

# GDD 1200 Project Increment 2

## Help Policy

The help policy I have for project increments is significantly more strict than for the programming assignments. Some people describe the policy as an “empty hands” policy; basically, you can talk to people about problems you’re having, but you’re not allowed to type on your computer or write anything down to take away with you after the discussion ends.

## Overview

In this project increment, you're adding a moving asteroid to the game.

## Getting Started

Open GitHub Desktop and make sure the Current Repository on the upper left is set to gdd-1200-<your GitHub user name>. Click Fetch origin near the upper middle to make sure you have the most recent copy of your repo.

Use File Explorer (Windows) or Finder (Mac) to navigate to your local repo. Navigate into the PI2 folder in your repo.

The folder contains a snippet of code for you to use for the assignment.

To start this project increment, you can download my solution to Project Increment 1 (available on Canvas) and extract it into the PI2 folder in your repo or you can copy your Asteroids folder from the PI1 folder in your repo into the PI2 folder in your repo.

## 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 `PI2` folder in your repo if you get stuck.
10. Add the Asteroid script as a component to the Asteroid game object.
11. Use GitHub Desktop to commit your changes to your local and remote repos.

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. Use GitHub Desktop to commit your changes to your local and remote repos.

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 increment, 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. Use GitHub Desktop to commit your changes to your local and remote repos.

**Caution 1:** 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 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.

**Caution 2:** Because of the way the Unity Editor and the `ScreenUtils` class work, you should NOT run your game as full screen in the editor.

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.

## Solution Requirements

Your solution to this problem must:

- Implement all the functionality described above
- Comply with the GDD Coding Standards

## Turning In Your Assignment

Use File Explorer (Windows) or Finder (Mac) to navigate into the `PI2\ Asteroids` folder in your repo. **Important: Delete the Library and Logs folders before continuing.** Navigate up one folder. Compress the `Asteroids` folder into a zip file and submit it in the appropriate assignment on Canvas.

Use the default Windows compression utility for this; the grader may not own WinZip, 7Zip, or whatever other program you're using to build your zip file. If you use something other than the

Windows compression utility and the grader can't unzip your submission, you'll get a 0 on the assignment.

### **Late Turn-ins**

- Turn-ins are due at the beginning of the scheduled class time on the specified due date.
- No late turn-ins will be accepted.