Challenge 1 - Scripting Needs

Refer to “About the Challenges and Solutions” for more information about readings of this type.

Throughout Courses 1 and 2 of this series, we will be developing *AsteraX*, a modern take on the classic arcade space shooter *Asteroids*. Of course, you may have developed a similar game on your own, but in this project, we have worked to specifically include concepts that will be covered on the Unity Certified Programmer exam. By working through the challenges in these courses and those that follow, you will encounter nearly all of the concepts in the exam and be better prepared to take it with confidence.

Scenario:

All game development needs to start somewhere, and often that is something like the Requirements Document that we provide here (as the next item, and also available to download in the Session Resources). This document describes the core gameplay and all the features you will need to implement over Courses 1 and 2 of this sequence.

Referencing these requirements, start thinking about your project architecture: what scripts and classes will you need to make this work, and how will they work together?

Challenge:

Draw a "Components and Classes Diagram" (CCD) that covers all of the Script Components and C# classes you will need to implement the *AsteraX* requirements in a Unity project; a map of the code that you can use to guide you through all the programming tasks ahead. There is no exact right answer for to this challenge (or any of the programming challenges in this course), but do take some time to think through the scripting needs and—after you have finished creating your document—the instructor will walk through one valid solution.

Your solution should include:

- A list of Unity components that your spaceship, asteroids, and other GameObjects will need
- An outline and/or diagram of the classes and methods that you will need to use or create
- Details on how information is passed between GameObjects, when required
- This does not need to be formal, just make it something that you could refer to or that you could use to explain your approach to another programmer.

Assume that you will use Unity 2017.4 LTS to implement the game.

When you’re finished, complete the Self-Evaluation coursework before continuing to the instructor’s solution.

Hints and Tips:

There are many ways to complete this challenge, so we've provided you with an informal example based on a simplified *Pac-Man*-like game, also available in the Session Resources. This example includes both a Requirements Doc and a Components and Classes Diagram (CCD).