COMP30019 – Graphics and Interaction

Lab 1:
Introduction to Unity

Lecturer: Dr. Jorge Goncalves
Tutor:

THE UNIVERSITY OF MELBOURNE

Welcome!

- This is a fun, but also very useful subject! Good choice!
- There will be 10 two-hour labs for the semester
- Labs will largely involve Unity – the lab computers have a very recent version installed (2018.1).
- In the first half of the semester you will learn the tools to develop powerful interactive applications – in the second half, you'll learn how to *evaluate* them from a human-computer interaction (HCI) perspective.
- More than just a "game development" subject – although games will be the primary focus of the main project.

Strongly advised…

- Play around with Unity **in your own time**! It is assumed you will use many online resources outside of class to learn the engine in detail. It is impossible to cover everything in-class.
- Object oriented programming experience is important! Expect to do a lot of extra work if you have not taken an OOP subject before.
- Please use the same version of Unity on your personal computers as that on the lab computers, particularly for assignments.

Unity

- Unity is a powerful 3D games and simulation engine
- Major titles have been developed using Unity
- Scripting in C#, shaders in Cg/HLSL
- Rapid development and maximal re-use
- Cross-platform
- Component based architecture

Component Based Architectures

- Arguably the most flexible engine architecture type for games
- Mitigates the tight coupling between different sections of code to greatly enhance reusability
- Based on the idea of preferring “composition over inheritance”
- Unity utilises a component based architecture!

Unity's Architecture

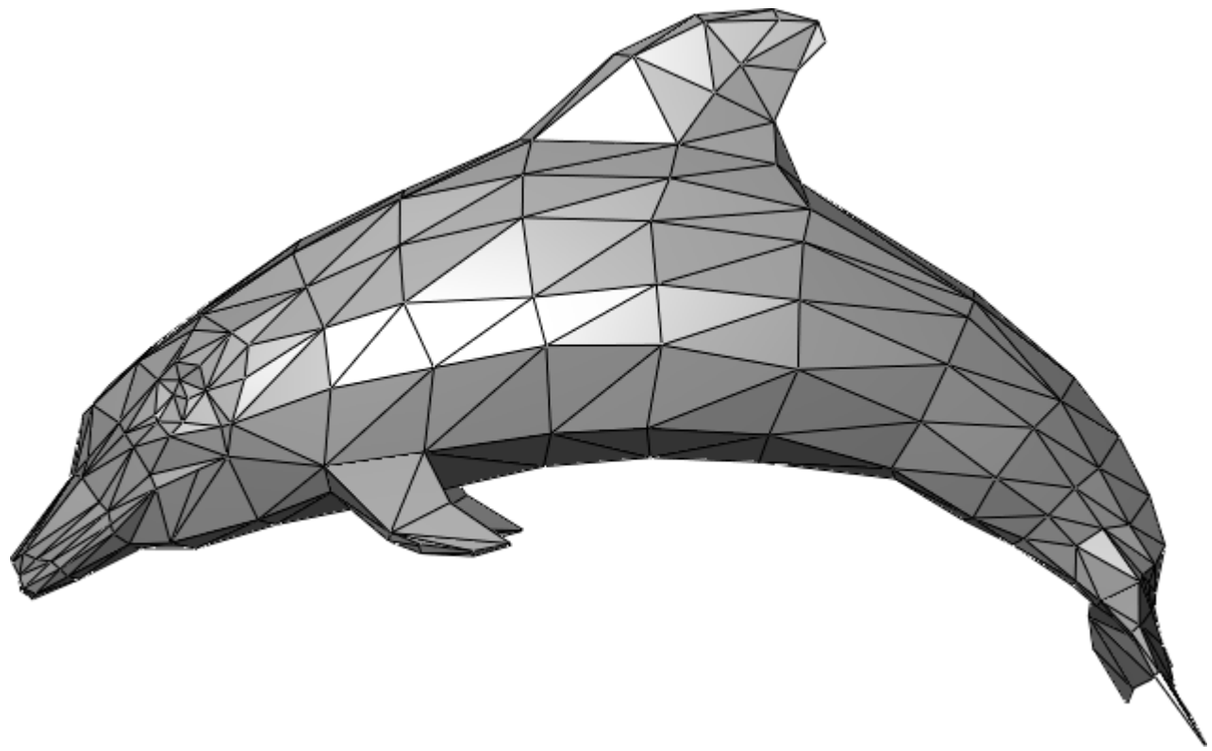
- A *scene* is made up of entities called "game objects"
- All game objects have a "Transform" component, describing their position, rotation and scale in the game world.
- A game object can have 0 or more additional ***optional*** components attached to it
- Some example entities/game objects:
  - <Transform>
  - <Transform, Mesh Filter, Mesh Renderer>
  - <Transform, Camera>
  - <Transform, Spot Light>
  - <Transform, Mesh Filter, Mesh Renderer, HealthScript>

3D objects

- How do we represent 3D objects in a scene?

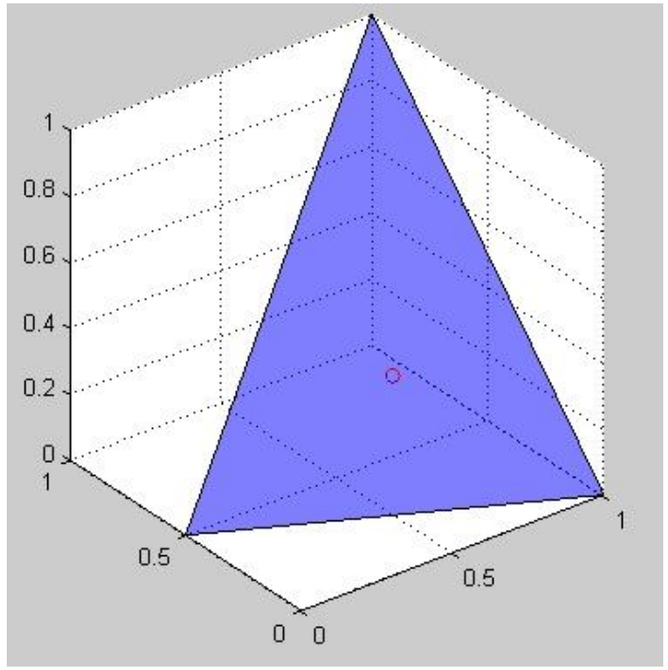
3D objects

- How do we represent 3D objects in a scene?



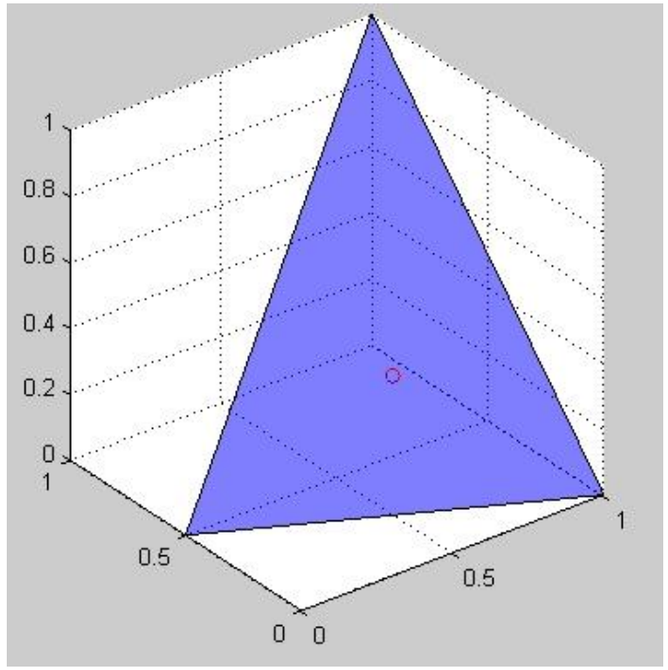
Triangles

- A triangle can be defined in 3D space by means of three points (or *vectors*).
- For example:
- <0.5, 0.0, 0.0>, <1.0, 1.0, 0.0>, <1.0, 1.0, 1.0>



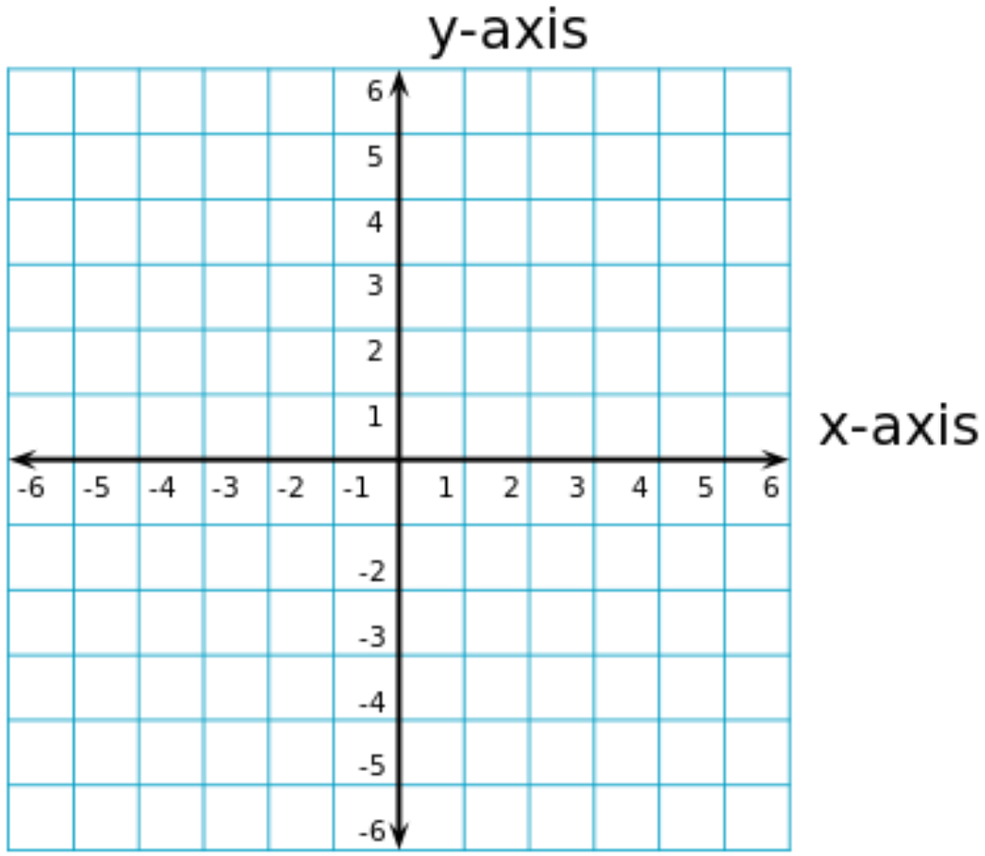
Triangles

- A triangle can be defined in 3D space by means of three points (or *vectors*).
- For example:
- <0.5, 0.0, 0.0>, <1.0, 1.0, 0.0>, <1.0, 1.0, 1.0>

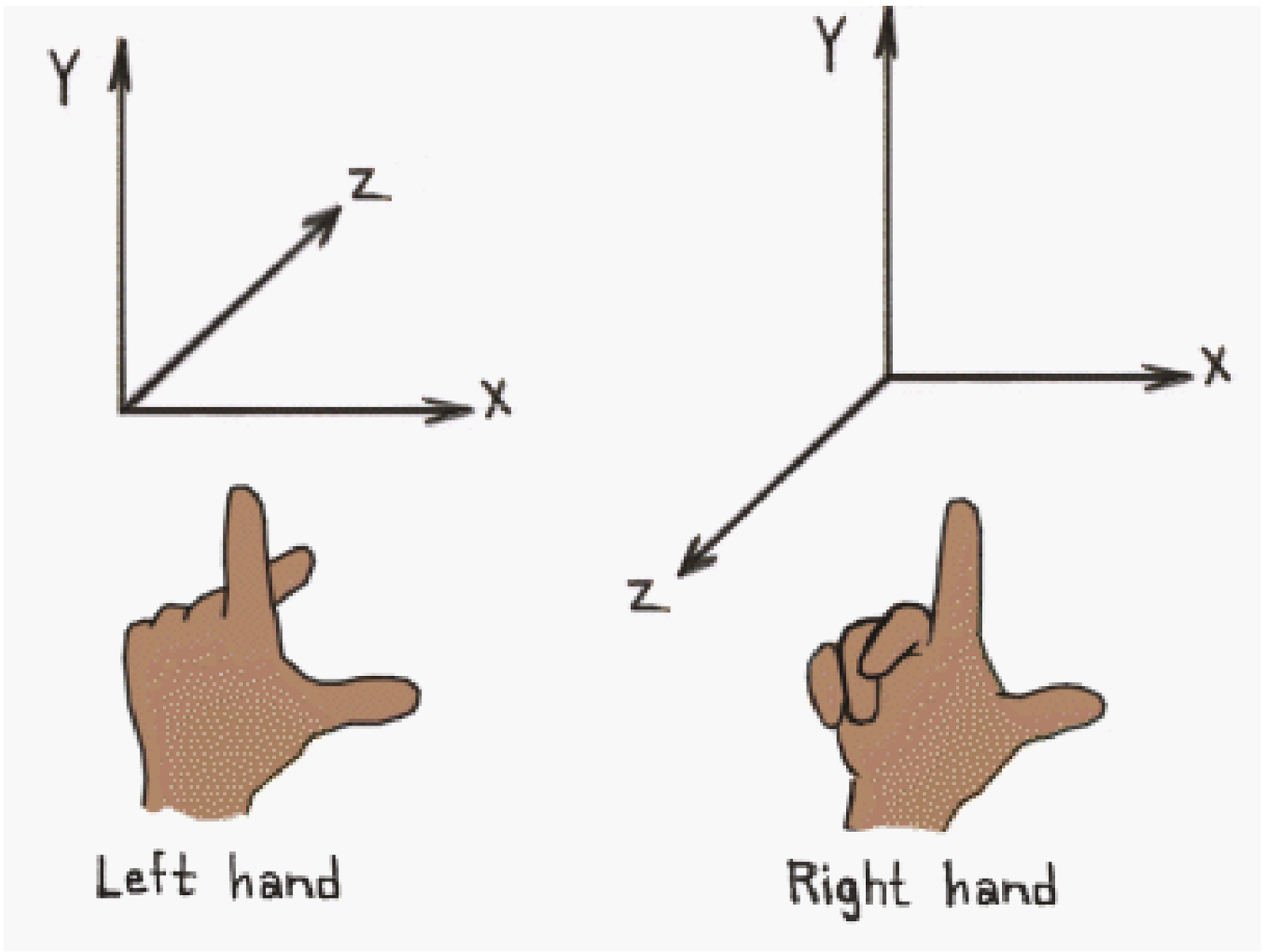


Coordinate systems

- In order for sets of vertices to be meaningful, we need to define some axes
- But in 3D we also have a z-axis. Which way does the z-axis go?



Left vs right handed coords

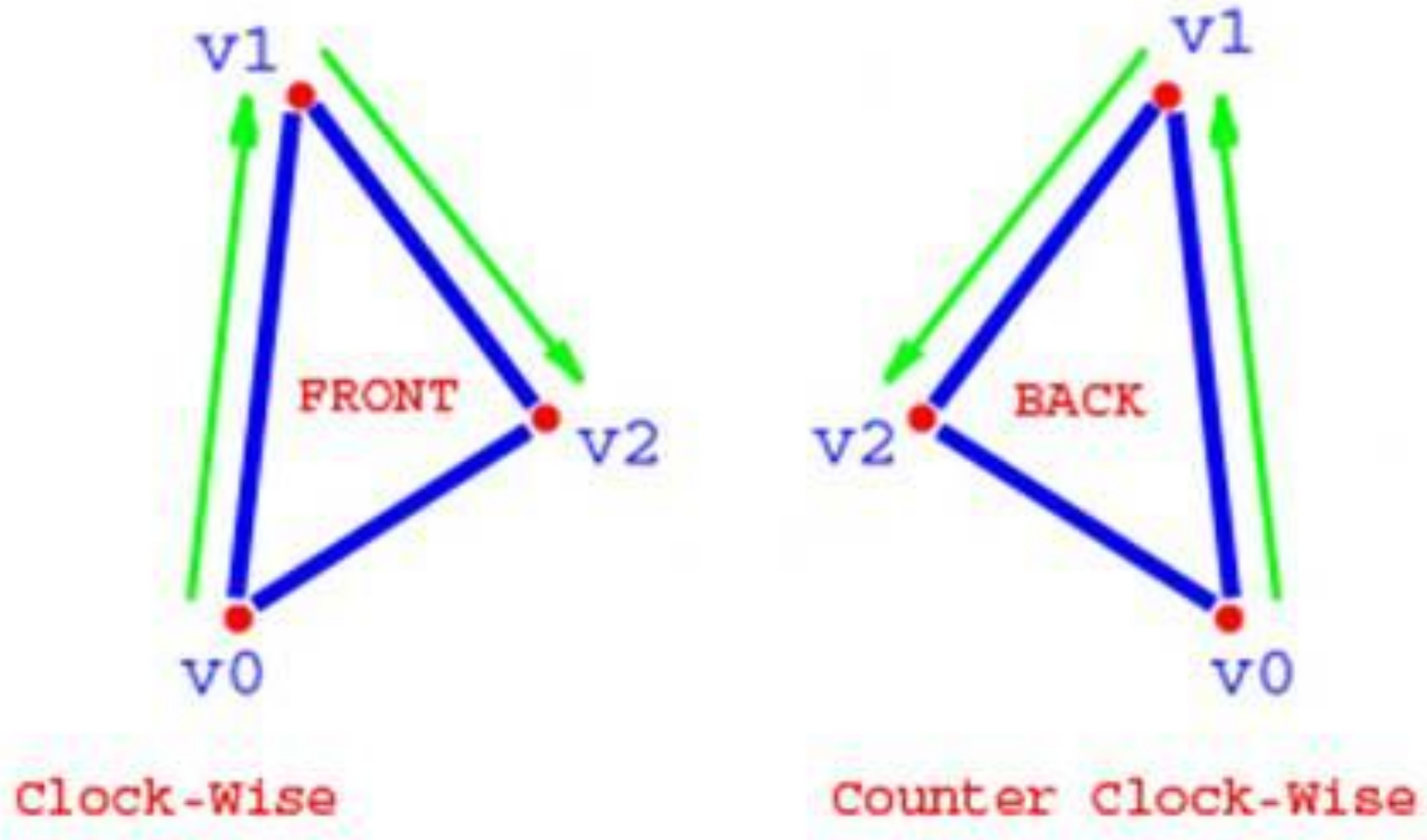


http://viz.aset.psu.edu/gho/sem_notes/3d_fundamentals/html/3d_coordinates.html

Front/back faces

- 3 vertices define a triangle.
- What defines which side is the “front” of the triangle?

Vertex winding order



Front/back faces

- Why do we care which side is front facing?

Culling

- We usually cull the back faces! Most objects are *closed,* so there's no point in rendering the back faces.