

---

Quinn Roemer & Logan Hollmer

Dr. Daryl Posnett

CSC 133 MWF 9-9:50am

April 4, 2020

### **Tower Defense Documentation - Part 1**

We decided to work on the Space Defense Design posted in Piazza. Currently, the game is running and has several features that we like, and some we will change in the future. As of today, the game has two buttons, one to start/pause the game and one to place a Plasma tower. Clicking the plasma tower button will subtract 50 dollars from the player's money and it will then place a tower at the next location the player touches. A player cannot place a tower if they do not have the money for it. They also cannot place a tower over a button or on the path the aliens take. If a placed tower is touched a grey ring will be shown around it indicating the range of the tower. Currently, there are 3 waves of aliens that spawn and make their way toward the player's base. Aliens have a health bar that changes as they are damaged. If an alien reaches 0 health it is removed from the map. If an alien reaches the base they are removed from the map and the player loses a life. If the player reaches 0 lives at any time the game ends and they receive a message that they lost. If the player can survive all 3 waves of enemies they are presented with a message saying that they won. The towers turn to face the alien they are shooting at, we like this visual effect but want to change it in the future. Currently, we are rotating the entire image, this causes some inconsistencies to happen and makes the turret look odd as it rotates. Long term we want to animate the turret's rotation rather than rotating the entire image.

We built several classes to act as the "backbone" of the game. The first of these being GameActivity. This class collects the necessary data on the host device and creates the gameEngine object. This class is very similar to the "activity" classes that were in our other projects. The next class that we use to form this "backbone" is the GameEngine class, this is our "controller" class. This object is primarily used to hold the game loop. It performs necessary thread management and calls updates on all the game objects when it determines an update is required (currently 30 times a second). This object also detects when a touch event occurs, calling the input handler in our game. Lastly, this object holds several game objects. These being GameWorld, GameView, and GameMap. GameWorld is responsible for holding all of the game objects that are currently in use. These range from towers, enemies, projectiles, game lives, and currency. In addition, it also holds a number of Booleans that are used to outline the current game state. For example, we have Booleans for whether or not the game is paused, over, and/or won. In addition, we also track whether it is okay to draw and if the thread is currently running. Inside this object we have methods for starting a new game, ending the current game, and getters/setters for certain game objects/variables. GameView is responsible for drawing all of the Games objects currently in play. We wrote this class to simplify the call to draw items inside of GameEngine. Since a certain order is required when objects are drawn, GameView enforces this order and draws everything as needed.

---

GameMap was created to hold unique information about the game's map and path. It holds the background image, as well as certain assets (base, path, etc.). It also defines how many waves spawn on the map you are playing, and what makes up each wave.


The HUD is a class that takes care of all the buttons and text that is displayed. It contains the code for creating and drawing all the controls that are seen in-game. The UIController handles any and all input from the user. There is a `handleInput()` method that decides what to do based on where the user touches and other variables in the GameWorld. The UIController is an observer, which is part of an observer pattern. The observer pattern uses several interfaces to allow the UIController to observe the GameEngine and react when a touch event happens.

Circle is a custom class for creating and drawing a circle. Currently, this is used exclusively for drawing a circle around a tower to indicate it's range.

Movable is an abstract class that inherits from `GameObject`, it has no concrete methods, rather it has an abstract method `move()` to ensure all the objects that inherit from it can move. It also has a heading and a speed. Speed is simply an integer, however, heading is an object of type `Angle`. `Angle` is a class we created to contain an angle. It contains an integer that is guaranteed to be between 0 and 360. Note that for movement, 0 degrees is heading up toward the top of the screen and 90 degrees means heading right on the screen.

Our enemies are built using a factory pattern. We have an interface entitled `Alien` that enforces the necessary methods that all enemies must implement. `AlienFactory` builds the desired enemy based on this interface and a passed type. This passed type is currently a `String`. We plan to change this in the future to a different type of variable (most likely an enumeration). When the type is determined the alien is built using the `EnemyBuilder` class built into the `Enemy` class. We decided to do this to simplify the construction of enemies while making it easy to add new enemies in the future. This `EnemyBuilder` allows you to customize certain aspects of the enemy. In the end, it returns an enemy of your desired type. This is sent back to `AlienFactory`, which defines the enemy to be built, which in turn returns the completed `Enemy` object to the class that called for the creation of an enemy. While building an enemy, a movement strategy is chosen based on the passed type. Currently, only one strategy has been developed, but we plan to add more strategies as new enemies are implemented.

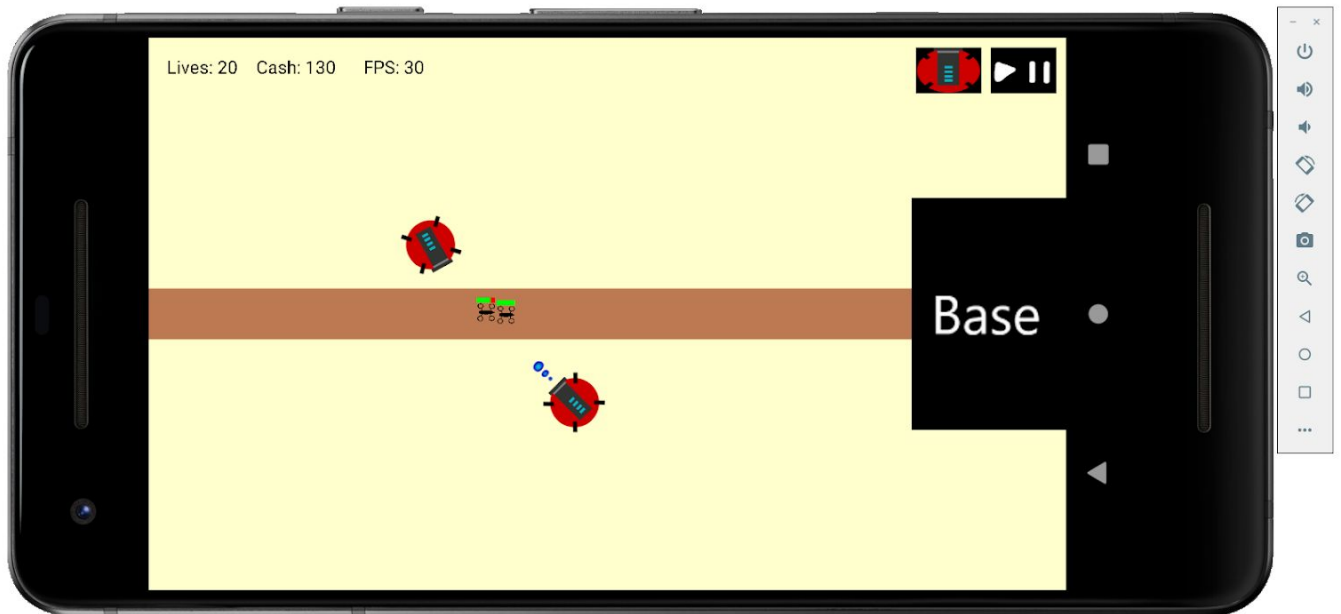
`Tower` is an abstract class that inherits from `GameObject`, it contains concrete methods for checking if an alien is in range and if a point is located inside of a tower's graphic. All towers have a `TowerData` object which is a container for the data about the tower. This contains data such as the range and cost of the tower, it also has a `ProjectileData` object. `ProjectileData` is another container for holding the information about a tower's projectile. This includes stuff such as the projectile's graphic, speed, damage, etc. Both `Tower` and `TowerData` are created using a factory pattern. `PlasmaTower` is a specific type of tower that inherits from `Tower`. It uses the `TowerDataFactory` to create the tower's data. `PlasmaTower` has a concrete method for how it shoots. When it



shoots it creates a projectile object and adds it to the list of projectiles in the GameWorld. A Projectile is an object that inherits from Movable. It takes a ProjectileData object, an origin, and a destination as arguments. With this data, it sets up its graphic and also creates a speed and heading. Projectile has two concrete methods, move() updates the location of the projectile based on the speed and heading. Collision() detects if the projectile has run into something or has left the tower's range and returns a boolean.

As our project progresses we plan to implement more enemy and tower types as the game design requires. We also will implement several different maps with a map selection screen appearing when the user starts the game. As you can tell from the screenshot of our game below, our assets are very basic. We plan to further develop these and refine them as we move forward. Lastly, we feel that some of the ways that we implemented certain items in the game will need work and refactoring. For example, we are currently applying damage to an enemy as soon as the projectile spawns. This can cause the enemy to disappear while the projectile is still moving to its target. This will require some reworking on our end as we find another implementation for this game mechanic. Overall, we are happy with our current progress and are excited to continue moving forward as we develop our game.

Image of the game running



Class Responsibility Collaboration Cards

|   |  |
|---|--|
| Tower   |  |
| <ul style="list-style-type: none"> <li>• Checking if the tower contains a point.</li> <li>• Checking if an alien is in range of the tower.</li> <li>• Setting the size of the tower.</li> </ul> | <ul style="list-style-type: none"> <li>• GameObject</li> <li>• TowerData</li> </ul>  |
| Plasma Tower  |  |
| <ul style="list-style-type: none"> <li>• Shooting.</li> <li>• Tower initialization.</li> </ul>  | <ul style="list-style-type: none"> <li>• Tower</li> <li>• TowerData</li> <li>• GameWord</li> <li>• TowerDataFactory</li> </ul> |
| TowerFactory  |  |
| <ul style="list-style-type: none"> <li>• Create a tower of a specific type.</li> </ul>  | <ul style="list-style-type: none"> <li>• PlasmaTower</li> </ul>  |

|  |   |
|--|---|
| TowerDataFactory   |   |
| <ul style="list-style-type: none"> <li>• Create the default data for the towers</li> </ul> | <ul style="list-style-type: none"> <li>• TowerData</li> </ul> |

|  |   |
|--|---|
| Projectile   |   |
| <ul style="list-style-type: none"> <li>• Set the heading of the projectile.</li> <li>• Check if there is a collision.</li> <li>• Move the projectile.</li> </ul> | <ul style="list-style-type: none"> <li>• ProjectileData</li> <li>• GameWorld</li> <li>• Movable</li> <li>• TowerData</li> </ul> |

|   |  |
|---|--|
| Circle  |  |
| <ul style="list-style-type: none"> <li>• How and where to draw the circle.</li> </ul> |  |

|  |  |
|--|--|
| Angle  |  |
| <ul style="list-style-type: none"> <li>• Set the angle</li> <li>• Get the angle</li> </ul> |  |

|  |   |
|--|---|
| HUD  |   |
| <ul style="list-style-type: none"> <li>• Create the controls</li> <li>• Draw the controls</li> <li>• Add graphics to the controls</li> <li>• Display text</li> </ul> | <ul style="list-style-type: none"> <li>• GameWorld</li> </ul> |

|   |  |
|---|--|
| UIController  |  |
| <ul style="list-style-type: none"> <li>• Decides what to do based on player input.</li> </ul> | <ul style="list-style-type: none"> <li>• HUD</li> <li>• GameWorld</li> <li>• TowerFactory</li> </ul> |

|   |  |
|---|--|
| AlienFactory  |  |
| <ul style="list-style-type: none"> <li>• Build an Enemy based on a passed type.</li> <li>• Return an Alien enemy</li> </ul> | <ul style="list-style-type: none"> <li>• Alien</li> <li>• Enemy</li> <li>• EnemyBuilder</li> </ul> |

| AlienHealthBar  |   |
|---|---|
| <ul style="list-style-type: none"> <li>• Contain information related to the enemies health.</li> <li>• Designate the point where the healthbar is drawn.</li> </ul> | <ul style="list-style-type: none"> <li>• Enemy</li> </ul> |

| DroneMovementStrategy   |  |
|---|--|
| <ul style="list-style-type: none"> <li>• Define the speed of an enemy.</li> <li>• Define how the enemy moves across the map.</li> </ul> | <ul style="list-style-type: none"> <li>• Movable</li> <li>• Enemy</li> </ul> |

| Enemy  |   |
|--|---|
| <ul style="list-style-type: none"> <li>• Define health and resistance.</li> <li>• Hold references to the attached healthbar and movement strategy.</li> <li>• Handle collision with game objects.</li> <li>• Handle health and resistances.</li> </ul> | <ul style="list-style-type: none"> <li>• GameObject</li> <li>• Alien</li> <li>• Movable</li> <li>• DroneMovementStrategy</li> <li>• AlienHealthBar</li> </ul> |

| EnemyBuilder   |   |
|--|---|
| <ul style="list-style-type: none"> <li>• Build an Enemy based on a builder pattern.</li> </ul> | <ul style="list-style-type: none"> <li>• Enemy</li> </ul> |

| GameActivity  |  |
|---|--|
| <ul style="list-style-type: none"> <li>• Handle the creation of the game activity.</li> <li>• Start the game engine and run the game.</li> <li>• Handle pausing and resuming the game.</li> </ul> | <ul style="list-style-type: none"> <li>• GameEngine</li> </ul> |

| GameEngine  |  |
|---|--|
| <ul style="list-style-type: none"> <li>• Contain game loop.</li> <li>• Handle updates to the game object</li> <li>• Coordinate drawing when to draw with GameView.</li> </ul> | <ul style="list-style-type: none"> <li>• GameView</li> <li>• GameWorld</li> <li>• HUD</li> <li>• UIController</li> </ul> |



|  |  |
|--|--|
| <ul style="list-style-type: none"><li>● Handle OnTouch events.</li></ul> |  |
|--|--|

|   |  |
|---|--|
| GameWorld   |  |
| <ul style="list-style-type: none"><li>● Contain game state variables.</li><li>● Handle starting/ending game.</li><li>● Manage game state variables.</li></ul> | <ul style="list-style-type: none"><li>● GameMap</li><li>● GameView</li></ul> |

|   |   |
|---|---|
| GameMap   |   |
| <ul style="list-style-type: none"><li>● Contain all map bitmaps.</li><li>● Contain information on the predefined path.</li><li>● Handle spawning enemies.</li></ul> | <ul style="list-style-type: none"><li>● GameWorld</li></ul> |

|   |   |
|---|---|
| GameView  |   |
| <ul style="list-style-type: none"><li>● Handle drawing all game objects</li></ul> | <ul style="list-style-type: none"><li>● GameWorld</li></ul> |

|  |   |
|--|---|
| Movable  |   |
| <ul style="list-style-type: none"><li>● Hold values for heading and speed.</li><li>● Enforce movement on subclassed types.</li></ul> | <ul style="list-style-type: none"><li>● GameObject</li><li>● DroneMovementStrategy</li><li>● Projectile</li></ul> |

|  |  |
|--|--|
| GameObject   |  |
| <ul style="list-style-type: none"><li>● Hold location and attribute size and bitmap.</li><li>● Handle setting the attribute size.</li><li>● Handle drawing the game object</li></ul> | <ul style="list-style-type: none"><li>● Movable</li><li>● Enemy</li><li>● Projectile</li></ul> |