Course 2 Exercise 4: Teddy Bear Explosions

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.

Clone your repository

1. Click on the appropriate link below and accept the assignment to create your repository with the starter code and for submitting your work:

    a. Gallant AM: https://classroom.github.com/a/ySb2x87w
    b. Gallant PM: https://classroom.github.com/a/qZZ4Mh_D
    c. Nunn AM: https://classroom.github.com/a/KEIsTnQf
    d. Nunn PM: https://classroom.github.com/a/jdwzHTyA
    e. Wijaya AM: https://classroom.github.com/a/EstNDiC-
    f. Wijaya PM: https://classroom.github.com/a/hu88by-p

2. In GitHub Desktop, clone the repository you just created to your desktop.

3. Open the Exercise4 project in Unity.

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.

Problem 1 - Get the explosion working

1. Drag the Explosion prefab from the Prefabs folder in the Project window onto the Hierarchy window. Run the game and watch the explosion play and play and play ...

2. Drag the **`Explosion`** script from the Scripts folder in the Project window onto the Explosion prefab in the Prefabs folder in the Project window; this also attaches the script to the instance of the prefab in the scene.

3. Open the **`Explosion`** script in Visual Studio and add code below the comment in the Update method to destroy the game object if the explosion has finished its animation. You can use the Unity Scripting Reference to explore how to use the Animator **`GetCurrentAnimatorStateInfo`** method to figure out how to do this (the **`anim`** field is an **`Animator`** object).

4. Run the game. The explosion animation should disappear after it finishes playing the first time. You'll have to watch closely because this happens pretty quickly. If you look at the Hierarchy window as the game runs, you'll see the Explosion game object disappear from the Hierarchy window when that game object is destroyed.

5. Delete the Explosion game object from the Hierarchy window.

6. In GitHub Desktop, commit your work with message: “Completed problem 1”

Problem 2 - Get teddy bears spawning in scene

1. Create a TeddyBear prefab using the provided teddy bear sprite and TeddyBear script.

2. Add a Rigidbody 2D component to the prefab, making sure you freeze rotation in Z using the Constraints.
3. Add a Box Collider 2D component to the prefab, making sure you set the Material field to the TeddyBearMaterial in the materials folder in the Project window.
4. If you want to, you can edit the box collider for the TeddyBear prefab. To do that:
    a. Drag the prefab onto the Hierarchy window.
    b. Click the Edit Collider button in the Box Collider 2D component in the Inspector, then drag the green squares on the collider in the Scene view to make the collider fit the sprite more tightly.
    c. Click the Edit Collider button again to stop editing the collider.
    d. Click the Overrides dropdown next to Prefab near the top of the Inspector and click the Apply All button to apply your changes to the prefab.
5. Delete the TeddyBear game object from the Hierarchy window (if you have one there).
6. Remove gravity from the game.
7. Attach the provided `TeddyBearSpawner` script to the Main Camera, select the Main Camera, and drag the TeddyBear prefab onto the Prefab Teddy Bear field in the script in the Inspector.
8. Run the game and watch the teddy bears spawn and bounce off the edges of the screen and each other.
9. In GitHub Desktop, commit your changes with message: “Completed problem 2”.

Problem 3 - Kill teddy bears multiple ways

1. Add code to the `TeddyBear` `Update` method to destroy the teddy bear if the death timer has finished. No explosion here, the teddy bear just "goes gentle into that good night."
    a. From within a script, we can access the game object the script is attached to using `gameObject`. That means we can destroy the TeddyBear game object the TeddyBear script is attached to using `Destroy(gameObject);`
2. Play around with the TeddyBear TeddyBearLifespanSeconds field to change how quickly the teddy bear dies.
3. Add a TeddyBear tag to the TeddyBear prefab. To do this,
    a. Select the TeddyBear prefab in the Project window and click the Untagged dropdown next to Tag at the top of the Inspector.
    b. Select Add Tag ..., click the + below and to the right of the List is Empty message, type TeddyBear as the New Tag Name, and click the Save button.
    c. This adds a new tag to the game, but doesn't add it to the TeddyBear prefab.
    d. Select the TeddyBear prefab in the Project window again, click the Untagged dropdown next to Tag at the top of the Inspector, and select the TeddyBear tag at the bottom of the dropdown.

4. Set the Prefab Explosion field of the Teddy Bear (Script) component of the TeddyBear prefab to the Explosion prefab.

5. Add code to the body of the if statement in the `TeddyBear` `OnCollisionEnter2D` method to instantiate the `prefabExplosion` and destroy the teddy bear. You already learned how to destroy the teddy bear above. To instantiate the explosion prefab at the teddy bear's location (which is, of course, where the explosion should be) you can use `Instantiate<GameObject>(prefabExplosion, transform.position, Quaternion.identity);`

6. The Instantiate method is called a generic method because we tell it what type of object we're instantiating (in this case, a `GameObject`) between the < and the >.

    a. The first argument to the `Instantiate` method is the prefab we're instantiating.

    b. The second argument is the location where we want to instantiate the game object; `transform.position` is the location of the teddy bear.

    c. The third argument tells how much rotation to apply when we instantiate the game object, where `Quaternion.identity` means don't rotate at all.

7. Run the game again and watch both teddy bears explode when they collide with each other.

Submit your work

8. In GitHub Desktop, commit your changes with the message: “Completed problem 3”.

9. Push your changes to the remote.

    a. By committing and pushing your updates to GitHub you have submitted your assignment on GitHub Classroom.

10. Return to CodeHS and respond to the prompt.