Final Project: Tower Defense (Part 1)

Out: Monday, October 30th 2017

Due: Monday, November 6th 2017 at Noon (11/6/17 at 12:00 PM)

There is no starter code for this assignment.

This is a group assignment. You may divide up the work however you please, but credit will be shared among the three of you. You will have a chance to indicate if a group member was uncooperative after you submit this assignment.

Introduction

For this assignment, the three of you will be making a 3D tower defense game from scratch. Tower defense games have been around for a long time (arguably since 1990). They hit their peak either in 2007-2008 or in 2011-2012, depending on whom you ask. They're still quite popular, though - plenty of new TDs are being released on a regular basis. As far as game design elements go, there are a lot of different ways to make a tower defense game, but for this assignment we're going to stick with a rather vanilla formula. Here is a rough outline of the game elements that you'll eventually implement:

- a 5x5 grid on which to place towers and over which enemies will walk
- a base with 100 hit points to defend at one side of the grid
- enemies which spawn in waves on the opposite side of the grid and walk towards your base, pathfinding their way among your towers
    - a "normal" enemy, which has middling stats and does a moderate amount of damage to the base
    - a "fast" enemy, which is weak but very fast and does a small amount of damage to the base

    - a “strong” enemy, which is powerful but slow and does a large amount of damage to the base
- towers which can placed in open tiles on the grid and which block the paths of enemies (towers cannot be placed in a position that will prevent enemies from ever reaching the base, i.e., in a position that makes pathfinding impossible)
    - a “normal” tower with balanced stats
    - a “freeze” tower which has an area of effect attack that deals little damage but slows enemies in the area
    - a “shock” tower which has a lightning attack that can chain between enemies near one another
- money that is earned for destroying enemies and which can be used to purchase and upgrade towers
- victory if the player defeats all of the enemy waves, and defeat if their base is destroyed (i.e., if its HP reaches 0)

We’ll be doing this project in multiple parts, so you don’t have to worry about doing this all in one week. :)

Specifically, for the first part of this project, we’re going to focus on:

1. The Grid
2. The Base
3. Spawning Enemies
4. Basic Towers

The Grid

To start off, create a new **3D** project in Unity. Name it however you like.

The first thing that you need is a 5x5 grid of cells to support everything. There are lots of different ways of doing this, either on the code side or on the editor side, and you should feel free to do it any which way you like. One simple way is to create 25 evenly spaced cubes with attached “cell” components that handle clicking, building, etc. Make sure that your camera can see everything!
However you do it, here’s what you need:

1. A 5x5 grid of “cells” (whatever that means to you)

2. Each cell will later need to support being clicked on (which will open up a menu for building towers), so keep that in mind when creating any relevant components/helper classes
3. The cell in the “bottom right” of your grid will be your base, and the cell diagonally opposite (i.e., in the “top left”) will be the spawn point for the enemies

Adding a Base

In the “bottom right” (whatever that means, dependent on your camera orientation) cell of your grid, add a model of your base. Feel free to just use a green cylinder or something for now; we’ll add nicer looking models in at the end of the project. You also need some way of indicating the base’s current health (it starts with 100 hit points and takes damage as enemies come into contact with it).
However you do it, here’s what you need:

1. A base in the bottom right of your grid. It can be any 3D shape that you want, as long as it’s clear that there’s something special there.
2. A health bar in the top left. You can use sliders (a Unity UI component) to make decent looking meters. The meter is full at 100, and empty at 0.
3. You’re going to want to link up the base to the health meter on the code side
4. You’re going to want to provide provisions for stuff colliding with the base later, so be sure to add the necessary components to your base to accommodate this

Spawning Enemies

A Basic Enemy

Time to make your first enemy. Place a cylinder on your grid somewhere far away from your base. You’ll notice that it seems huge in comparison to the total size of the grid. Fix that by reducing the x, y, and z scales of its transform down to (0.2, 0.1, 0.2), respectively. This should leave you with a squat disc shape. This will be your first enemy. It needs to make its way across the map towards the base.
Here’s what you need:

1. The enemy should look different from your base (and, later, from all of your towers)

2. The enemy should move in a straight-line path (proper pathfinding will come later) towards the base at all times
3. When the enemy hits the base, it should disappear and reduce the base's health by however much damage you've assigned to it (your choice)
4. The enemy should have knowledge of its current health (we're going to be shooting at it later). You don't have to add health bars or anything (although you're welcome to), just remember to keep track of health somewhere.
5. Finally, turn this enemy into a prefab. We'll be using it a lot in the coming steps.

The Spawner

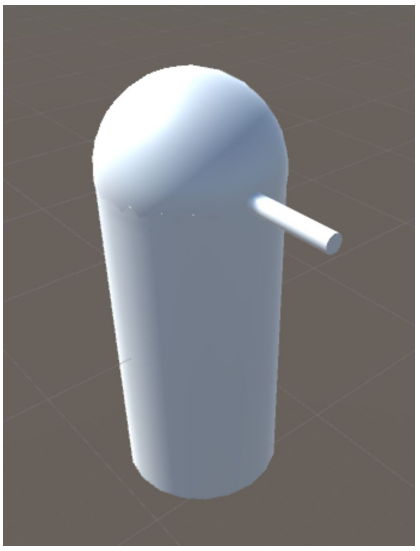
At the moment, we only ever have one enemy. This isn't very exciting. Let's add some more. Place a spawner on the cell diagonally opposite the base (i.e., in the top left). Don't make it a physical object, since things will appear underneath it and it looks ugly to have stuff appear inside of other things. Give your spawner some means of knowing what the "normal enemy" prefab is. Every few seconds, have the spawner create a new instance of that enemy. Since you set the enemy's behavior in the previous step, each new enemy should walk to the base and collide with it, reducing its health.
Here's what you need:

1. A spawner GameObject that knows about the "normal enemy" prefab, e.g., using Resources.Load or a public variable.
2. Every few seconds, the spawner should create a new instance of that enemy

Eventually, the base's health should reduce to zero (since you're spawning infinite enemies and have no way of destroying them yet).

A Basic Tower

Apropos of which, let's build some towers.
Make a basic tower model. One option is to put a sphere on top of a cylinder with a smaller cylinder sticking out, like this:



You don’t have to do it this way, just be sure to make sure that it looks totally different from your other models and that you can tell which direction it’s facing. Place your new tower somewhere on your grid (but not on the straight path from your spawner to your base!). Like the enemy, make it smaller than a full grid cell, although not quite as small.
Here’s what it needs to do:

1. Rotate towards the nearest enemy (if there is one). See the doc pages on Quaternions.LookRotation for some useful tips.
    a. One way to find nearby enemies is to use a spherecast
2. Fire at the enemy on a regular basis. You’ve seen plenty of bullet examples already, so this should be simple enough.
3. Enemies that get hit with a bullet should take a corresponding amount of damage, whatever that amount is. Enemies with 0 (or less) health left die.

Submitting

This should be enough for this part of the assignment. In the next segment, you’ll add a lot more. Your current project should have:

1. A 5x5 grid of cells
2. A base in one corner, which takes damage whenever enemies touch it
3. A spawner for those enemies, which periodically makes enemies
4. Enemies which walk from the spawner to the base in a straight line path

5. One (1) basic tower, somewhere off the path of the enemies, that regularly shoots at the nearest enemy, doing damage to it
    a. Enemies with 0 or less health die

Once you have all of these things, zip up your project as per usual (Assets and ProjectSettings directories only) and submit **one copy** of the .zip file as your *Canvas group*.