Course 2 Exercise 3: Shrink and Grow

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 below and accept the assignment to create your repository with starter code and for submitting your work:
    a. Gallant AM: https://classroom.github.com/a/qnFcmTRf
    b. Gallant PM: https://classroom.github.com/a/vjaDl1jQ
    c. Nunn AM: https://classroom.github.com/a/X0ZDdxDb
    d. Nunn PM: https://classroom.github.com/a/yhZEWLDi
    e. Wijaya AM: https://classroom.github.com/a/XprF7BB8
    f. Wijaya PM: https://classroom.github.com/a/DlOhYxQz
2. In GitHub Desktop, clone the repository you just created to your desktop.

Create your Unity project and prepare for GitHub tracking

1. Use Unity Hub to create a new 2D Unity project named ***Exercise3***. Save the project in your new repository folder.
2. Once the project opens in Unity, go to File Explorer and move the **_UnityProjectRoot.gitignore** file into the Unity project folder and rename it to **.gitignore**
3. Go to GitHub desktop and commit your changes with the message: “Create initial Unity project”. Make sure there are only about 30 files being committed.
    a. If you have thousands of changed files, return to step 2 to make sure you’ve named the gitignore file properly and that it is placed at the root of the *Unity* project not in its original location.
    b. Ask for help if you are unsure.
4. Push your changes to the remote.

At this point you are ready to proceed with this assignment. 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 - Grow out of control!

1. Rename the SampleScene as Scene0.
2. Add a new Sprites folder to the project window and use your OS to copy a sprite of your choosing into that folder.
3. Drag the sprite into the Hierarchy window to create a game object in the scene.
4. Run the game and watch nothing happen.
5. Create a new Scripts folder in the Project window and create a new C# script in that folder called Resizer.

6. Open the new script in Visual Studio and add a documentation comment for the class. The **`Resizer`** class (script) will shrink and grow the game object over time.

7. We need fields to support both our standard timer and the resizing process. Write a line comment (just above the comment for the **`Start`** method in the script) indicating that the fields you're declaring next are for timer support. Declare a **`float`** **`TotalResizeSeconds`** constant and set it to 1 just below your new comment.

8. Declare a **`float`** **`elapsedResizeSeconds`** variable and set it to 0 just below your new constant.

9. Make sure your code compiles without errors (don't worry about warnings -- for example, a warning saying you're not using the **`elapsedResizeSeconds`** field yet).

10. Write a line comment indicating that the fields you're declaring next are for resizing control.

11. Declare a **`float ScaleFactorPerSecond`** constant and set it to 1. Setting this constant to 1 will make the game object double in size in the first second, triple in size over the first 2 seconds, and so on.

12. Declare an **`int scaleFactorSignMultiplier`** variable and set it to 1. We'll use this variable to either grow (when it's 1) and shrink (when it's -1) the game object as appropriate.

13. Make sure your code compiles without errors.

14. Delete the **`Start`** method, including the comment above it; we don't need that method for this problem.

15. Add a line comment and code to the **`Update`** method to resize the game object.

    a. You can remind yourself how to change the **`localScale`** of the game object by looking at the **`YellowTeddyBear`** script from Exercise 11 in the previous course, but of course you can't just multiply **`x`** and **`y`** by 4.

    b. Instead, you need to use the resizing control fields you declared above and the **`Time.deltaTime`** value to change the scale properly. Whatever you come up with for your solution should make the game object twice as large in approximately a second.

16. You can just see if it "looks right" at this point; we'll use the debugger to carefully check it soon.

17. Run the game. Swear because nothing happens. Attach the Resizer script to your game object in the Hierarchy window. You should see the game object grow (out of control!) as the game runs.

18. Play around with the **`ScaleFactorPerSecond`** constant to make the game object grow faster or slower.

19. Save your work and go to GitHub Desktop to commit with message: “Completed problem 1”.

Problem 2 - Shrink and grow

1. Add code to the **`Update`** method to update the timer and check if it's finished. Look at the solution to Exercise 1 if you need help with this.

2. In the if body of the code you wrote in the previous step, add code to reset the timer and change the sign of the **`scaleFactorSignMultiplier`** field. In other words, if it's 1 make it -1 and if it's -1 make it one. Do this with a multiplication, don't write an if statement to do it!

3. Run the game and watch the weirdness.

4. Save your work and go to GitHub Desktop to commit with message: “Completed problem 2”.

Problem 3 - Use the debugger

1. We said in Problem 1 that we'd use the debugger to check the resizing, so let's do that now. In your IDE, set a breakpoint on the line of code that resets the timer. You set a breakpoint by clicking in the gray bar to the left of the line of code, at which point a red circle appears at that location.

2. Select Attach to Unity in the menu bar near the top. Go to the Unity editor and run the game.

3. After the game has run for approximately a second, you'll be back in Visual Studio at the breakpoint (you may have to open Visual Studio yourself).

4. Hover the mouse over **`transform.localScale`** a few lines of code above the breakpoint. If you implemented the scaling properly, the value should be **`(2.0, 2.0, 1.0)`**. Select Debug > Stop Debugging or click the red square in the menu bar near the top to stop debugging.

5. Remove the breakpoint by clicking on the red circle.

6. Stop the game in the Unity editor.

7. Play around with the **`TotalResizeSeconds`** constant to change the duration of the growing and shrinking.

8. Save your work and go to GitHub Desktop to commit with message: “Completed problem 3”.

Submit your work

9. Make a final test of your code to make sure it is working as expected.

10. If you made any additional changes to your code, make sure you commit them before pushing your changes to the Remote.

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

11. Return to CodeHS and respond to the prompt to complete this assignment.