CS 319 - Object-Oriented Software Engineering

Final Report

Iteration 2

Medieval Tower Defense

<u>Group 1-A</u>

Buğra Aydın

Berke Deniz Başaran

Mahir Özer

Table of Contents

How the Implementation Went:

First Iteration

Since the doctrine of iterative development was adopted, there were two iterations made to complete the project. In the first iteration, more primitive use cases were planned and implemented since the main goal was producing the software with its essential ingredients of game engine, GUI components, basic function of game objects such as shoot projectile function of Tower. The implementation in the first iteration of the game was not challenging since the use cases were basic, and time was not a big constraint. Only concern in the first iteration time period for us, developers was to get to know each other and what are the qualifications of each of us individually in the production process.

Second Iteration

In the second iteration phase, existing diagrams are improved, and the existing use cases are extended. In the time period of second iteration, main goal was to polishing the existing software and extend the functionalities that game objects can perform. The process of testing and improving the software system was frustrating, and more time consuming. However, division of tasks and planning for the implementation was not a big concern anymore. Due to this case, frustration was not very big. The importance of soft skills such as communication, cooperation, professionalism, and flexibility are not underestimated. One of the most important lesson to learn from this development cycle is that soft skills matter as much as hard skills such as coding, and testing. Also, in the implementation phase, we needed to abandon some design patterns in order to catch the deadline, however, this abandoned design patterns will be implemented to the software later on.

We also learned the importance of the testing software. In order to have a bug free software working as intended, there has to be a lot of unit and behavioral testings.

Requirements for Using the Software

Operating System: An operating system that supports the JRE is required[1]. This game is designed for PC.

JRE: Java SE Runtime Environment version 8 or more is required. Without this software installed on the computer, game's executable jar file will not work.

Mouse and Keyboard: These devices are required so user can control the application.

Monitor: A monitor with at least 1280 width and 1024 height dimensions is required so that game frame can fit on the screen.

Disk Space: At least 150 MB of disk space in the computer.

User's Guide

Game Control

The main control mechanism specified for Medieval Tower Defense game is the mouse. Player can navigate through main menu by left clicking the button elements. There are two additional mechanisms for control which are spacebar key on keyboard. The spacebar key is used to move back from gameplay to the main menu. User is able to left click tower icons in the shop menu then by left clicking on a tower grid slot, tower will be placed. If the user clicks an already placed tower, it will level up and gain increased attributes.

Main Menu

Once the requirement for Java Runtime Environment is completed, player can open the game application by double clicking the "Medieval TD.jar" file provided. This file contains the main functionality of the game. When the game is opened, player can interact with different sub menus which is done by clicking one of the 6 buttons.
These 6 buttons and their functionings are provided below:

1. **Play Game:** Starts the game by generating a random map, initializing the game objects, and initializing the manager classes which rule the corresponding game objects.
2. **Settings:** Allows player to modify the game difficulty and sound level. Sound level can be changed with the slider that resides on the right of the *Sound Level* label. By

[1] http://www.oracle.com/technetwork/java/javase/certconfig-2095354.html#os

moving the slider cursor right, sound is increased and by moving it to the left, sound is decreased. The game difficulty is set by clicking one of the buttons on the right of the *Difficulty* label named as: “Sandbox”, “Easy”, “Medium”, “Hard”, and “Hardcore”. Sandbox being the easiest and hardcore being the hardest, game difficulty scales between the 5 buttons. There is also a “Default Settings” button, which sets the sound level and game difficulty as medium. It resides beneath the difficulty buttons. Due to the deadline’s arrival, the setting operations couldn’t be implemented yet.

3. **Information:** In this section, the player can get information about enemy and tower types. Before visualizing the types for each, player needs to identify the game object of concern. For example, the user wants to get information about the arrow tower. Initially in this case, *Tower* option must be chosen from the menu bar at top and then, via the second menu bar player can finally choose the *Arrow Tower.* Same method shall be used to choose a specific enemy type. This information screen is a really important aspect in the strategy games, also called the tech tree. User can gather information about the game and come with different strateties from here.

4. **Help:** This section consists of two buttons, which display different texts on the pane beneath once clicked. These buttons are named as: “Control” and “Goal”. The text of control button reminds player to use the mouse for control mechanisms, such as buying a tower from shop menu, upgrading an existing tower in game or navigating through the main menu. On the other hand, the text of goal button provides information about the victory conditions.

5. **High Scores:** The panel in the section of high scores will display the 10 highest scores and the players achieved them.

6. **Credits:** Name of the developers, game version, and date updated will be displayed in this section.

Beside these buttons, there is also an “Exit Game” button for terminating the game, and “Back” button in each panel for navigating back to the main menu. There is an exceptional case for navigating back from the gameplay panel and it’s by pushing the space button.

Gameplay

The game starts when player clicks on the “Play Game” button on the main menu. The game map consists of two types of ground slots and it’s randomly generated. Towers can be placed on the green slots, and enemies will follow the brown path that disallows any tower placement***(See image .1 and image .2 in Appendix)***.

Player can see the game status from the bar at the bottom of the screen from left to right.

There are four game status visualized at total, and these are described with images corresponding each of them:

- The shield icon shows the amount of health points that player beholds.
  (see image .3: shield icon)
- The golden coin icon shows the amount of gold that player beholds.
  (see image .4: gold coin icon)
- The watch icon shows the elapsed time in the format of mm:ss.
  (see image .5: watch icon)
- The helmet icon shows the current enemy wave.
  (see image .6: Helmet icon)

At the section slightly upper of this bar, shop section stands in which icons for each tower are shown***(see Appendix image .7)***. By clicking one of those icons, and clicking the location of slot on map, player can buy a tower. After a tower is bought, cost amount fixed for this tower will be reduced from the total amount of gold. Existing towers can be upgraded as well by left clicking on the tower of choice. When the game starts, first wave of enemies will appear in the time mark of 0:3. The location with pentagon image represents the spawn point of enemies***(see Appendix image .8)***, while the location with castle image represents the enemy destination***(see Appendix image .9)***. The player must act quick and place the towers so that enemies can be dealt with before they reach to the end location. If the player allows trespass of enemies 10 times, the game will be lost. Instead, if player endures against every enemy wave, game will be won. In both cases, the consequence of the game will be notified to player . Tower placement shall be made carefully, so that more enemies can be harmed with less number of towers, which means by using less amount of gold. A gameplay instance is shown with *appendix image .8*. There is a randomly created map, enemies with health bars, and towers placed to defeat the enemies.***(see Appendix image .10)***

Changelog

Added & Upgraded Functionality

We turned from the grid-based tower ranges to pixel-based ranges, this decision made our implementation much easier and we were able to get much more flexible results when implementing variety into towers.

In our on-hit-effects for the particles, we use a variety of animations using different framerates. We do this by a single function which takes the animation framerate, and the directory for the 2d-animation files. Note that we use no arrays for this implementation, the animation sequences are determined from the image file names, which makes adding new animations much easier for the developers and there is no memory requirement for an array either, i.e. it's a 1 minute job almost to add a 2-D animation with any framerate(There is the limitation of 1000 FPS(i.e. an animation can't be more than 1000 fps) because we were working with integers). This implementation was for the 2-D animations over 10 FPS(frames per second). We used Java's Timer class for this implementation.

For the animations with a lower framerate than 10fps, we used a simpler method with if statements and executed animation sequences each time the remainder of gameFrameCount with the animation fps equals to 0.

Unlike in the old version, high scores can be displayed in this version if player fulfills the condition which is to achieve a high score. The high scores are kept in and read from an external txt file the scores of the previous players are also kept.

New sound effects for enemies and towers are added with the update. Each different tower plays a different sound effect, and each different enemy plays a different sound effect when they die, or when a projectile hits on.

In the new iteration, Health bars are also added to the enemy units. Health points of each enemy unit on map are visualized with a bar that appears above them.

New enemy units are added to the game with new version. There was only a single type of unit implemented in the previous version, yet now, there are 16 new units added, each have moving animations. Also, the enemy classes in the first iteration and second design was changed since we used the tilesets from Warcraft 2 and the corresponding enemy classes was not matching.

For a balanced gameplay, attributes of towers and enemy units are updated. In the current version, each tower has different characteristics just like each enemy unit does have.

For each enemy unit, there are new information texts written in the upgraded version.

Random map generation is added with the new version. This functionality is provided with a new class called RandomMapGenerator which modifies the game map in each new game.

GridSlot images are constructed from 10 random tower grid slot and 10 random enemy grid slot images now. One grid slot image consists 4 of these images. Ending point of the enemy line has a castle image now, and the starting point has a enemy spawn image also. This changes upgraded the visual quality of the game.

Game declarations such as gold, time, current wave and time was made and game logic such as game over, buy tower etc. were implemented using them.

Shop interface is refurnished with a new background and new icons that represent different towers.

Changes Due to Complications:

In the first iteration main game logic was embedded into the gameplay panel which beholds the main function as a GUI component. In the current version, there is a new class implemented that holds the game logic as a non-GUI component. Now, gameplay panel calls the instance of this new class to initiate the gameplay.

Game objects, which are towers, enemies, and projectiles used to be managed by the arrays. In the second iteration phase, arrays are replaced with arraylist objects, which provide better functionality and better dynamism.

Enemies are used to follow a pre-determined path in the first iteration. This path was defined in the GameManager class. In the upgraded version, enemies now move according to the enemy grid slots specified as enemy path.

WaveManager and Wave classes are removed due to the unnecessary calls they make. Their functionality is embedded to the EnemyManager class in the new version.

A new arraylist called graveyard is defined so that dead enemies can be finalized better.

Status of the Implementation

Traps were introduced in the 2nd iteration, yet due to the deadline date's arrival and absence of corresponding image sets for them, they had to be abandoned for the current version of the MTD.

Settings are left undone and still not integrated to the gameplay yet. Yet as GUI components, they still exist and can be accessed via the "Settings" button in the main menu.
In the current state of the implementation, pause menu is absent.

Other than these three requirements specified in the analysis, rest of the functional requirements specified are implemented and complete. The game units such as towers and enemies need some balance on their health and thier damages and it will be patched according to the user feedback.

Current functionalities: Enemy AI, Random Map Generation, Sound Effect and Music, Play Game with different enemies and upgradeable towers,shop, different waves, boss waves, high scores, help and information, credits, complete UI, exit.

Credits

Game tileset images are taken from the Warcraft 2 and Warcraft 3, with a permission from Blizzard Entertainment. I contacted them on their live support and asked them if it would be a problem if I use their sources for educational purposes and they told me that their game license allows me to use it.

Game sounds were taken from this website, which is an open source game element sharing platform
https://opengameart.org/

Appendix

(image .1: enemy grid)

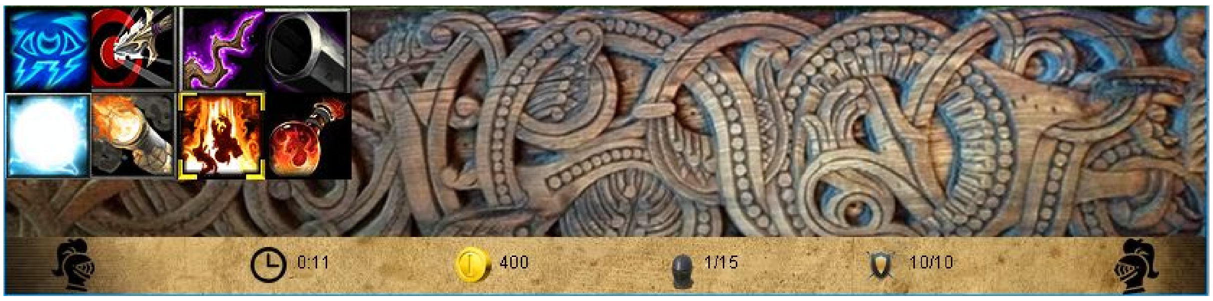
(image .2: tower grid)

(image .3: shield icon)

(image .4: goldicon)

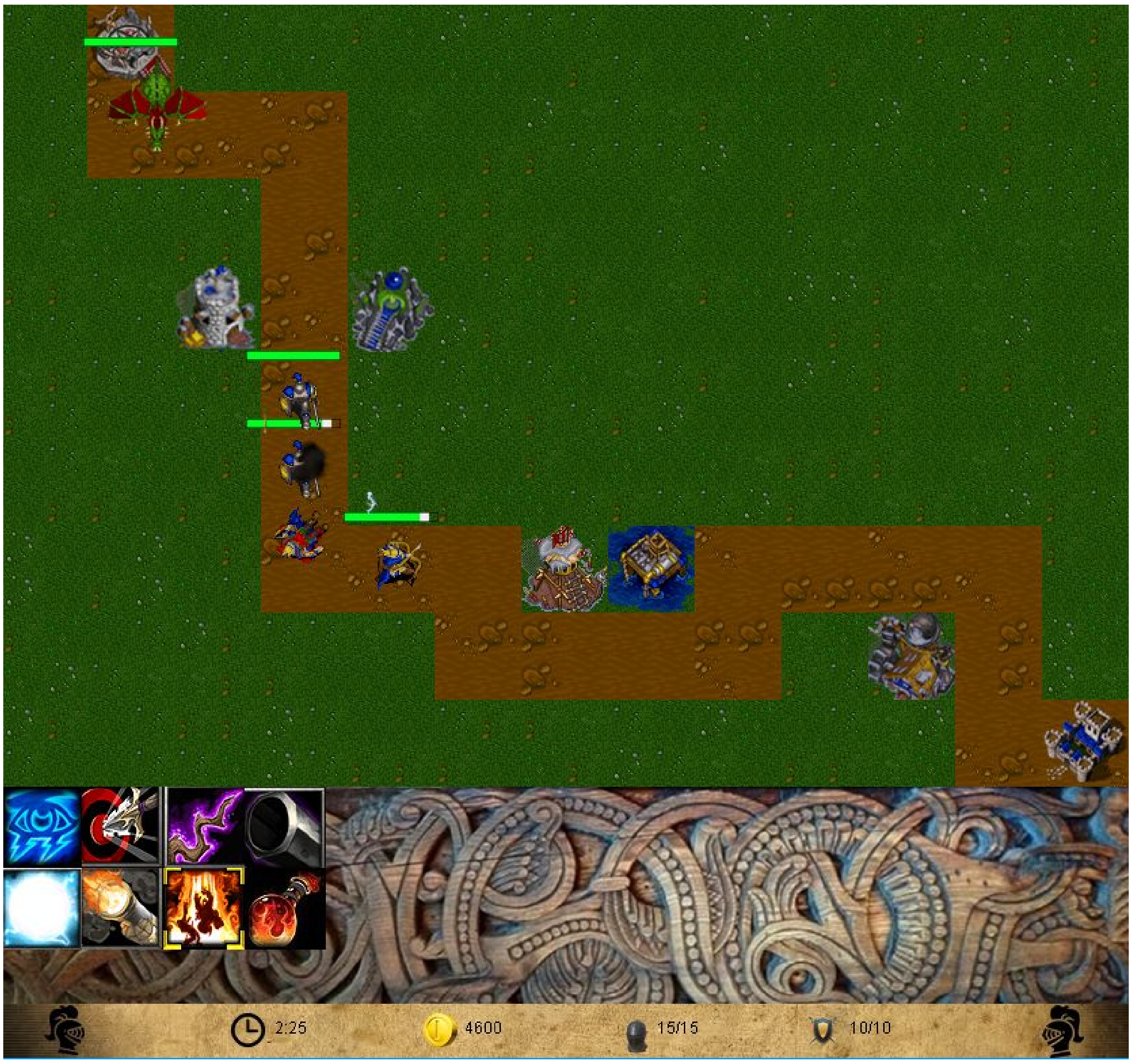
(image .5: watch icon)

(image .6: Helmet icon)



(image .7: Shop menu)

(image .8: Enemy destination) *(image .9: Enemy Spawn Location)*



(image .10: During gameplay)