# COMP30019 – Graphics and Interaction
# Lab 1

**The University of Melbourne**

**Introduction:**

In this lab you will be introduced to Unity, a powerful games and simulation engine which facilitates the construction of rich interactive scenes.

You will need to work with three files to start off with:

- *MainScene.unity* – A Unity scene containing a partially formed cube entity. You will be working with this scene.
- *CubeScript.cs* – The C# script which constructs and renders the cube geometry, attached to the cube entity.
- *VertexColorShader.shader* – A Cg/HLSL shader used to render the cube. Note that you do not need to worry about the workings of shaders for this lab. We will briefly modify the file for the purposes of some demonstrations, but you do not need to understand how the code in it works.

**Tasks:**

1) Open *MainScene.unity* in Unity. Press the 'Play' button and you should see a partially rendered cube. Switch to the 'Scene' tab so you can navigate the view using the mouse: The middle button/scroll wheel allows you to drag/zoom the view and dragging with the right mouse button allows you to rotate the view.
    a. Navigate the view to look at the cube from different angles. It may help to right-click the 'scene gizmo' located in the top-right of the viewport in order to select different angles to view the cube from. Which two faces are missing?
    b. Open *VertexColorShader.shader* and remove the line `Cull Off` and save the file. Make sure you keep this code removed for the remainder of the lab. Examine the cube again in the Unity editor. What did `Cull Off` do?
2) Add vertex definitions to *CubeScript.cs* to complete the cube (add the missing faces). Culling is now on, so make sure you use the correct vertex winding order such that the triangles you define are visible from the outside of the cube.
3) Create a new script component called *PyramidScript.cs* to define an upward facing square pyramid. The tip of the pyramid should be located at (0.0, 1.0, 0.0). You may wish to copy the cube script as a starting point. Give each face a unique colour. Remember to create a new empty game object to attach the script to, and make sure you set its transform component so it does not overlap the cube in any way.
4) Navigate the view to look from the inside of the cube. Since back-face culling is on, you won't be able to see the inside faces of the cube. Modify the index generation code in *CubeScript.cs* (near the end of the script) so that you can see the interior of the cube instead of the exterior. What other ways could you achieve this effect (there are at least two)?

Constructing 3D meshes by manually defining vertices is tedious and rarely done in practice. However, the ability to define vertices *in code* is useful if we wish to procedurally generate a mesh based on some sort of algorithm.

5) ***Challenge.*** This exercise is a great preparation for assignment 1. Write a script to *procedurally generate* a closed cone. Use public class attributes to allow parameters of the cone to be set from the editor, such as the radius. Hint: You can think of a cone as a generalisation of the pyramid you manually defined in this lab. The vertices of the base will form a circle rather than a square, and there'll be one vertex for the tip. Maths functions like `sin()` and `cos()` should help.

An alternative to defining meshes programmatically is to use a *3D modelling tool* to construct them using a graphical user interface (a free, open-source tool you might wish to try is called *Blender*). Models can be saved as various file formats, many of which can be read by the Unity engine. A model can be imported to a Unity project simply by placing it in the "Assets" folder. Note that a "model" is typically more than just a "mesh" (although it doesn't have to be). This is because meshes alone cannot capture the complexity of some objects.

6) ***Extension.*** Search the web for some 3D models and try to import them into your Unity project. Models with the .fbx extension are generally well handled by Unity, but there are other formats work well too. Always keep in mind copyright and licensing when including such resources in your projects – there are many freebies out there, but licenses sometimes prohibit usage under certain circumstances (i.e. commercial uses).