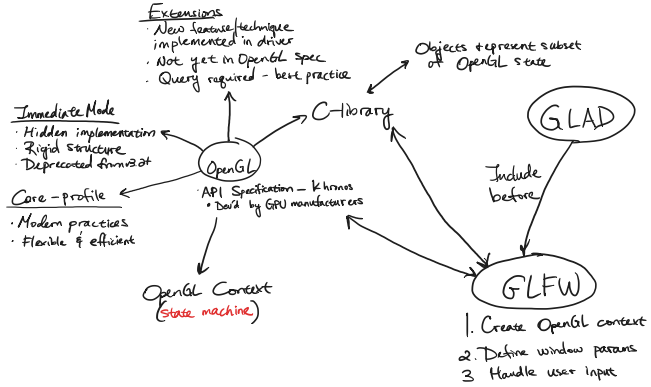
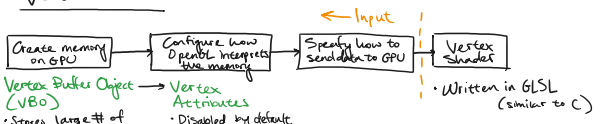
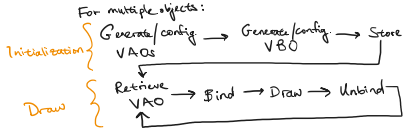
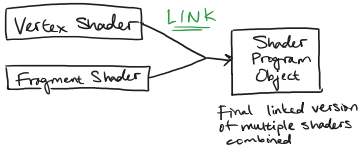


Vertex Shader



Vertex Array Object (VAO)
- Stores state by storing vertex attributes calls
  - Only have to call vertex attributes once
- Core OpenGL requires VAOs





Final linked version of multiple shaders combined

Fragment Shader

- Calculates color output of pixels (RGBA)

Graphics Pipeline: 3D coordinates → 2D pixels

※ 1. Vertex Shader:
- 3D coors. → 3D coors.
- Basic processing on vertex attributes

2. Primitive Assembly:
- Assembles all point(s) into primitive shape (eg. triangle)

3. Geometry Shader:
- Generates other primitives using vertices

4. Rasterization Stage:
- Maps primitive(s) to pixels on screen
- Clipping: Discard all out-of-view fragments

※ 5. Fragment Shader:
- Calculates the final color of a pixel

6. Alpha Test & Blending:
- Checks depth of fragment and discards accordingly
- Blends according to alpha (opacity) values

※ Most often working w/ these stages
No defaults for these!!!
Must define!

Shaders: Small programs implementing steps of graphics pipeline run on GPU processing cores

GLSL: OpenGL Shading Language

Vertex: Collection of data per 3D coordinate
↳ Vertex Attributes: Representation of vertex data

Primitives: Hints passed to OpenGL about render types

Fragments: All data required to render a single pixel

Normalized Device Coordinates: 3D coordinates b/w -1.0 & 1.0 on all 3 axes (x, y, z)
↳ NDC → Viewport Transform → Screen-space coordinates → Fragment Shader

Element Buffer Objects (EBO): Buffer (like VBO) that stores indices OpenGL uses to decide what vertices to draw
↳ Indexed Drawing
- VAO also keeps track of EBO bindings