CPEN 321 - Team 38 Open Source Project: Tower Defense Game Improvement

Quentin Fu, Jerry Zhang, Kevin Chan

1.Introduction

This project involved improvements made to an open-source simple Tower Defense game built on Processing 2 in Java. The game is designed by developer Heilion and uploaded onto the github repository https://github.com/Heilion/Tower-Defence. The workload distribution of each team member is as follows:

1.2 Individual Contributions

Kevin Chan added second tier tower upgrades with different functionalities, updated game graphics by importing newer background, shields, upgrade buttons and tower visual effects.

Qiantong Fu designed and added new types of minions in the game to diversify the gameplay experience. He also wrote the project report and acted as the scrum master of the group during weekly meetings.

Jerry Zhang created a menu that allows player to select difficulty of easy, normal or hard. The game starts on the menu and once game over (whether win or loss) will also redirect player to menu after a mouse click. Jerry also organized the presentation and acted as the main speaker for the presentation.

1.3 Run-time Environment

The reason that we chose this specific project is that compared with other programming languages, we are more familiar with Java. We also used Processing 2 in EECE 281 project course in second year, so familiarity with the IDE is also an incentive to work on this project. Furthermore, many of our peers plays tower defense games such as Bloons Tower Defense so we found interest in working on such a program. Previously we had considered several other possible project choices, but had to abandon them either because we are unfamiliar with the run-time environment or we are inexperienced with the programming language.

1.4 Added Features

The game originally has 4 different towers, 5 types of minions. On top of that, we added 3 more upgrades for the towers and 4 more minions. We also implemented a start menu with 3 difficulty options and improved the graphics of the game.

2.Work Accomplished

2.1 Challenges faced

Since the whole game is built on Processing 2, the biggest challenge for us is to understand the relatively complicated layers of the game, then modify on top of those layers. Since the game is uncommented and hardcoded, it is difficult for developers such as ourselves to fully comprehend the implications of each class. So there’s limited room for improvements and changes across the program. As a result, we learned through this project the importance of coding format and consistency across a program so that future developers are provided with a convenient way of providing changes.

Another challenge that we faced involves the high memory usage of software programs, especially tower defense games. Based on empirical observation of other tower defense games such as Bloons Tower Defense, when many towers are active, the many processes leads to mild or even extreme lag in frame updates. This is the same case with our tower defense game and it leads to subpar gameplay experience for users.

2.2 Software Engineering Process adopted

Our team used a combination of extreme programming and scrum methodology disciplines. In terms of extreme programming we utilized pair programming and casual continuous integration. In terms of scrum, we held frequent informal scrum meetings to share updates and brainstorm ideas on issues. We also created user story and scenarios to solidify our objectives in this project. Finally we used requirements elicitation to gather ideas to form objectives on the project.

2.3 Works Accomplished

We managed to design 4 more towers with distinct features. The first one, called arrow, can fire at multiple targets. The second one, called Arnold, fires a line of bullets at the wherever the mouse pointer is at. The third one, called laser, fires a beam that does more damage. The fourth one, called cripple, slows that enemies that it hits with a slow multiplier. We also improved the game graphics such as the background and visual effects by importing pictures.

The new minions are added by extending the existing enemy class and we made different models for each of the new minions using the draw() method. Each minion consists of 3 parts: base, hat, and healthbar. For each minion class, we override those methods in order to produce different appearance. Those minions follow the path generated from the AIpath() class as well. Apart from the appearance, we modified movement speed, health, shield and regenerating by overriding as well. More specifically, the first minion we added, called Demacia, is golden and after eliminating will grant the player a huge reimbursement. The second one we added, called Corki, has high mobility. The third one, called Voodoo, has ability to heal the nearby minions. The last one, called Leviathan, is an upgraded boss which possesses both shield and health regenerating abilities. The reason for the upgraded boss is to reconcile with the different difficulties.

This project is built on top of a fully functioning simple tower defense game. Our team managed to add new minions, towers, visuals, and menu interface. Since the IDE is Processing 2, it is not feasible to use unit tests and code coverage plugins to obtain statistical contributions by our group. However, as a qualified estimation, we would say that we have significantly improved the overall experience of the game in terms of visual experience, increased player autonomy and depth of the game.

2.4 Role and collaboration

To reiterate the introduction of this report, members of our team split up the actual coding of the project into 3 objectives: menu interface and graphics, minion improvement and tower improvement. We worked together often in person and committed our changes incrementally to github in a continuous integration manner. Each of our member also had a role outside of purely coding. Quentin was responsible for creating the project report, Jerry was responsible for organizing logistics of meetings and Kevin determines whether our updates to the project was sufficient enough.

3. Conclusion:

Processing may not be the most ideal IDE for game developing. However throughout the project we begin to understand the importance of having a dedicated program for future developers to conveniently understand and work on code. Applying engineering process to our work helps accelerate our progress and is certainly helpful for our future developing process.

Through this project report, we have gone through the 3 main objectives of our project and how we managed to complete these goals. We also talked about the methods that we used during development to improve communication and organization. Finally we addressed challenges that were faced when working on this project. Although we were not able to resolve all challenges, we believe that there was much to learn from understanding and analysing the implications of challenges. Which ultimately helps us mature as software engineers.