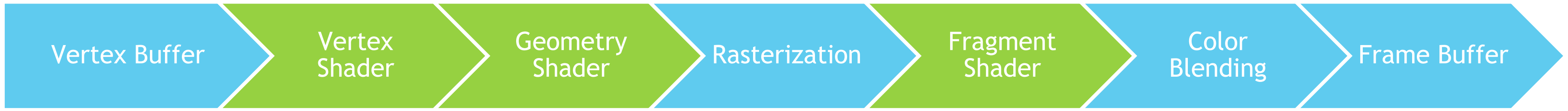
OpenGL - Shader & GLSL

Sandbox

- http://glslsandbox.com/

Rendering Pipeline Overview



Hello Shader!

- Shader Programing in OpenGL
- Data Connection
  - VBO
  - Uniform
  - Texture
- GLSL Syntax

Shader Programing in OpenGL (Packaged in shader.h)

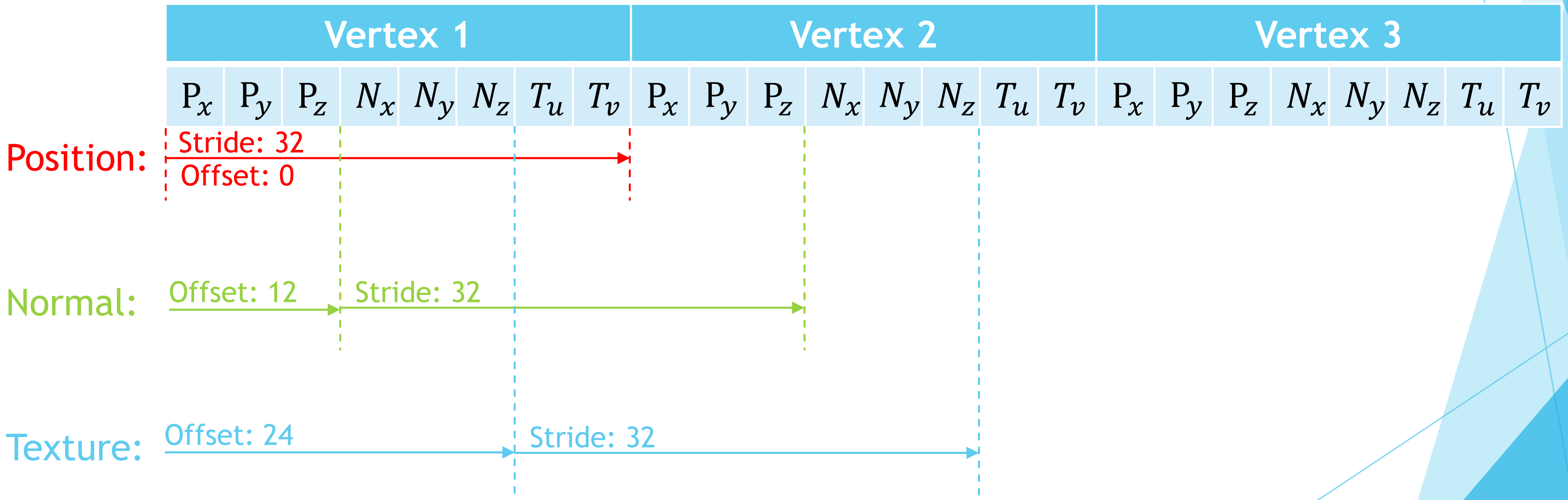
- GLuint glCreateShader(GLenum shaderType);
  - Creates an empty shader object
  - GL_COMPUTE_SHADER, GL_VERTEX_SHADER, GL_TESS_CONTROL_SHADER, GL_TESS_EVALUATION_SHADER, GL_GEOMETRY_SHADER, GL_FRAGMENT_SHADER
- void glShaderSource(GLuint shader, GLsizei count, const GLchar **string, const GLint *length);
  - Replaces the source code in a shader object
- void glCompileShader(GLuint shader);
  - Compiles a shader object

Shader Programing in OpenGL (Packaged in shader.h)

- GLuint glCreateProgram(void);
  - Creates a program object
- void glAttachShader(GLuint program, GLuint shader);
  - Attaches a shader object to a program object
- void glLinkProgram(GLuint program);
  - Links a program object
- void glDetachShader(GLuint program, GLuint shader);
  - Detaches a shader object from a program object to which it is attached

Data Connection (VBO)

- Vertex Buffer Object



Vertex 1
Vertex 2
Vertex 3
P_x P_y P_z N_x N_y N_z T_u T_v P_x P_y P_z N_x N_y N_z T_u T_v P_x P_y P_z N_x N_y N_z T_u T_v
Position:
Stride: 32
Offset: 0
Normal:
Offset: 12
Stride: 32
Texture:
Offset: 24
Stride: 32

Vertex Attribute

```
struct VertexAttribute
{
  GLfloat position[3];
  GLfloat normal[3];
  GLfloat texcoord[2];
};
```

Implementation in OpenGL

```
glGenBuffers(1, &vboName);
glBindBuffer(GL_ARRAY_BUFFER, vboName);

VertexAttribute *vertices;
glBufferData(GL_ARRAY_BUFFER,
    sizeof(VertexAttribute) * vertices_length,
    vertices,
    GL_STATIC_DRAW);
```

Link to GLSL

🟩 🟦 Must Be Same!!!

```
glEnableVertexAttribArray(0);
glVertexAttribPointer(0,
    3,
    GL_FLOAT,
    GL_FALSE,
    sizeof(VertexAttribute),
    (void*)(offsetof(VertexAttribute, position)));
```

OpenGL

```
layout(location = 0) in vec3 pos;
```

GLSL

Data Connection (Uniform)

- Uniform
  - Act as parameters that the user can pass to the program.
  - They do not change in shader

- ~~Attribute~~ (deprecated now)
  - Alias to in

- ~~Varying~~ (deprecated now)
  - Alias to out

Implementation in OpenGL

☐ Must Be Same!!!

```
GLfloat pmtx[16];
glGetFloatv(GL_PROJECTION_MATRIX, pmtx);
GLint pmatLoc = glGetUniformLocation(program, "Projection");

glUseProgram(program);
glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, pmtx);
glUseProgram(0);
```

OpenGL

```
uniform mat4 Projection;
```

GLSL

Data Connection (Texture)

🟩 🟦 Must Be Same!!!

```
GLint texLoc = glGetUniformLocation(program, "Texture");

glUseProgram(program);
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texObj);
glUniform1i(texLoc, 0);
glBindTexture(GL_TEXTURE_2D, NULL);
glUseProgram(0);
```

OpenGL

```
layout(binding = 0) uniform sampler2D Texture;
in vec2 texcoord;
out vec4 outColor;
void main() { outColor = texture2D(MyTexture_1, texcoord); }
```

GLSL – Fragment Shader

GLSL Syntax

- Basic Variable Types
  - vec2, vec3, vec4, …
  - mat2, mat3, mat4, …
  - float, int, bool, …
  - sampler2D, …
- Basic Functions
  - max, min, sin, cos, pow, log, …
  - dot, normalize, reflect, …
  - transpose, inverse, …

Reference

- https://www.khronos.org/registry/OpenGL-Refpages/gl4/