Computer Graphics

COMP3421/9415
2021 Term 3 Lecture 2

What are we covering today

How do Computers make Graphics?

- Hardware - Monitors and GPUs
- What's in the screen? Pixels and colours
- What's the GPU? A computer inside your computer
- What is "rendering" (Polygon Rendering)?
- What is OpenGL?
- What are Shaders?
- How are we coding in this subject?

Graphics Hardware - Monitors

A two dimensional array of lights

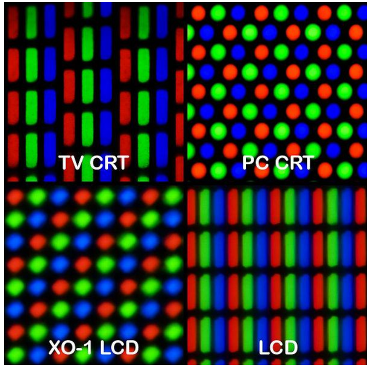
- A panel with an array of light sources . . .
  - LCD, LED, OLED etc etc . . . Cathode Ray Tube :P
- Each little light (called a pixel) has separate red, green and blue capability
- The image on screen can refresh multiple times a second
- Current standard is around 1920 x 1080 at 16-24bits per pixel and 60hz

A cathode ray tube monitor from the 1990s
Image credit: Daniel Christensen

Monitors - Illusions of Reality

None of this is real

- The illusion of colour
- The illusion of complete objects
- The illusion of movement



1mm of pixels in different formats
Image credit: Wikipedia user Pengo

Graphics Hardware - Graphics Cards

Specialised hardware

- A computer in your computer
- Has a processor . . .
- . . . and its own memory
- Receives data through the motherboard
- Only outputs video to the monitor *(we're disregarding GPGPU for the moment)*

Image credit: Nvidia

Graphics Cards - Historical Perspective

Why is this a separate piece of hardware?

- From a bandwidth perspective:
    - A sample monitor has 1920 x 1080 pixels in 24bits of colours = approx 6MB
    - Refreshing 60 times a second = approx 365MB a second
    - PCI Express bus bandwidth is approx 266MB a second
- Without a graphics card, we lack the bandwidth to refresh a monitor
- Historically, this was one of the first reasons why separate graphics hardware was created (back when the numbers were more like 640 x 480)
- However, it's not just a bandwidth issue!

Graphics Hardware vs the Computer's CPU

A massively parallel floating point calculator

Regular CPU (Intel Core i9)	GPU (Nvidia 3090)
8 processing cores @ 3.6GHz	10,496 processing cores @ 1755MHz
Each core runs different processes independently	Every core is running the same code at the same time
Large range of possible instructions	Limited set of calculations available

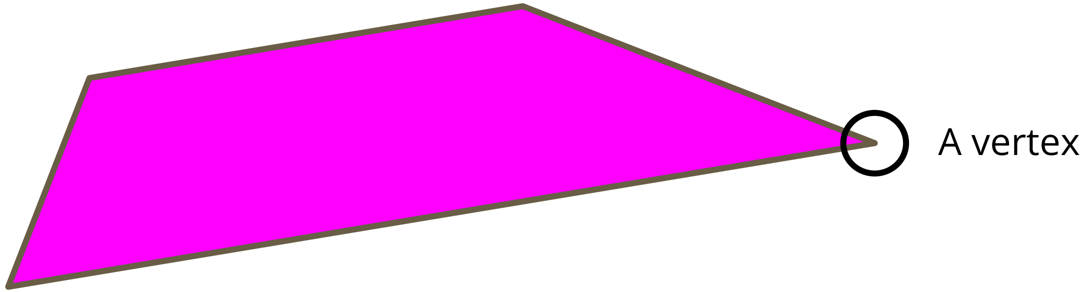
A funny take on this from an Nvidia conference: https://youtu.be/-P28LKWTzrI

Polygon Rendering (an introduction)

Polygons

Taking a concept of a scene and turning it into an image

- This is a huge concept . . . we'll start simple
- A polygon is a shape, made up of vertices (corners) and edges (lines)



Polygons into Meshes

A complex object can be made up of many vertices and edges

- Pieced together, we call this a "mesh"
- A series of polygons where some of them share vertices
- This creates a "surface"

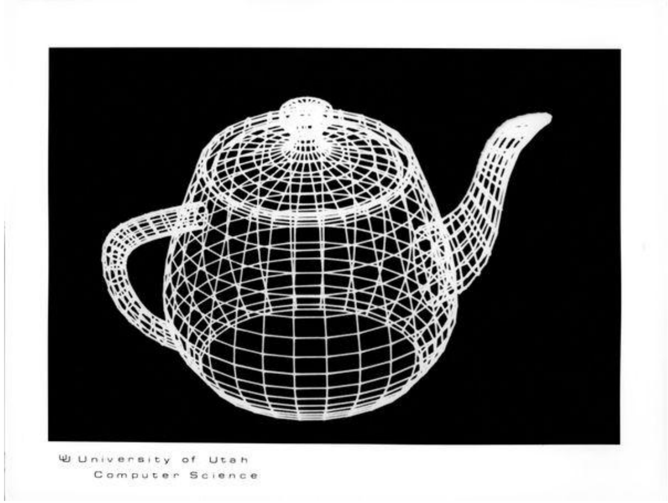


Image credit: School of Computing, University of Utah

Vertices and Coordinates

Keeping track of vertices

- Vertices have coordinates, like x,y,z
- A group of floating point numbers
- Data Structures!
- Vectors of vertices
- Coordinates also allow us to apply techniques from linear algebra (transform matrices)

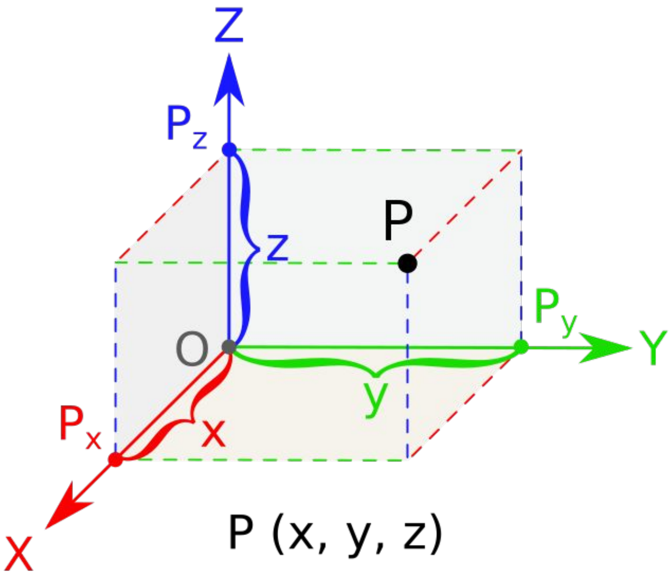


Image credit: Wikipedia user Андрей Перцев

View Projection

How do we "look" at a virtual scene?

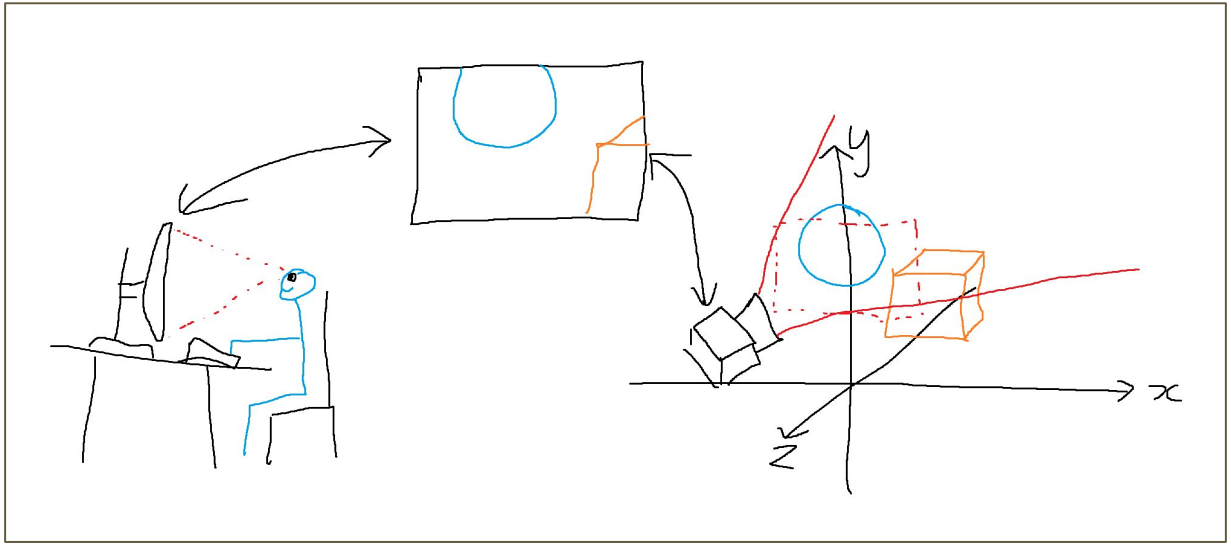


Image credit: Marc Chee

Virtual to Real

Projection to Pixels

- Imagine every pixel being in the virtual world
- Imagine a line drawn from the virtual camera through a pixel
- That line meets an object in the world
- Whatever colour that object is . . .
- Is the colour the pixel ends up

3D Projection

Mapping coordinates

- We're mapping 3D coordinates onto our 2D monitor
- If we have a viewpoint in the same coordinate system (the camera)
- We can tell what that camera should see using maths
- More detail in future lectures!

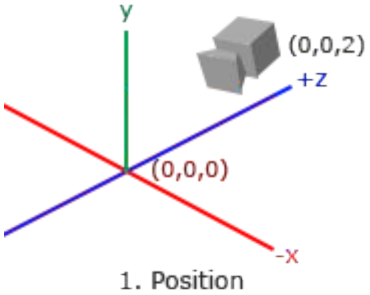


Image credit: learnopengl.com

Break Time

5 minute break

- Graphics can be excessively technical
- It can also involve a LOT of maths
- We're going to use a lot without necessarily knowing it in extreme detail
- But we will still definitely want to understand how things work at a theoretical level

Coding Graphics

What is OpenGL?

The Open Graphics Library

- A big API/Library
- Gives us access to the Graphics Card
- Also provides a lot of functionality so that we can do graphics without low level programming
- Is not a language itself!

Shaders

Code that runs on the GPU

- C++ runs on the CPU (once compiled)
- GLSL (OpenGL Shader Language) runs on the GPU
- Vertex Shader
  - Runs once per vertex
  - Can manipulate vertices (and more later!)
- Fragment Shader
  - Runs once per pixel
  - Can manipulate the colour of the pixel
  - Usually receives information from the vertex shader

C++ Features in this Course

A specific subset of C++ for Graphics Purposes

- A compromise between needing a prior C++ course
- and being able to get involved with Graphics quickly
- Want a primer? We got you fam: https://youtu.be/3DStoqQnUxc
- Want a reference project? https://gitlab.cse.unsw.edu.au/COMP3421/21T3/cpp101

Our Code Setup

C++ with OpenGL

- Cmake project works in different operating systems
- We're supporting Windows, Linux and MacOS
- Not IDE specific, CLion and VSCode recommended
- You don't need to buy a RTX 3090 to learn how to code Graphics!
- Want help getting set up? https://gitlab.cse.unsw.edu.au/COMP3421/21T3/opengl_cmake_setup

What did we learn today?

Our first step into the details

- Graphics Hardware - monitors and graphics cards
- Polygon Rendering - an overview
- Graphics Development in this course
    - OpenGL
    - Shaders
    - C++
    - Cmake