Open Graphics Library (OpenGL) - Continued

Images were taken from the book: “Interactive Computer Graphics” by Angel and Shreiner
OpenGL Shaders - Reminder

• **Vertex shaders** process each of the vertices

• **Fragment shaders** process each pixel in the relevant range
  – For example, if we draw a line between two vertices, the **vertex shader** will process the two vertices and the **fragment shader** will process the pixels on the line
OpenGL Shaders - Reminder

- The vertex shader needs vertex position (any frame), outputs position in clip frame
- The fragment shader gets the output of the vertex shader, interpolated to each pixel
  - Pixel coordinates are obtained using gl_FragCoord
Why is Fragment Shader Important?

- per vertex lighting
- per fragment lighting
What is Wrong Here?

- Vertex shader code (vertices in different colors):

```glsl
#version 150
in vec4 vPosition;
in vec3 in_color;
in mat4 translation;
in mat4 rotation;
out vec3 out_color;
void main() {
    gl_Position = translation * rotation * vPosition;
    out_color = in_color;
}
```

- Sent to the GPU for each vertex separately
- Computed for each vertex
Uniform Variables

- Solution: compute the model-view matrix on the CPU, pass it to the GPU once using a **Uniform** variable

```glsl
#version 150
uniform mat4 modelview;
in vec4 vPosition;
in vec3 in_color;
out vec3 out_color;
void main() {
    gl_Position = modelview * vPosition;
    out_color = in_color;
}
```
Uniform Variables

• CPP code:

```cpp
GLuint modelviewLoc = glGetUniformLocation(program, "modelview");
glfwUniformMatrix4fv(modelviewLoc, 1, GL_FALSE, mvMat);
```

Do not transpose

```cpp
GLfloat*
```