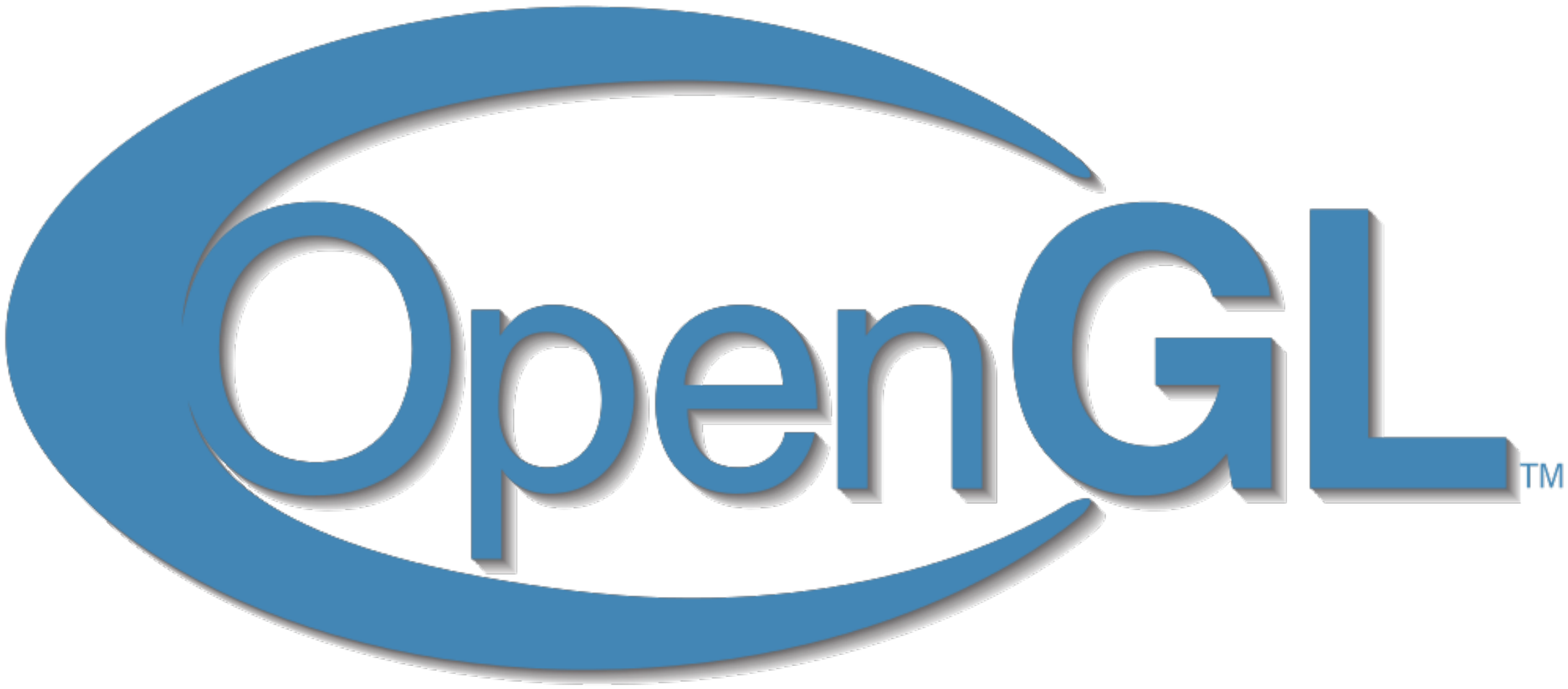

OpenGL
TM

Resources

Modern Tutorial:

https://open.gl/

API Docs

http://docs.gl/

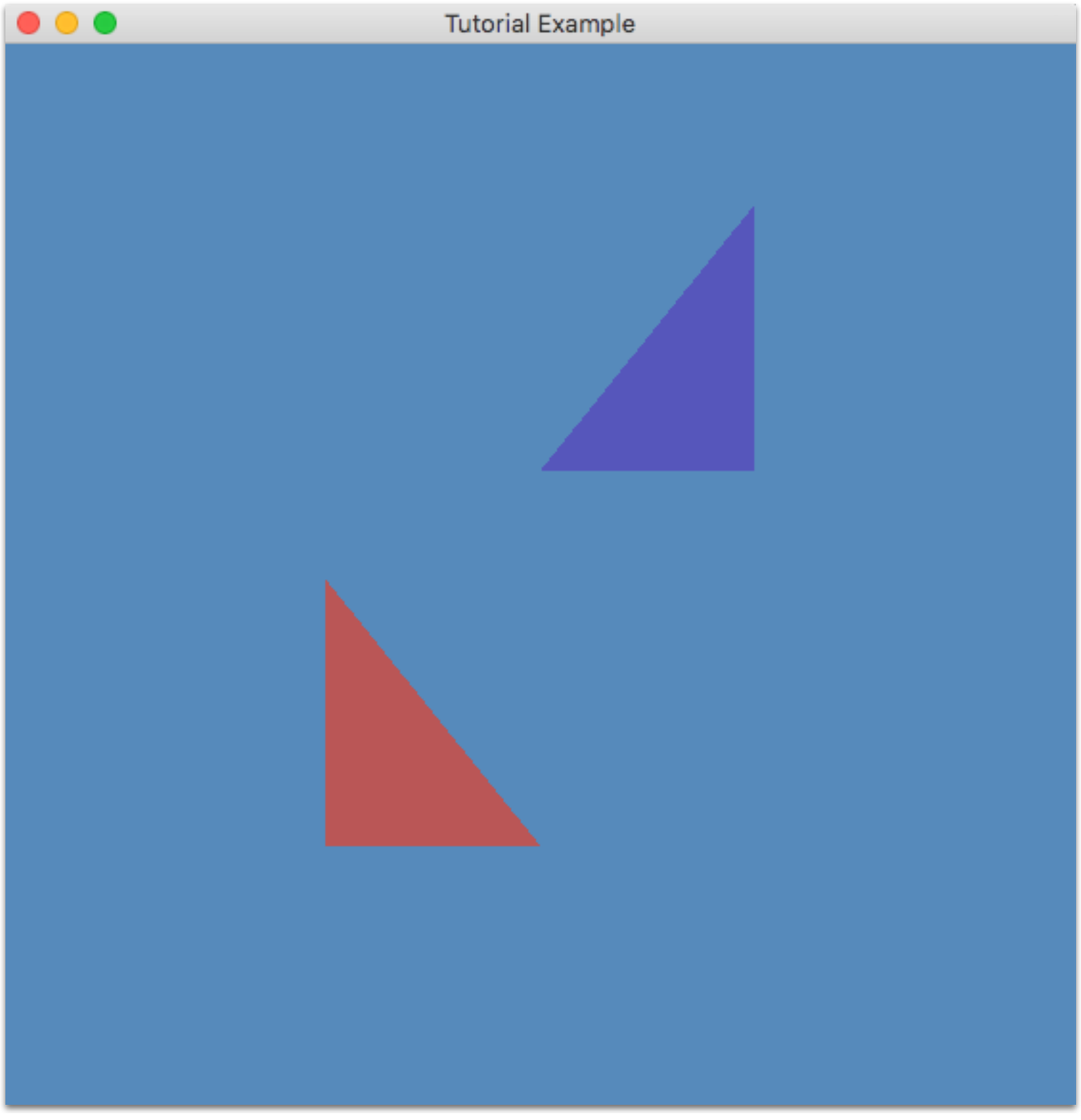
Wiki

https://www.opengl.org/wiki

Big Thanks!

Topics

- VAOs: Vertex Array Objects
- VBOs: Vertex Buffer Objects
- Vertex Attributes
- Data Mapping
- Shader Basics



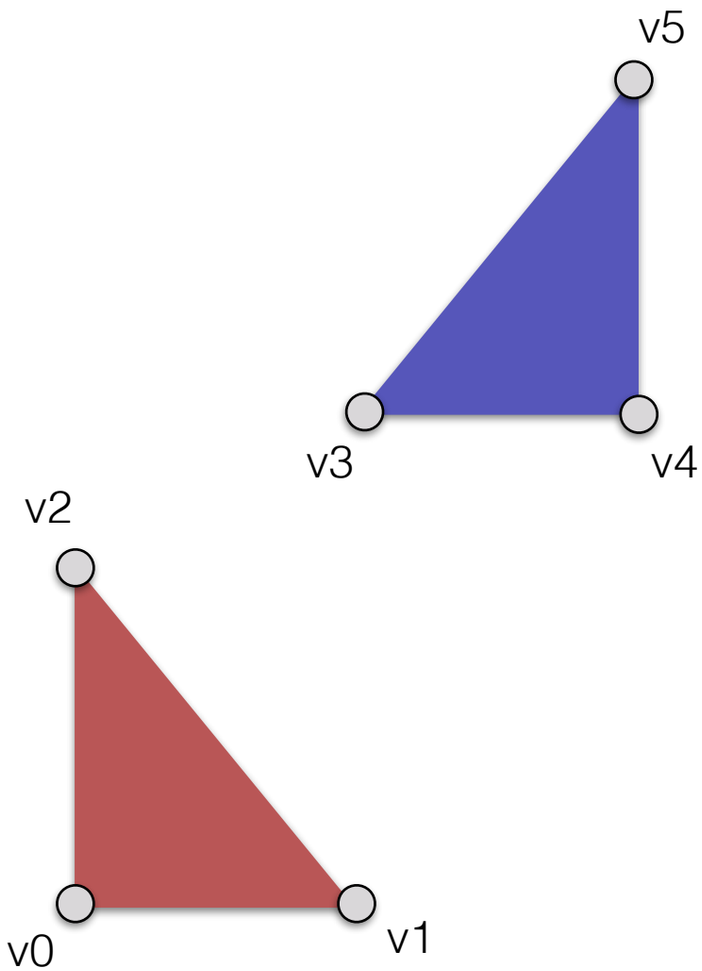
Check Piazza for Tutorial Example code.

Tutorial Example

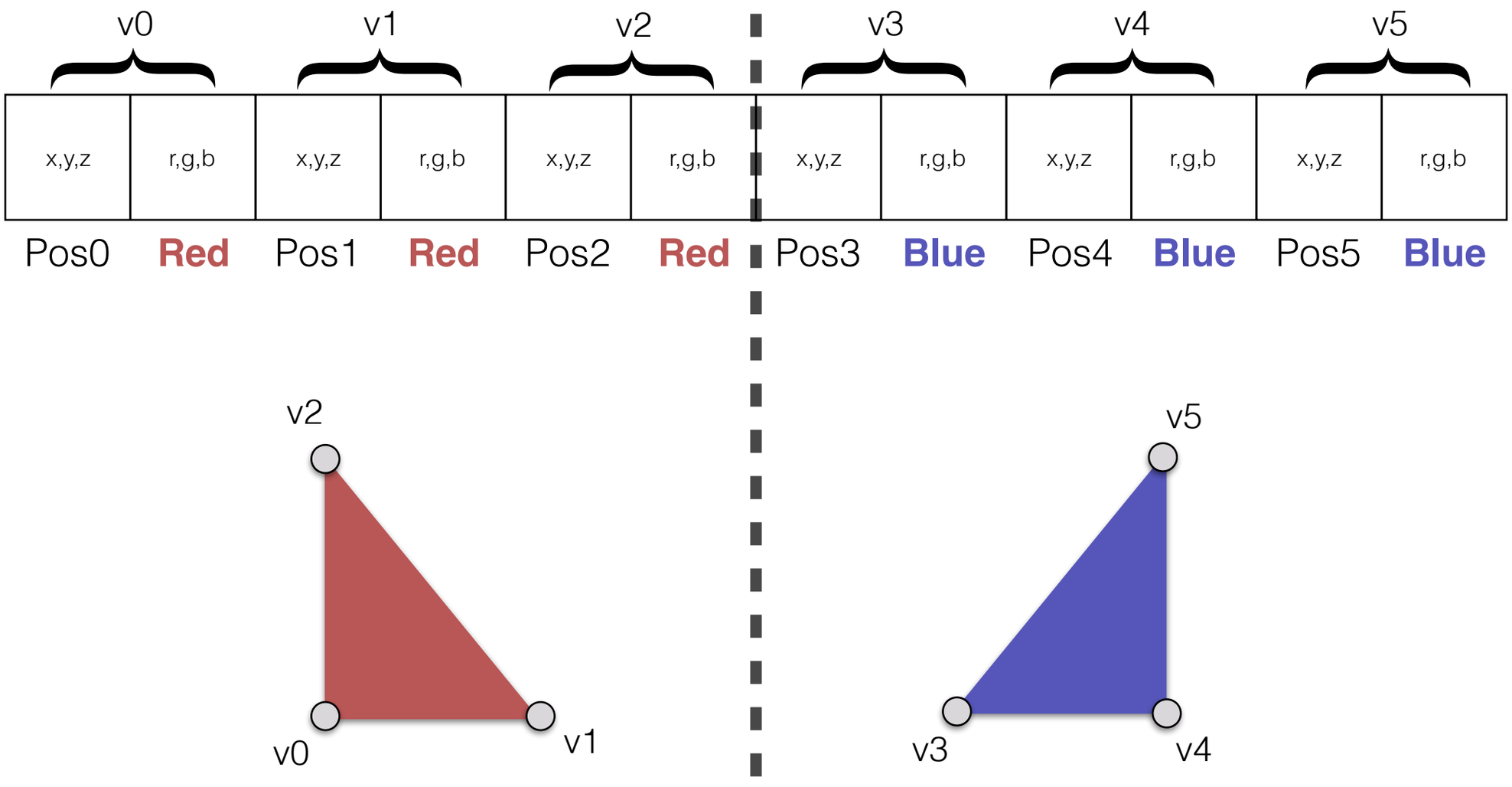
```
void TutorialExample::init() {
    // Create and link Shaders…

    // Generate VBO, and upload triangle data.
    uploadVertexDataToVbo();

    // Generate VAO, and create VBO-to-shader mapping.
    mapVboDataToShaderAttributeLocations();
}
```

Allocate OpenGL resource up front when program starts.

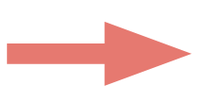
Vertices



Vertex Data Layout



Tutorial Example



```
void TutorialExample::init() {
    // Create and link Shaders…

    // Generate VBO, and upload triangle data.
    uploadVertexDataToVbo();

    // Generate VAO, and create VBO-to-shader mapping.
    mapVboDataToShaderAttributeLocations();
}
```

Tutorial Example

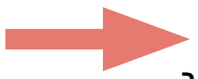
void uploadVertexDataToVbo( )

```
vec3 red(0.7, 0.3, 0.3);
vec3 blue(0.3, 0.3, 0.7);

vec3 triangleVertices[] = {
//   Positions                      Colors.
    vec3(-0.4f, -0.5f, 0.0f),     red,
    vec3(-0.0f, -0.5f, 0.0f),     red,
    vec3(-0.4f,  0.0f, 0.0f),     red,

    vec3( 0.0f,  0.2f, 0.0f),     blue,
    vec3( 0.4f,  0.2f, 0.0f),     blue,
    vec3( 0.4f,  0.7f, 0.0f),     blue,
};
```

```
glGenBuffers(1, &m_vbo);

// Upload triangle data to the vertex buffer.
glBindBuffer(GL_ARRAY_BUFFER, m_vbo);

glBufferData(GL_ARRAY_BUFFER,
             sizeof(triangleVertices),
             triangleVertices,
             GL_STATIC_DRAW);

// All done with GL_ARRAY_BUFFER target.
glBindBuffer(GL_ARRAY_BUFFER, 0);

CHECK_GL_ERRORS;
```

Tutorial Example

```
void TutorialExample::init() {
    // Create and link Shaders…

    // Generate VBO, and upload triangle data.
    uploadVertexDataToVbo();

    // Generate VAO, and create VBO-to-shader mapping.
    mapVboDataToShaderAttributeLocations();
}
```



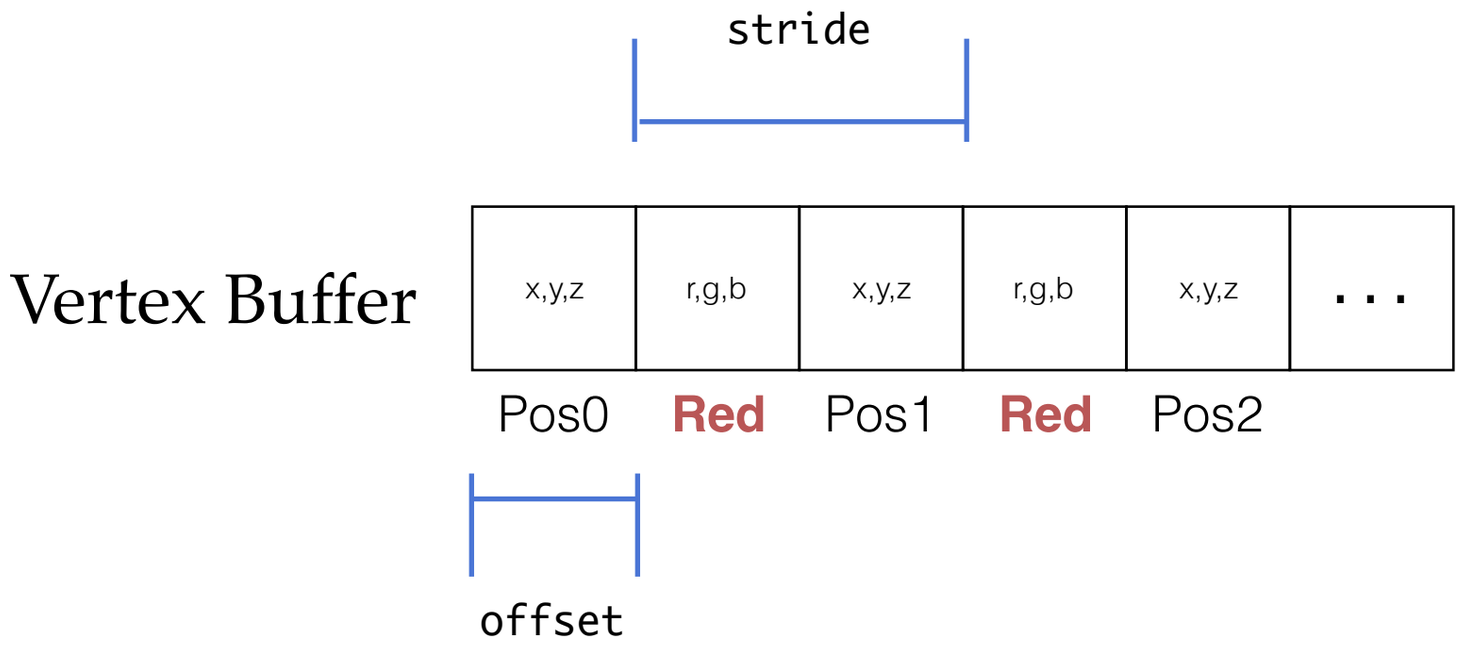
Tutorial Example

void mapVboDataToShaderAttributeLocations( )

```
glGenVertexArrays(1, &m_vao);

// Bind the VAO that will record the mapping.
glBindVertexArray(m_vao);

// Bind the VBO that contains data to be mapped.
glBindBuffer(GL_ARRAY_BUFFER, m_vbo);

// Map position data from VBO to vertex attribute slot:
stride = sizeof(vec3)*2;
offset = 0;
glVertexAttribPointer(
    m_position_attrib_location,
    3, GL_FLOAT, GL_FALSE, stride, offset
);

// Map color data from VBO to vertex attribute slot:
stride = sizeof(vec3) * 2;
offset = sizeof(vec3);
glVertexAttribPointer(
    m_color_attrib_location,
    3, GL_FLOAT, GL_FALSE, stride, offset
);
```

Stride: Num bytes from start of data element to next data element of the same type.

Offset: Num bytes from start of vertex buffer to first data element of a given type.



- Example here shows the `stride` and `offset` for our color elements
- Position elements have the same `stride`, but an `offset` of `0` since the first position element starts at the beginning of our vertex buffer.

Tutorial Example

void mapVboDataToShaderAttributeLocations( )

```
glBindVertexArray(m_vao);

// Enable position vertex attribute slot.
glEnableVertexAttribArray(m_position_attrib_location);
```

```
// Enable color vertex attribute slot.
glEnableVertexAttribArray(m_color_attrib_location);


// Unbind VAO, and check for errors.
glBindVertexArray(0);
CHECK_GL_ERRORS;
```

Tutorial Example

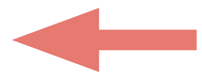
Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

Tutorial Example

Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

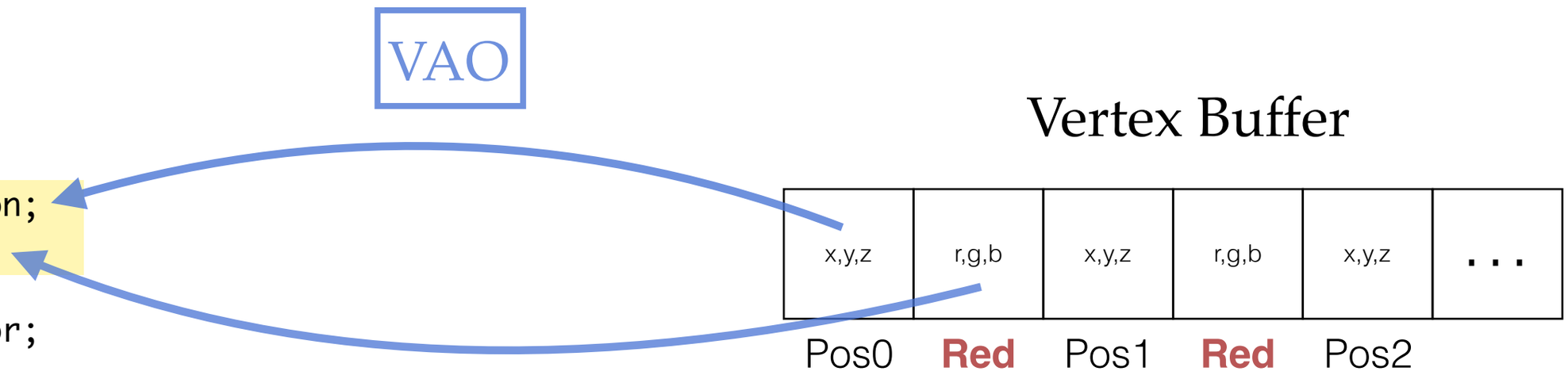


Vertex Attributes

Tutorial Example

Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```



Tutorial Example

Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

Fragment Shader

```
#version 330

in vec3 vsColor;

out vec4 fragColor;

void main() {
    fragColor = vec4(vsColor, 1.0);
}
```

Tutorial Example

Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

Fragment Shader

```
#version 330

in vec3 vsColor;

out vec4 fragColor;

void main() {
    fragColor = vec4(vsColor, 1.0);
}
```

- Type must match.
- Name must match.

Tutorial Example

Vertex Shader

```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

Fragment Shader

```
#version 330

in vec3 vsColor;

out vec4 fragColor;

void main() {
    fragColor = vec4(vsColor, 1.0);
}
```

Tutorial Example

Vertex Shader

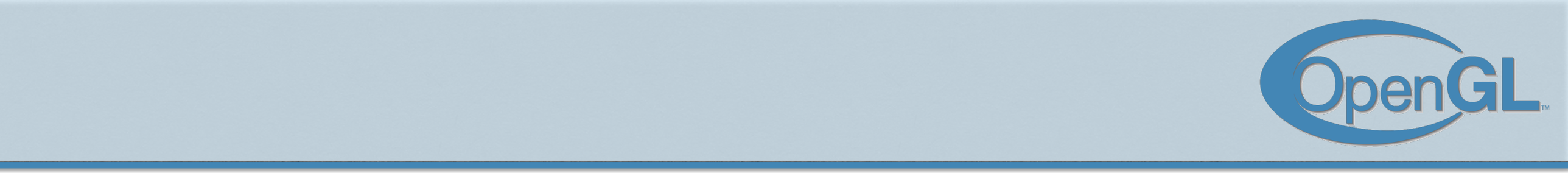
```
#version 330

in vec3 position;
in vec3 color;

out vec3 vsColor;

void main() {
    gl_Position = vec4(position, 1.0);

    vsColor = color;
}
```

Fragment Shader

```
#version 330

in vec3 vsColor;

out vec4 fragColor;

void main() {
    fragColor = vec4(vsColor, 1.0);
}
```

- Used to interpolate vertex outputs (e.g. vsColor) at screen-space fragment locations.
- Typically in 3D you want to set `gl_Position` equal to the clip space coordinate of the vertex in order to do perspectively correct interpolation.

(`gl_Position = Projection * View * Model * vec4(position, 1.0);`)



Questions?