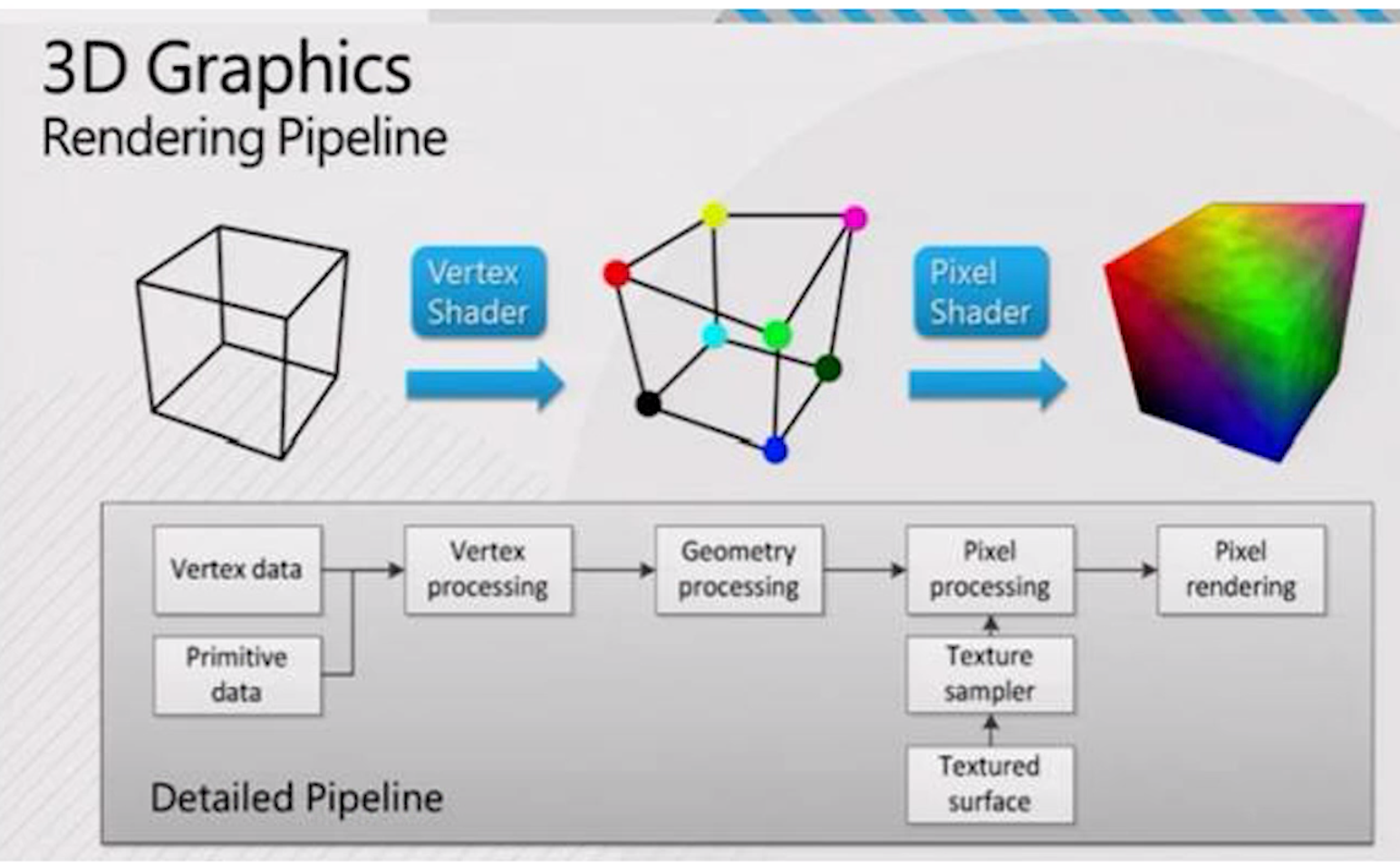
Shaders



<https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Minimal_Shader>

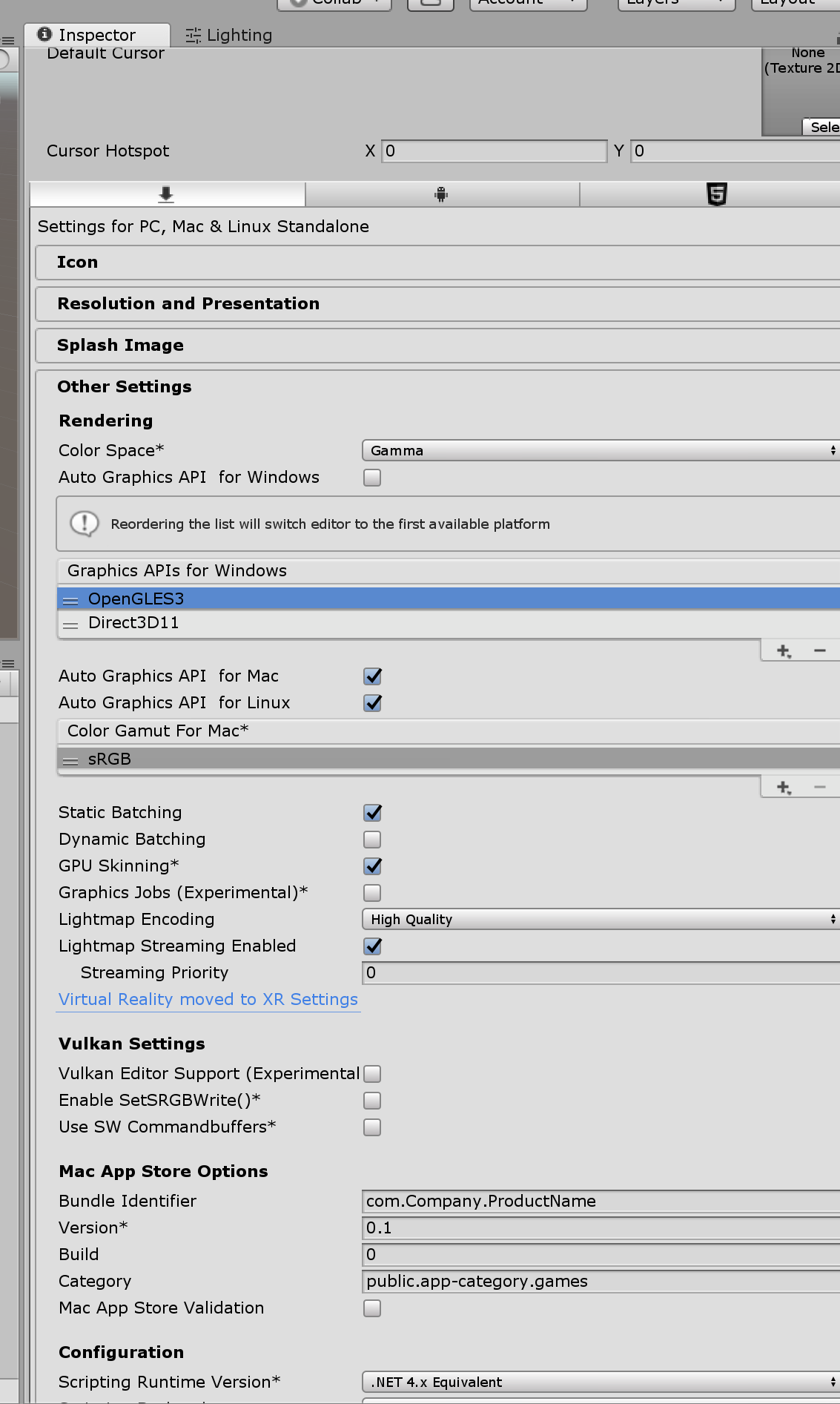
Need to attach the shader to a material.

Vertex shader: change the position of the vertices in the rendering.

Pixel shaders: changes every pixel.

Important:

Configuration in Unity:



Shader "GLSLShader" {

SubShader{ // Unity chooses the subshader that fits the GPU best

Pass { // some shaders require multiple passes

GLSLPROGRAM // here begins the part in Unity's GLSL

#ifdef VERTEX // here begins the vertex shader

void main() // all vertex shaders define a main() function

{

gl\_Position = gl\_ModelViewProjectionMatrix \* gl\_Vertex;

// this line transforms the predefined attribute

// gl\_Vertex of type vec4 with the predefined

// uniform gl\_ModelViewProjectionMatrix of type mat4

// and stores the result in the predefined output

// variable gl\_Position of type vec4.

}

#endif // here ends the definition of the vertex shader

#ifdef FRAGMENT // here begins the fragment shader

void main() // all fragment shaders define a main() function

{

gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0);

// this fragment shader just sets the output color

// to opaque red (red = 1.0, green = 0.0, blue = 0.0,

// alpha = 1.0)

}

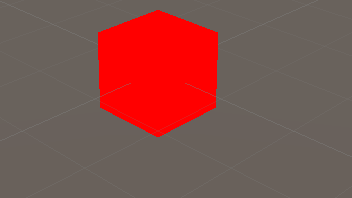
#endif // here ends the definition of the fragment shader

ENDGLSL // here ends the part in GLSL

}

}

}



 gl\_Position can only be set from within the vertex shader, since it's a per-vertex attribute. Loading the shaders in correct order should get rid of that problem though.

 In GLSL, a “shader” is either a vertex shader or a fragment shader. The combination of both is called a “program”.

Unfortunately, Unity refers to this kind of program as a “shader”, while in Unity a vertex shader is called a “vertex program” and a fragment shader is called a “fragment program”.

<https://docs.unity3d.com/Manual/SL-VertexFragmentShaderExamples.html>

each Pass represents an execution of the vertex and fragment code for the same object rendered with the material of the shader

The **Vertex Shader**  
 is a program that runs on each vertex of the 3D model. Quite often it does not do anything particularly interesting. Here we just transform vertex position from object space into so called “clip space”, which is what’s used by the GPU to rasterize the object on screen. We also pass the input texture coordinate unmodified - we’ll need it to sample the texture in the fragment shader.

The **Fragment Shader**  
 is a program that runs on each and every pixel that object occupies on-screen, and is usually used to calculate and output the color of each pixel. Usually there are millions of **pixels**  
 on the screen, and the fragment shaders are executed for all of them! Optimizing fragment shaders is quite an important part of overall game performance work.

<https://en.wikibooks.org/wiki/GLSL_Programming/Unity>

<https://en.wikibooks.org/wiki/GLSL_Programming/Unity/RGB_Cube>