CMSC427
OpenGL and JOGL

Step 1: Configuring and compiling

- In Eclipse following previous instructions
- Get from web page CMSC427OpenGLCode.zip
- Add graphics3Dlib.jar to JOGL project
- If on Mac, follow errata

- From command line
- Add jar files to CLASSPATH
- Use .bat file cmds
- javac code/Code.java
- Java code.Code

Step 2: Creating a GLCanvas (Gordon 2.1)

```
public class Code extends JFrame implements GLEventListener
{       private GLCanvas myCanvas;

        public Code()
        {       setTitle("Chapter2 - program1");
                setSize(600,400);
                setLocation(200,200);
                // Added for Mac
                GLProfile glp = GLProfile.getMaxProgrammableCore(true);
                GLCapabilities caps = new GLCapabilities(glp);
                myCanvas = new GLCanvas(caps);
                // Replacing this line
                //myCanvas = new GLCanvas();
                myCanvas.addGLEventListener(this);
                getContentPane().add(myCanvas);
                setVisible(true);
        }
```

Step 2: Creating a GLCanvas (Gordon 2.1)

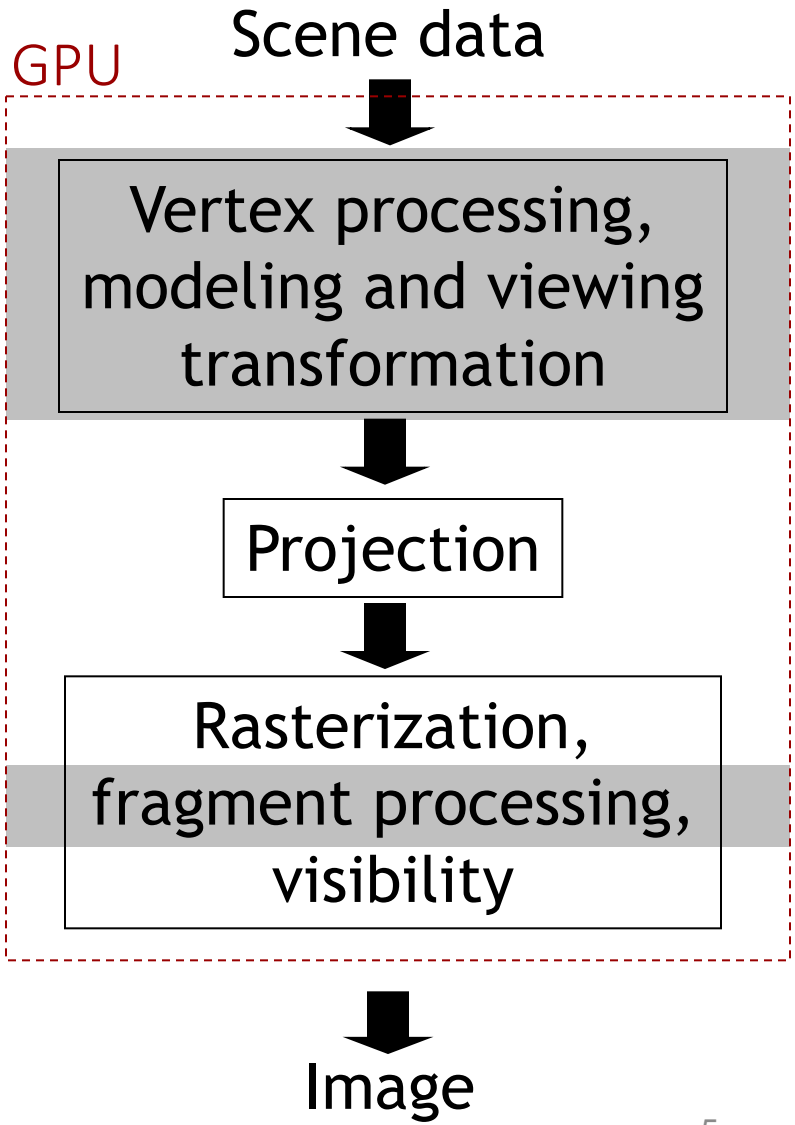
```
public void display(GLAutoDrawable drawable)
        {       GL4 gl = (GL4) GLContext.getCurrentGL();

                float bkg[] = { 1.0f, 1.0f, 0.0f, 1.0f };
                FloatBuffer bkgBuffer = Buffers.newDirectFloatBuffer(bkg);
                gl.glClearBufferfv(GL_COLOR, 0, bkgBuffer);
                System.out.println("display");
        }
```

If not OpenGL version 4, change to GL3 or GL2

Println helps track display events

Programmable pipeline and shaders

- Functionality in parts (grey boxes) of the GPU pipeline specified by user programs
- Called shaders, or shader programs, executed on GPU
- Not all functionality in the pipeline is programmable



Step 3: Compiling shaders

Shader programs are compiled and linked in Java program, sent to GPU

Appear as strings in Java program

```
String vshaderSource[] =
{       "#version 410    \n",
        "void main(void) \n",
        "{ gl_Position = vec4(0.0, 0.0, 0.0, 1.0); } \n"
};

String fshaderSource[] =
{       "#version 410    \n",
        "out vec4 color; \n",
        "void main(void) \n",
        "{ color = vec4(0.0, 0.0, 1.0, 1.0); } \n"
};
```

Step 3: Compiling shaders

```
// This code can be copied verbatim between programs – only write once
int vShader = gl.glCreateShader(GL_VERTEX_SHADER);
int fShader = gl.glCreateShader(GL_FRAGMENT_SHADER);

gl.glShaderSource(vShader, vshaderSource.length, vshaderSource, null, 0);
gl.glCompileShader(vShader);

gl.glShaderSource(fShader, fshaderSource.length, fshaderSource, null, 0);
gl.glCompileShader(fShader);

int vfprogram = gl.glCreateProgram();
gl.glAttachShader(vfprogram, vShader);
gl.glAttachShader(vfprogram, fShader);
gl.glLinkProgram(vfprogram);

gl.glDeleteShader(vShader);
gl.glDeleteShader(fShader);
return vfprogram;
```

Step 4: Reading shaders

Vert.shader: This shader sets a fixed location for every vertex.

```
#version 410
void main(void)
{
        gl_Position = vec4(0.0, 0.0, 0.5, 1.0);
}
```

Frag.shader: This shader sets a fixed color for every pixel.

```
#version 410
out vec4 color;
void main(void)
{
        color = vec4(0.0, 0.0, 1.0, 1.0);
}
```

Step 5: Reading shaders, second example

Three positions for three vertices => triangle

```
#version 410

void main(void)
{ if (gl_VertexID == 0)
        gl_Position = vec4( 0.25,-0.25, 0.0, 1.0);
  else if (gl_VertexID == 1)
        gl_Position = vec4(-0.25,-0.25, 0.0, 1.0);
  else
        gl_Position = vec4( 0.25, 0.25, 0.0, 1.0);
}
```

Step 6: Animation

Animator object in Java:

```
FPSAnimator animator = new FPSAnimator(myCanvas, 30);
animator.start();
```

Animation in shader:.

```
#version 410

uniform float inc;

void main(void)
{
  if (gl_VertexID == 0)
            gl_Position = vec4( 0.25+inc,-0.25, 0.0, 1.0);
  else if (gl_VertexID == 1)
            gl_Position = vec4(-0.25+inc,-0.25, 0.0, 1.0);
  else
            gl_Position = vec4( 0.25+inc, 0.25, 0.0, 1.0);
}
```

Step 6: Animation

Animator object in Java:

```
FPSAnimator animator = new FPSAnimator(myCanvas, 30);
animator.start();
```

Animation in shader:.

```
#version 410

uniform float inc;

void main(void)
{
  if (gl_VertexID == 0)
            gl_Position = vec4( 0.25+inc,-0.25, 0.0, 1.0);
  else if (gl_VertexID == 1)
            gl_Position = vec4(-0.25+inc,-0.25, 0.0, 1.0);
  else
            gl_Position = vec4( 0.25+inc, 0.25, 0.0, 1.0);
}
```

Step 6: Animation

The value of x is passed into the Shader through the glProgramUniform1f call.

The glGetUniformLocation() call gets a ptr to the shader variable

(and, yes, we're passing x here to inc in the shader, so a variable name confusion).

```
x += inc;
int offset_loc = gl.glGetUniformLocation(rendering_program, "inc");
gl.glProgramUniform1f(rendering_program, offset_loc, x);
```

Step 7: Passing in vertices

Finally, if we want to pass vertices into the Shader we need to pass VBOs – Vertex Buffer Objects – that contain vertex information such as

Position
Normal
Color
Texture coordinates