Homework 5: Tank!

Out: Monday, October 23rd; Due: Monday, October 30th at Noon (10/30/17 @ 12:00 PM)

There is no starter code for this assignment.

This is a pair-programmed assignment. You may divide up the work however you please, but there will be a *teammate evaluation* follow-up in which your partner will have a chance to complain about you if you slack off.

Introduction

For this assignment, you'll be making a clone of the classic game *Tank!*. Specifically, you and a partner will make a two-player, cross-platform, minimalist version of the game. This README serves not as specific instructions but rather as a set of guidelines for you to use as scaffolding.

As we've said above, we're not giving you any starter code, or even any starter anything. You'll be going from clicking the "New Project" button all the way to making a feature-complete (albeit for a short list of features) playable game.

As we also said above, you'll be working with another person, either of your own choice or assigned to you by us if you couldn't find a partner.

Collaborating Effectively With Unity

You have a couple options when it comes to collaborating:

- old school: you can use version control; we'll post a *.gitignore* up to Canvas/Files for you to use (you can also find useful versions online if you're too good for mine :( )
- Unity ships with a collaboration suite built right in. Full disclosure: I've never used it and I have no idea how straightforward it is to use.

- I guess you can also just email files back and forth/use a cloud-sharing service, but Tim Berners Lee will get sad over the way you're using the Internet

First Steps

Let's get started. First, we need somewhere to work:

- create a new **2D** Unity Project (call it whatever you like, although "Captain Pooper's Love Castle" might get confusing later)

Adding a Player

Aesthetic description:

- triangular, but with one side shorter than the other so that we can tell their orientation
- a solid color (player two will be a different color)
    - yes, you may choose the color

Some components that they need:

- a rigidbody2D
- a collider (Google "Unity polygon collider")

Behavior:

- movement is "Asteroids" style
    - they have thrust (positive or negative)
    - they can turn (left or right)
    - we'll handle screen boundaries later
- pulling the ***right trigger*** (as in, right-hand) should fire a bullet (more on bullets later)

Shootin' Stuff

As we said, pulling the right trigger fires a bullet. Here are some bullet descriptions:

- a triangle the same color as the player that shot them
- moves around
- dies when it hits something other than another bullet
    - *id est*, when it hits ***either*** player or the boundaries of the level

    - we'll handle score later
- ***does not collide with other bullets at all***
    - Google "Unity collision layers"

Here are some components that bullets should have:

- a RigidBody2D
- a PolygonCollider2D

We Build the Best Walls

Okay, so we've mentioned walls a couple of times. Let's actually make them so that our player's/bullets actually touch them.

There are two basic ways to make walls:

1. You can make four separate walls, each with its own box collider, and position/stretch them correctly so that they cover the four sides of the screen.
2. You can use an *EdgeCollider2D*, which is a polyline with no internal volume (i.e., things can hit its edge, but can never get stuck "inside" it, wherever that is).

A constraint:

- the walls need to position themselves correctly based off of screensize (Google "Unity get screen size")
- no matter what screen I play it on, the tanks should hit the edge of the screen and not pass through

Adding Another Player

As *Three Dog Night* says, one is the loneliest number. Player two should behave/look pretty much the same as player one, with some exceptions:

- they should be a different color
    - their bullets should be this same color (i.e., the one that's different from player one's color)
- they should be controlled by the other controller

Scoring

Add some score!

- put the score for Player One in the top left and the score for Player Two in the top right
- when a bullet hits a player:
    - if that bullet came from *the other player*, *the other player* **gets two (2) points**
    - if that bullet came from *the same player*, *that player* **loses one (1) point**

In other words: if Player One shoots Player Two, Player One gets two (2) points. If Player One shoots herself, she loses one (1) point.

Cross-Platform Capability

If you've been working cross platform (i.e., you and your partner run two different operating systems), you may have already done this. If not:

- make your game cross platform
    - for all three major operating systems

Submitting

Double-check that everything's good to go, and submit ***one copy of your project***. Name it: "yournetid-theirnetid-homework5.zip"