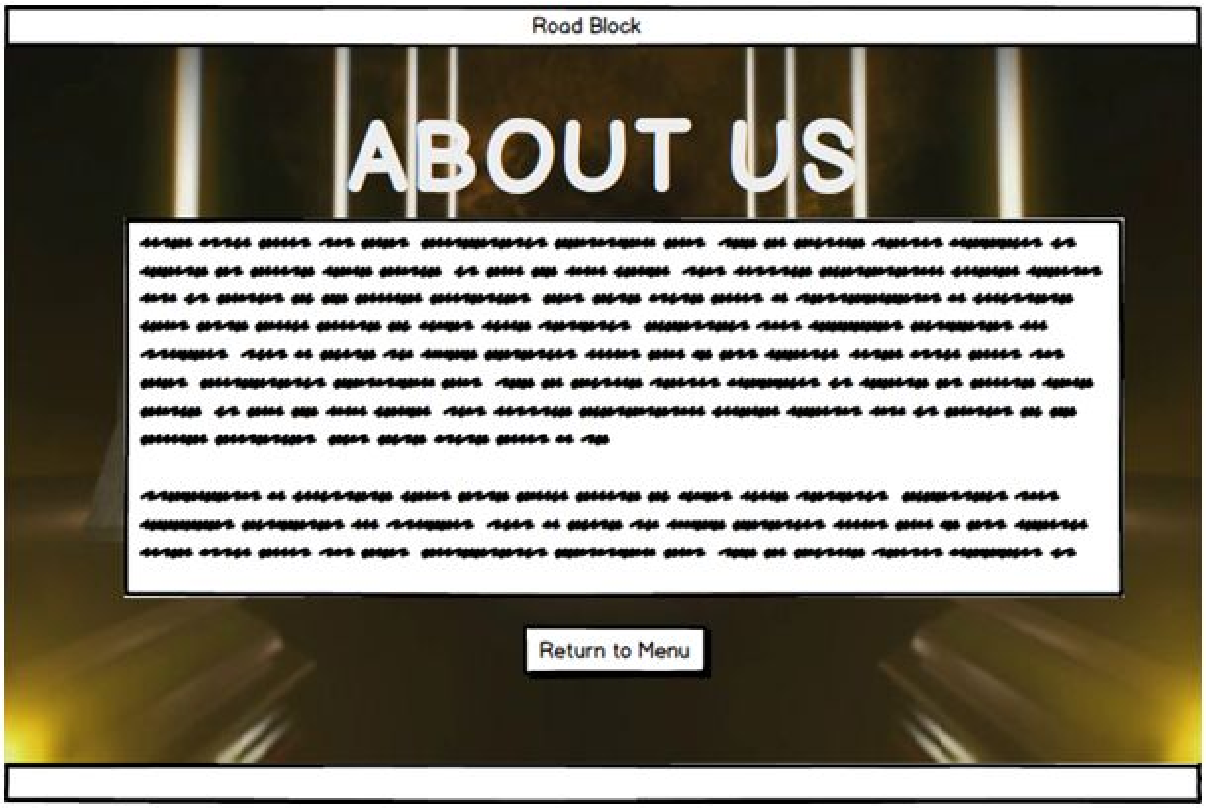
This is the main screen of the game. Player plays the game in this screen and places the police cars in order to block the paths which the thief may use in order to get away from the police.


Road Block
BLOCKS
wasted
Try Again
Return to Menu
POLICE LINE DO NOT CROSS
Reset
Return to Menu

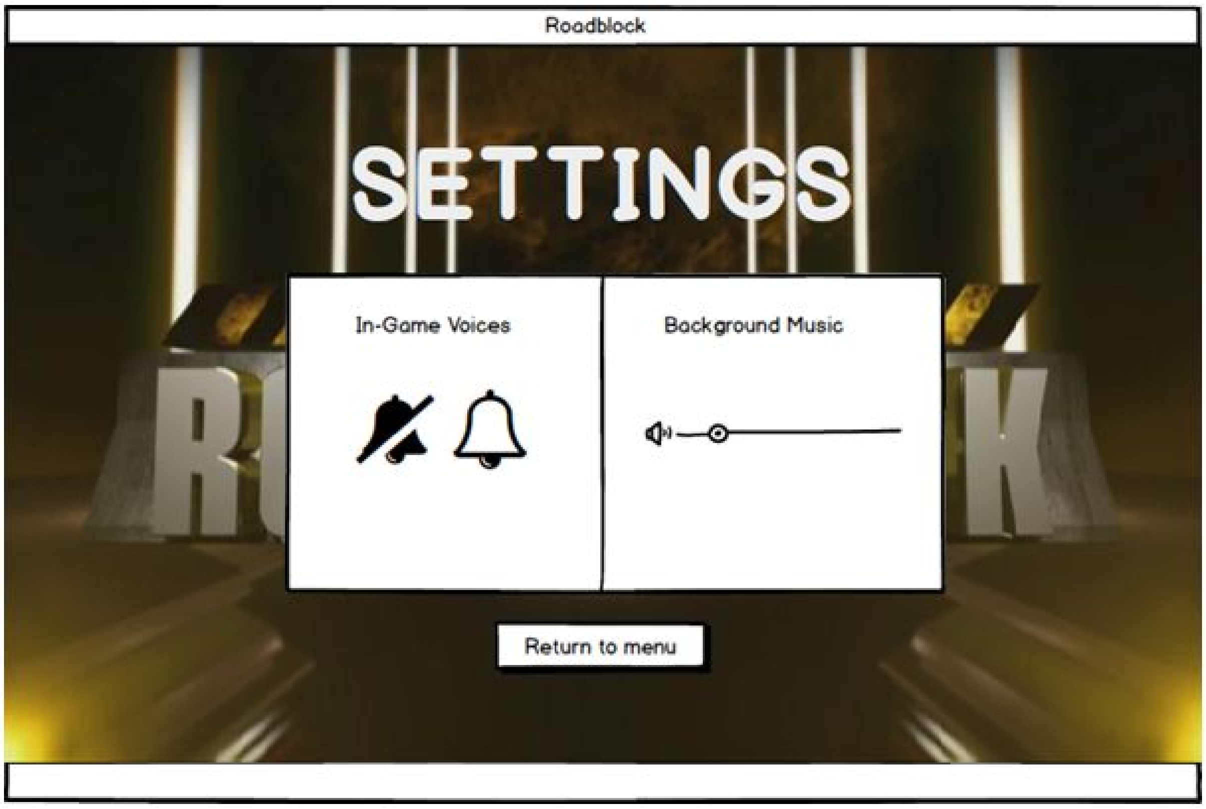
This page occurs when the player loses the game. The player can try it again or return back to menu.



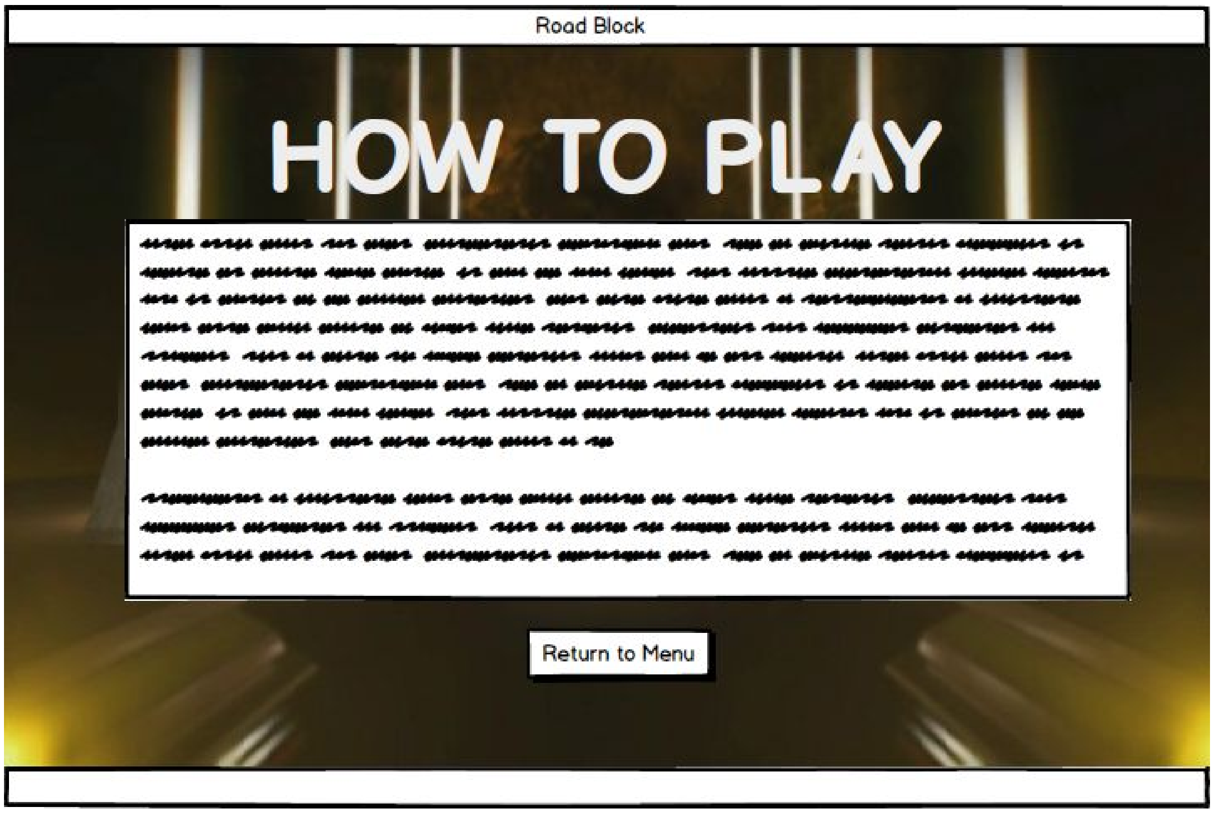
This page occurs when the player wins the game. The player can play the next level or return back to menu.



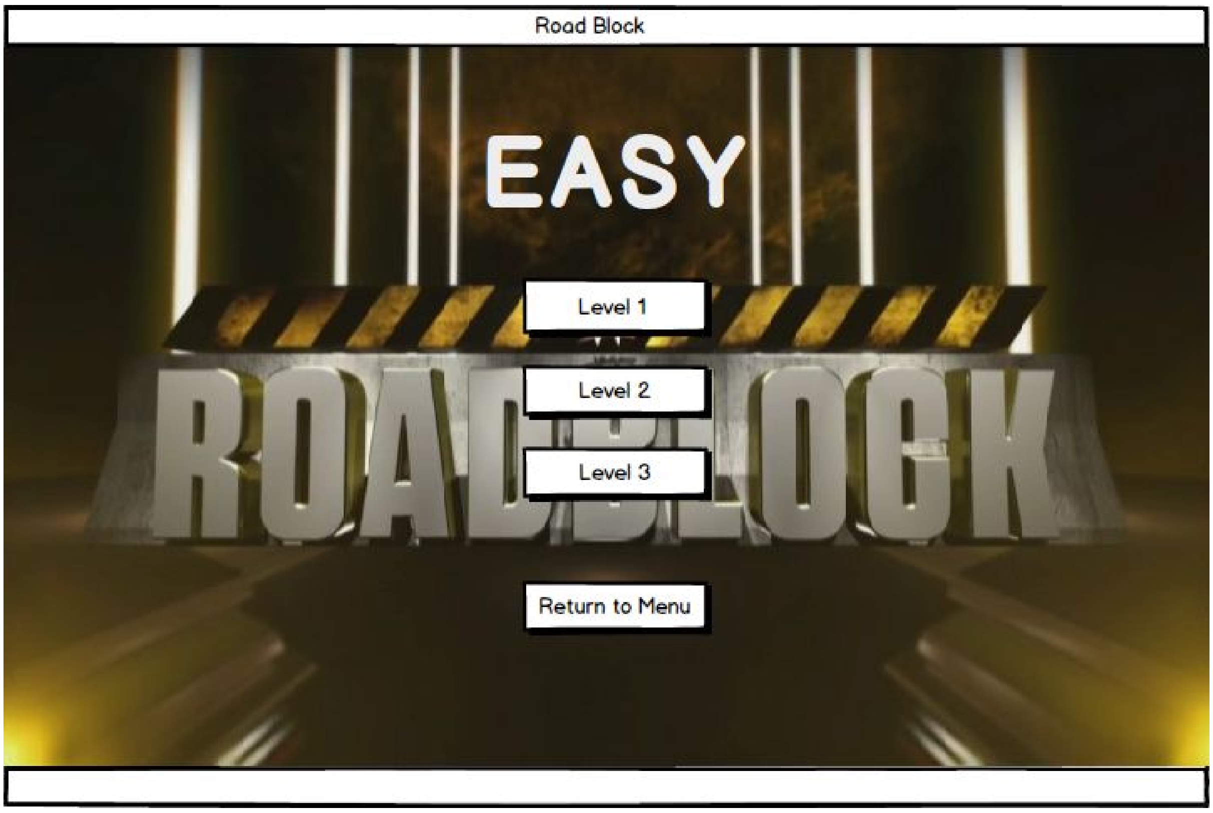
In this page, it is possible learn about the developers of the game and it also provides the option to return to the main menu.



These settings allows the user to turn on or turn off in-game voices. Also it is possible to adjust the background music volume.



In this page,user can learn the basics of the game.This page enables users to Return to Menu.



In this page,User may select 3 different levels of the Easy difficulty level.User may also return to the menu by clicking Return to Menu.

7. Glossary & references

Lethbridge, Timothy C., and Laganière R. *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*. McGraw-Hill, 2002.