3D Graphics Programming

T163 - Game Programming

Instructors

Alex Richard (Lectures)

- arichard6@georgebrown.ca
- Other contact info on Blackboard

Hooman Salamat (Labs)

- Hooman.Salamat@georgebrown.ca
- Other contact info on Blackboard

Evaluation System

Assessment Tool:	Description:	Outcome(s) assessed:	EES assessed:	Date / Week:	% of Final Grade:
Assignment 1	Practical coding OpenGL exercise	1, 4-8	1-11	3	**10**
Assignment 2	Practical coding OpenGL exercise	1, 4-8	1-11	5	**10**
Assignment 3	Practical coding OpenGL exercise	1, 4-8	1-11	7	**10**
Assignment 4	Practical coding OpenGL exercise	1, 4-8	1-11	11	**10**
Assignment 5	Practical coding OpenGL exercise	1, 4-8	1-11	13	**10**
Midterm Exam	Test on code and theory	2, 3, 6	2, 4-7, 11	7	**30**
Project	Practical coding OpenGL project	1, 4-8	1-11	15	**20**
				TOTAL:	**100%**

Course Outcomes

1. Create various 3D programs using OpenGL
2. Explain the basic concepts of 3D programming
3. Explain the fixed and programmable graphical pipelines
4. Manipulate and animate 3D objects to produce games
5. Apply textures and lighting to 3D objects to produce realistic and/or stylized effects
6. Apply special effects to enhance the visual quality of 3D scenes
7. Use primitive 3D objects as well as complex models to produce games
8. Format all deliverables to comply with Canadian laws and policies

Books 🕮

Interactive Computer Graphics - A top-down approach with shader-based OpenGL - 6th edition

By: Edward Angel & Dave Shreiner

ISBN: 978-0-13-254523-5

OpenGL 4.0 Shading Language Cookbook

By: David Wolff

ISBN: 978-1-849514-76-7

D2L Brightspace

- You will submit all classwork, i.e. labs, assignments, etc. via the related link on D2L
  - Could be in Assignments folder or Week folder for labs

- This course for T163 is delivered on campus
  - Follow the D2L link in the Content page

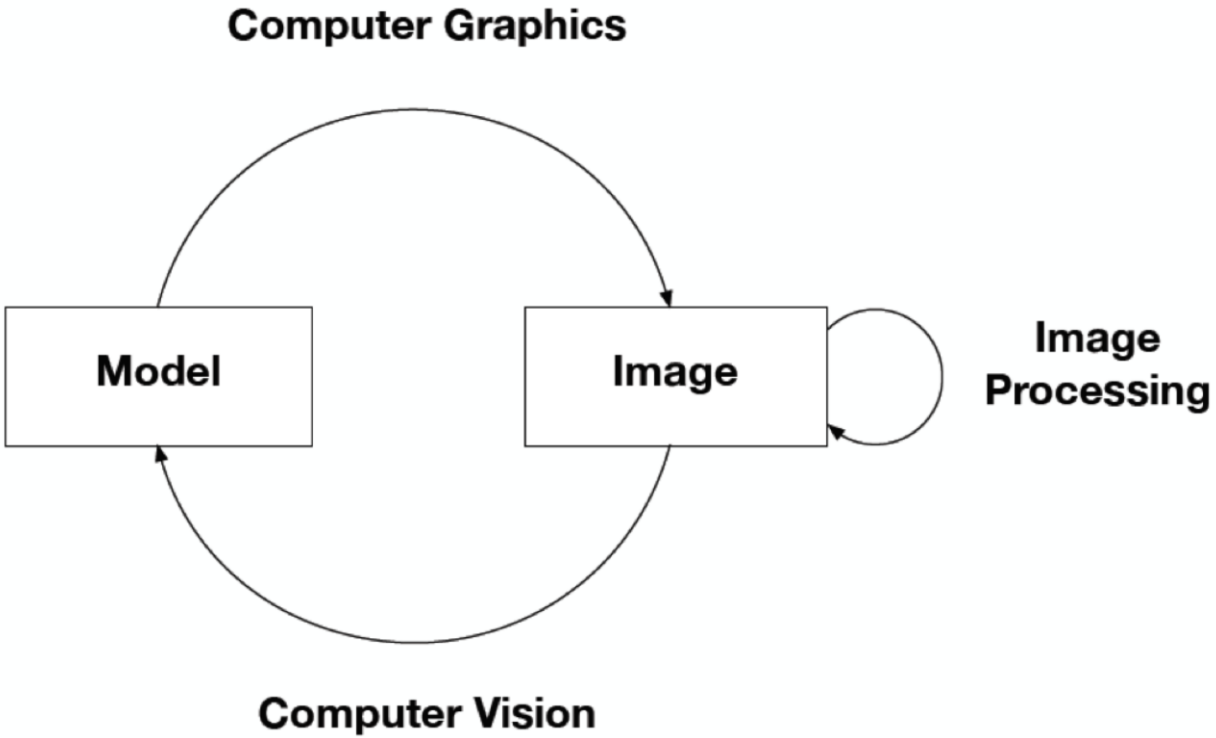
Week 1

Intro to OpenGL & Useful Tools

Framework Options (GLFW/GLUT/SDL)

Math Review

What is Computer Graphics?



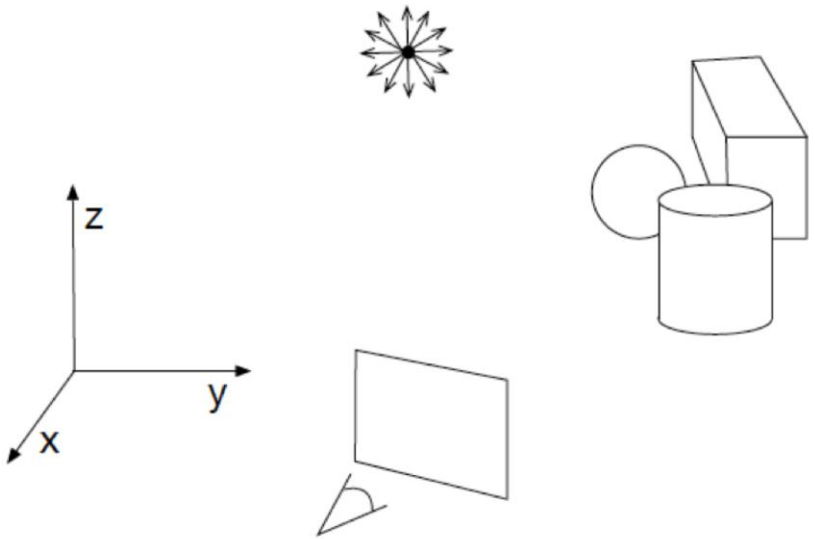
Applications?

- Display of information
- Design
- Games
- Simulation
- Animation
- User Interfaces

Process

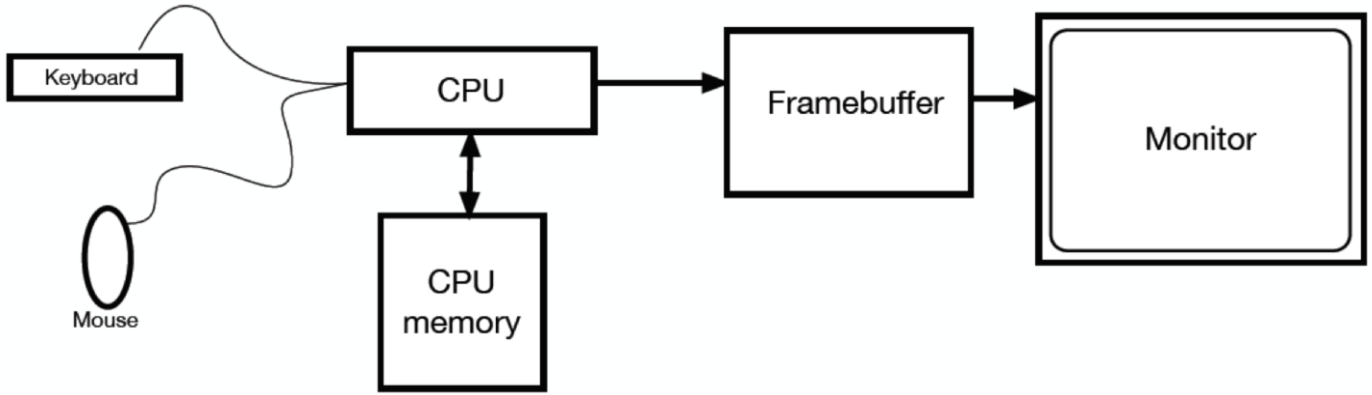
<input>: Given 3D model, material properties, eye, camera, lights

<output>: Generate 2D image



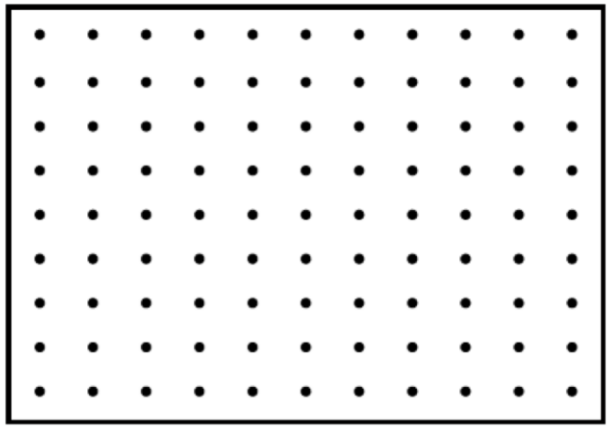
Early Hardware

- CPU does all the work



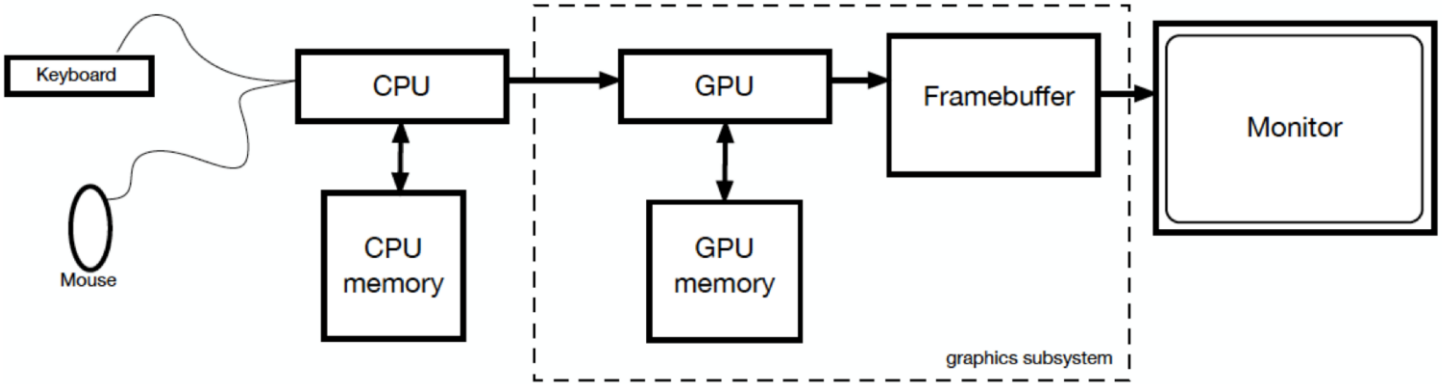
What is a Frame Buffer?

- 2D array of colorful pixels
- Intensity and color (R, G, B, A)
- Number of pixels (resolution)
- Bits per pixel: 1, 8, 24, 36

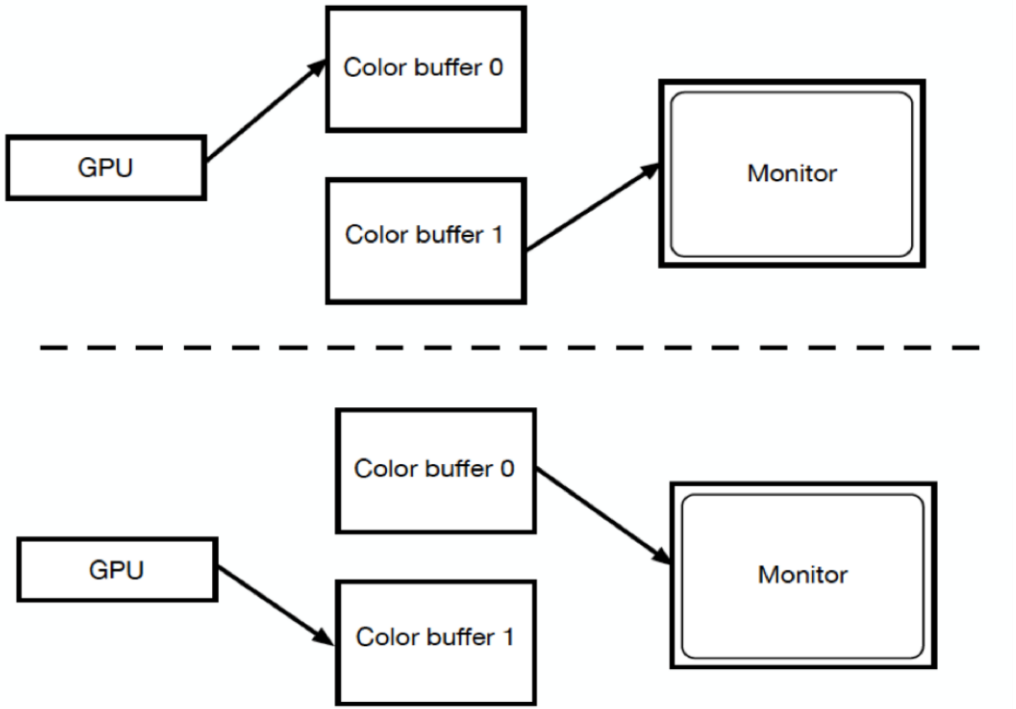


Modern Hardware

- Enter the GPU!



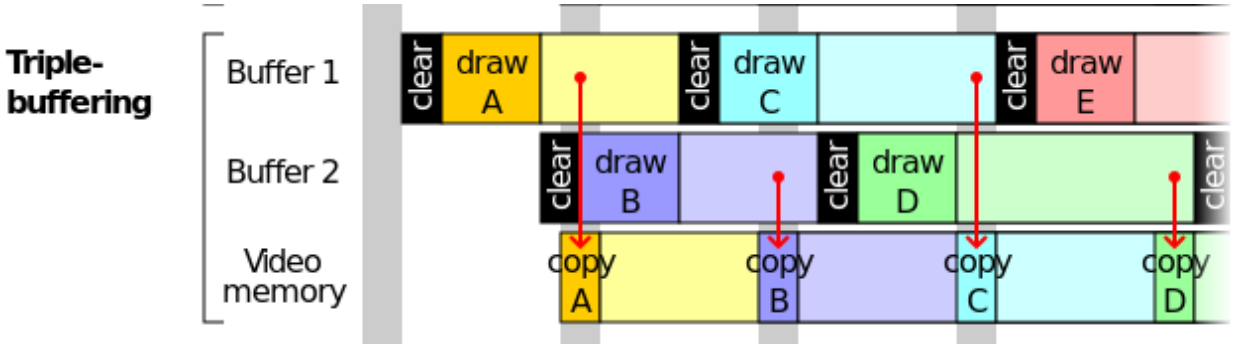
Double Buffering



Triple Buffering

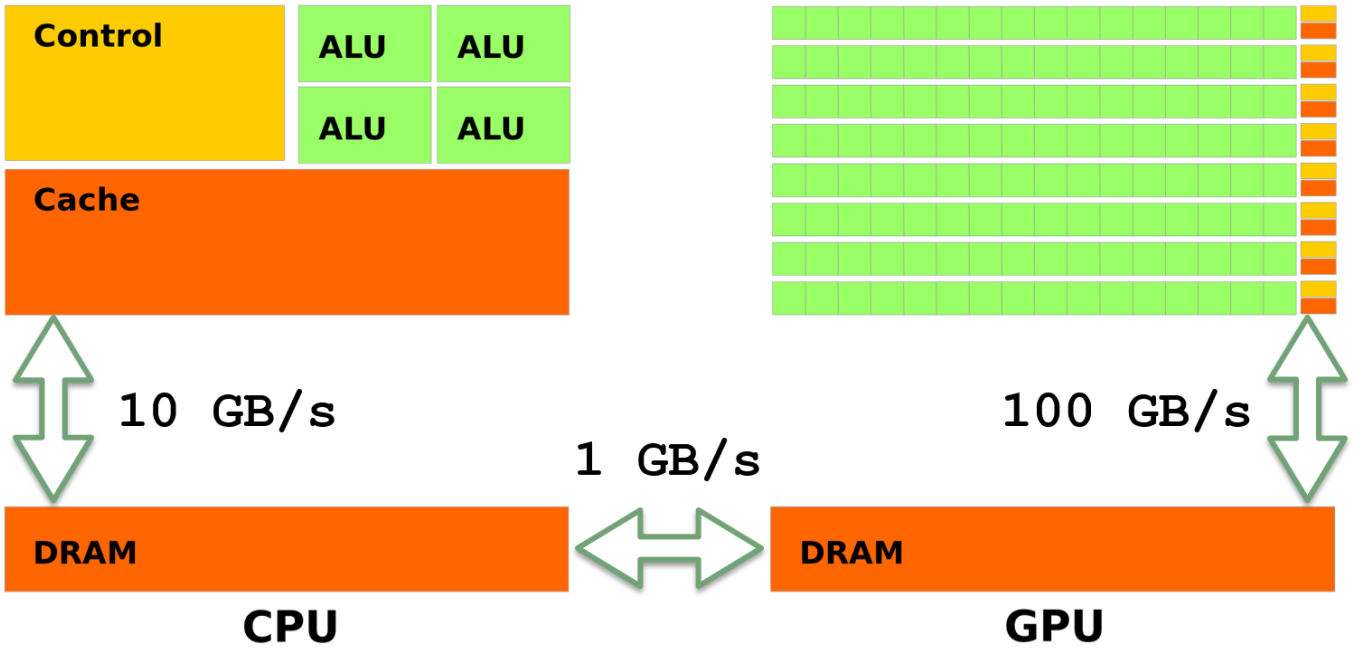
Note: This was a joke from the former instructor who doesn't work here anymore...

Triple Buffering



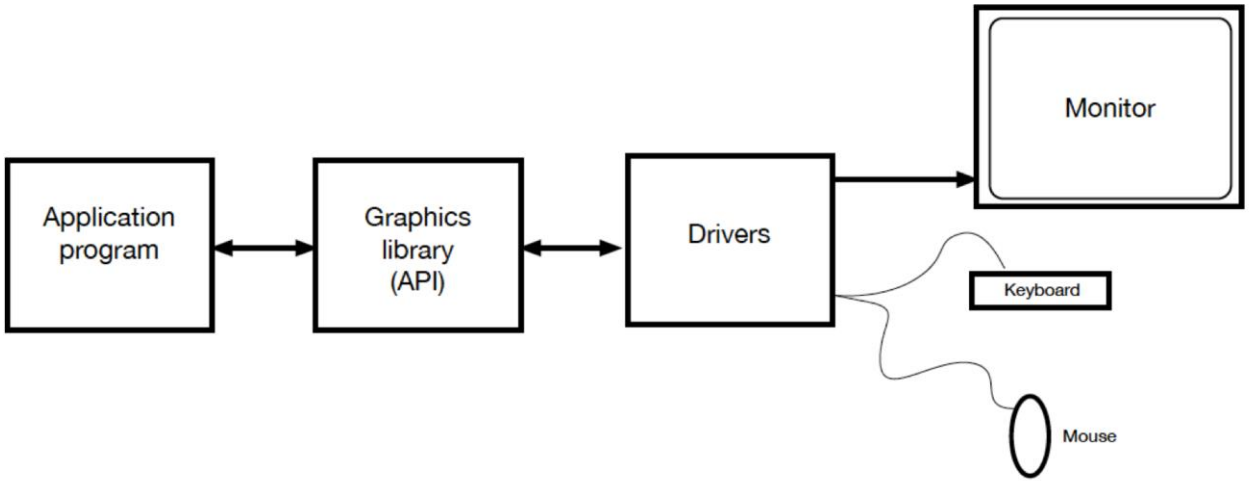
- Higher framerate potential
- Requires extra GPU memory

CPU vs GPU


Control
ALU
ALU
ALU
ALU
Cache
10 GB/s
1 GB/s
100 GB/s
DRAM
DRAM
CPU
GPU

GPU

- Graphical Processing Unit
- API:

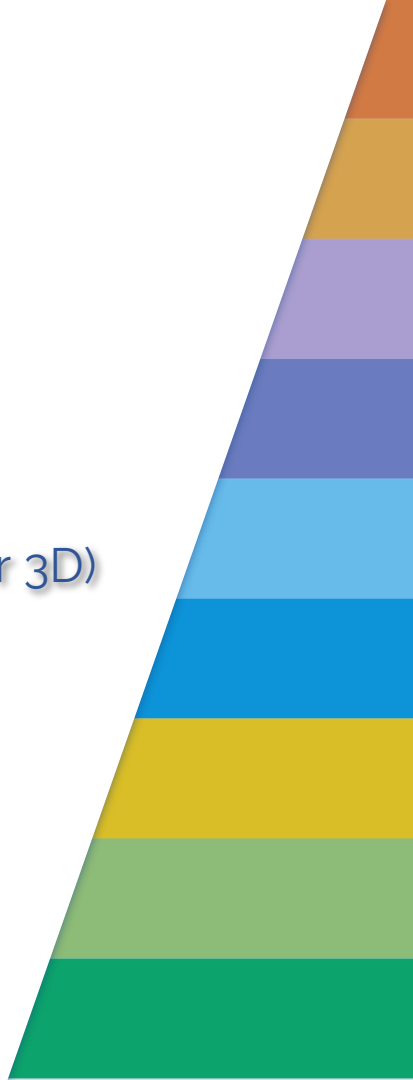


GPU

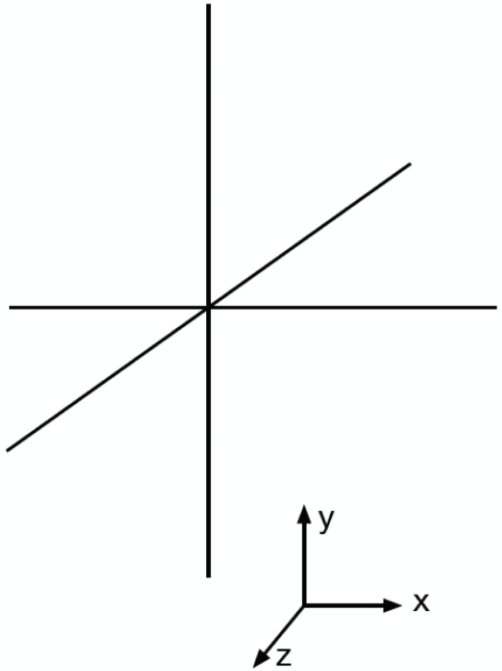
- API Major Tasks
  - Specify objects to be viewed
  - Specify properties of these objects
  - Specify how these objects to be viewed

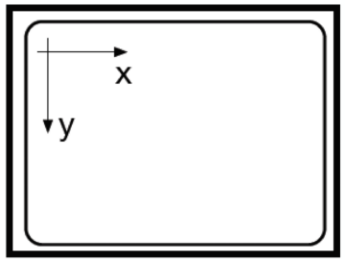
Coordinate Systems

- Model/Object coordinate system
  - Where you define the object
- World coordinate system
  - Where objects are placed relative to each other (2D or 3D)
- Screen coordinate system
  - Device specific coordinates



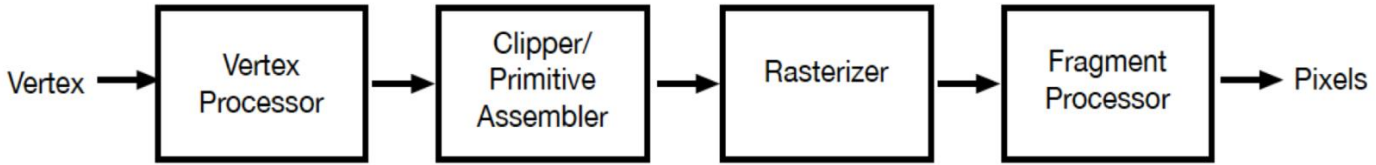
Coordinate Systems





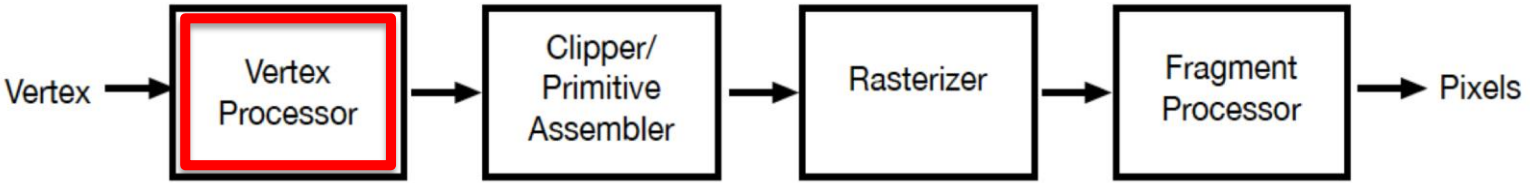
Graphics Pipeline

- World to screen…



- Transform and project graphics primitives onto projection screen
- Determine what's inside (clipping)
- Determine what's visible
- Break down into pixels
- Shade approximately

Graphics Pipeline

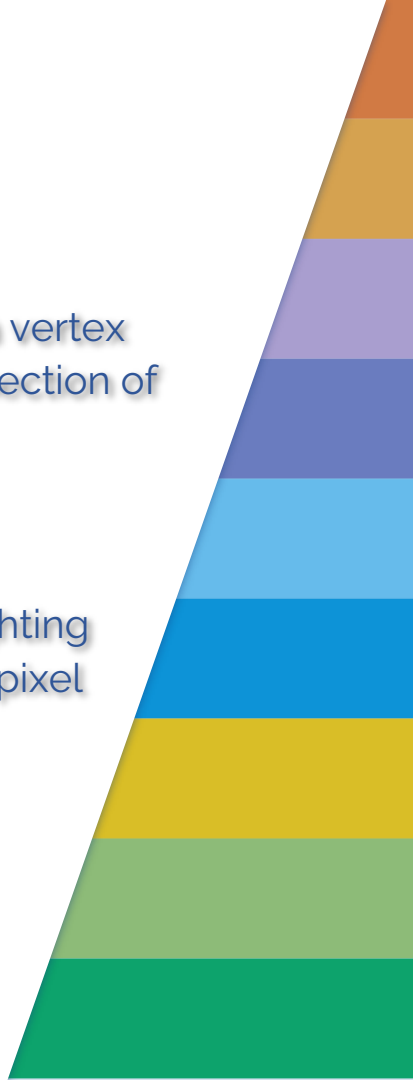


- Vertex Specification
  - VAOs (Vertex Array Objects)
  - VBOs (Vertex Buffer Objects)

- Vertex Shader (programmable)
- Tessellation (programmable)
- Geometry Shader (programmable)

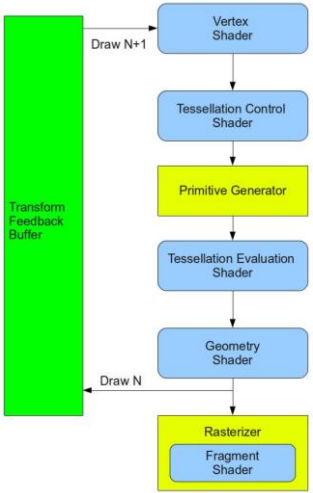
Shaders

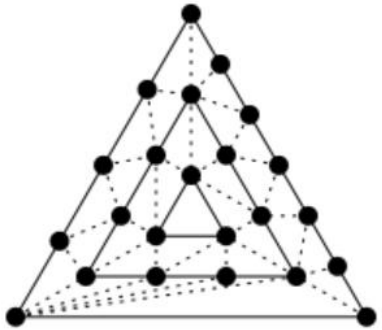
- Vertex Shaders
  - Programs that describe the traits (position, colors, and so on) of a vertex
  - The vertex is a point in 2D/3D space, such as the corner or intersection of a 2D/3D shape
- Fragment Shaders
  - Programs that deal with the per-fragment processing such as lighting
  - The fragment is a WebGL term that you can think of as a kind of pixel and contains color, depth value, texture coordinates, and more
- All shaders run on GPU!



Shaders, cont'd

- Tessellation Shaders
  - Occurs in the Vertex Processing stage in the OpenGL rendering pipeline
  - Patches of vertex data are subdivided into smaller primitives
  - https://www.ogldev.org/www/tutorial30/tutorial30.html

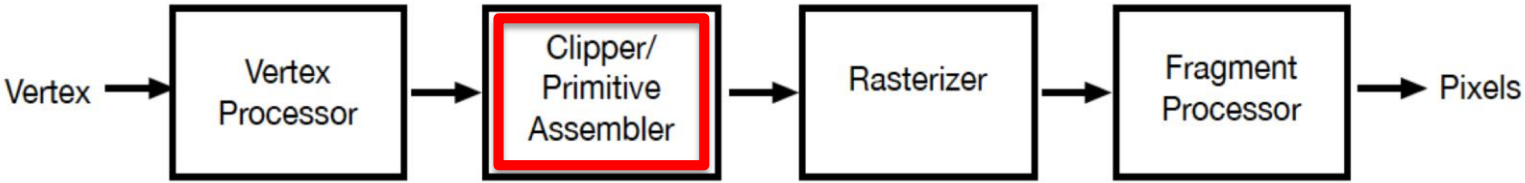




Shaders, cont'd

- Geometry Shaders
  - Takes as input a set of vertices that form a single primitive
  - Transform these vertices before sending them to the next shader stage
  - Converts the original set of vertices to completely different primitives
    - Can even add additional vertices
  - https://learnopengl.com/Advanced-OpenGL/Geometry-Shader

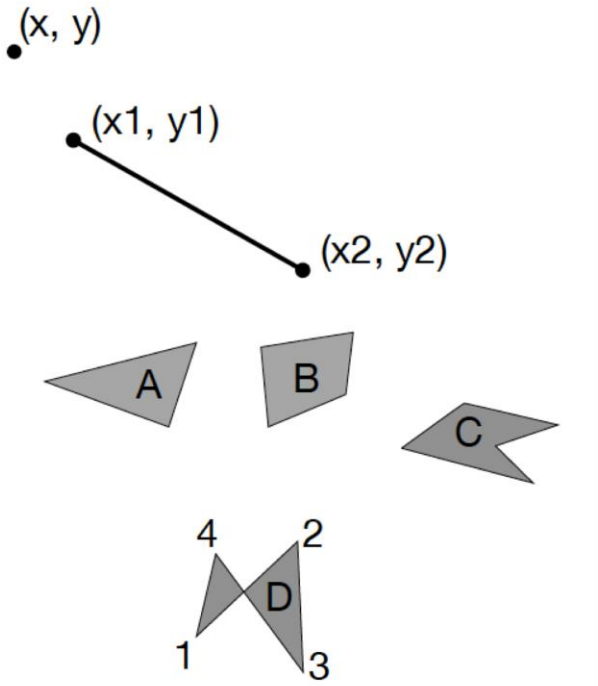
Graphics Pipeline



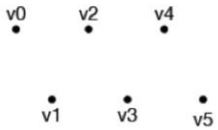
- Vertex Post-Processing
- Primitive Assembly
  - Vertices converted in to a series of primitives
  - Remove primitives that can't be seen (culling)

Primitives

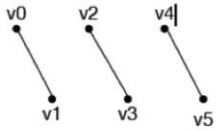
- Points
- Lines
- Triangles
- Quads
- Polygons
- Curves
- Surfaces



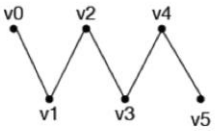
OpenGL Primitives



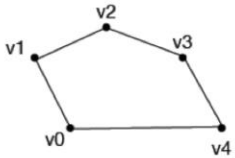
gl.POINTS



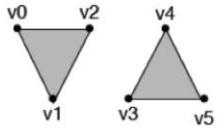
gl.LINES



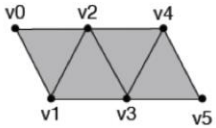
gl.LINE_STRIP



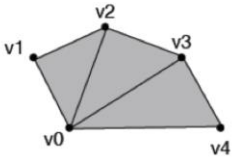
gl.LINE_LOOP



gl.TRIANGLES

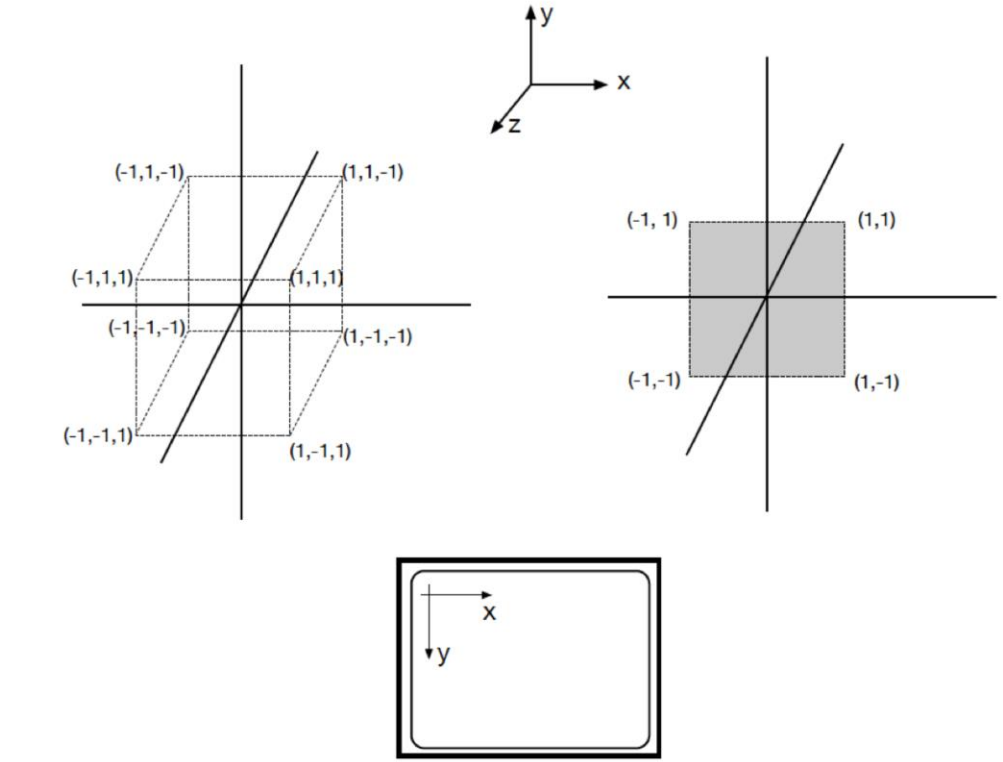


gl.TRIANGLE_STRIP



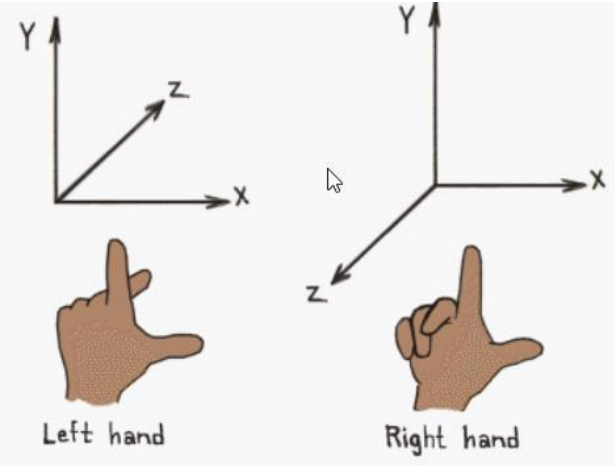
gl.TRIANGLE_FAN

OpenGL Defaults



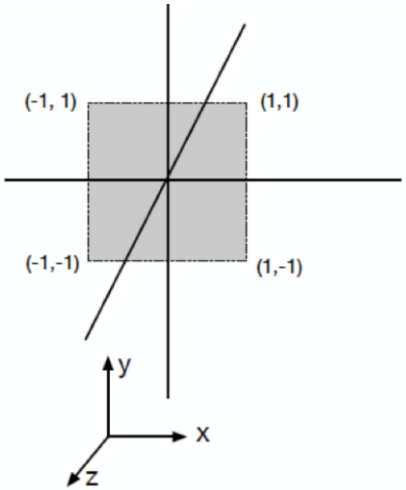
OpenGL Defaults

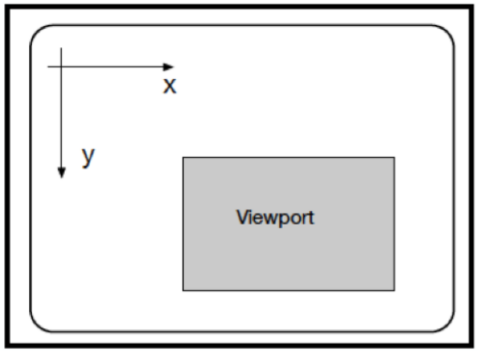
- So are OpenGL coordinates left-handed or right-handed?
- What does that even mean?



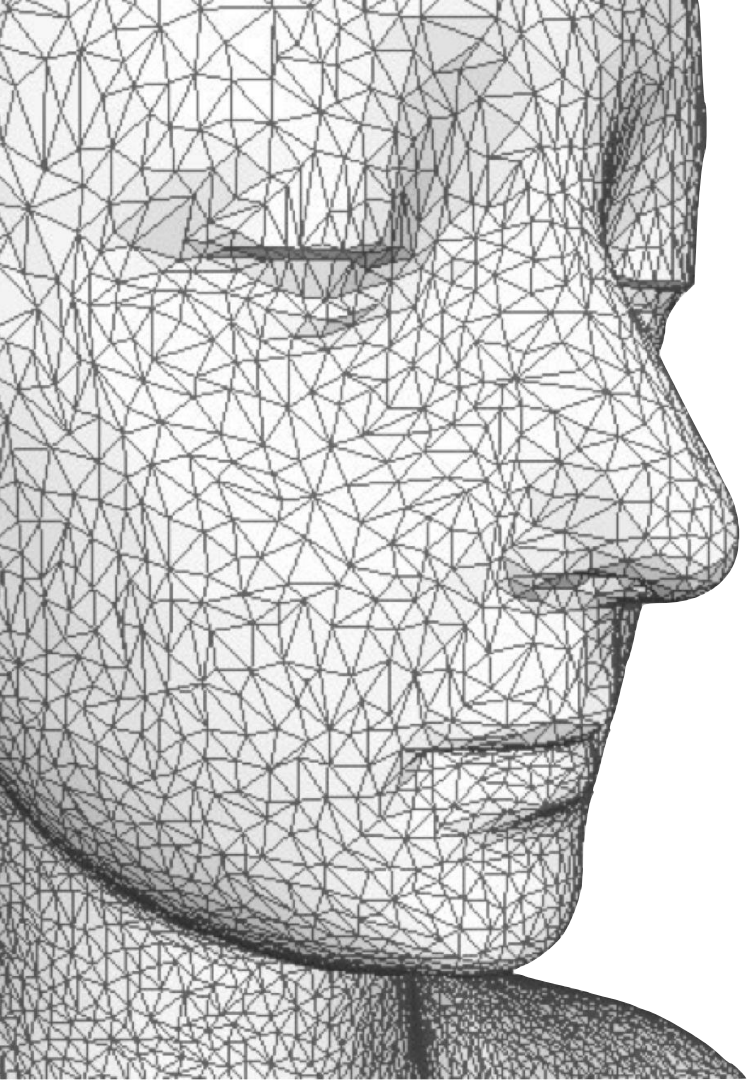
Viewport

- Specifies where on the screen the window will appear





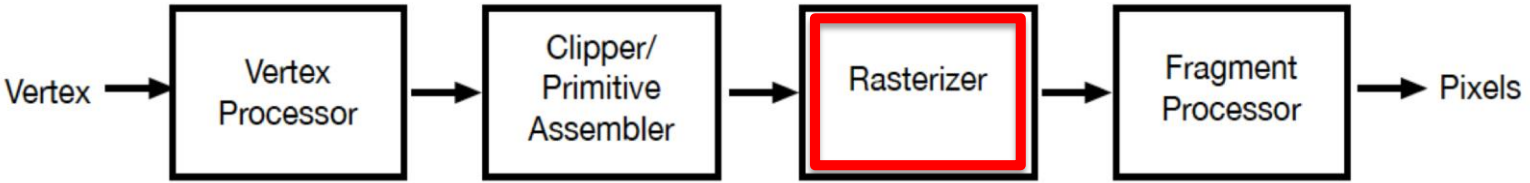

```
gl.viewport(x, y, width, height)
```



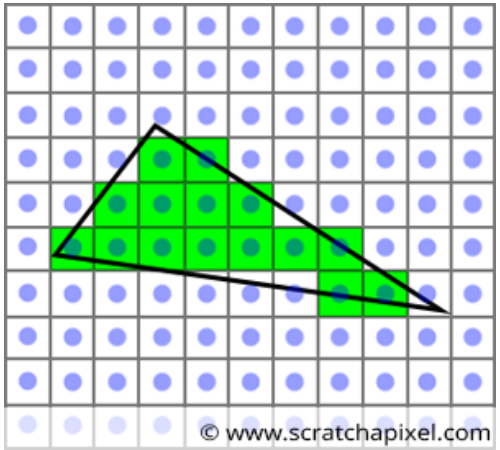
Polygonal Meshes

- - List of vertices
- - List of polygons
  - Each polygon has list of vertices
  - V: 1, 2, 3, 4, 5
  - P: A(1,2,5), B(2,3,5), C(3,4,5)

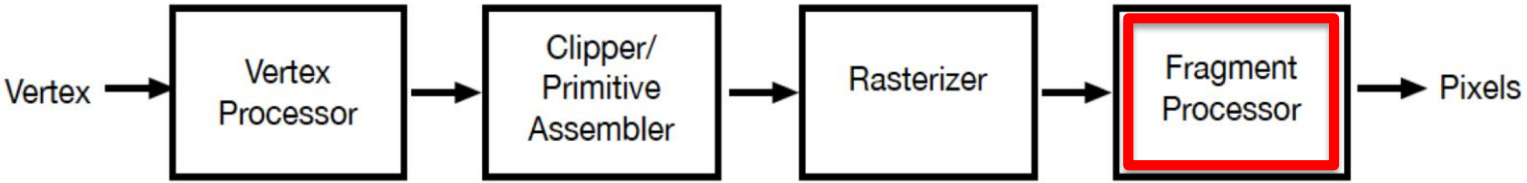
Graphics Pipeline



- Rasterization
  - Converts primitives to fragments (Pixel data)



Graphics Pipeline



- Fragment Shader (programmable)
- Per-Sample Operations
  - Depth test, color blending
  - Swap the buffers and done!

Graphics Pipeline

- The normal graphics pipeline has limitations, however
  - So We will look at some advanced rendering techniques much later

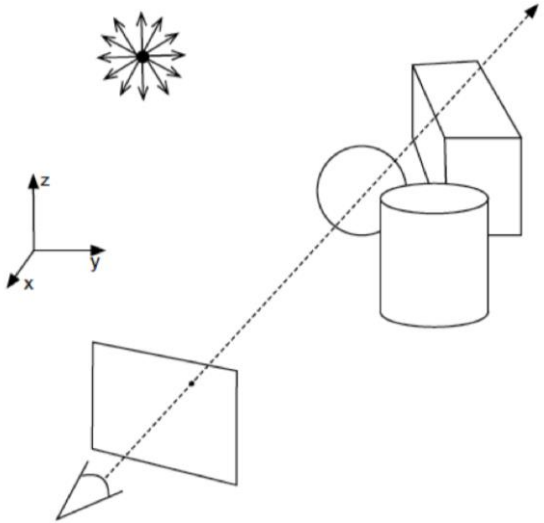
- What are some of the most impressive graphics and/or graphical details you've seen in games?

Ray Tracer

Radical obscure reference!

Ray Tracing (Screen to World)

- Shoot ray from eye through screen into world
- Intersect objects with ray
- Find closest intersection
- Do shading/lighting calculation



Week 1

Lecture Example

Week 1

Lab Activities

Week 1 Lab

- For the lab, see Hooman's material
- OpenGL examples covered:
  - GLUT and GLFW basics
  - Fixed and programmable pipeline examples
  - Rendering different shapes
  - Using vertex and fragment shaders

Week 1

End