CS 319 - Object-Oriented Software Engineering
Design Report
Iteration 2

Defenders Of The Kingdom

Group 3-H

Alp Ege Baştürk

Barış Eymür

Emre Gürçay

Öykü Ece Ayaz

Table Of Contents

1. Changes in the Implementation

Classes from the design report were mostly the same however contents of the classes were changed according to implementation and improvements. Factories for levels 3 to 5 were implemented. There were only 2 factories for the first two levels in the design diagram. Also storage type of attackers and towers in the GameMap was changed from static array to ArrayList<Attacker>. Also towers are stored as a LinkedList<Tower>. This made dynamic control the objects in the game map easier and safer. We have implemented flags to determine if attackers are killed or not entered the map yet. With change to LinkedList, this became obsolete. However we did not change the remnants from the previous implementations because there were parts working according to this implementation and they continue to work with the new implementation. Also factories still return an array for attackers which is converted to an array list. We have not changed the factory implementation because converting the returned array to ArrayList was an easier one-line operation provided by Java libraries.

It was observed that FileController which was removed after the first iteration had some useful functions. For this purpose, new class with the name mapMatrixReader was implemented. This class only provides a static method to read a map matrix from the .mat file. This was implemented to reduce code repetition.

Furthermore Strategy pattern which was used for attack behaviour improved to include all attack operations in the game. Three new implementation of the interface, SingleAttackUpgraded1, HeroAttack, AttackerAttack, were added to the project. Currently Tower, Hero and Attacker classes use AttackBeheaviour implementations. This made it easier to manage attack behaviours of the objects and made dynamic changes easier. AreaAttackTower, Hero and Attacker classes have only one algorithm implemented. SingleAttackTower has two implementations for its algorithm. This allows dynamic switch when the tower gets upgraded. Tower, Hero and Attacker classes were implemented according to the pattern.

Base building was implemented as an object instead of a special tile location. Health of the base building is shown above the object as a healthbar instead of health outside of the game map. Attackers attack the base by the new behaviour implemented. This attack type also notifies Gamecontroller that attacker is dead. This results in the same functionality in the analysis.

Another important change was the storage of the parent classes in the children. GameObjects; Tower, Hero, Attacker stores the reference of the GameMap they are currently in. This allows them to send notifications to the map. Also GameMap and InputController now store references to the GameController. They notify changes in the map or mouse inputs to the GameController if necessary.

Also a class named Particle implemented for the attack animations. This class takes location and type information as parameters. Towers create instance of this class when they attack and add them to GameMap by calling addParticle() method of the GameMap from the reference. These instances are drawn by the game map. Particles notify game map similar to other objects when they need to be destroyed.

TowerListController class was improved to accommodate new operations. Changes were minimal; Array of the rectangles and images were increased in size and detected operations were changed accordingly.

InputController was upgraded to detect more operations. Now it also listens to the main game panel. Also this class stores the reference of the GameController which created it. It checks for the

new operations on the List , it checks the type of the operation and calls the add methods of the GameController by the stored reference.

GameMap implementation changed to provide methods for addition and deletion of the game objects mentioned before. Also methods to check if clicked position on the map is available or to get the type of the selected tower were implemented. These two operations are called by the Game Controller when user tries to add, upgrade or remove.

GameController was updated to provide changes mentioned before. Methods to check if tower or hero are deployable to the map were implemented. Methods to check if game is running or paused are implemented. Game loop is controlled according to these methods.

2. Status of the Project

Completed Parts and Bugs

Core functionalities of a tower defence game are implemented. There are functional towers attackers and a base to be defended. Player has a finite resource to spend. In addition, some extended features we planned to implement like upgrading towers, selling towers and deploying heroes are added.

Particle class was implemented for particle operations however animations are primitive. They are lines for targeted attacks and circles for area attacks. Animations may be upgraded for better experience.

Two types of towers implemented. One of them can be upgraded which changes its attack algorithm implementation with the use of the strategy pattern. There are 2 types of attackers and 5 levels. Each level has a different map and different set of attackers. Primitive attack animations for towers and heroes were implemented. However hero attack animation has bugs and it sometimes works. Also Mouse listener detects mouse passed on the main game panel. This results in background changes in the right panel when mouse moves in the top left corner of the game map. Mouse motion detection mimics the one in the right panel when it should not do. This is a cosmetic bug.

While game is running there can be exceptions and errors due to concurrent modification. We have tried to prevent null pointer exceptions occurring when an object is destroyed and another object tries to access it. Also we have tried to prevent concurrent modification exceptions thrown when more than one object changes one of the list of gameobjects. If one of these errors are catched, then one operation is skipped without any visible effects. However such operations were the main source of the exceptions and there may be unknown bugs.

In the main menu, if player clicks PlayGame button there will be a null pointer exception however this does not break the program. Player can continue by selecting the level before pressing Play Game. Also background color of the game buttons turn to black when they are pressed. This is a cosmetic bug and does not effect the functionality of the game.

Level scores and playable levels are read from and stored to a text file.

Parts Need to Be Done

Most notable requirement left is the animations. Game functionality is sufficient for a game however animations are basic. Also we have tried to implement better tiles however this resulted in a considerable drop in the performance, thus we have reverted back to basic implementation.

Game cannot be played from the jar file because we cannot read from the path in the jar file. Research showed that file paths and image types should be changed for reading from a jar file.

3. User's Guide

3.1 Requirements and Installation

Defenders of the Kingdom is a tower defense game implemented in Java, hence the in order to run the game the computer must have installed Java Run Environment (JRE). Game was built and tested on Java 1.8, thus this version is recommended. However game was written with Java.awt and Java.Swing, thus older or newer versions of java are expected to be supported.

The project is implemented using IntelliJ IDEA as the IDE. We have tried to generate a .jar file. We managed to do the generation by using command line and IDE tools, however files are not loaded from the file system properly if game is started from the jar. Thus an IDE like IntelliJ is required to run the game. Project file is on the GitHub page in the master branch, test1. Project folder is *CS-319_Group-3H/test1/.*

3.2 Overview of the Game

When the user runs the game, the main menu will be displayed. The levels will be displayed for the user to select. User will select the level which he wants to play by pressing on the button of the level. Then he or she will click on the Play Game button to play that level. User will be able to deploy towers. These towers can be sold for a fraction of their price or they can be upgraded. Hero can be deployed in front of the base. Hero attacks the targets and block the road of the attackers. It pushes attackers back. Hero stays on the field for a limited number of attacks to provide an advantage as an expensive last resort defence. Player receives a gold bounty for each attacker killed. When attackers reach the base they attack it. This attack reduces the health of the base by one and also kills the attacker. Game finishes when there are no attackers or the base health is zero.

3.2.1 Controls

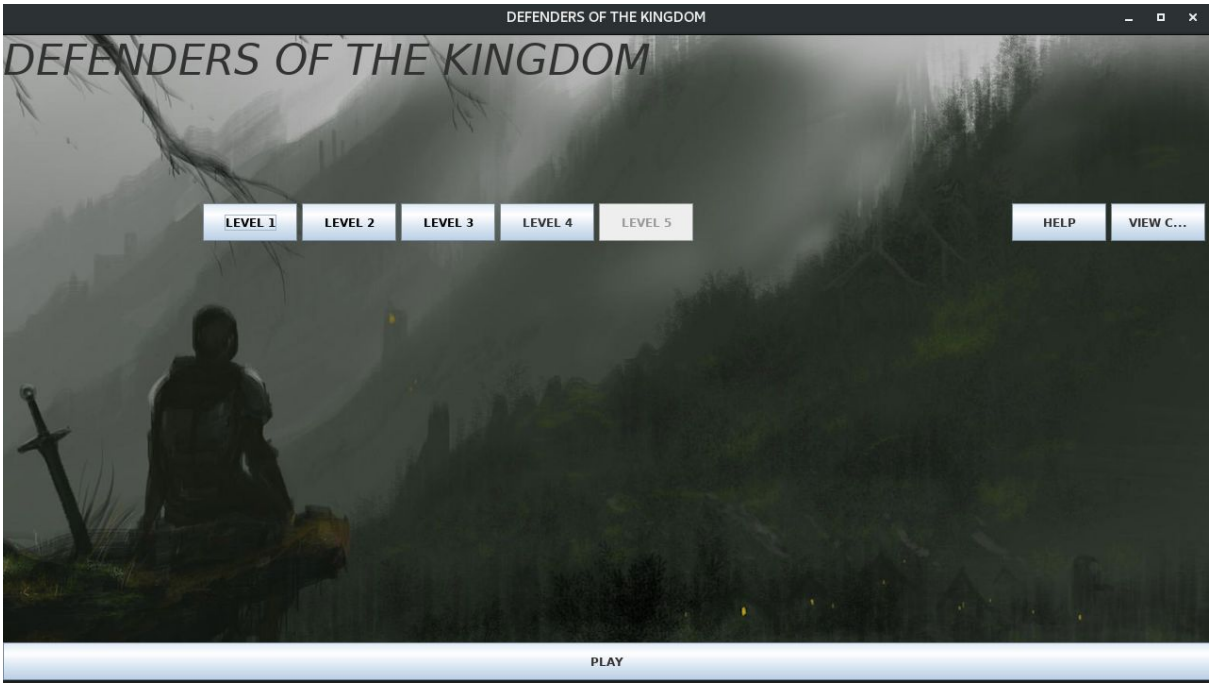
The mouse will be used to interact with the towers. Player will select towers, upgrade and sell options from the list and select deploy location on the map with the mouse. If a player chooses a hero hero will be deployed immediately on the map.

3.2.2 Game Screens and Menus

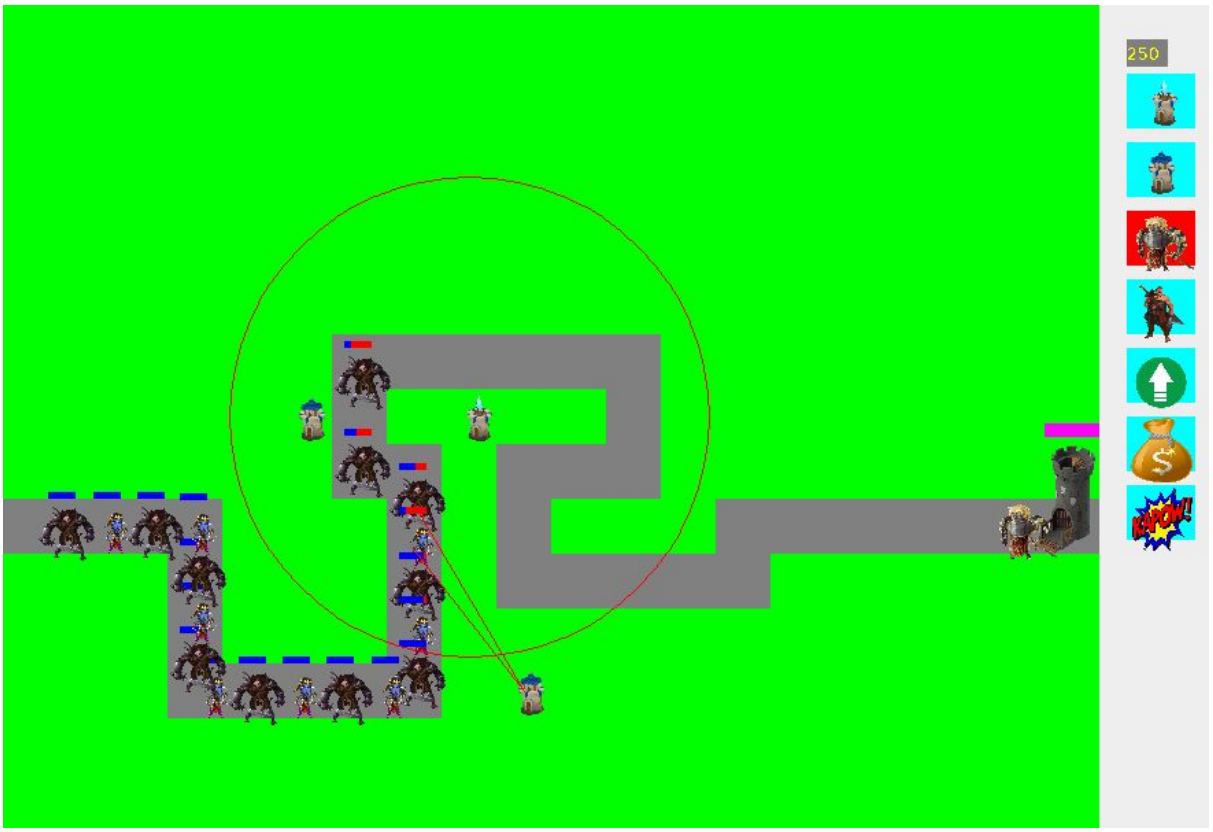
Main Menu

When the player runs the game, the below screen will be displayed. Only the First 4 buttons are enabled in the screenshot. Level 5 is disabled because player has not completed the Level 4 with at

least 1 star. Player will be selecting a level and continue by clicking the “Play” button at the bottom of the page to play the game. Help and Credits buttons show Help and Credits respectively.



Game Play



Game map is shown above. Attackers seen on the path can move. Path and green fields are made from Tile objects with the information read from the matrix file.

Right Panel

The towers, heroes, up arrow and coin bag images on the right can be selected. If mouse is on an element on the right panel, background changes to red. If a player selects a tower then he can place it on the map by clicking on a position in the map. Player cannot place on top of other towers or any location in the path. Heroes are deployed in front of the base when they are selected. Player's gold amount is shown on the top of the right panel. Player should have enough gold to deploy to map. Up arrow and coin bag work similar to tower placement. Up arrow is upgrade which upgrades a towers. Currently only single attack towers can be upgraded. Coin bag sells a tower and returns a fraction of its cost. Button below the coin bag pauses the game.

Game Objects

Attacker

This is the enemy which player is defending against. There are two types shown in the image above. If they move to the same tile as the base, they decrease the health of the base by one and die.

Area Attack Tower:

This is the tower in the middle of the circle. It attacks all attackers in its range. It can be deployed from the right panel by selecting the first tower in the right panel.

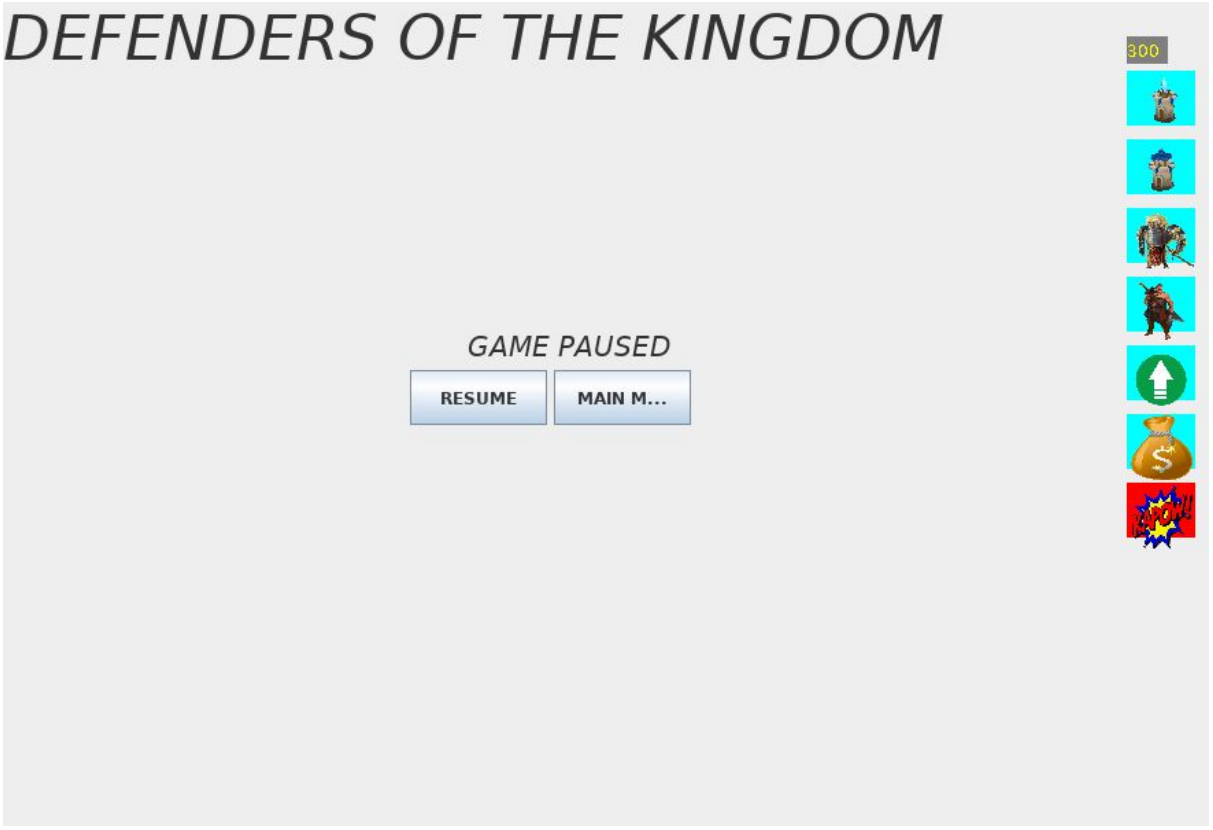
Single Attack Tower:

This is the tower is the one at the one end of the attack lines. It can be deployed by selecting from the right panel. It also can be upgraded by selecting the up arrow from the right panel and clicking on the tower to upgrade. Upgraded version can attack two targets at once.

Hero

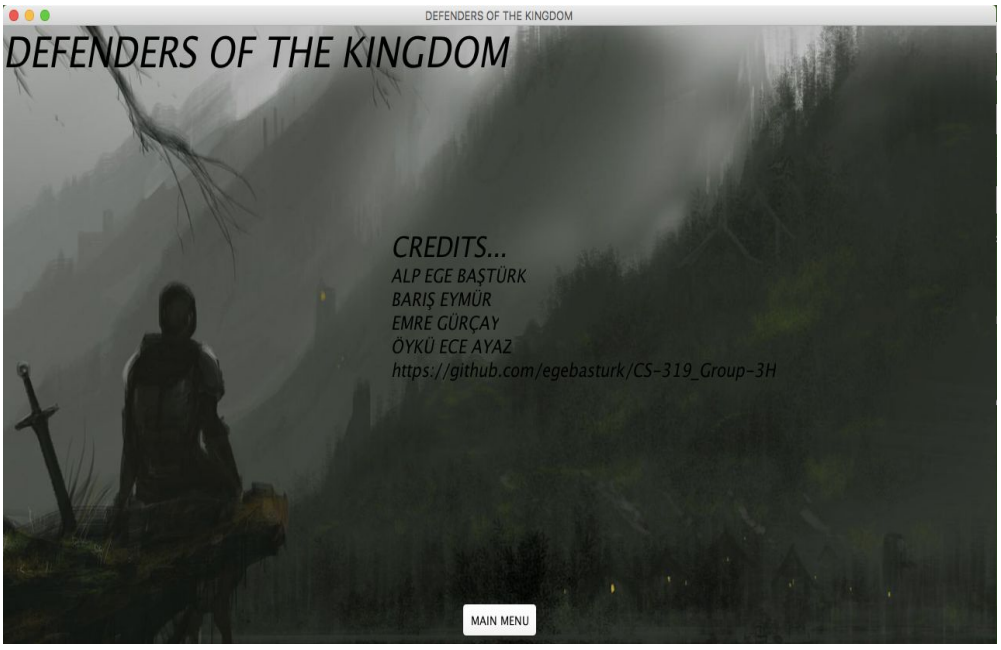
There are two types of hero. They stay deployed for a finite number of attacks. They block the path of the attackers, attack them and knock them back. First type is the one deployed and the one with the red background in the right panel. Second type is the one below it.

Pause Menu



Pause menu is shown when player presses the last element in the right panel, which is the one with red background in the image above. This pauses the game. Player may click resume game to continue playing or go back to main menu.

View Credits



Credits page in which the developers of the game and the GitHub link is presented is below.

View Help

Help page aims to help the user in the gameplay.