Drive By Defense

*James Fitzgerld, Justin Rodriguez*

Section I: Overview

*Drive By Defense* takes the classic tower defense game and provides a twist by allowing the player to constantly change the placement of their defenses! Our game is a top down strategy game that separates itself from others in its genre by focusing on a unique core mechanic while keeping true to tower defenses players love. With a solid, single input devise, control scheme and a clean user interface, the player can focus on their strategy and play the game how they want to. The core of this game is that it gives the user the option to improve their gameplay by utilizing the tower repositioning system without taking away from the users who simply want to focus on building their line of defenses.

Section II: Post-Mortem

Our vision for the game was to take a genre we both enjoyed and do something new with it. Though we accomplished this to an extent, the overall feeling of the current state of the game it that it lacks depth. The most noticeable change from the original design document is the power up/upgrade system. We decided to change from tech trees unlocking things in exchange for tokens to tokens that were consumed to increase a towers strength permanently, We felt this fit our game better due to all the towers being the same but with different stats. If the towers were to have different features in the future, we might consider adding in some kind of research system in addition to the upgrade system we currently have. Though we have not done the entire math behind our current system. We feel that it will allow for many different play styles dude to certain power points towers will reach in addition to players attempting to best optimize the gold to strength ratio in addition to the normal positioning optimizations found in traditional tower defense games.

The most significant changes to our plans revolved around not being able to accomplish everything we wanted to. Though the code is in a functional position for it, we never fully integrated an enemy armor/tower attack type system. There is no way currently for the user to know what type of towers would have worked well against different types of enemies due to unimplemented tower feedback. Both visual queues for the towers and for the health of the enemies appearing were scrapped due to time. There is still an audio feedback but it has no diversity, which causes it to be functional but not extremely useful for the user who is attempting to figure out the game. Due to lack of depth, we also did not have the opportunity to properly balance the game. This led to use scrapping the different difficulty modes and left the user with only the sandbox mode (Which is more a balancing mode than a traditional sandbox). We did not even attempt our stretch goal of randomly generated, endless waves.

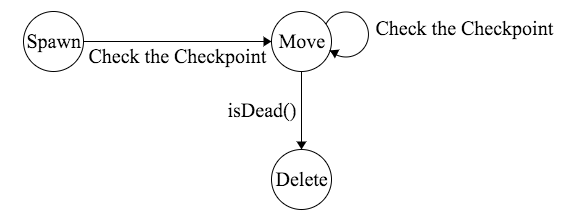
Additionally we made changes to our control scheme and UI. We decided to limit the inputs to once device under the though that the user could (in theory) attach the controller of their choice and bind the different keys to buttons on their controller. As for our user interface, with the changes made to the power up system requiring so much space, we decided to remove the information on the next wave. It made the UI take up too much of the screen and hindered the user experience.

If we were to attempt this task a second time, the first thing we would do is attempt to fix the primary issue the game has in its current state: the pace. The game can feel slow and clunky because the user is forced to use the mechanic whenever they wish to add a tower. We would change the game from having the player walking around at all times to a toggle option between waves and give the user a secondary more classic control scheme using the mouse to manage towers. We feel the core feature we added is healthy for the game but it feels forced upon the user as it. Though it keeps the user active during waves, it does slow down the user between waves. Additionally we would have liked to been able to add more depth to the game in terms of different types of towers and enemies. This might have been possible had we properly allocated time; however, integration issues took up the time that was set to add in this part. Finally, we would take another look at the design of the store UI and attempt to find a way to include a description of each tower so the user has more information than simply the gold cost.

Section III: Bugfixes and Efficiency Upgrades

Bug fixes from the feature complete version are as followed: a bug with the sprites of enemies not appearing until the player hits the spawn wave button, a bug where moving a tower sometimes left behind an invisible wall causing unit collision, a bug where moving a tower was not changing where the tower was shooting from, and a bug where the range of the towers was being calculated incorrectly. These bugs with the collision detection and range are from using two separate coordinate systems across the group and integration being done incorrectly (One system was absolute position and one was relative to the camera). One party worked on the vehicle/tower was done by and another did the collision detection and range (A range indicator was added that should more accurately represent the tower range). As for efficiency, we attempted to keep the project in a good state for efficiency though out the design process. The bug with the sprites was causes by a last second change to improve efficiency before feature complete.

Section IV: Artificial Intelligence

The AI for this game is found within the enemy. Checkpoints are placed along the path made for the enemy to travel. Upon spawn, the enemy will check what checkpoint it is at and the move accordingly. It will continue to check and move until it is dead. This will happen when either the enemy hp drop to 0 or the enemy reaches the final checkpoint. If the hp reaches 0 the player gains all the gold from the enemy’s inventory. If the enemy reaches the final checkpoint, the player’s lives go down by 1. The improvement we would look to add is to have different types of enemy’s follow different checkpoints. For example we would like to include a flying enemy that would follow checkpoints across the screen ignoring those set on the path.

Appendix A: Milestone Calendar

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| Week 3 |  | 19  January | 20  Register Group | 21 | 22 | 23 | 24 |
| Week 4 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Week 5 | 1  February  Outline Weekly Goals | 2 | 3  Design Document | 4 | 5 | 6 | 7 |
| Week 6 | 8 Outline Weekly Goals | 9 | 10 | 11 | 12 | 13 Progress Meeting | 14 |
| Break | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| Week 8 | 22 Outline Weekly Goals | 23  Write Report | 24  Progress Report | 25 | 26 | 27 Progress Meeting | 28 |
| Week 9 | 29 Outline Weekly Goals | 1  March | 2 | 3 | 4 | 5 Progress Meeting | 6 |
| Week 10 | 7 Progress Meeting – Prep Slice | 8 | 9  Vertical Slice | 10 | 11 | 12 | 13 |
| Week 11 | 14 Outline Weekly Goals | 15 | 16 | 17 | 18 | 19  Progress Meeting | 20 |
| Week 12 | 21 Progress Meeting – Finalize | 22 | 23  Feature Complete | 24 | 25 | 26 | 27 |
| Week 13 | 28 | 29  Class Demo | 30 | 31  Class Demo | 1  April | 2 | 3 |
| Week 14 | 4 | 5 | 6  Final Report |  |  |  |  |

­­­­

Appendix B: Self/Peer/Group Assessment

For this project, out labor was mainly split into tasks that could be accomplished individually and then integrated together. We used github for version control, which worked really well. The main issue we ran into with this approach was the fact that we had a miscommunication in how to do the coordinate system. This lead to a quiet a few bugs we had to work around to get the different code blocks to work together.