T.O.W.E.R.S.

Design Document

TEAM 7

Garrett Kizior

Keith Droll

Nicolas Bratton

Ryan DeSalvio

Table of Contents

T.O.W.E.R.S.©

1. Purpose

T.O.W.E.R.S. is a first person shooter/tower defense hybrid incorporating the best features from FPS and tower defense gameplay to create a unique experience not yet seen in the video game market.

The purpose of T.O.W.E.R.S. is to create a system that responds to player input and allows the player to complete objectives such as fighting against AI enemies, protecting a main base, and strategically placing defenses all the while letting the player choose how to complete these objectives.

To achieve this we will implement an Unreal Modular Architecture, including classes to representing the various aspects of the game such as the players, enemies, items, materials, and animations.

1.1 Functional Requirements:

As A User:

- As a user, I would like to start a new game
- As a user, I would like to pick up dropped currency from destroyed enemies
- As a user, I would like to buy turrets, and ammunition
- As a user, I would like to know what my current health, current currency count, and round number is
- As a user, I would like to choose upgrades for myself
- As a user, I would like my character attributes to be shown in an easy to read menu
- As a user, I would like to fight in either ranged and melee styles (If time allows for melee)
- As a user, I would like to view an end-game statistics screen
- As a user, I would like to pause the game
- As a user, I would like to be able to place turrets from a top down perspective
- As a user, I would like to be able to see turrets current ammunition
- As a user, I would like to be able to see my base's health
- As a user, I would like to be able to upgrade turrets
- As a user, I would like to be able to view a well-designed upgrade tree for my player

As A Developer:

- As a developer, I would like to like to include animations for character/enemy movement
- As a developer, I would like to like to include animations for character/enemy attack
- As a developer, I would like to be able to store high scores
- As a developer, I would like for enemies to spawn at an increasing rate as the rounds progress
- As a developer, I would like the gameplay to be balanced and encourage player experimentation
- As a developer, I would like to implement an achievement system (if time allows)
- As a developer, I would like to incorporate a start menu that allows for starting games, and viewing high scores, etc.
- As a developer, I would like to allow for cooperative play (if time allows)
- As a developer, I would like for some enemies to follow the character blindly
- As a developer, I would like for some enemies to shoot the character

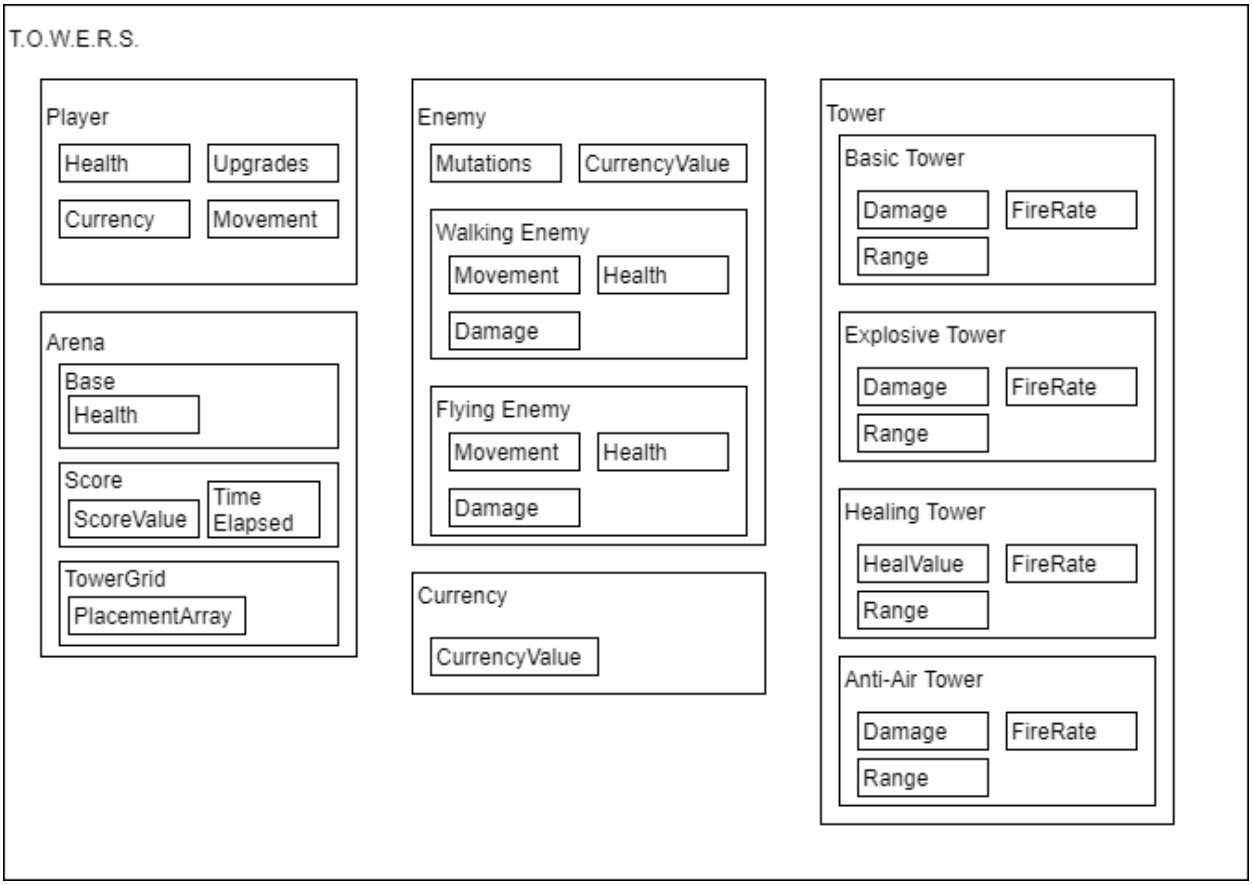
- As a developer, I would like to mutate the enemy's attributes (health, speed, strength) each round based on enemy performance
- As a developer, I would like to allow the character to equip multiple weapons
- As a developer, I would like to include a credit scene at the end of game (if time allows)
- As a developer, I would like to include a store to purchase character skins, etc. with in-game tokens earned from completing each play through (if time allows)

1.2 Non-functional Requirements:

- As a user, I would like to have fast response times with user input
- We must be able to play this game on PC
- The menus and interface needs to be simple, intuitive and responsive
- We must have fast frame rates, and minimize computer resources
- Art style must be attractive and simple
- Game must include catchy and fitting soundtrack
- Game must include sound effects for different events
- Game is stored in an executable file – this prevents people from viewing source code
- As a developer, I would like the game to support multiple platforms (if time allows)
- As a user, I would like to be able to use a controller or keyboard

2. Design Outline

2.1 Architecture Outline



For our architecture design we will use the Unreal Modular Architecture, because we felt it represented most what we wanted to accomplish. This pattern integrates closest to our vision of the final product; we have a overarching superclass that contains each modular subsystem beneath it. These subsystems would control each aspect of our game, from the player to each enemy to the towers and how they can be placed. These classes will interact with each other in Unreal Engine; for example, when a Player will shoot an Enemy, the two classes will interact by the Enemy losing health according to what weapon the Player shot it with. Another example interaction would be the Arena controlling where and when to spawn Enemies, as well as where Towers can and cannot be placed.

3. Design Issues

3.1 Functional Issues

Functional Issue #1: Since the game is going to be a game based on repeated quick play-throughs, should the user be able to save his or his progress mid-game?

Option A: Yes, integrate a save game feature. Where player can resume a game exactly where they previously were

Option B: No, Player must finish the play though without saved progress in one sitting

Option C: Yes, integrate a save game feature. Where the player can resume a game at the beginning of the round they were currently on.

Decision: Option B. This is not in scope of the design of the game. Saving a game would encourage slowing the pace of the game down, when our intentions as developers are to based gameplay around quick individual playthroughs. Also saved games might encourage players to reload past saves to repeat a favorable playthroughs.

Functional Issue #2: In game, when a player is to pick up currency, how should this be done?

Option A: Currency is automatically added to total currency Whenever the player walks over a currency item

Option B: Player only acquires the currency that is walked over at the end of each round

Option C: Player acquires currency automatically when a enemy dies. No item drop.

Decision: Option A. Currency should automatically be added right as the player walks over the currency. This allows the player to have an idea of how much currency he has before the end of the round upgrade/turret placement UI appears. The player should have an idea of what upgrades and turrets he could afford before entering this menu.

Functional Issue #3: Should we provide a feature to view previous local high-scores?

Option A: Yes. Available in the options menu and at the end of each playthrough

Option B: No

Option C: Only view high-scores at the end of a completed playthrough

Decision: Option A. This will give players the ability to compare previous runs and the ability to share scores with friends. We believe this will promote re-playability, self-to-self competition, and self-to-friend competition

3.2 Non-Functional Issues

Non-Functional Issue #1: What language and game development engine are we going to use?

Option A: Unity engine with C#

Option B: Unreal engine 4 with C++

Decision: Option B. Because we have little experience with Unity, and because our team has collectively a lot of experience with Unreal, we decided to go with Unreal and C++.

Non-Functional Issue #2: What will we utilize for a soundtrack?

Option A: Create our own soundtrack

Option B: Purchase a soundtrack from 3rd party

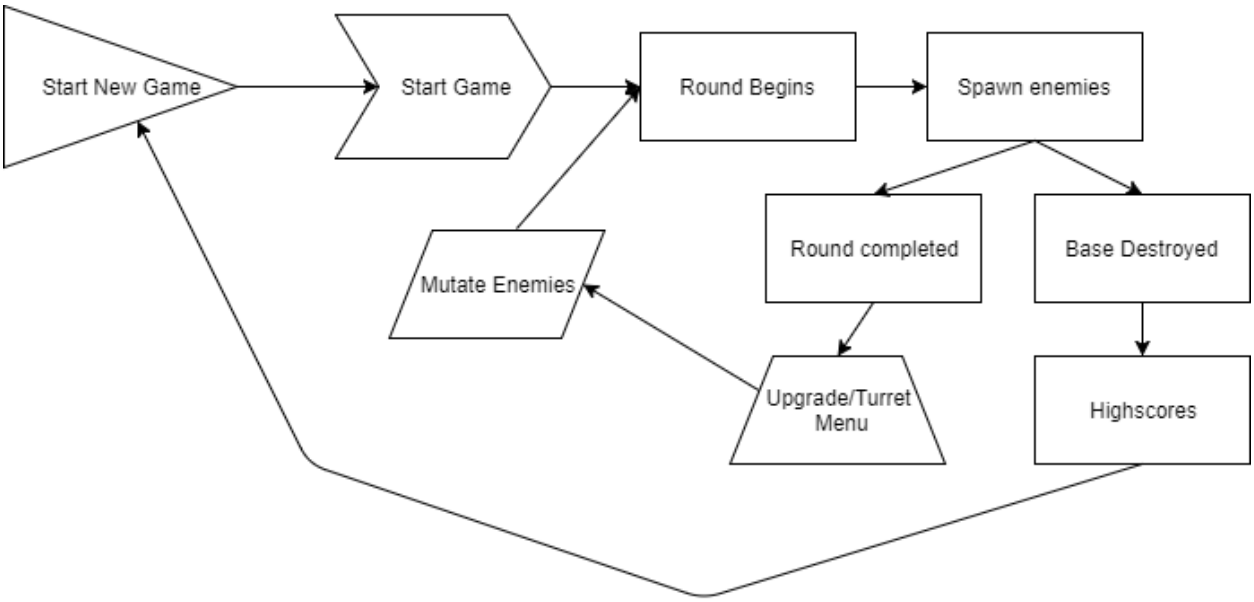
Option C: No soundtrack. Sound is solely from sound effects in-Game

Option D: Mix of creating our own sounds and purchasing other sounds

Decision: Option D. We believe creating our own sounds will increase the originality of our game. And we also feel that having purchased sounds interlaced with our own created sounds will make the game experience more complete.

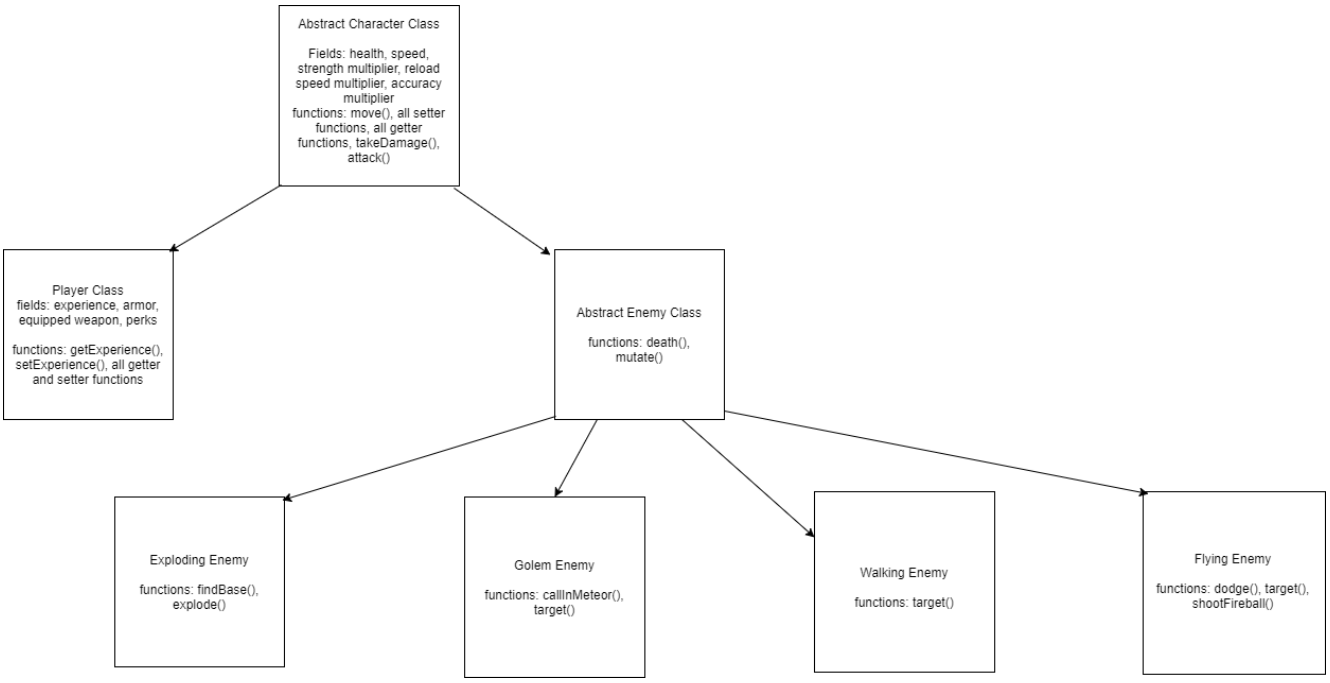
4. Design Details

4.1 General Game Flow Outline



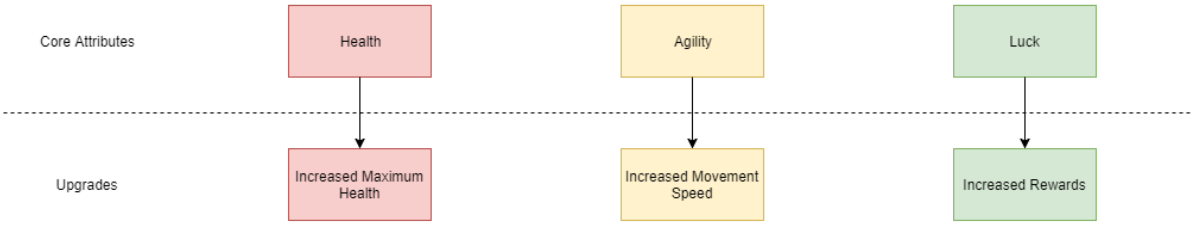
The general game flow outline starts at the ‘Start New Game’ node. From there we start the game and begin the first round. During a round the game will spawn a number of enemies. During the round, the player can either defeat all the enemies without their base being destroyed, or the base is destroyed before defeating all the enemies. If the base is destroyed during the round, the high scores from other previous playthroughs will become visible and you will be redirected back to the main screen. If the player is to complete a round without their base being destroyed, then the upgrade menu along with the turret placement option will become available. After selecting upgrades and placing turrets the game will determine what enemies will be mutated and how to mutate them. And then the next round will begin.

4.2 Basic Character Class Outline



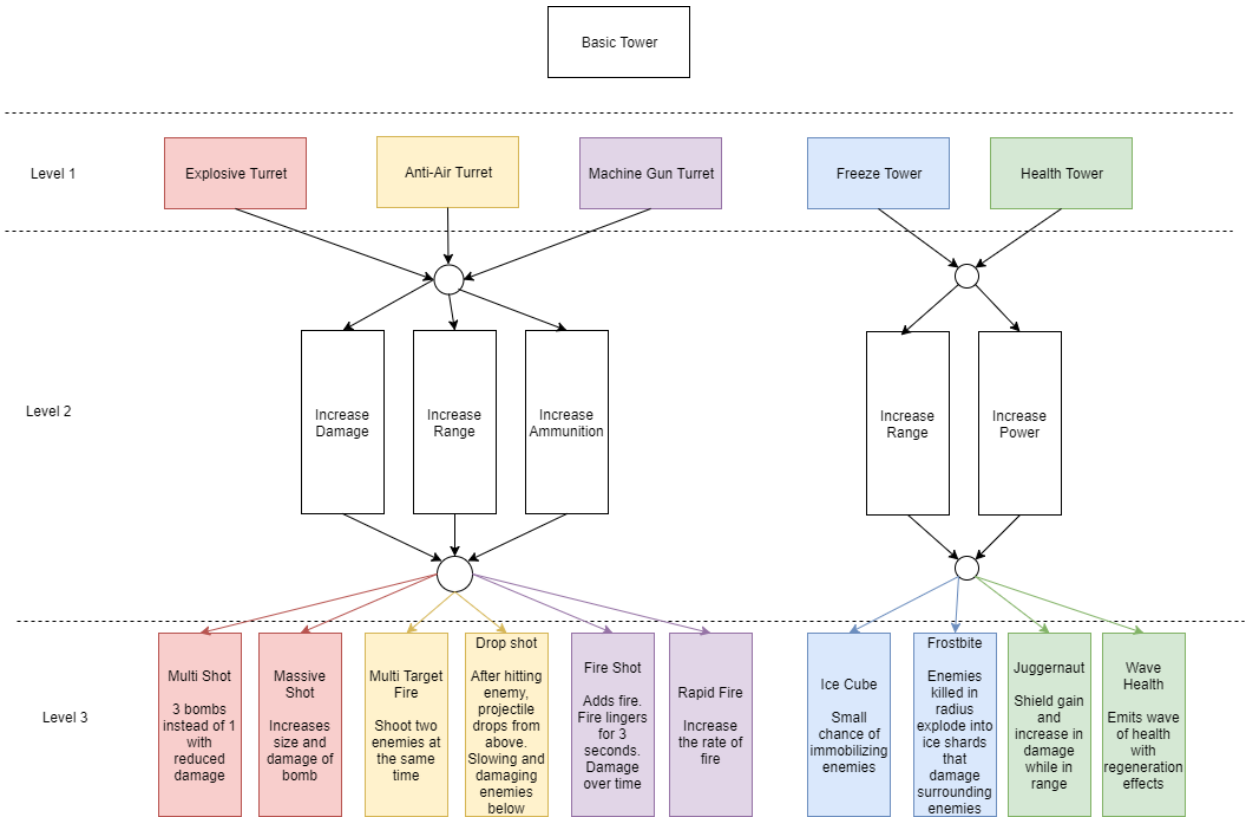
All actors/characters in the game belong to the Abstract Character class. A character is decided to be either a Player (the playable character that the User plays as) or an Enemy (the opponents faced by the User while playing). If the Character is a Player, they gain all of the attributes appropriate to the Player class, such as experience, equipped weapon, perks, among others. These are attributes that are unique to the Player class, and Enemy Characters may not have these same characteristics. Likewise, Enemies have variables that Players do not, such as rangeOfAttack and methods such as follow() and target(). There are other subclasses of the abstract Enemy superclass that are not listed here, as it is meant to be a flexible class so that we may add and remove different types of enemies at will, as we come up with them.

4.3 Character Upgrade Path



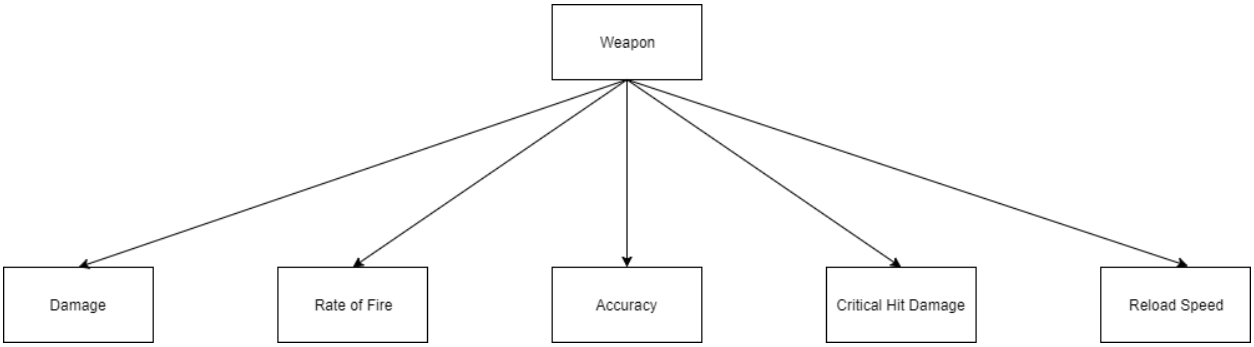
The player in the game will be able to upgrade their intrinsic attributes of health, agility, and luck. When health is improved, the player maximum amount of possible health will increase. When agility is improved, the player's movement speed will increase. And when luck in increased, the amount of currency and other rewards received from the game will increase. These upgrades will be chosen by the player in the "strategic" phase of the game, between rounds.

4.4 Tower Upgrade Path



There are several towers types to be implemented in the game. Each tower will have unique properties and have unique upgrade paths culminating in a “Level 3” status which will add some fundamental change to the way the turret operates making it more powerful and useful. The outline, which can be seen above, follows a patter. Level 1 is the creation of the base turret which has a slow fire rate and weak damage. Turret upgrades can then be purchased with in game currency. Once level 2 is within a players budget, a decision must be made between 3 basic buffs like increasing range, damage, or ammunition of the turrets. Level 3 is then unlocked and the individual towers gain some personality by offering interesting buffs like the chance to freeze enemies or cause splash damage.

4.5 Weapon Upgrade Path



In addition to being able to upgrade and improve the player and the buildable turrets/towers, the weapons that the player will use will also be upgradable. To accomplish this we will give each weapon five upgradable attributes: damage, rate of fire, accuracy, critical hit damage, and reload speed. Damage will increase the per-shot damage done to enemies. Rate of fire will increase the rate at which the weapon can be fired. Accuracy will decrease the spread of the weapon's shots. Upgrading critical hit damage will further increase the damage done to an enemy when the player scores a hit on a vulnerable spot of an enemy. And upgrading reload speed will decrease the time needed to reload the weapon.