ICCS-404
COMPUTER GRAPHICS AND ~~AUGMENTED~~ REALITY

EXTENDED

Pisut Wisessing, PhD

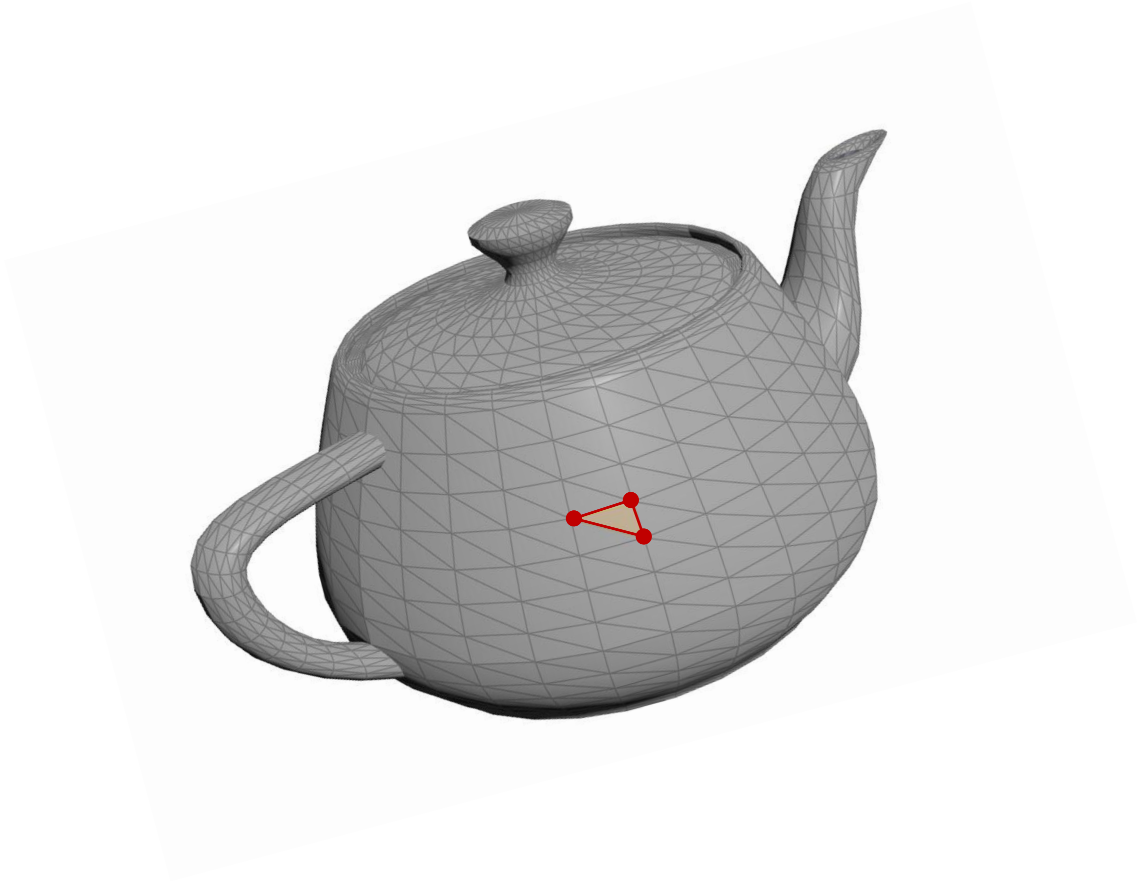
Assistant Professor

CMKL University

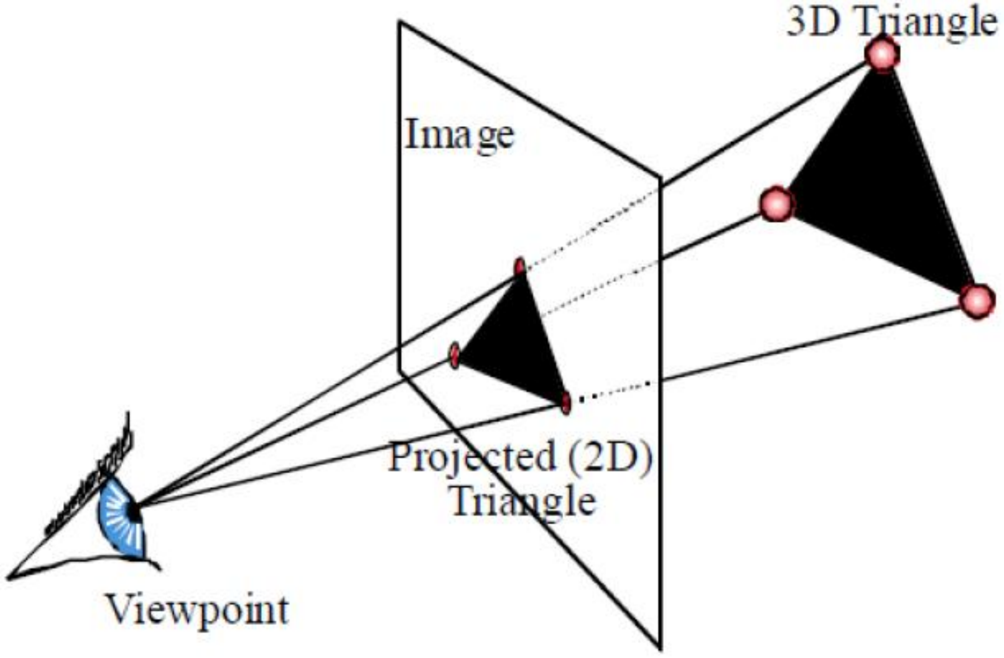
WebGL

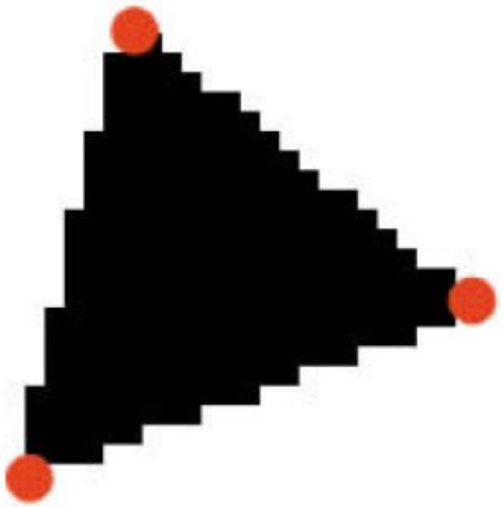
INTERACTIVE CG PIPELINE

- How will OpenGL render this teapot (or just a single triangle, step-by-step?

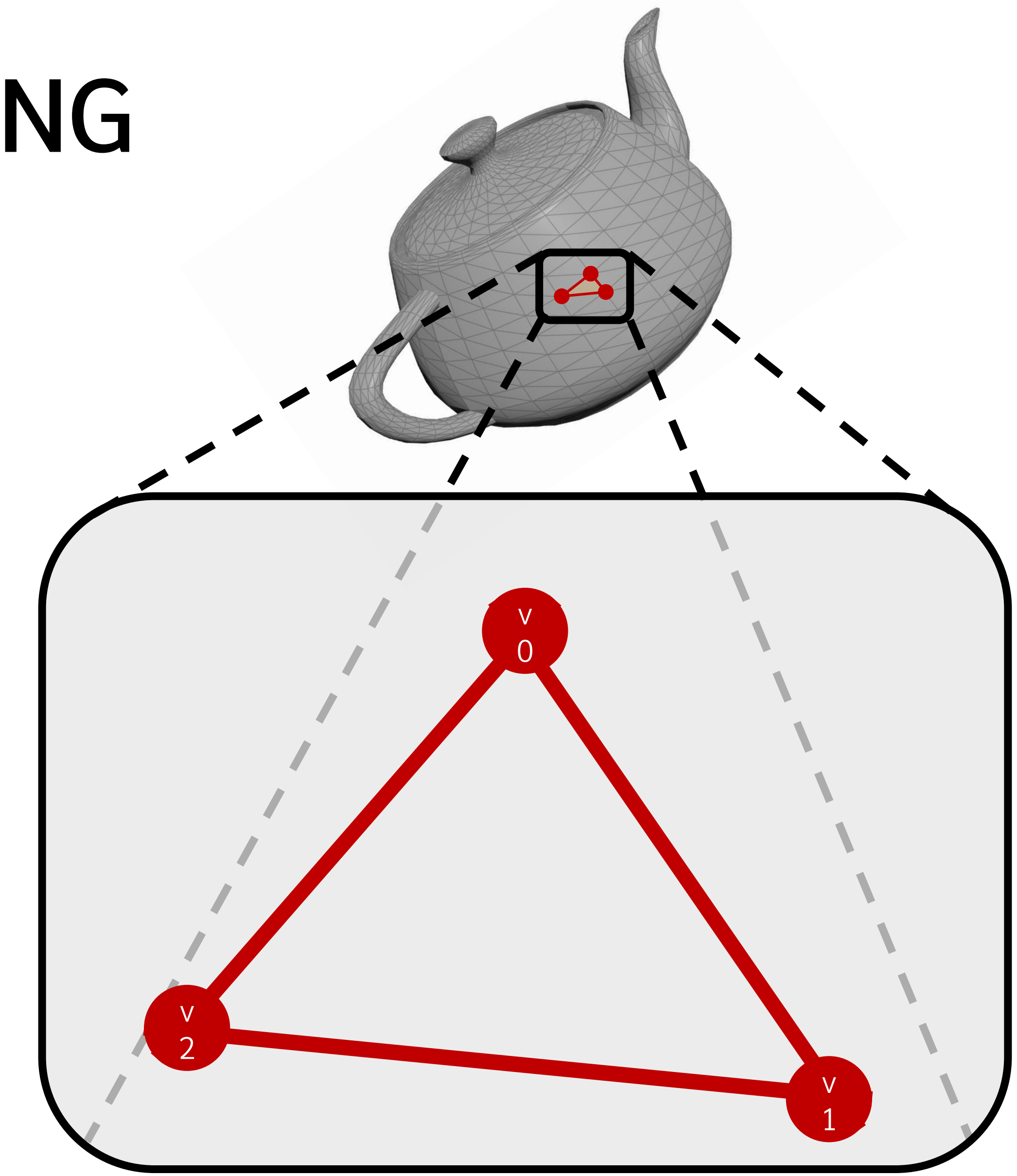


RASTERIZER

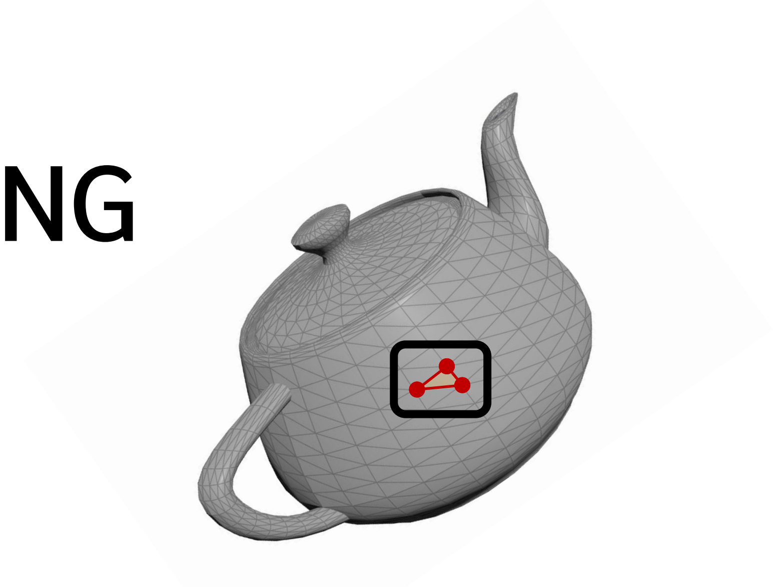


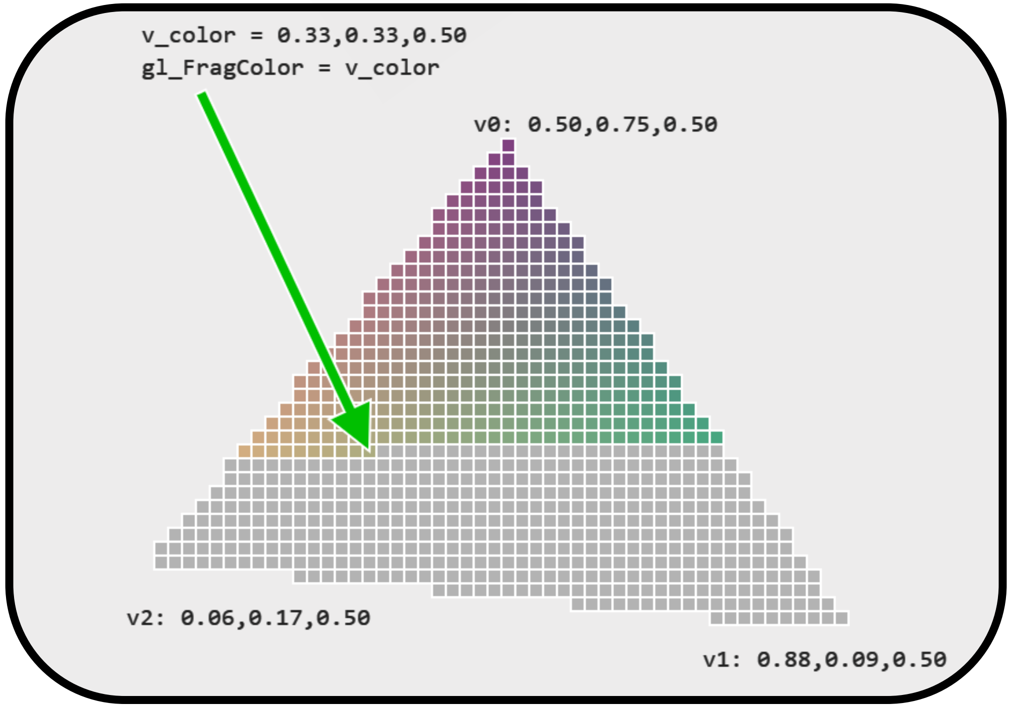


VERTEX PROCESSING



FRAGMENT PROCESSING





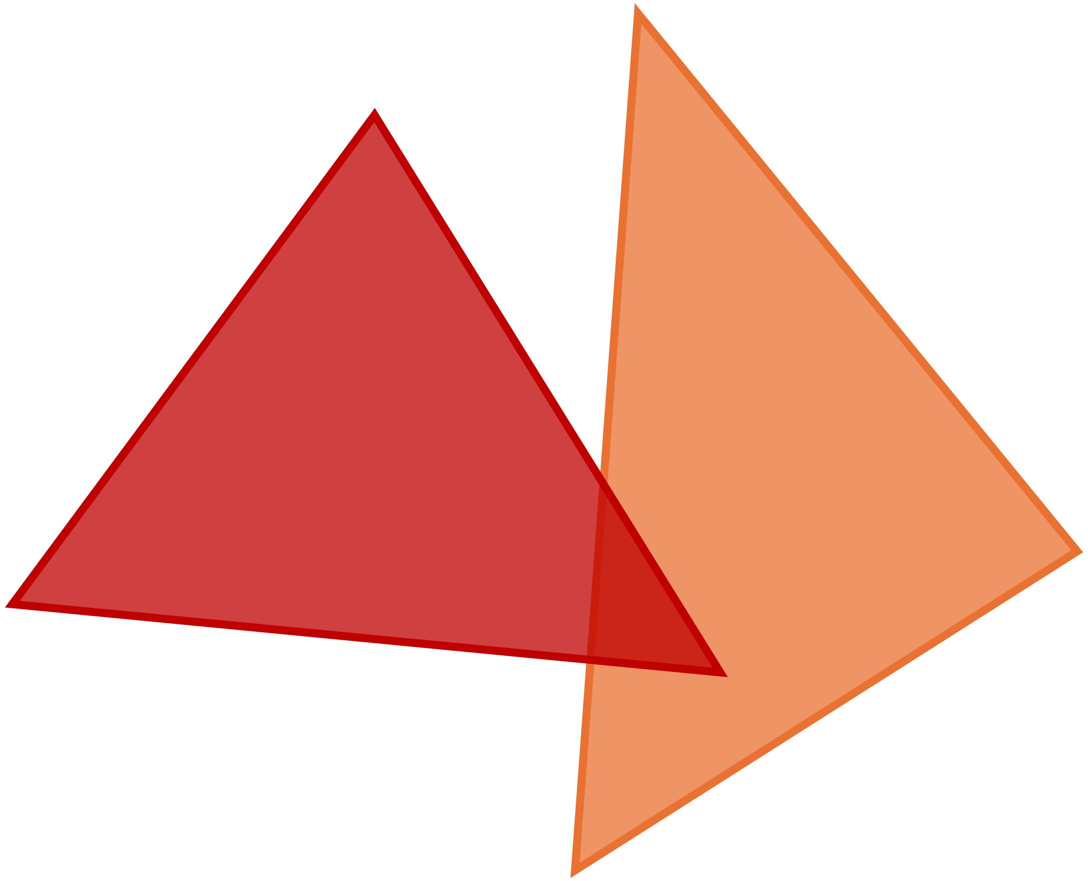
```
v_color = 0.33,0.33,0.50
gl_FragColor = v_color
```

v0: 0.50,0.75,0.50

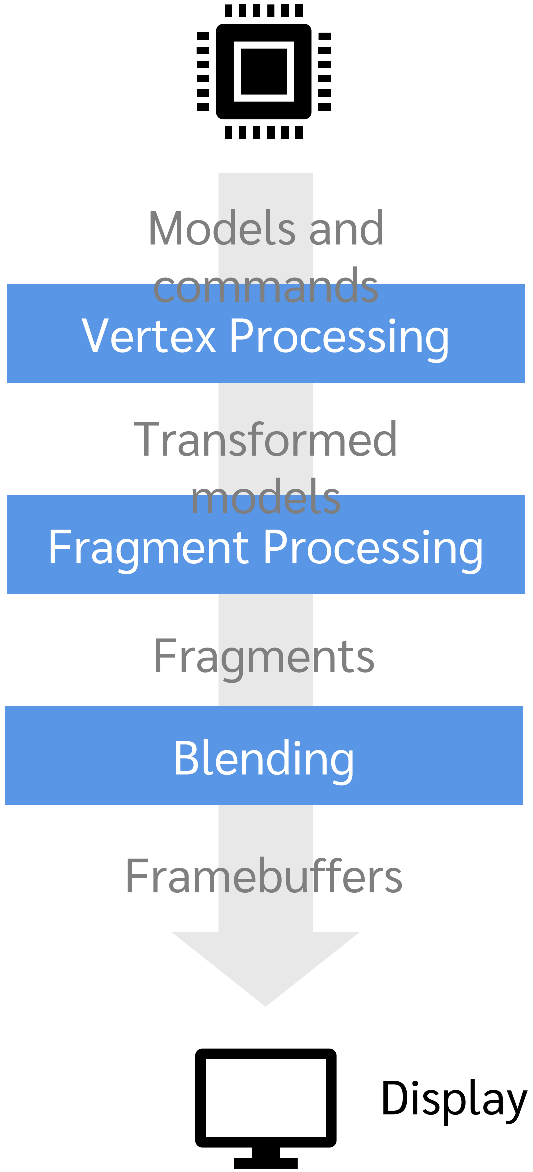
v2: 0.06,0.17,0.50

v1: 0.88,0.09,0.50

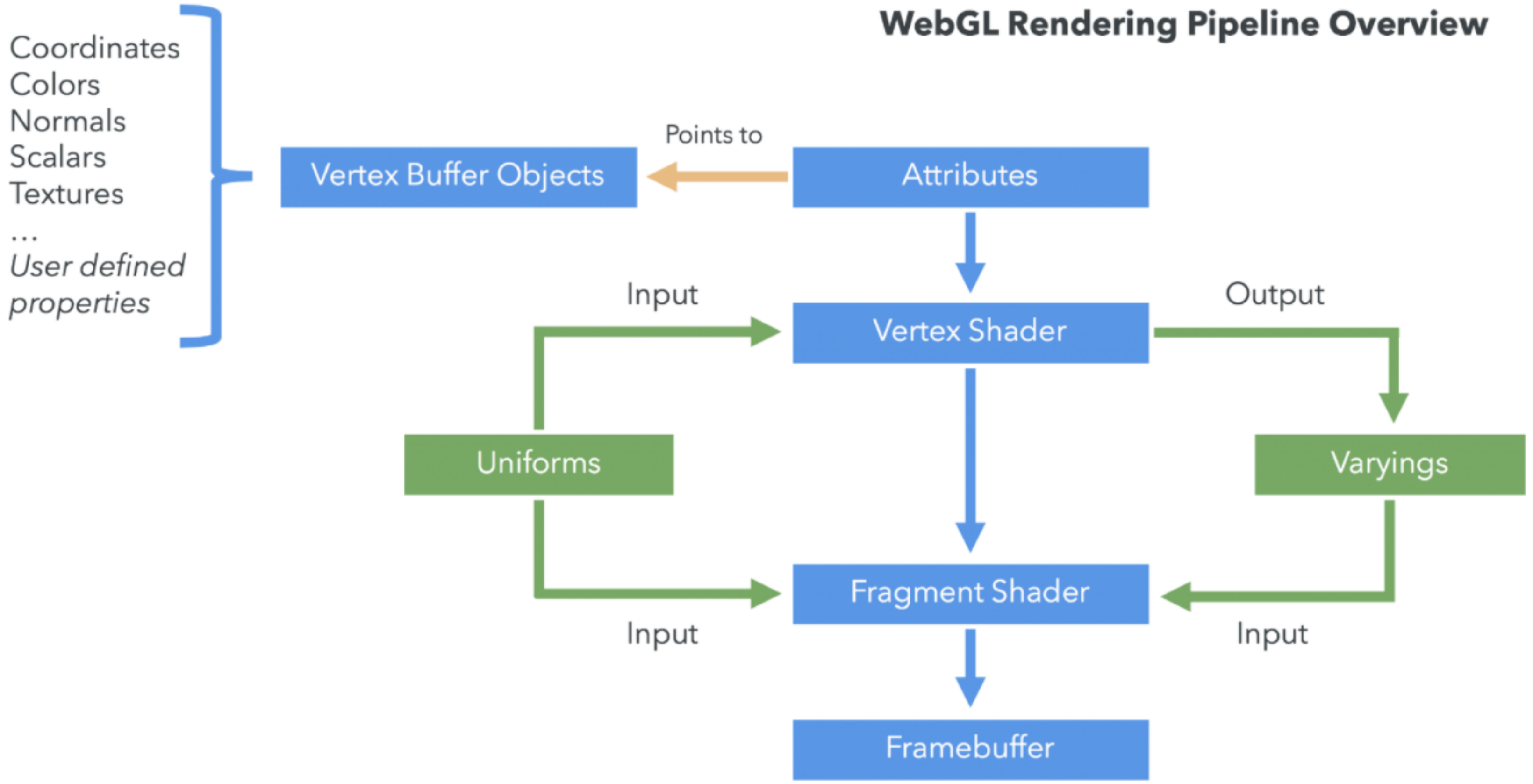
BLENDING



CG PIPELINE


Models and commands
Vertex Processing
Transformed models
Fragment Processing
Fragments
Blending
Framebuffers
Display

WEBGL PIPELINE



WebGL Rendering Pipeline Overview

Coordinates
Colors
Normals
Scalars
Textures
...
User defined properties

Vertex Buffer Objects

Points to

Attributes

Input

Vertex Shader

Output

Uniforms

Varyings

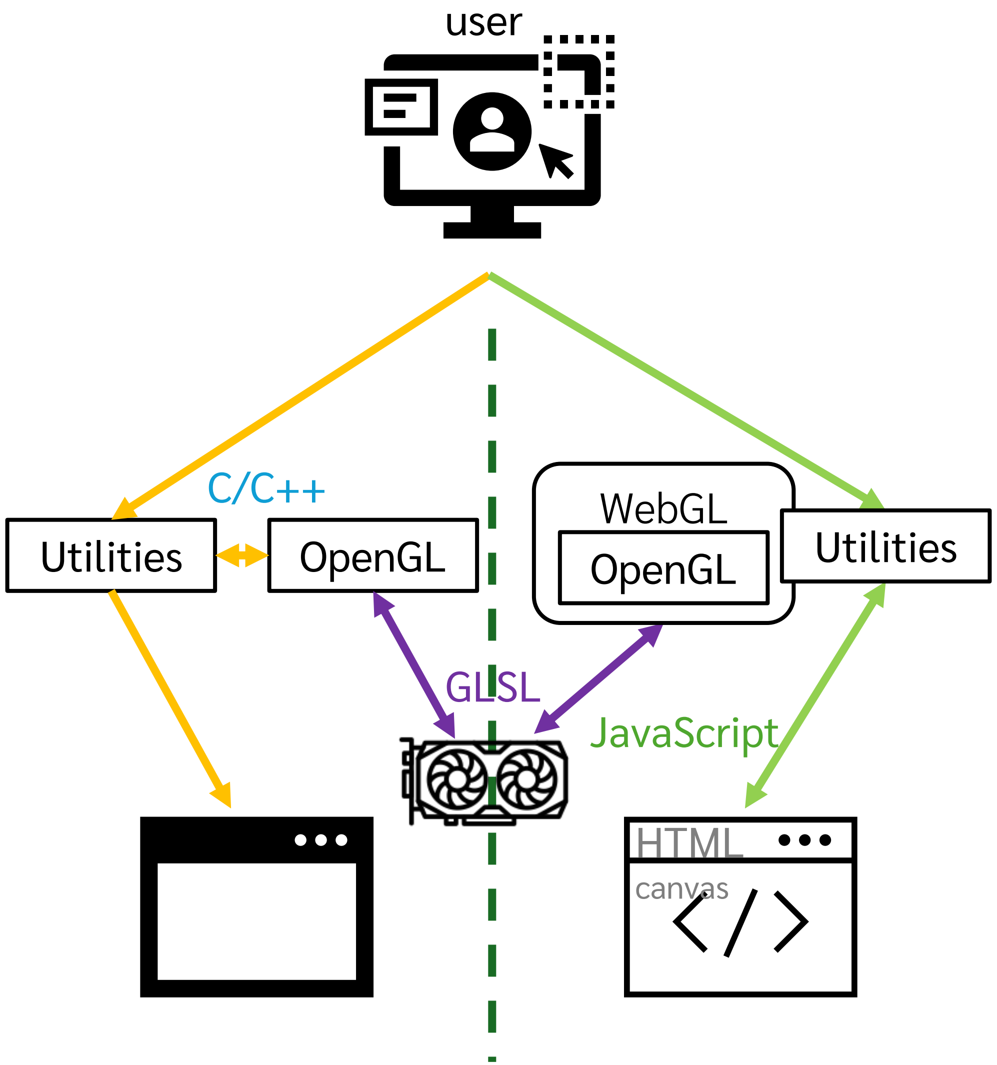
Input

Fragment Shader

Input

Framebuffer

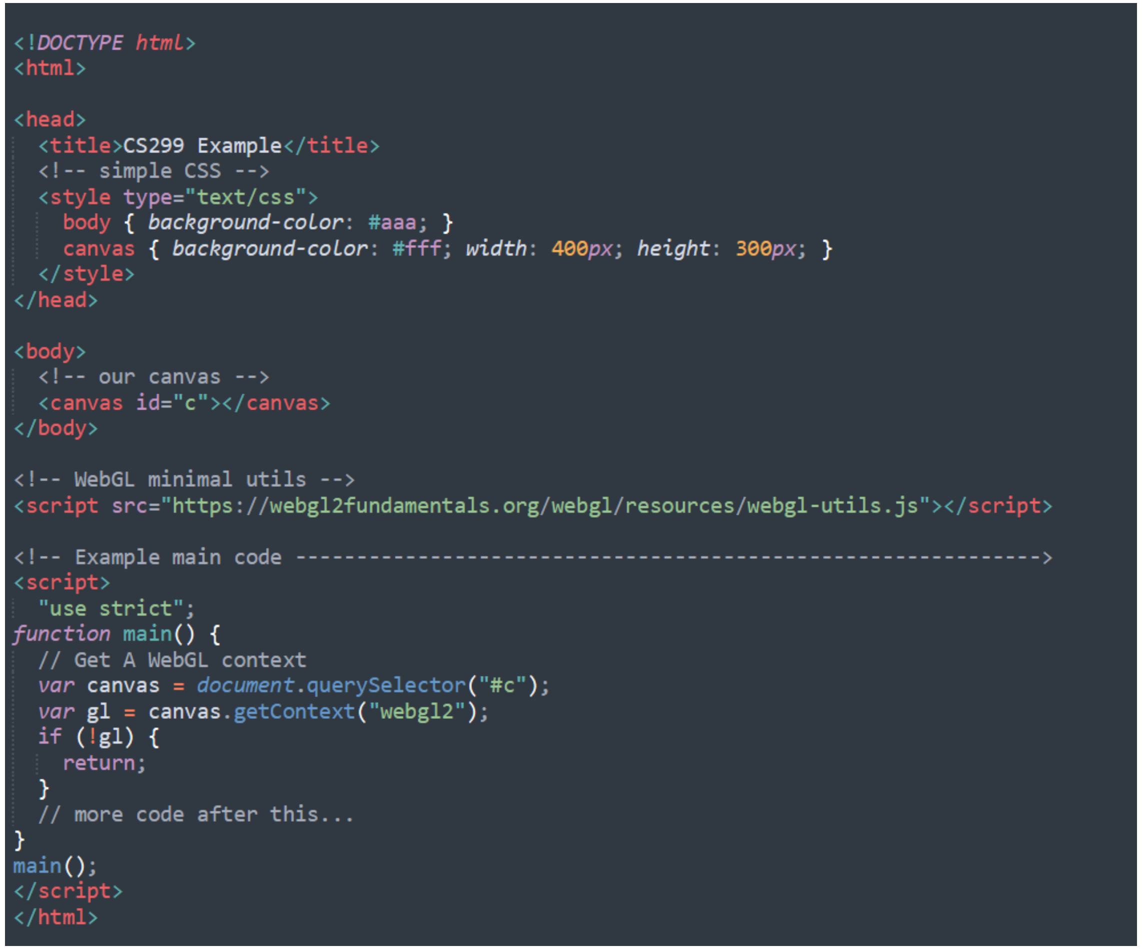
OPENGL VS WEBGL



HTML, CSS, JAVASCRIPT

DOM (Document Object Model) – API for HTML/XML

- **HTML** (Hyper Text Markup Language) – content (static)
- **CSS** (Cascading Style Sheets) – presentation
- **JavaScript** – dynamic content, functions or actions



```
<!DOCTYPE html>
<html>

<head>
  <title>CS299 Example</title>
  <!-- simple CSS -->
  <style type="text/css">
    body { background-color: #aaa; }
    canvas { background-color: #fff; width: 400px; height: 300px; }
  </style>
</head>

<body>
  <!-- our canvas -->
  <canvas id="c"></canvas>
</body>

<!-- WebGL minimal utils -->
<script src="https://webgl2fundamentals.org/webgl/resources/webgl-utils.js"></script>

<!-- Example main code ------------------------------------------------------------>
<script>
  "use strict";
function main() {
  // Get A WebGL context
  var canvas = document.querySelector("#c");
  var gl = canvas.getContext("webgl2");
  if (!gl) {
    return;
  }
  // more code after this...
}
main();
</script>
</html>
```

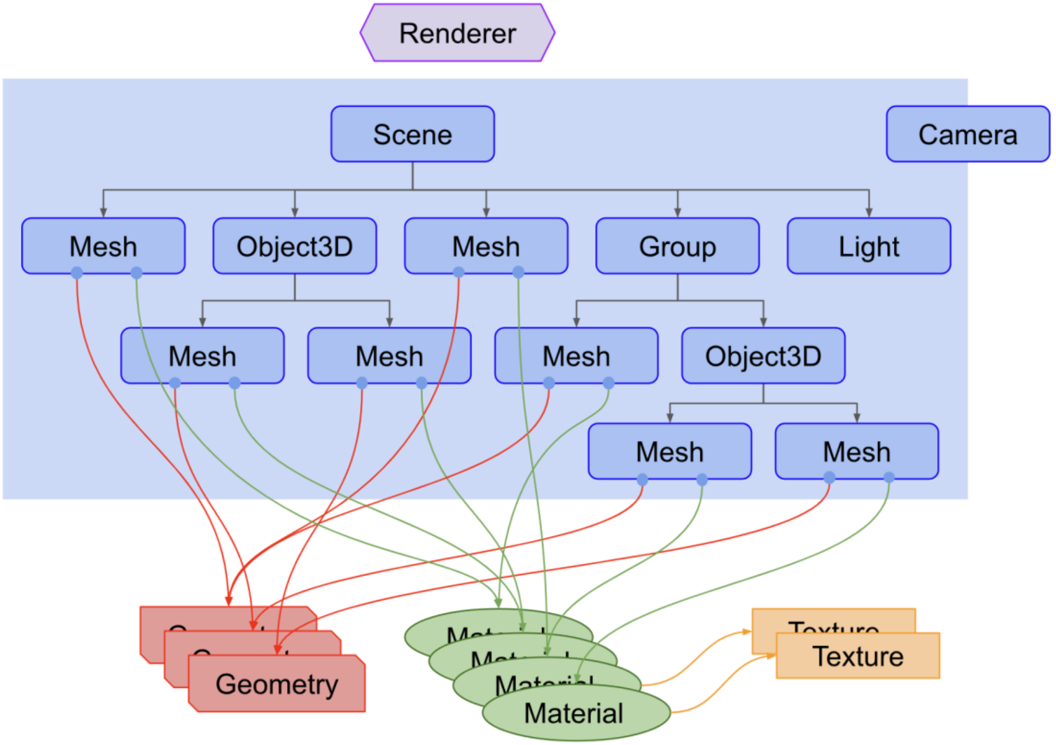
EXERCISE

Drawing a simple triangle with WebGL

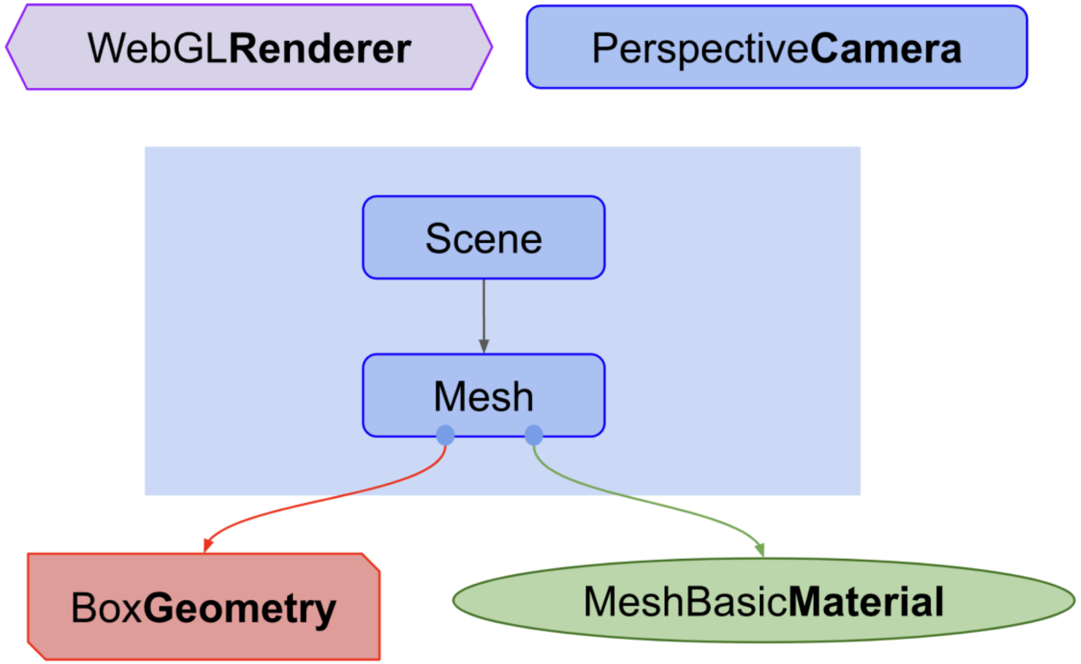
Three.js

THREE.JS FUNDAMENTALS

- Renderer
- Camera
- Scene



HELLO CUBE!



```
import * as THREE from 'https://threejsfundamentals.org/threejs/
resources/threejs/r132/build/three.module.js';

function main() {
  const canvas = document.querySelector(‘#c');
  const renderer = new THREE.WebGLRenderer({canvas});
  // camera inputs: fov, aspect, near, far
  const camera = new THREE.PerspectiveCamera(75, 2, 0.1, 5);
  camera.position.z = 2;
  const scene = new THREE.Scene();
  // box inputs: width, height, depth
  const geometry = new THREE.BoxGeometry(1, 1, 1);
  // greenish blue
  const material = new THREE.MeshBasicMaterial({color: 0x44aa88})
;
  const cube = new THREE.Mesh(geometry, material);
  scene.add(cube);
  renderer.render(scene, camera);
}
main();
```

EXERCISE

Drawing a simple cube with Three.js

BLENDER

Please watch

Imphenzia's Learn Low Poly Modeling in Blender 2.9 / 2.8

https://www.youtube.com/watch?v=1jHUY3qoBu8