Course 2 Unit 3 Project 3: Let It Rock

Project Description

You've decided to continue building the simplified version of the classic arcade game Asteroids that we started in the previous week. You won't actually complete the full version of the game until the beginning of the next course, but game developers typically break their game development into a number of increments. This project is the second increment in your development of this game.

In this project, you'll implement a moving asteroid for the game, including screen wrapping.

Why do we care?

Because it's cool to actually build a real game, even a simple one! This project is the second increment in your implementation of a simplified version of Asteroids.

Learning Goals

The learning goals for this assignment include many of the learning objectives from all the units in the course. The overall learning goal is:

- Create a Unity game that processes keyboard input.

Overview

In this project (the first increment of our Asteroids game development), you're building the Unity project and adding the player ship to the game. **This increment of the project usually takes around 3 hours to complete, so you should expect it to take you two class periods to complete.**

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/C508YDMZ
    b. **Gallant PM**: https://classroom.github.com/a/a5_cbXqY
    c. **Nunn AM**: https://classroom.github.com/a/Di-D_5ps
    d. **Nunn PM**: https://classroom.github.com/a/FOdpHWz0
    e. **Wijaya AM**: https://classroom.github.com/a/_3S5LMEH
    f. **Wijaya PM:** https://classroom.github.com/a/OG-I4Pko
2. In GitHub Desktop, **clone** the repository you just created to your desktop.
3. The repository contains a number of useful materials, including my Asteroids project with the solution to the previous increment.

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.
b. If you prefer, you may build onto your version of the last game, but make sure it's working properly first. Make sure you include the same **.gitignore** file that is in the Asteroids project provided.

Implementation Requirements

I've provided detailed step-by-step instructions below for you to follow if you'd like to have a "guided development experience" for this project.

Of course, if you would like more of a challenge, you're also welcome to implement your solution without following those instructions. However, you still need to implement all the required functionality for the project. Specifically, your solution needs to:

- Include a moving asteroid in your game that starts moving in a random direction with one of 3 random sprites
- Screen wrap the asteroid. For example, when the asteroid leaves the bottom of the game window it should re-appear at the top of the game window

Screen Wrapping

Caution: Even if the screen wrapping is working properly, so a player could play the built game and screen wrapping would work perfectly, it may not seem to be working in the Unity Editor. The best thing to do is to build the game and play the built game. If, however, you want to just stay in the editor, double click the Main Camera in the Hierarchy window, then use Ctrl + Middle Mouse Wheel to zoom in on the Scene view until the box that shows the edges of the camera view just disappears from view.

Testing Your Game

Here are the rubrics to confirm that your game functions correctly:

1. Asteroid is in the scene when the game starts
2. Asteroid starts moving in a random direction
3. Asteroid is one of 3 random sprites (you'll have to play the game multiple times to check this)
4. The asteroid screen wraps properly. Wrapping should work correctly for the left, right, top, and bottom sides as well as all four corners.
5. The ship screen wraps properly. Wrapping should work correctly for the left, right, top, and bottom sides as well as all four corners.

Increment 2 - Detailed Instructions

In this Unity project (the second increment of our Asteroids game development), you're adding a moving asteroid to the game, including screen wrapping.

Step 1: Add a moving asteroid

For this step, you're adding a moving asteroid to the scene.

1. Add sprites for three different asteroids to the Sprites folder in the Project window. **CAUTION: Make you asteroid sprites the correct size for your game; don't scale your game object in Unity.**
2. Drag one of the asteroid sprites from the Sprites folder in the Project window onto the Hierarchy window.
3. Rename the resulting game object Asteroid.
4. Add a Rigidbody 2D component to the Asteroid game object.
5. Add a new C# script named Asteroid to the Scripts folder in the Project window.
6. Double click the Asteroid script to open it in Visual Studio.
7. Add a documentation comment above the line declaring the class.
8. Delete the `Update` method from the `Asteroid` script.
9. Add code to the `Start` method to apply a random impulse force to the asteroid to get it moving. You should try to do this on your own first, but feel free to use the code in the **random impulse force.txt** file included in the Instructions folder if you get stuck.
10. Add the Asteroid script as a component to the Asteroid game object.
11. In GitHub Desktop, commit your changes with message: "Completed step 1".

When you run your game, you should see the asteroid move in a random direction from the center of the window.

Step 2: Make moving asteroid a random sprite

For this step, you're making the moving asteroid randomly pick from the three asteroid sprites you included in the game.

1. Double click the Asteroid script to open it in Visual Studio.
2. Add three fields to the class to hold the three different asteroid sprites. Be sure to mark the fields with `[SerializeField]` so you can populate them in the Inspector.
3. Add code to the Start method to randomly pick one of the three asteroid sprites for the asteroid. You'll need to access the `SpriteRenderer` component of the asteroid so you can change the sprite for that component.
4. In the Unity editor, populate the new fields in the Inspector with the three asteroid sprites.
5. In GitHub Desktop, commit your changes with message: "Completed step 2".

When you run your game, you should see the asteroid, using one of the three asteroid sprites, move in a random direction from the center of the window.

Step 3: Make moving asteroid wrap

For this step, you're making the moving asteroid wrap when it leaves the screen instead of disappearing from the game forever.

You should immediately realize that we already added screen wrapping functionality for the ship in the previous iteration, and the code here should work exactly the same way. Instead of copying and pasting the screen wrapping code from the Ship script into the `Asteroid` script – that approach is almost always a horrible idea! – we'll remove the screen wrapping code from the Ship script and include it in a new `ScreenWrapper` script. We can then add our new script to the Ship and Asteroid game objects and they should both work fine.

1. Add a Circle Collider 2D component to the Asteroid game object. Edit the collider as you see fit, making sure the circle collider is completely inside the sprite for your asteroid.
2. Create a new C# script in the Scripts folder in the Project window and name it ScreenWrapper.
3. Double click the ScreenWrapper script to open it in Visual Studio.
4. Delete the `Update` method from the `ScreenWrapper` script.
5. Cut the field that stores the radius of the collider from the Ship script and paste that field as a field in the `ScreenWrapper` script.
6. Cut the code that retrieves the CircleCollider2D component and saves its radius into your field from the Ship Start method and paste it in the `ScreenWrapper` `Start` method.
7. Cut the `OnBecameInvisible` method from the Ship script and paste it in the `ScreenWrapper` script.
8. Add the ScreenWrapper script as a component to both the Ship and Asteroid game objects in the Unity editor.
9. **Caution: Even if the screen wrapping is working properly, so a player could play the built game and screen wrapping would work perfectly, it may not seem to be working in the Unity Editor. The best thing to do is to build the game and play the built game. If, however, you want to just stay in the editor, double click the Main Camera in the Hierarchy window, then use Ctrl + Middle Mouse Wheel to zoom in on the Scene view until the box that shows the edges of the camera view just disappears from view.**
10. Test the game, checking against the rubic to make sure it is working correctly and apply fixes as necessary.

When you run your game, you should see the asteroid, using one of the three asteroid sprites, move in a random direction from the center of the window. When the asteroid leaves the screen it should wrap back into the screen. You should also make sure ship wrapping still works properly.

Note that if you left your asteroid on top of the ship in the scene, when the scene starts up they're immediately in collision so they both start moving. That's fine at this point; we'll make the required changes later to actually kill the ship when it collides with an asteroid. I moved my ship out of the way so I could see the asteroid movement without that collision.

Submit your work

1. In GitHub Desktop, commit your changes with the message: “Ready for grading”. If you didn’t need to make additional changes, move on to the next step.
2. Push your changes to the remote/origin.
    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.