Course 2 Unit 2 Exercise 6: Spawn and Explode

Although this exercise isn't worth any points, it gives you valuable programming experience. You're almost definitely going to have to complete the exercises to succeed in the course.

Getting Started - Clone your repository

1. Click on the appropriate link then accept the assignment to create your repository for submitting your work:
   a. Gallant AM: https://classroom.github.com/a/emA8xglG
   b. Gallant PM: https://classroom.github.com/a/na1XicFV
   c. Nunn AM: https://classroom.github.com/a/wkUxu18B
   d. Nunn PM: https://classroom.github.com/a/zwsGzAoT
   e. Wijaya AM: https://classroom.github.com/a/kHKGhxwZ
   f. Wijaya PM: https://classroom.github.com/a/dKLFXpsg
2. In GitHub Desktop, clone the repository you just created to your desktop.
3. Open the project in Unity.
   a. **Note:** When you open the provided Unity project, it will probably start in an Untitled scene. Double-click Scene0 in the Scenes folder in the Project window to open the correct scene.

We encourage you to make interim commits as you go. Use your commit message to indicate which step (e.g.: "Completed through step 5").

Problem 1 - Setting up the mouse button processer

1. Drag the MouseButtonProcessor script from the scripts folder in the Project window onto the Main Camera in the Hierarchy window.
2. Left click the Main Camera in the Hierarchy window to select it. Expand the prefabs folder in the Project window and drag the Explosion prefab onto the Prefab Explosion field in the Mouse Button Processor (Script) component in the Inspector.
3. Drag the TeddyBear prefab onto the Prefab Teddy Bear field in the Mouse Button Processor (Script) component in the Inspector.
4. In GitHub Desktop, commit your work with message: "Completed problem 1".

Problem 2 - Spawn teddy bear on left mouse button

1. Open the Input Manager, expand the Axes area if necessary, and add 1 to the value next to Size.
2. Expand the bottom input axis (the one you just added) and change the Name to SpawnTeddyBear. Change the Positive Button to mouse 0 and delete the Alt Positive Button.
3. Open the MouseButtonProcessor script in Visual Studio.
4. Add code to the `Update` method to instantiate the teddy bear prefab at the mouse location when there's input on the SpawnTeddyBear axis.

    a. The Mouse Button Processing Revisited lecture discusses how to only respond on the first frame of input on an axis; you should use that approach here to make sure you only spawn a single teddy bear when the player presses the left mouse button.
    b. The Mouse Location Processing lecture shows how to determine the location of the mouse in world coordinates.
5. Test your code to make sure it works properly and debug as necessary.
6. In GitHub Desktop, commit your code with message: “Completed problem 2”.

Problem 3 - Explode a teddy bear on right mouse button

1. Open the Input Manager, expand the Axes area if necessary, and add 1 to the value next to Size. Expand the bottom input axis (the one you just added) and change the Name to ExplodeTeddyBear. Change the Positive Button to mouse 1 and delete the Alt Positive Button.
2. Open the MouseButtonProcessor script in Visual Studio.
3. Add code to the `Update` method to explode one of the teddy bears in the game.
    a. The Mouse Button Processing Revisited lecture discusses how to only respond on the first frame of input on an axis; you should use that approach here to make sure you only explode a single teddy bear when the player presses the right mouse button.
    b. I already tagged the TeddyBear prefab with a TeddyBear tag, so you can get one of the teddy bears in the game using the `GameObject FindWithTag` method. Because that method can return null (if there are no teddy bears in the game), check for null before trying to blow up the teddy bear.
    c. If the teddy bear isn't null, instantiate the explosion prefab at the teddy bear's position and destroy the teddy bear.
4. Test your code to make sure it works properly and debug as necessary.

Submit your work

1. In GitHub Desktop, commit your changes with the message: “Ready for grading”.
2. Push your changes to the remote.
    a. By committing and pushing your updates to GitHub you have submitted your assignment on GitHub Classroom.
    b. If auto-grading is enabled, this will also check your code and provide automatic feedback on your code.
3. Return to CodeHS and respond to the prompt.