FIT2096 Assignment 3 2020

Game Project - Tower Defence Game (20%)

Figure 1 Plants vs Zombies

Due: Week 12, Friday 12th of June , 11:55 pm

Platform: Unreal Engine 4

Task

Your task in this assignment is to create a simple Tower Defence game using the Unreal 4 game engine. Enemies will spawn on one side of the level and try to reach a 'base' at the other side of the level. If the enemies destroy the 'base' they win. If the Towers successfully defend the base they win.

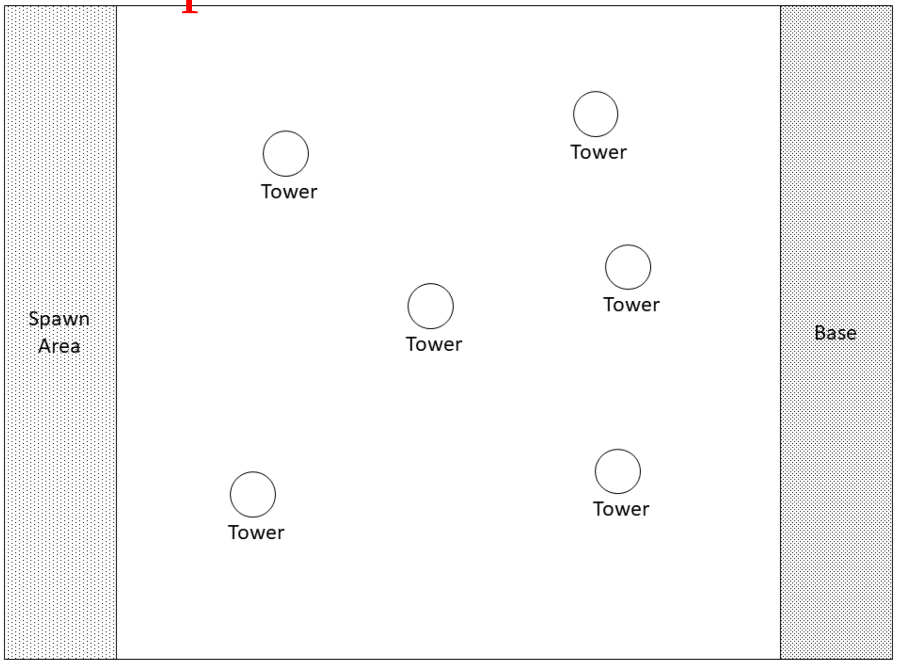

Figure 2 Level Layout Design

The Implementation

Game Balancing: you must provide a way to easily experiment with variables such as tower and enemy attributes, bullet damage, shooting frequency and bullet speed etc. to help you balance the game and make it a re[illegible] experience (i.e. No 'magic numbers'). These values must be editable from the e[illegible] changing any C++ code or blueprint script.

Specifications

Game Project

Use the 3rd person ga[illegible]

Game Level/Map

The game level will be made from a simple plane scaled to the required size with a suitable texture applied. See Figure 2 Level Layout Design for an example of how your game level should be laid out. The position of the towers is indicative only, you can place them anywhere in the level.

Camera

You will need to implement another camera, known as a top-down camera. This camera should be able to view the entire scene from an above perspective. The user needs to be able to press the Space Bar key to switch between the 2 cameras implemented within the scene (3rd person, top-down).

Towers

As in a traditional Tower Defence game the role of the Towers is to defend your base against the attack of a group of Enemies.

The specifications for the towers are:

- Coding
  - The core functionality of the agents must be coded in C++. Variables can be exposed as a Blueprint class.
- Tower Placement: the position and attributes should be hard coded
  - Extra: enable the player to place the towers
  - Extra: enable random tower placement
- Towers should be represented by capsule objects, or another suitable model from the Unreal Starter Content, with different colours/materials indicating the different types of towers
- The different types of towers you will create are:
  1. Radius effect
     - These types of Towers do damage to enemies within a given radius.
     - You will need to specify:
       - The radius of effect
       - Minimum and maximum damage
       - Times per second damage is dealt to enemies
  2. Bullet Shooter
     - These types of Towers must each select an Agent that comes within a given radius, and lock onto the Agent so the Tower rotates to always face it. The Tower must shoot at the Agent until it is destroyed or moves out of range.

- Bullets should be represented by the a suitable model from the Unreal Starter Content
- A random element must be included making bullet accuracy from 70% - 80%
- You will need to test for a collision between the bullet and enemy
- n enemy it must do a range of damage between a mum value
- riable that sets rate of fire for the tower which can be e editor.

Enemies

The enemies will mov el from their spawn point towards the base on the opposite side of the level. If they reach the base they deal a set amount of damage to the base. Once all the hit points of the base have been damaged the enemies have won the game.

- Coding
  - The core functionality of the agents must be coded in C++. Variables can be exposed as a Blueprint class.
- Enemy model
  - Use any of the 3D Models provided in the Unreal Starter Content
- Waypoint Navigation
  - Your enemies must use a set of waypoints to navigate their way across the game level. Each waypoint contains a location (x,y,z) in the game level.
  - Waypoints will be hard coded and stored in an array.
    - Extra: randomise waypoints
  - You will have at least 4 paths your agents can navigate across the level
- Spawning
  - Enemies should spawn randomly at the start of one of the waypoint paths.
  - You will need to specify the enemy spawn rate
- Speed, Size, Hit Points
  - You must have four different types of enemies denoted by different colours.
  - Larger enemies are stronger with a high number of hit points, but move slower, while smaller enemies are faster but have fewer hit points. If you get time you can experiment with your balancing your game by changing these attributes.
- Enemy Material
  - The material applied to the Enemy will reflect how much health it has
  - As the enemy takes damage the colour will change until its health reaches zero and is destroyed
  - (You will need to do some of your own research on how to do this!)
- Damage if the enemy reaches the base
  - If four enemies reach the base it is destroyed and the game is over

Game Audio and Particle Effects

You will add simple sounds and particle effects (from the Starter Content) for the following:

- Bullets firing and hitting a target
- Explosions when enemies are destroyed
- Enemies spawning

HUD

A heads-up display (HUD) displaying the following information:

- HP of the Base
- The Score
- Number Enen

Game Menus and

Your application must ing controls:

- A Start Screen d Screen displayed at the appropriate times in the game and providing basic functionality
  - Start Screen
    - Text displaying your Name and Student number
    - A "New Game" button that starts a new game
  - Pause Screen
    - A "Resume" button that resumes the game
    - An "Exit" button that takes the player back to the Start Screen
  - End Screen
    - Appears when the game is over
    - Displays the winner and game stats
    - A "Replay" button which starts a new game
    - An "Exit" button which takes the player back to the Start Screen
- 'p' – Pause
  - Pauses the game and displays the "Pause Screen"

UE4 Gameplay Framework & Classes

Your game must use the correct Unreal Engine gameplay classes such as: Game Mode, Game State, Actors, UObjects etc.

Additional Functionality (20%)

The final part of this assignment requires you to add some additional functionality of your own choosing. You MUST discuss your proposed functionality and get approval from your tutor by the end of week 11.

Examples of additional functionality may include but are not limited to:

- Experimentation with custom materials.
- Exploration of procedural content and level generation.
- Random placement of Towers
- Player placement of Towers
- Enhanced gameplay elements such as
  - a difficulty and level-up system,
  - in-game currency with purchasing,
  - and the ability to save high scores.
- Anything else you can think of! Personal exploration of concepts not formally taught in this unit are welcome.

Marking Rubric

Functionality – 60%					
	N	P	C	D	HD
Level					
The game level and c[illegible]ctly (5)					
Towers					
Tower position and at[illegible]d represented by a model with different colours indicating the different types of towers? (2)					
Does the Radius effect work as specified? (8)					
Does the Bullet Shooter work as specified? (15)					
Enemies					
Are the enemies and movement implemented correctly with 4 paths across the map? (10)					
Does the material change colour to reflect the health of the enemy? (5)					
Game Audio and Particles					
Are the game audio and particles implemented for all required events? (5)					
HUD & Game Menu					
Has the HUD been implemented (5)					
Adheres to the Model-View-Controller pattern (5)					

Additional Functionality – 20%					
	N	P	C	D	HD
Is there additional functionality? (20)					

Code Quality – 20%					
	N	P	C	D	HD
Does the code exhibit good object-oriented design and use of Gameplay Framework classes? (10)					
Is the readability and style of the code to a high standard? Are **appropriate comments** included throughout? (10)					