Space Game Workshop
Level – Beginner
Focus: Introducing Game Development
Goals:

1. **Explore Game Programming Fundamentals**
2. **Encounter mild (intentional) bugs, and identify solutions**
3. **Identify other potential uses for common components and programming practices**
4. **Explore relationship between art and programming in Game Design**

Steps:

- Introduce Entity-Component Structure of Game Engine

(This is a GameObject! It has components)

What is a Component?

A script that is attached to a GameObject to give it new features. Sort of like a MOD.

What are the different types of Components?

Transform
Renderer (multiple types... Sprites, Mesh, etc)
Rigidbody + Collider
Custom Scripts

- Introduce the Ship Component

We are going to use a custom component called 'Ship' to move our player, and shoot the targets. This script is already attached to our ship, but it's a bit incomplete. We'll need to fill in some of the gaps to get it to work.

Open Ship.cs

- Breakdown of C# Programming (examples are in the script)

What is a ***class****?*
What are ***variables***?
What are ***functions***?
Optional: write an Add(int x, int y) function

Monobehaviour functions

Special functions provided by the game engine, that it calls automatically. This is what makes the game run!

Modifying the Ship Script

HandleMovement() is being called on a loop by Update. But it doesn't do anything yet! Why not?

It has a bunch of IF statements with nothing in them! We need to fill these.

Filling The Conditionals

Movement can be measured using coordinate values stored in a **Vector3**. Every frame we'll modify the directionThisFrame by adding in a Directions based on our key-presses.

EXAMPLE:
directionThisFrame = directionThisFrame + Vector3.right;

Show off moving in each direction after adding in some code.

Spacebar will need to use the provided **Fire()** function instead!

Assigning References

Oh no! The game engine throws errors when we try fire the blaster. What's going wrong?

We have unassigned references! Unity is very picky, and if you are trying to use unassigned variables it gets confused and scared and DIES.

The laserbolt prefab is not assigned in the editor. You need to drag it in from the assets folder.
The firePointA and firePointB values need to be assigned from the hierarchy. They are **child** GameObjects to the **parent** Ship, and will follow it around everywhere!

Rigidbody Failure

Even with our references filled, we still need to change a few settings.

1. The LaserBolt prefab's Rigidbody has gravity enabled!
   a) Try setting its gravity scale to zero.
2. Too many LaserBolts?
   a) Swap Input.GetKey(KeyCode.Space) to Input.GetKey**Down**(KeyCode.Space)
   b) There are other, more complicated ways to manage a staggered, repeating fire rate. But this is probably the most intuitive solution for a beginner.

- Customizing Artwork by Creating/Editing Sprites in Piskel

Piskel

The sprites developed for this workshop were made in a web-based application called Piskel (**piskelapp.com**). The source files have been included in the Unity Package and can be used for editing.

Importing a Sprite to Piskel

Import the piskel file into the editor using the sidebar.

Editing Your Sprite

On the right hand size, choose the layer you'd like to paint on. Selecting the layer will highlight the paint that exists on that layer.

Choose the brush tool (or the symmetrical brush tool) on the left hand side to start painting pixels! Your color palette can also be selected towards the bottom left. The left/right mouse buttons use the foreground/background paint swatches respectively.

A transparent paint swatch works the same as the eraser tool, so you may want to simply leave the background swatch as transparent for more convenient erasing.

Exporting a Sprite From Piskel

When you're finished, you can export your sprite as either a single image, or by layers.

Hit the 'export' button on the right toolbar to start the process.

- Single Image:
  - Choose **PNG** from the image types, and choose the top download option.
- Layers:
  - Choose **Zip** from the image types, and select 'split by layers'. Then, download the Zip file.
  - You will need to extract the files from the zip file to use them. Drag them from the zip to the desktop or another folder on your computer, then import them into Unity through Assets > Import (or simply drag them into the project's Assets folder)