

Bilkent Universtity

Department of Computer Engineering

CS 319 Term Project

Group 1D  
Wars & Warriors  
Iteration 2

Final Report

Project Group Members:

Samet DEMIR  
Ahmet MALAL  
Mahammad SHIRINOV  
İbrahim MAMMADOV  
Huseyn ALLAHYAROV  

Supervisor: Eray TUZUN

Table of Contents

[1 Introduction 3](#_Toc533359836)

[2 Design Changes 4](#_Toc533359837)

[2.1 Added Parts 4](#_Toc533359838)

[2.1.1 Singleton Pattern 4](#_Toc533359839)

[2.2 Removed Parts 4](#_Toc533359840)

[3 Lessons Learnt 4](#_Toc533359841)

[4 User’s Guide 5](#_Toc533359842)

[4.1 System Requirement 5](#_Toc533359843)

[4.2 Installation 5](#_Toc533359844)

[4.3 How to Play 6](#_Toc533359845)

# Introduction

In the 2nd iteration, we aimed to aid the aesthetics of the game and implement our additional features. We luckily could manage our time well and implemented the additional features as well as other missing parts from the 1st iteration. We have changed the maps which were simple maps of the board game. We replaced these maps with our own maps which are bigger in size and host many new game objects such as forest, lake, warships, new wall shapes and chain objects as we mentioned in our additional features part before. Game now has 4 maps which get more sophisticated each level. Maps are loaded from an external file with the necessary information for the initialization of the levels. We added a new hint feature to aid the player while playing. When player uses a hint by clicking the hint button, a wall is added to the map and hint button gets deactivated to limit the help. To implement the hint feature we now use a new external file named solution. Both map and solution files are encrypted to avoid cheating. To use the external files, we have implemented a file system to the project as well. GUI of the game have also been improved with new music and sound files, background images and game textures. Players can now use keyboard to play the game with dedicated buttons to select/deselect the walls, place or remove them and rotate them in any direction; left or right. Settings page have also been added to the game which allows user to enable or disable the background music, sound effects and change and remap the keyboard keys for gameplay. Now player can also use right button of the mouse while dragging the wall to rotate it on the go. With the addition of new maps, game now checks the game state to verify if the game is over or not instead of using fixed solutions. We have implemented the mobile warriors that can change place. We also added our immovable objects which are lake and forest. The lake hosts warships of both red and blue warriors. Chain objects should be used in addition to the walls to pass the level which have both.

We changed many parts of the code as well as our way of organization. We started to use the GitHub as the main source of the project to be able to use concurrently and have the final version of the project in every one of our computers while working. For the 2nd iteration we started by changing and managing code with a better organization. We have met every week with an increasing frequency to be able to manage our time well enough to accomplish all the challenges we had with the implementation.

To conclude, we implemented all the additional features we intended to add. We also improved the game’s GUI both in the game and menus. We have faced a lot of problems in this period and fixed them as soon as possible and all the decisions have been made by the vote of all group members.

# Design Changes

## Added Parts

**FileDataStructure Class** - FileDataStructure object class that manages the primitive types that has been read from the external files.

**FileSystem Class** – FileSystem object class that manages the object types that has been read from the external files.

**Forest Class** – Forest object class that manages the forest object on the map.

**GameButton Class** – GameButton object class that extends the JButton class and game use these buttons instead of JButton.

**GamePanel Class** – GamePanel object class that manages the layout of all the panels.

**LevelView Class** – LevelView object class that manages the game and its levels with the map and other objects on it.

**SoundManager Class** – Manager class for the game that manages the audio files and implements appropriate listeners for other classes to use.

### Singleton Pattern

**SoundManager Class** is the one that can be used by every class that has been made on the structure of Singleton Pattern. Every other class may call static methods of the SoundManager to play and pause the background music as well as the other sound effects. It also has mouse and window listeners to be used by the other views.

## Removed Parts

**StableEnemy Class –** This class has been removed and replaced by the Enemy and EnemyArmada classes.

**StableAlly Class –** This class has been removed and replaced by the Ally class.

**LandForces Class –** This class has been removed and replaced by the Ally and AllyArmada classes.

# Lessons Learnt

* + - * 1. This project made us realize the importance of certain things pertaining to project management:
* Time Management and Work Distribution: Arranging meetings, and more importantly, spending the meeting time efficiently turned out to be a nontrivial task. We learnt that working on the project at different times by different groups of members (whoever is available) is quite effective.
* Communication: This project made us realize the importance of communication and breaking down of decisions and problems to be handled, and also their documentation.
* UML diagrams: These were an effective way of storing decisions made by the group and referring back to them whenever needed. Also, they were helpful to see the “big picture”.
* We also started to use GitHub more effectively and every group member learnt to use the GitHub for proper work flow and version control. It aided our work organization and allowed us to have the latest versions of the project in our machines and fix the bugs and implement new features concurrently without waiting for one another.

# User’s Guide

## System Requirement

Walls & Warriors is a Java based game, thus, Java Run Environment (JRE) and Java Development Kit (JDK) must be installed prior to start the game. 15 MB free disk space and 256MB of RAM and 256MB of VRAM are the minimum requirements to run and play the game smoothly.

## Installation

The GitHub repository does not contain a .jar executable file. To play the game a .jar file should be created after the compilation of the project. To do so, following steps should be taken in the given order:

**Windows:**

* Open a **Command Prompt** in the main folder.
* You can do it by **Shift + Right Click** -> Open PowerShell window here.
* Or you can do it by **Windows + R** -> Search for **cmd**, then change the directory to the main folder you are in with **cd** command.  
    
  *\*Main folder is the extracted folder* ***cs319-Walls-and-Warriors-master***
* Run **mkdir bin** *(Creates a folder named bin)*
* Run **javac -d bin src/\*.java** *(Compiles the files)*
* Run **jar cvfm Walls^&Warriors.jar Manifest.txt -C bin .** *(Creates a .jar executable file named Walls&Warriors)*

If **javac** or **jar** commands do not work for you or return an error message as follows:

*'jar' is not recognized as an internal or external command, operable program or batch file.*

*‘javac’ is not recognized as an internal or external command, operable program or batch file.*

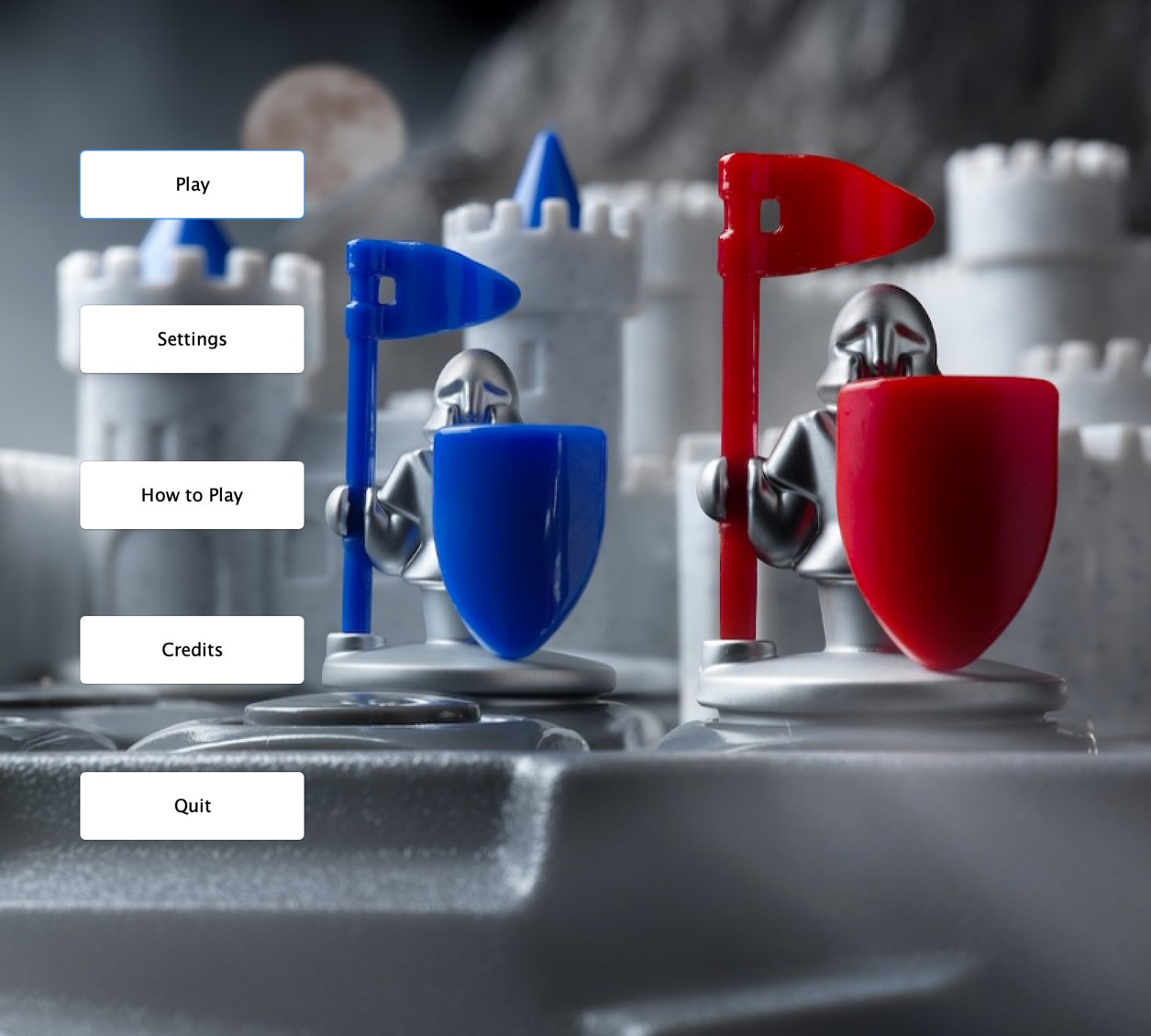
* Open a **Command Prompt**
* Change the directory to your boot disk (for ex. C:/) with the **cd** command. To go back in the path, use **cd ..** command.
* Run **dir jar.exe \s**
* Copy the file path that is as follows:
  + **[Disk]:\Program Files\Java\jdkX.X.X\_X\bin**(*for ex.* C:\Program Files\Java\jdk1.8.0\_191\bin)
* Run the commands as follows:
  + Run **“FilePath/javac.exe**” **-d bin src/\*.java**(*for ex.* “C:\Program Files\Java\jdk1.8.0\_191\bin\javac.exe” -d bin src/\*.java)
  + Run “**FilePath/jar.exe**” **cvfm Walls^&Warriors.jar Manifest.txt -C bin .**(*for ex.* “C:\Program Files\Java\jdk1.8.0\_191\bin\jar.exe” cvfm Walls^&Warriors.jar Manifest.txt -C bin .)

**Mac & Linux:**

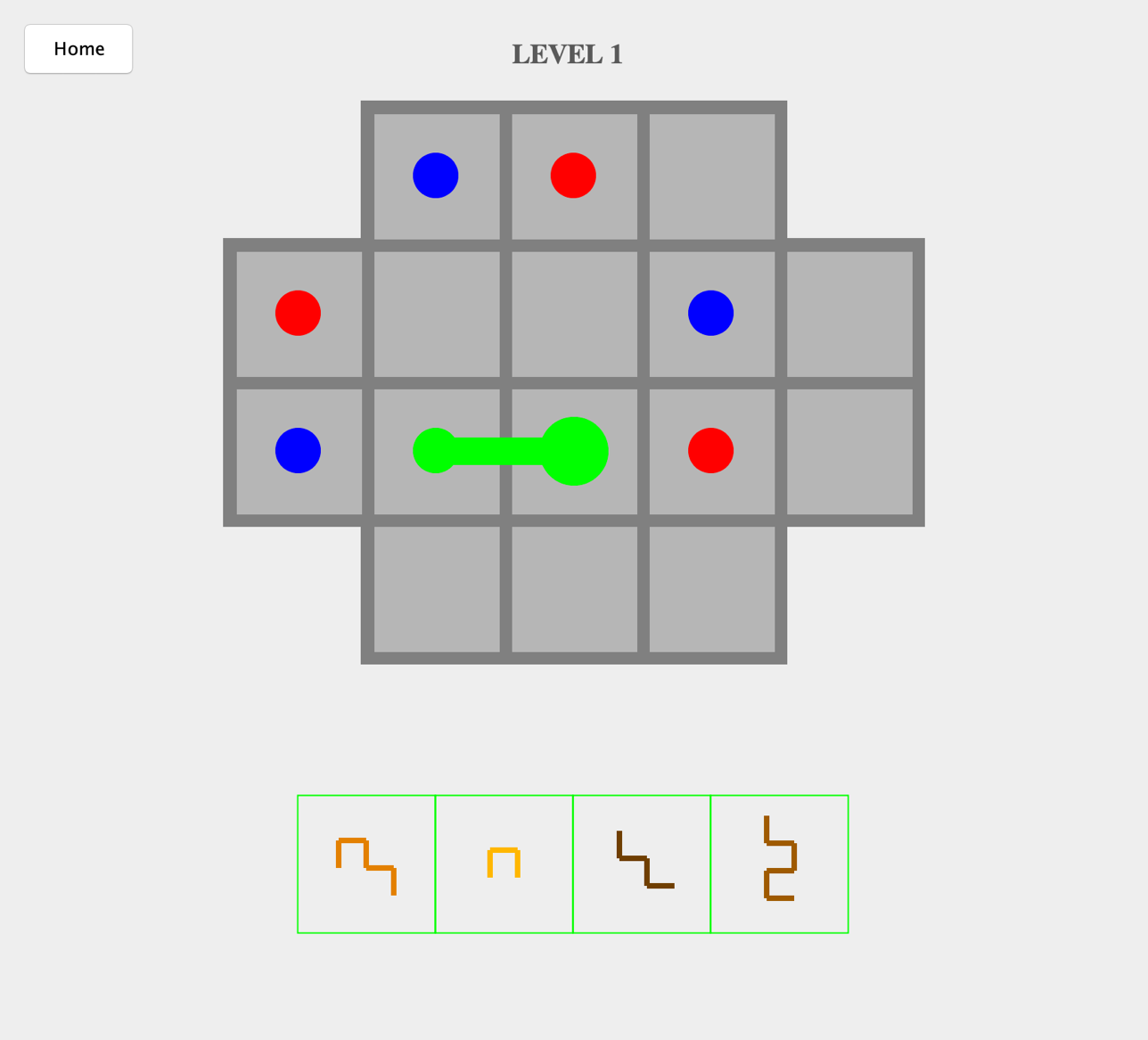
* + Open a terminal in the main folder.
* You can do it by **Command + Space** and search for terminal and press enter and then change the directory to the main folder with **cd** command.  
    
  *\*Main folder is the extracted folder* ***cs319-Walls-and-Warriors-master***
* In some distributions of Linux, you can right click in the folder and choose **Open in Terminal** option.
  + Run **mkdir bin** *(Creates a folder named bin)*
  + Run **javac -d bin src/\*.java** *(Compiles the files)*
* Run **jar cvfm Walls\&Warriors.jar Manifest.txt -C bin .** *(Creates a .jar executable file named Walls&Warriors)*

## How to Play

At the start of the game, the game menu is displayed, with 5 buttons as below.



The How To Play and Credits buttons display information about the gameplay and creators. Settings button opens the settings menu, and Quick button exits the game. Finally, the Play button takes the player to the Level Menu, where the player selects a level and starts playing.



In the game screen, the board and the available walls are displayed. The user can select the walls from the panel and drag them onto the map. The wall displays a clicking sound if the placing is successful. The walls can be rotated while in the panel by right clicking. When the level is completed, a victory sound is played and the player has two options to select from a popped up window: go to the next level and return to menu.