Bilkent University
Dept. of Computer Engineering

Object Oriented Software Engineering Project

RoM: Redeemers of the Monarchy

Final Report

Project members: Efe Ulas Akay Seyitoglu, Erkam Berker Senol, Nashiha Ahmed
Course Instructor: Ugur Dogrusoz

Final Report
December 24, 2016

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object Oriented Software Engineering Project, CS319

Contents

Final Report

Figure 1: Our Game. (Finally!)

1 Introduction

Our project is a "Tower Defense" type game. It is a Java desktop application. Tower Defense is a strategy game. In Tower Defense games, players aim to defend their territories or possessions from enemies by destroying them before they reach the endpoint. Popular Tower Defense games include "Missile Command," "Plants vs. Zombies," and "Kingdom Rush." Our CS319 project uses this idea for inspiration.

In our game, there are separate "waves" of monsters with distinct powers and lifespans that try to reach the end point. By placing obstacles in their path, the player tries to prevent the monsters from reaching the endpoint. These obstacles can be towers with special powers that kill the monsters passing by them. If the player is still alive after all the waves, the player wins.

This report is the Final Report of the project. It includes both exposed and unexposed documentation. The user's guide, status of the implementation, and general API documentation are especially detailed in this report.

2 User's Guide

The user must have a Java environment to use our game as stated in the Design Report. The user can either download a zip file of our game from our github repository (https://github.com/ebsenol/CS319-Group23) so that all the additional files, such as sound files or images, are availble for the game to run. Then the user can run the program as he or she chooses, such as using an IDE. The user can also run the .jar file but must keep all the additional files in the same folder. This should be accessed from our github repository as well. Other than this, the

game requires no setup. Simply running the game should ensure that it works. If the game for some reason runs into an error, we request the user to simply restart the program ensuring the user that we have done our best to avoid and remove bugs and error. The user should not change the code of the game, unless the user coding experience. The game should be played like any other game. If the user has any confusion, the user can refer to the tutorial. If the user has any issues, concerns, suggestions, or simply wants to reach out to us, the user can find contact information in the credits menu of our game.

3 Changes to Implementation

We tried our best to keep the implementation as similar to the previous reports as possible. One of the changes is that the game look was improved upon. The following background image was used instead of the previous one showed below the new game background.

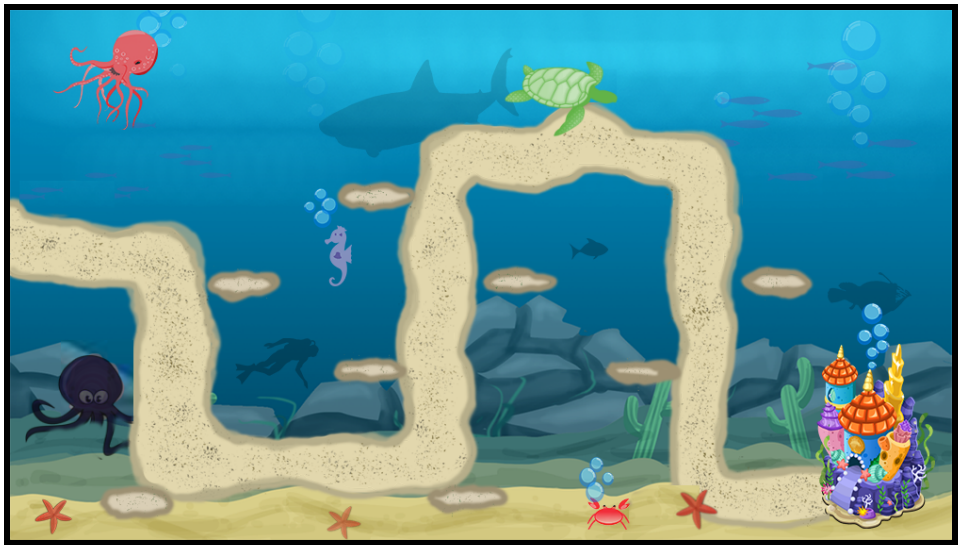

Figure 2: The New & Improved Game Background [3] [4] [5] [6]

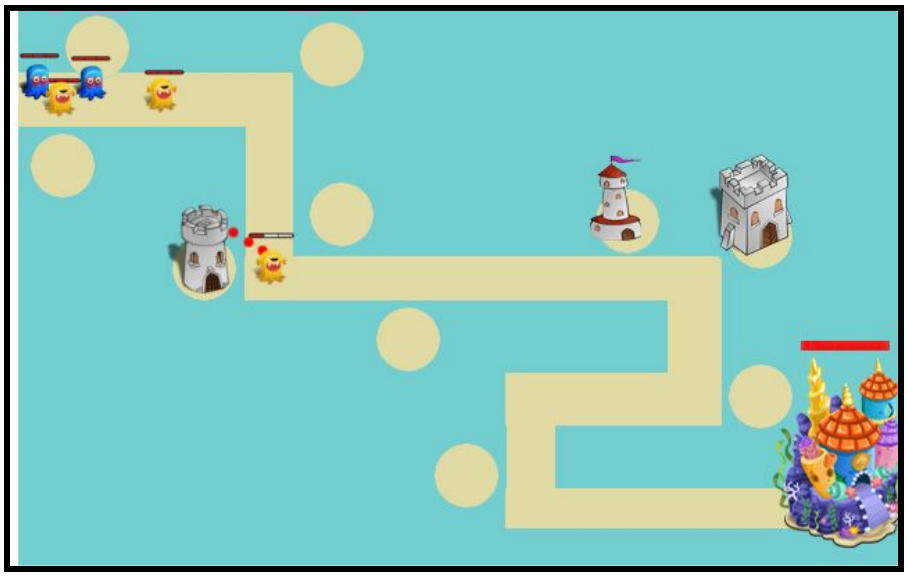

Figure 3: The Old Game Background [4] [7] [8] [9] [10] [11] [12]

Many of the use-cases are met. The player can play the game, view and change the settings, the high score table, and credits. However, unfortunately, some

things could not be implemented due to shortage of time. We did not have enough time to include sound effects, for example. However, music was added. The pause menu was not implemented completely. The coin, health, and wave values did not increase correctly either. Ten levels of different number and kinds of monsters was implemented. The player can add different kind of towers as well and may upgrade the tower according to the coin count of the player. The towers successfully hit the monsters and kill them when their lives are finished.The user can view the settings, credits, and so on and return back to the Main Menu. The menus look almost exactly as planned in the previous reports.

We followed our design pattern and object model almost exactly, but realized that some classes were better off not being used, such as the GameObject class. The UI used CardLayout() to implement the navigational path of the UI Objects.

4 Complications During Implementation

Although we tried our best, in general the time shortage was the biggest limiting factor of our implementation not being perfect. However, we ran into several other complications as well. The movement of the monsters was difficult to implement on the path as it was time dependent but also independent of the background image. Sound was also difficult to integrate with our program. Another complication was pulling and pushing from/to Github. It was difficult to integrate our code for example, when a team member did not push the most recent version of their code or pull the most recent version of the program. Placing the tower buttons on the map and towers was also difficult because the towers were independent of the background image which made coding to place them quite tedious. In general, designating different parts of the program to different group members and integrating them all together afterwards can be quite problematic even if this gap can be made smaller through a more thorough design. This is because different team members can have different ways of implementing even if the same design was agreed upon.

5 General API Documentation

A general guide for developers is explained briefly. As stated in previous reports, our program follows and MVC pattern. Relevant code of Entity Objects such as Monster and Tower can be found in the Model Subsystem. All the GUI components can be found in the View Subsystem. Logic behind the game and other control classes can be found in the Control Subsystem. Most panels use either a GridBagLayout or a BoxLayout, except for the MainFrame which uses CardLayout so that it can switch between panels.

6 References

[1] "CS 319 Object-Oriented Software Engineering." Accessed November 11, 2016. http://www.cs.bilkent.edu.tr/~ugur/teaching/cs319/.

[2] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[3] "UnderWater Game Background" <http://img04.deviantart.net/7cae/i/2013/049/b/2/underwater_prairie_by_toastwafflea rt-d5vdonj.png>.

[4] "Castle"<http://vignette1.wikia.nocookie.net/tinyzoo/images/b/b9/Underwater_Castle.png/revision/lat est?cb=20130105133327 >.

[5] "Octopus" <http://clipartix.com/wp-content/uploads/2016/09/Octopus-clipart-free-images-5.png>.

[6] "Turtle"<http://dbclipart.com/turtle-clip-art-image-7661/>

[7] Tower Clipart, Viewed 10 October 2016, <http://www.clipartpanda.com/categories/tower-clipart>.

[8] Tower Clipart, Viewed 10 October 2016, <http://www.clipartpanda.com/categories/tower-clipart>.

[9] Tower Clipart, Viewed 10 October 2016, <http://www.clipartpanda.com/categories/tower-clipart>.

[10] Tower Clipart, Viewed 10 October 2016, <http://www.clipartpanda.com/categories/tower-clipart>.

[11] Slimy Blue Monster, Viewed 10 October 2016, <http://iconbug.com/detail/icon/2063/slimy-blue-monster/>.

[12] Yellow Cyclops Monster, Viewed 10 October 2016, <http://iconbug.com/detail/icon/2048/yellow-cyclops-monster/>.

[13] Green Monster With Eye Patch, Viewed 10 October 2016, <http://iconbug.com/detail/icon/2059/green-monster-with-eye-patch/>.