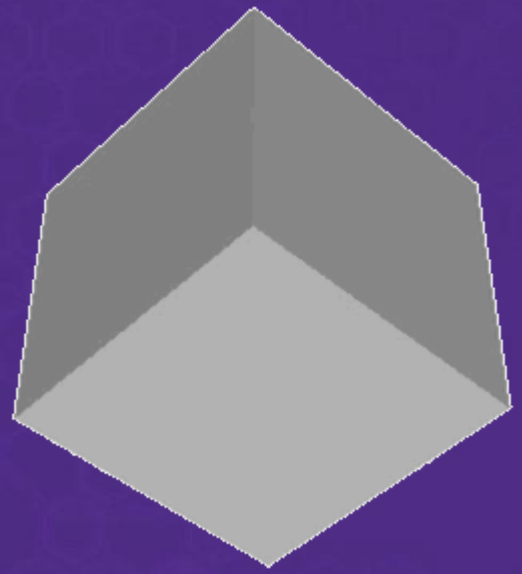
Developing Graphics Frameworks with Java and OpenGL



Part 08: Vertex Colors

Passing Data to Shader Programs

- data passed in and out of shaders by variables declared with *type qualifiers*: keywords that modify variable properties
- **`in`** indicates variable value supplied by previous pipeline stage
- **`out`** indicates variable value will be passed to next pipeline stage
- vertex shader:
  - **`in`** values supplied from a buffer
  - **`out`** values passed to fragment shader
- fragment shader:
  - **`in`** values supplied from vertex shader (interpolated by rasterizer)
  - **`out`** values stored in a buffer (color, depth, stencil)

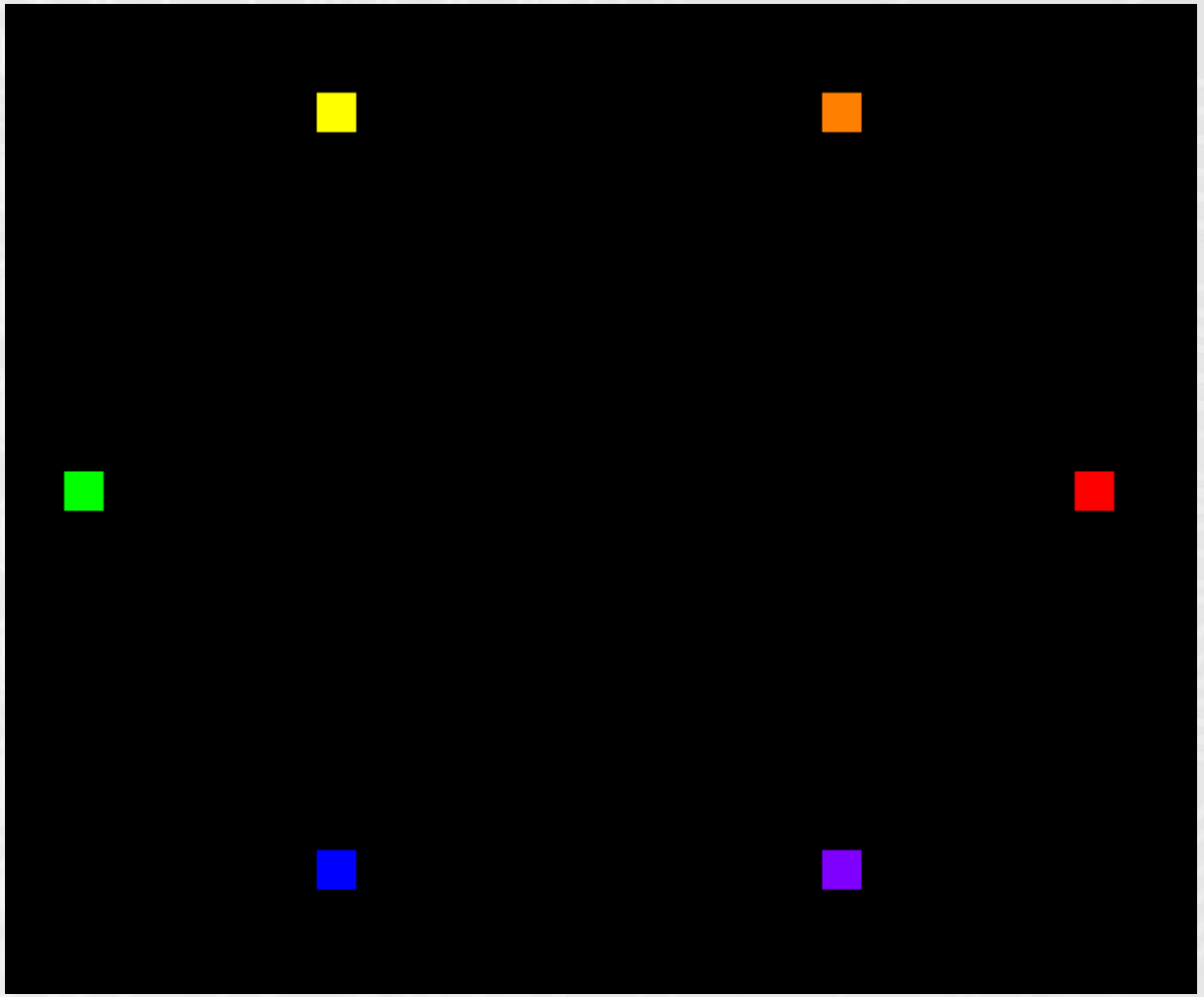
Assigning Color to each Vertex

- Create a second buffer to store colors, then write shaders:

```
in vec3 position;
in vec3 vertexColor;
out vec3 color;
void main() {
    gl_Position = vec4(position.x, position.y, position.z, 1.0);
    color = vertexColor;
}
```

```
in vec3 color;
out vec4 fragColor;
void main() {
    fragColor = vec4(color.r, color.g, color.b, 1.0);
}
```

Result (Points)



Interpolated Vertex Colors

- When using vertex colors and rendering lines/triangles, a weighted average is used to calculate interior colors