GPU Side: Create shader
Step 1. Vertex shader

```
#version 330 core

// 1. Vertex attributes
layout(location = 0) in vec3 position;
layout(location = 1) in vec3 color;

// 2. Uniforms
uniform mat4 uniform_mvp;

// 3. Outgoing variables (to fragment shader)
out vec4 vertex_color;

void main(void) {
    vertex_color = vec4(color, 1.0);

// 4. Required output
    gl_Position = uniform_mvp * vec4(position, 1.0);
}
```

Step 2. Fragment shader

```
#version 330 core

// 1. Required write locations
layout(location = 0) out vec4 out_color;

// 2. Uniforms
uniform vec4 uniform_color;

// 3. Incoming variables (from vertex shader)
in vec4 vertex_color;

void main(void) {
// 4. Write the output
    out_color = vertex_color;
}
```

CPU Side

Step 1. Init: Compile shader and create uniforms

```
// create a new shader object
ShaderGLSL* bgs_glsl = new ShaderGLSL("BasicGeometry");
// compile it
bool shader_loaded = bgs_glsl->LoadAndCompile();
if (!shader_loaded) return false;
// get its id
bgs_program_id = bgs_glsl->GetProgram();

// get id of uniforms
bgs_uniform_mvp = glGetUniformLocation(bgs_program_id,
"uniform_mvp");
bgs_uniform_color = glGetUniformLocation(bgs_program_id,
"uniform_color");
```

Step 2. Draw: Pass active shader data to GPU and draw

```
// set shader active
glUseProgram(bgs_program_id);

// pass uniform information (using the uniform id)
glUniformMatrix4fv(bgs_uniform_mvp, 1, false, &mat[0][0]);
glUniform4f(bgs_uniform_color, 0.0f, 1.0f, 0.0f, 1.0f);

// Other stuff here (bind vertex array objects, etc)
.
.
.

// Draw!
glDrawArrays(GL_TRIANGLES, 0, num_of_vertices);
glUseProgram(0);
```