CPSC 300: Software Engineering

CDR Tower Defense

Final Project Report

Colton Kenney

Dylan Calado

Raymond Strohschein

December 1st, 2017

Project Description (Scope and Purpose):

This report describes a software called CDR Tower Defense, which is a strategy video game written in Java. The game has been developed for the purpose of a final project, in CPSC 300: Software Engineering, during the 2017 fall semester at UNBC. It contains many of the standard features that tower defense games have generally and even includes some features we thought of ourselves. Our team's take on this type of game boasts a cool looking computer science theme and innovative features, such as combining different tower types, for a wide range of custom tower characteristics. This game is meant to be easy to learn how to play: It includes a short game guide to get new players up to speed on how the game works, as well as in-game descriptions, for the different defenses and their properties. The purpose of developing this type of product was to see if we could work efficiently together as a team, and challenge ourselves, to successfully develop a video game using the software engineering theory that we learned in this class. The development of CDR Tower Defence focussed heavily on computer graphics, as well as algorithms that required precise logical steps (such as path creation and path traversal).

Project Requirements:

- A main menu to start the game, view the game guide, or select a map to play on.
- A game play window to display what is happening in the game.
- Side bars/menus for purchasing or upgrading towers, and also for displaying text.
- Enemies who follow the (arbitrary) path and deal damage if they reach the base.
- Towers that can be placed on any tiles that are not path tiles. Fully upgraded towers that can be combined with other types of towers and upgraded further.
- Tower names that represent different defense mechanisms that computers utilize.
- Towers that attack enemies within a certain range, via a health draining affect.
- A tower range indicator that shows the player the radius of a specific towers attack zone.
- A health and currency system for enemies (enemies give you currency upon death)
- A health system for the CPU base (independent of the enemy health system).
- User friendly instructions to give new players tips on how to use the different features of the game.
- A sleek looking graphic design with pleasing colors and shapes (via a computer science theme).
- Sound effects, including background music (can be toggled) and enemy death sounds.

- A variety of possible map layouts to play on.
- Error checking code to ensure all data files consist of the valid format.
- Boss enemies (appear every n rounds as a single enemy with a large amount of health).
- Documentation containing all the information on the timeline of development as well as descriptions for the purposes of the packages and classes that make up the game.

Software Engineering Methodology Used:

For the duration of this project, we have been using the incremental development software engineering methodology. This style of documenting and coding has proven to be ideal for developing our video game. CDR Tower Defense started out with the most minimal features required to play it, and throughout the semester, we have added more fun and helpful features which have increased the complexity of what our game (and players) can do. Our initial assumption, that having several playable sub-versions would increase our overall chances of success, has shown to be very accurate, as we have successfully implemented all the core features of our game well ahead of the deadline. This left us ample time to casually increase the games complexity without having to work under a lot of pressure. In the end, this fact likely helped CDR Tower Defense become a better game than it would have if we were working under pressure, and rushing to meet the deadline.

Which development stage are we at?

As of December 1st, 2017, our team has the following features implemented and working (appending onto the features mentioned in the previous two progress reports):

- Error handling for dealing with things such as invalid inputs and incorrect (or corrupted) game data files.
- A dynamically generated list of maps to select from on the main menu (if new map data files are added to the games “maps” folder, the map list is automatically updated).
- Added a sound effect when enemies are killed.
- Added background music, which can be toggled on or off in-game (Credits: "Exit the Premises" Kevin MacLeod (incompetech.com) Licensed under Creative Commons: By Attribution 3.0 http://creativecommons.org/licenses/by/3.0/).
- Changed the names of defense towers to terms that related to computer defense (to better fit our computer science theme).

- Added an indicator for towers which, when a tower is selected, shows the attack radius of that particular tower on the map.
- Included an executable jar file in our project package for increased user friendliness.

What testing have we done?

A list of units tests have been produced using JUnit. The tests ensure that the proper exceptions are thrown when the map data file is wrong in any way. We have also done fairly extensive release testing with all our team members over the past few weeks. We believe this is the ideal way to test the GUI and the variety of buttons since formal unit tests may not accurately recreate the unpredictable behaviors of a human user. At this point the game is robust and dependable, since any game crashing bugs have been uncovered and fixed. It has also been played on several operating systems with no problems.

Did we need to modify the time table?

The time table has never had the need to be modified throughout the semester. We have been on (or ahead) of our schedule during the vast majority of the development process. From the beginning of this project, we knew it was key to not get behind schedule, as we had a large number of fairly complicated features to implement for CDR Tower Defense. Knowing this, we spent a lot of time writing the code for the core features of the game to be completely sure that we could have a functional product by the end of the semester.

What were some of the challenges faced during development?

During the requirement, documentation, and development processes of CDR Tower Defense, our teams has had several challenges to overcome. Luckily, all the challenges we faced, both minor and major, were overcome successfully!

One challenge was knowing how to write proper documentation during the process of development. The way we decided to do this was to specify date ranges throughout the semester, with each range specifying the requirements we were meant to implement during that time, and then document the actual work we were able to complete during a particular interval. For another form of documentation, we decided to write what general descriptions of what our Java classes are actually

doing, that way someone unfamiliar with the software could quickly get up to speed by looking at the code and simultaneously reading the class descriptions we documented.

Another challenge was not only figuring out how to do the graphics for moving enemies, but making that movement appear to be smooth and flawless, even when enemies are going around corners or entering from the edge of the screen. The first implementation of enemy movement, unsurprisingly, did not look anywhere near as smooth as it does in our finished product. We are quite happy with this aspect of the game.

Deciding how to encode the information for the maps in a text file was a major decision that had to be made for this game. Since the beginning, we have discussed this topic intensely, and have changed our minds on it several times. Eventually, we figured out exactly what was necessary to have in these map data files (and the format the map data should have), in order to focus on simplicity and to try not to over complicate things. In fact, throughout the creation of this game, we have tried very hard to not make the code a cluster of complexity. It is important to us that this game can be extended or modified without having to rewrite the core algorithms.

Did we modify the requirements?

Yes, the requirements of our game have been modified since our initial requirements were conceived. The only requirement that was actually changed was the mechanism for damage to enemies. Other than that, the other modifications were all additions of new requirements which we think make the game better. These new requirement additions were:

- An indicator for the selected tower's attack range.
- Background music and sound effects for enemy deaths.
- A list of selectable maps in the main menu.
- Error messages for a variety possible error detections (related to the map data files).

Summary:

Using various ideas and methods learned in Software Engineering, our team of three has managed to build a fully functional video game during a period of approximately 9 weeks. The game has met and exceeded all of its proposed requirements, and it resembles exactly what we had in mind when our team first imagined the idea. It was a great opportunity to gain different programming skills, like implementing GUIs and collaborating with other programmers, for example. On top that, the project

gave us several opportunities to present to the class. This is important because, the more chances one has to do presentations, the more comfortable they will become doing it. All in all, developing CDR Tower Defense was a positive experience and our team learned a lot doing it.