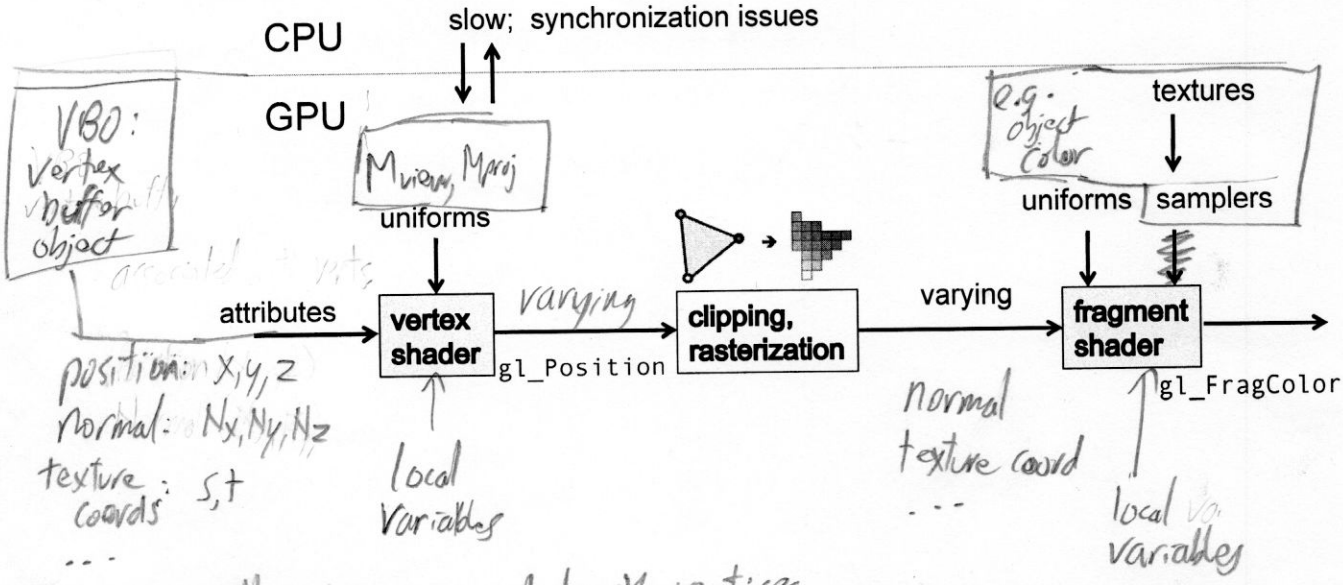
(live coding done in class)

Shader Overview



attributes: associated with vertices
uniforms: global variables that can be changed by user for each rendering call, e.g., each rendered object.
varying: variables to be interpolated across rendered triangles.

Example Vertex Shader

optional naming convention

```
attribute vec4 a_Position;
attribute vec4 a_Normal;
attribute vec2 a_TexCoord;

uniform mat4 u_ModelViewMatrix;
uniform mat4 u_ProjectionMatrix;
uniform float u_DistortionTime;

varying vec4 v_ViewPosition;
varying vec4 v_ViewNormal;
varying vec2 v_TexCoord;

void main() {
  vec4 view_pos = u_ModelViewMatrix * a_Position;
  vec4 proj_pos = u_ProjectionMatrix * view_pos;
  gl_Position = proj_pos;     // final assigned vertex position (in CCS)
    // variable attributes, interpolated across triangle, used by fragment shader
  v_ViewPosition = vec4(view_pos.xyz, 1);    // want interpolated VCS coords
  v_TexCoord = a_TexCoord;                   // want interpolated tex coords
  v_ViewNormal = a_Normal;                   // want interpolated normal
}
```

Vertex attributes, stored in VBO:

global variables, can be changed by user

to be interpolated across triangle

transform vertex.

setup for interpolation

VBO

P_1	N_1	T_1
P_2	N_2	T_2
⋮		

Note: should really also be transforming vertex normals before interpolation.

Example Fragment Shader

```
uniform vec4 u_FragColor;
uniform sampler2D u_AlbedoTex;

varying vec4 v_ViewPosition;
varying vec4 v_ViewNormal;
varying vec2 v_TexCoord;

void main() {

    vec4 texColour = texture2D(u_AlbedoTex, v_TexCoord);
    gl_FragColor = texColour;
}
```

} global variables, can be changed by user.

} interpolated results.

→ s,t texture coordinates

→ texture sampling method. + associated texture map

→ final output of every fragment shader.

Programmable Pipeline