Bilkent University

Department of Computer Engineering

**CS 319 Term Project**

*Section 1*

*Group 1A*

*Walls and Warriors*

Analysis Report

Project Group Members

1. Dawood Muzammil Malik
2. Deniz Yüksel
3. Faruk Oruç
4. Mert Soydinç
5. Muhammed Salih Altun

Supervisor: Eray Tüzün

Table of Contents

[1 Introduction 3](#_Toc527919883)

[2 Overview 3](#_Toc527919884)

[2.1 Game Objects 3](#_Toc527919885)

[2.2 Gameplay 4](#_Toc527919886)

[2.3 Ways to Play 5](#_Toc527919887)

[2.3.1 Classical Game 5](#_Toc527919888)

[2.3.2 Sandbox Mode 5](#_Toc527919889)

[3 Functional Requirements 6](#_Toc527919890)

[4 Nonfunctional Requirements 8](#_Toc527919891)

[5 System Models 8](#_Toc527919892)

[5.1 UML Diagrams 8](#_Toc527919893)

[5.1.1 Use Case Diagram 9](#_Toc527919894)

[5.1.2 State Diagram 14](#_Toc527919895)

[5.1.3 Sequence Diagrams 15](#_Toc527919896)

[5.1.3.1 CreateLevel Diagram 15](#_Toc527919897)

[5.1.3.2 PlayGame Diagram 16](#_Toc527919898)

[5.1.3.3 StartGame Diagram 17](#_Toc527919899)

[5.1.4 Class Diagram 18](#_Toc527919900)

[6 5.1.5 Activity Diagram 20](#_Toc527919901)

[6.1 User Interface & Screen Mockups 21](#_Toc527919902)

[7 Glossary & References 29](#_Toc527919903)

# Introduction

We are group 1A, 404 Name Not Found. We decided to design and implement the classic mind game, Walls and Warriors. This is a 1-player game that the player has a set of 4 walls with different shapes. The player can win the stage by placing those four walls in only one way. Therefore, for each stage there is a single solution. The original game board is a 5x4 rectangle, without the edges. There are 4 red and 3 blue knights and a blue tower that are initially placed on the board grids. One red knight is optional and does not have to be present in all stages. The player has to wall in the blue knights and keep the red knights out in order to win. However, while doing that the player cannot trap red knights inside walls.

We are going to adapt the classical board game to a desktop version. We will implement the board and the gameplay as 2D, like a chess board. The blue and red dots will represent the blue and red knights respectively.

Another feature that we plan on introducing is the sandbox mode. With the sandbox mode, players can specify the dimension of the board, choose number of knights and towers, create their own walls with their unique shapes and save the level. After saving the level, the particular level will be in the list view of the sandbox levels.

# Overview

# Game Objects

The game objects are the physical pieces that the player can interact on the screen.

* + 1. **Timer**

The timer is a game object that is used to measure high scores of the local players. The timer starts whence the tower and all the knights are placed to the grids.

* + 1. **Board**

The classical game board is a 5x4 rectangle without the grids on the edges. Therefore, there are in 16 grids in total that the knights and the tower can be placed. A knight takes 1 grid whereas the tower takes 2 grids. Between the grids, walls can be placed.

* + 1. **Knight**

A knight can be either blue or red. There cannot be more than 1 knight per grid. At the beginning of each stage, there are a combination of 3 and 4 knights placed on the board. All stages do not have to contain all knights. In fact, the first level contains only two blue knights and no more.

* + 1. **Wall**

Walls are the only game objects that player can interact. There are four predefined walls in the wall set.

* + 1. **Tower**

The tower is a game object that is initially placed on two grids and user cannot interact with the tower.

# Gameplay

The original board game has 80 levels. We did not decide yet, that how many levels we will have. However, each level will be hardcoded. The user can only interact with the walls. In the game, the tower and the knights are placed before the timer starts. The user has the options to choose one of the four walls with a mouse click, and after the click, the wall is selected and the user can rotate the wall 90 degrees with corresponding keyboard keys and place the wall into a space between the grids with the next mouse click. The wall then will be placed to the particular space. Once four walls are all placed and all blue knights are inside while the red knights are out, the timer will stop and the player will be asked a pop up to enter a name for collecting local high score. If the player places all the four walls and the game is not complete, the game will not give error messages. Therefore, the player has to figure out by his/her own the correct solution.

# Ways to Play

# Classical Game

The classical game is the style of the original game, adapted to a desktop application. There are 16 grids that the chosen knights and the tower are placed. In order to win the game, the player must place the walls in such a way that red knights would not have a path to enter the castle. While doing that, the player cannot trap a red knight by walling it by cornering it. See the “2.2 Gameplay” section for more details.

# Sandbox Mode

The sandbox mode features the player to create his/her own levels and become the editor. In this mode, the editor can specify the board dimensions. Also, there are infinite number of knights that the editor can use. Moreover, the player does not have to use the predefined 4 types of walls. By click and drop, the editor can draw walls between the grids. However, the solution has to be valid in order for the game engine to confirm the solution. Basically, we do not allow the editor to create a level that does not have a solution.

# Functional Requirements

**3.1 Starting a Game**

After starting the game, the player will encounter two tabs, which are level select and sandbox mode. Regardless of the selection made here the underlying gameplay is the same. The game board is a rectangle grid with five grids on one side and four grids on the other. The corners of the board are not grids. Therefore, the board is 16 grids in total. At the start of each level, the game will place a Tower, three Blue Knights and three or four Red Knights in the game board all except the tower is exactly one grid while the tower is two grids. The aim of the player is to use the walls in his possession to separate the grids consisting Blue and Red knight, so that the Blue knights are all walled up with the tower. The player will have different number of walls in different shapes depending on the level.

**3.1.1 Main Levels**

This can be accessed from the main Menu. In the main levels screen there is a fixed number of set levels that can be played in order. At first, the player will have access to a single level but as the player completes the levels, new ones will unlock. For each level, there is a timer to keep track of the players’ ability in solving the given puzzle. A number of high scores exists for all levels.

**3.1.2 Sandbox Mode**

This can be accessed from the main Menu. Sandbox mode is the level creator for the game. In the Sandbox mode, the player can choose the dimensions of the board, the number of Red or Blue knights and the number walls that can be placed in this level as well as the shapes of this level. The creator has to specify that there is a solution to the level he created if no solution is specified, the game checks to see if any solution is available. If no such solutions are found the newly created level cannot be saved.

**3.1.3 Game Algorithm**

The original Walls and Warriors game have set number of levels and a single solution for each level. This means that for the original game there is no need for an algorithm as long as you specify the solution for each level. However, our game features a sandbox mode for the player; the newly created levels in the sandbox mode can have vastly different properties like having more than one solution or having a much bigger board. This means that in order to have such a feature we are going to need an algorithm that can detect win conditions and illegal moves in unknown levels.

**3.1.4 Database**

The game will have a large number of levels and high scores related to each level. This makes it a must to have an efficient database in order to save our data.

**3.2 Options**

This screen can be accessed from the main menu. Here the player is able to choose between game settings as he pleases in order to enjoy the game to its fullest. The settings include the option the adjusting the volume, remapping the keys the game uses, changing the games language and activating a colour-blind Mode.

**3.3 How to Play**

This screen is accessed from the main menu. Here there is a number of instructions to help the player understand the game. These instructions include written sentences, informative pictures and link to tutorial videos.

**3.4 About Us**

This screen is accessed from the main menu. Here there is general information regarding the group members and our game. Here the player can report bugs and send their criticisms about the game to us.

# Nonfunctional Requirements

**4.1 User Friendly Game Interface**

One of the main features of every software system is the quality of its user interface. In order to make the game appeal to a large audience, the user interface of our game will be intuitive and easy to learn. It will also have convenient options such as having a colour-blind option and the ability to remap the keys for user convenience.

**4.2 Tool for Learning**

As a puzzle game, our product aims to make its players think as they solve problems and have fun. It is especially important for small children to engage in such activities from a young age so our game is a perfect tool for parents. However, this does not mean that people from all ages cannot enjoy the game.

# System Models

# UML Diagrams

# Use Case Diagram

**Use Case #1**Name:

* New Game

Participating Actor:

* Player

Stakeholder and Interests:

* Players wants to play the game or create his own level in the sandbox mode.

Entry Condition:

* Player has the game running.
* Player is on the main menu.
* Player clicks on the *New Game* button.

Exit Condition:

* Player clicks the *Quit Game* button.
* Player completes all the levels.

Main Flow of Events:

1. Player selects *New Game* option from the main menu.
2. The system responds by displaying the *New Game menu*.

Alternate Flow of Events:

1. If user wants to return to main menu:
   1. Player selects *Back* button.
2. If user wants to play the game:
   1. Player selects *Select Level* button (Use Case #2).
3. If user wants to create his own level in the sandbox mode:
   1. Player selects *Sandbox Mode* button (Use Case #2).

**Use Case #2**

Name:

* Select Level

Participating Actor:

* Player

Stakeholder and Interests:

* Player wants to play the game.

Entry Condition:

* Player is on the *New Game* menu.
* Player clicks *Select Level* button.

Exit Condition:

* Player clicks the *Back* button to return to main menu.
* Player quits the game.

Main Flow of Events:

1. Player clicks *Select Level* button.
2. System shows the levels the player is eligible to play.
3. Player selects the level he wishes to play.
4. The system loads and displays the game board and starts the timer.
5. Player tries to complete the challenge in the given time.
6. System displays player’s on screen when the game ends.
7. If the player’s score is one of the highest, the system asks the player to enter his name.

Exceptional Cases:

* User can modify audio settings while playing the game.
* User can view the current high scores while playing.
* User can view game instructions while playing the game.

**Use Case #3**Name:

* Select Sandbox Mode

Participating Actor:

* Player

Stakeholder and Interests:

* Players wishes to create a level of his own.

Entry Condition:

* Player is on the *New Game* menu.
* Player clicks *Sandbox Mode* button.

Exit Condition:

* Player clicks the *Back* button to return to main menu.
* Player quits the game.

Main Flow of Events:

1. Player clicks *Sandbox Mode* button.
2. Use enters the dimension of board he wants.
3. System responds by displaying the board of the dimensions entered.
4. User places the objects on the board.
5. When the user is done, he submits his challenge.
6. System runs the algorithm to check if the challenge is valid.
   1. If valid, the challenge is saved in the system.
   2. If invalid, user is asked to try again until his challenge is accepted.

**Use Case #4**Name:

* View Settings

Participating Actor:

* Player

Stakeholder and Interests:

* Players wishes to view/modify audio and/or video settings.

Entry Condition:

* Player is on the *Main* *menu*.
* Player clicks *Settings* button.

Exit Condition:

* Player successfully changed the settings.
* Player clicked the *Back* button.

Main Flow of Events:

1. Player clicks the *Settings* button.
2. Player selects the type of settings he wishes to modify.
3. Player applies the changes.

**Use Case #5**Name:

* View Options

Participating Actor:

* Player

Stakeholder and Interests:

* Players wishes to view information regarding the game or its creators.

Entry Condition:

* Player is on the *Main* *menu*.
* Player clicks *Options* button.

Exit Condition:

* Player clicks *Back* button.

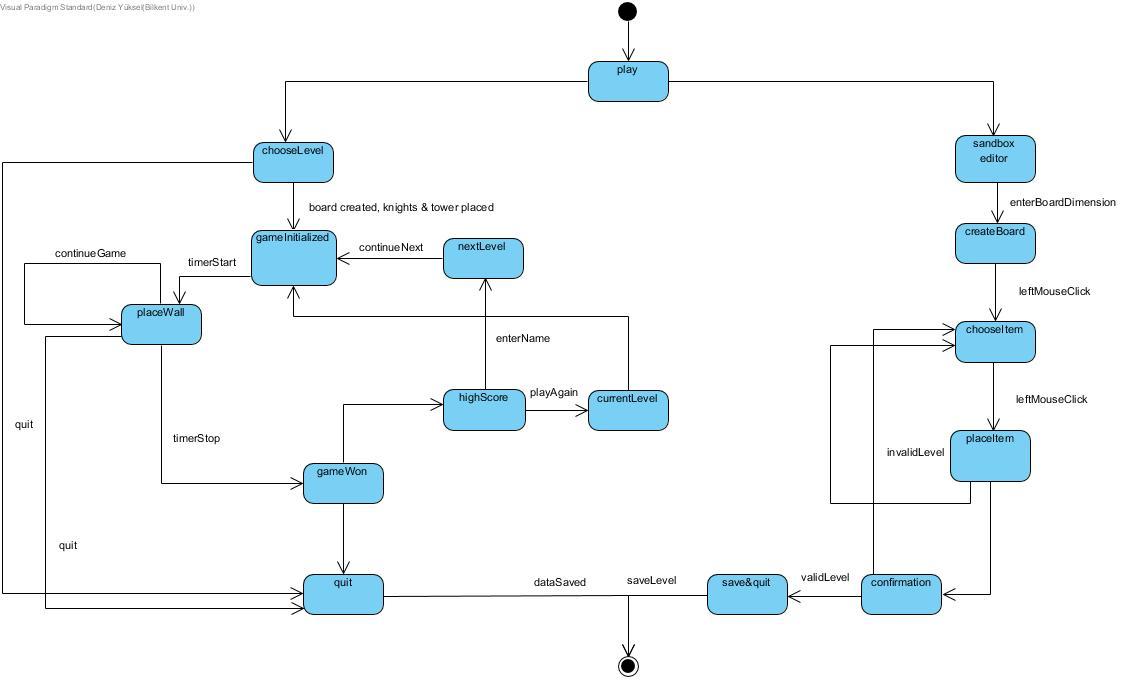
Main Flow of Events:

1. Player selects the *Options* button.

Alternate Flow of Events:

1. Player wants to check the high scores.
   1. Players clicks *High Scores* button.
2. Player wishes to see how to play the game.
   1. Players clicks *Game Instructions* button.
3. Player wants to see information about the game creators.
   1. Players clicks *About Us* button.
4. Player wants to see the game credits.
   1. Players clicks *View Credits* button.

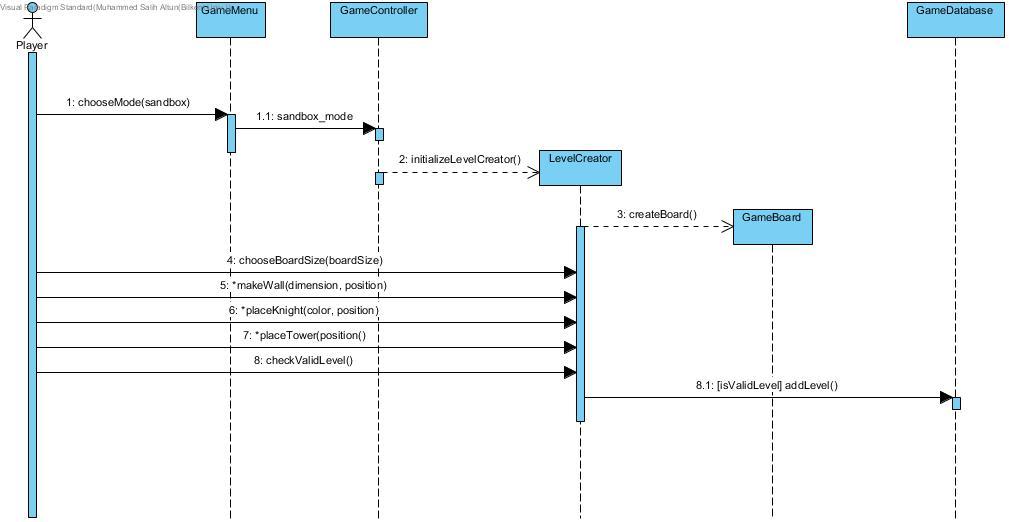
# State Diagram



The initial state of the game within the gameplay is the play state. Afterwards the player can either choose the sandbox mode or the classical game. If classical game is chosen, the player can choose a level that is either completed, or the one next to the last completed. After the player chooses the level, game is initialized with knights and tower. The timer starts as soon as initialization finishes. The player places walls between the grid spaces until the level is complete, while quitting is an option any time. If the game is won, the timer stops and the player is presented with a pop up for the local high score records. Afterwards, the player can either play the level again (and refuse to enter a name) or enter the name and continue to the next stage. When the next stage is chosen, the process repeats for the next level.

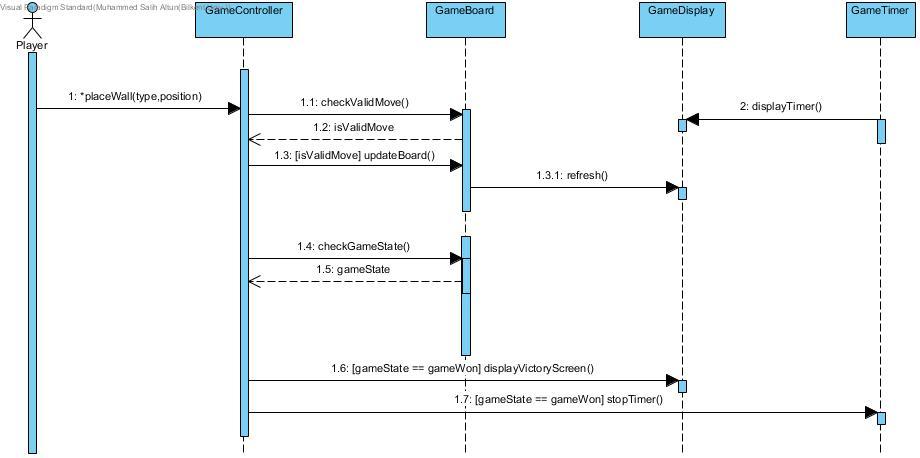
# Sequence Diagrams

# CreateLevel Diagram



In CreateLevel diagram, the player decides that they want to create a new level of their own. They choose sandbox mode from the menu. The menu gives this information to the game controller which initializes the level creator. The player needs to choose their board size first. After that, they are free to place however many of the game objects on the board. The player can also introduce their own new wall shapes to tackle these new puzzles. At any point, they can try to make their level a reality by asking the game to create the level. In this case the level creator checks if the self-made level has a valid solution or not. If the level has a solution, it is added to the game database.

# PlayGame Diagram



In the PlayGame Diagram, the player has started a level. They need to solve the puzzle. Game timer is created and starts when the level starts. Player starts trying to place the walls on the screen. The controller checks if the position that the user is trying to place the wall is valid or not. If the move is valid, then the controller tells the board to refresh with the new wall placed. Every time a wall is placed, controller looks at the board and checks the game state to understand if the player has completed the level. Meanwhile, the timer keeps ticking every second. This iterative process continues until the level is solved by the player and the game is won. In this case the controller tells the display to show the victory screen and the timer stops.

# StartGame DiagramD:\Deniz\Deniz\Deniz Önemli\Courses\CS\Cs-319\sequence\StartGameDiagram.jpg

In the StartGame Diagram, the player chooses to play a level from the game menu. The game gets the information of user's current level. Since the rest of the levels are locked, they can't play those. If the level they chose is playable then the game controller initializes the board with the properties of the level (position of knights and the towers). Game board tells the display to paint itself. Game controller also creates the timer and the timer starts ticking. The timer tells the game display to display the time passed and so the level has started.

# Class Diagram

The class diagram for our game can be seen above. It consists of nine classes but two of the classes are children of the GamePiece class.

* **GameController:** This class is the overseer of the whole class system. It lets other classes to talk to each other and gives necessary commands to them. Starting, loading and ending the levels are issued by this class.
* **GamePiece:** This is the superclass of all game pieces. Which are the things that can be put on the game board in the physical game. It has all the common functionalities of the pieces such as position and the ability to move.
* **Knight:** Child class of the GamePiece, both the Red and the Blue knights are represented by this class. It has a colour attribute to distinguish the different knights. As well as the functionality to get and set their colours.
* **Wall:** Child class of the GamePiece, all walls are made up of one side lines put together. The shape property specifies the formation of these lines to create a wall. The rotate functionality rotates the wall while keeping its shape intact.
* **GameBoard:** Represents the actual board of the game but in our game it is more like the map of each level. It has dimensions, which are 5 to 4 in regular levels but can be changed in sandbox mode. It also has the main game algorithm in it to keep track of the state of the game and block illegal moves.
* **LevelCreator:** A special tool used in order to create new levels by the player or the developers. It interacts with the database and the board to save and check the newly created levels.
* **GameDatabase:** Keeps all the data regarding the created levels, high scores and the progress of the main levels. It saves and loads data as controller instructs it.
* **GameTimer:** Keeps the track of time for each individual level. Has standard functionalities of a regular stopwatch.
* **GameDisplay:** This class is purely about displaying each component in the screen. It is connected to the rest of the game logic by the game controller and paints the game as the controller instructs it.

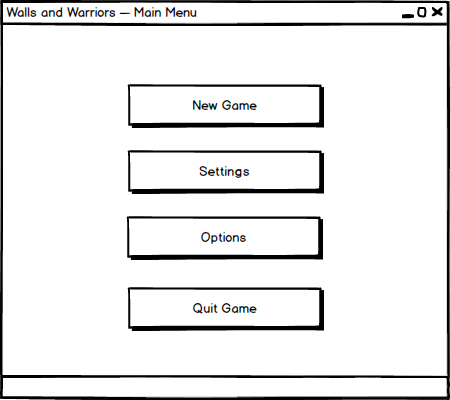
# 5.1.5 Activity Diagram

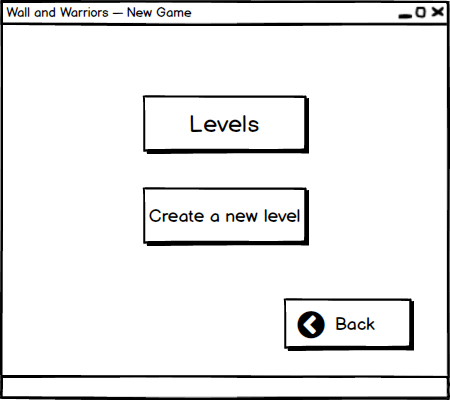
The activity diagram above illustrate how the system runs the game. In the beginning, the system waits for the user to select what he wishes to do. If the user decides to play the game, the system begins the game and starts the timer. It then waits for user to drag and drop all four wall pieces on the board. Once the fourth wall piece is placed, an algorithm is run and to check if the solution is valid. If the solution is valid, the game returns to the level selection menu so that the user can continue playing the game. However, if the solution is invalid, the game continues.

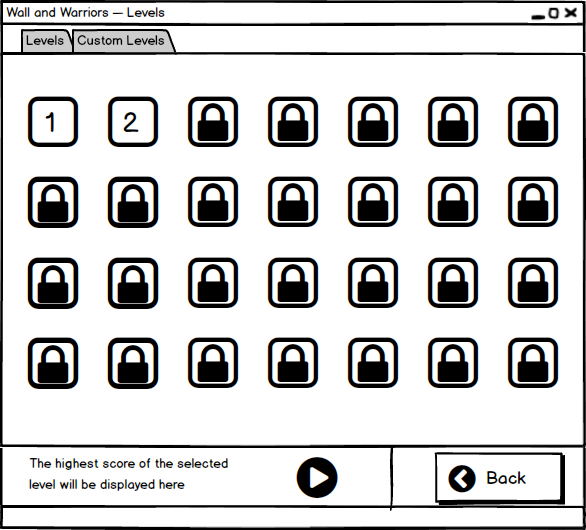
In the other mode, the Sandbox mode, the game allows players to create their own challenges. Once the user sets all the objects on the board and tries to submit the challenge, the same algorithm is run to check if the challenge is valid. If it is valid, the system saves the level. However, if the challenge is not valid, the system stays in the Sandbox Mode until the user either submits a valid challenge or cancels.

# User Interface & Screen Mockups

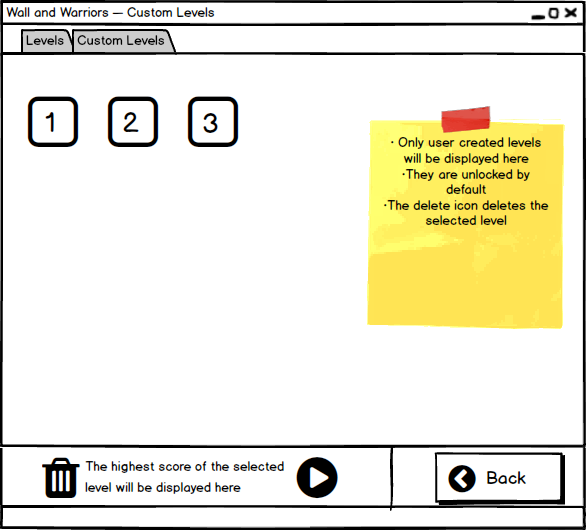
* **Main Menu**



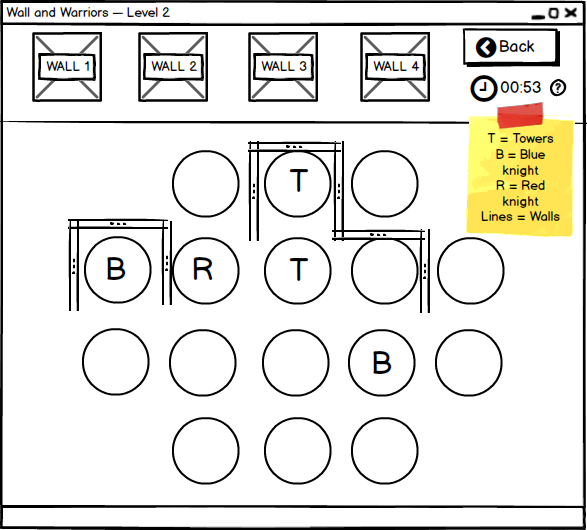
* **New Game**
* **Select Level**



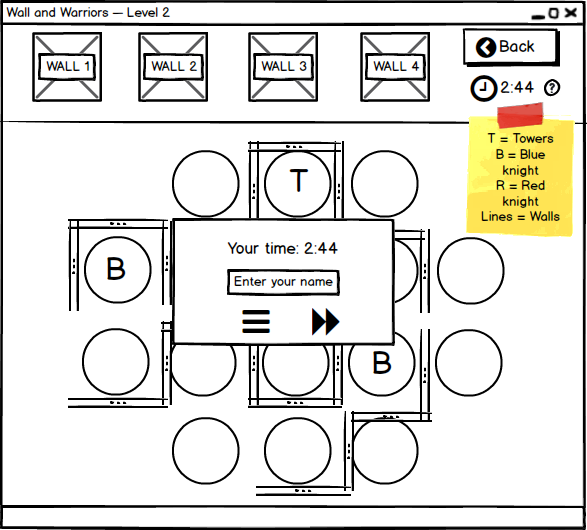
* **Select Custom Level**

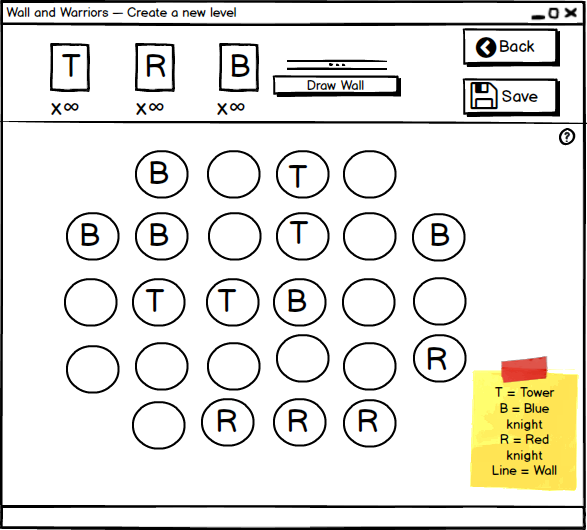


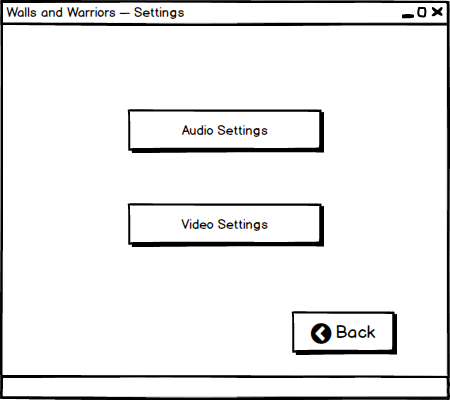
* **Play an Original Level**



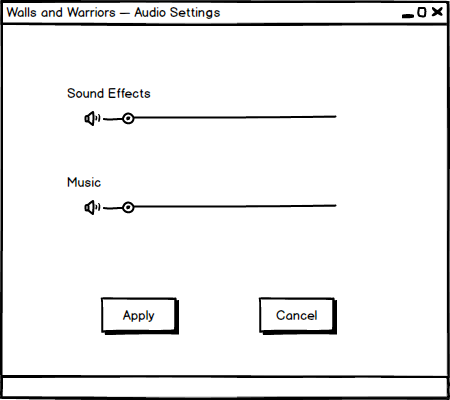
* **End of an Original Level**

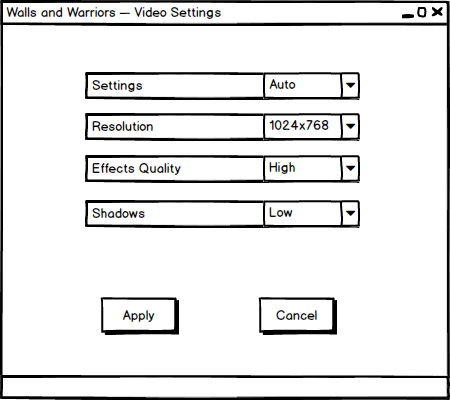


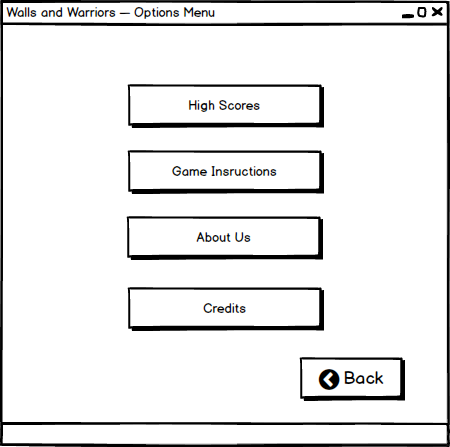
* **Create a New Level**
* **Settings**

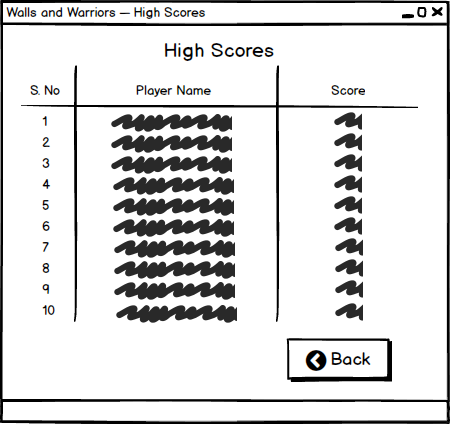


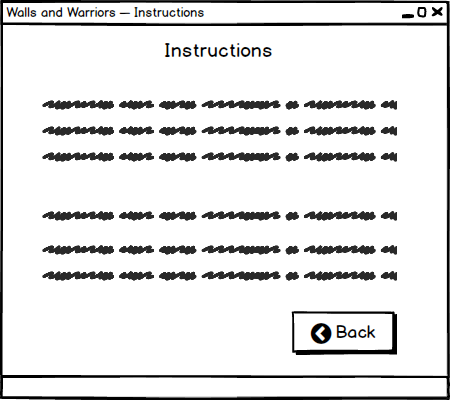
* **Audio Settings**

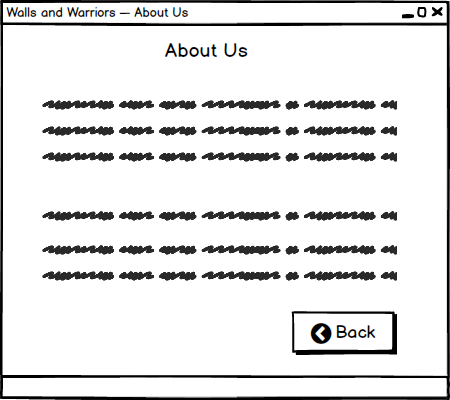


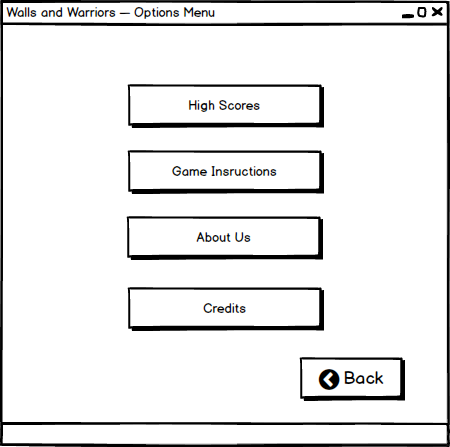
* **Video Settings**
* **Options**



* **High Scores**
* **Game Instructions**



* **About Us**
* **Credits**



# Glossary & References

**1)** https://www.smartgames.eu/uk/one-player-games/walls-warriors