# GLSL Code – Phong Shader

Unfortunately I failed to get the phong shader code to work properly in my ACW. It loads and compiles successfully, however the end result does not look as expected, and how it looked in RenderMonkey (it works fine in RenderMonkey). Below are snippets of code of the vertex and fragment shaders, with the key points explained.

## Vertex Shader

The vertex shader is very simple it defines the maximum number of lights available, and the actual number of lights to use. It calculates the light direction for each of the lights to use and passes it to the Fragment shader along with the eye position and normal

*varying vec3 normal, eyeVec;  
#define MAX\_LIGHTS 8  
#define NUM\_LIGHTS 6  
varying vec3 lightDir[MAX\_LIGHTS];  
void main()  
{  
 gl\_Position = ftransform();  
 normal = gl\_NormalMatrix \* gl\_Normal;  
 vec4 vVertex = gl\_ModelViewMatrix \* gl\_Vertex;  
 eyeVec = -vVertex.xyz;  
 int i;  
 for (i=0; i<NUM\_LIGHTS; ++i)  
 lightDir[i] =  
 vec3(gl\_LightSource[i].position.xyz - vVertex.xyz);  
}*

## Fragment Shader

The fragment shader is slightly more complex, and as a result longer in code than the vertex shader.  
For each light direction passed in from the vertex shader it is normalized. Because spotlights are been used the spotDirection is taken from the actual light source and normalized too. If the dot product of these two is greater than the spot cutoff (taken from the lightsource itself) then there is a pixel that needs to be illuminated. This process is then repeated via a for-loop for all of the light sources that are in use.

*varying vec3 normal, eyeVec;  
#define MAX\_LIGHTS 8  
#define NUM\_LIGHTS 6  
varying vec3 lightDir[MAX\_LIGHTS];  
void main (void)  
{  
 vec4 final\_color =  
 gl\_FrontLightModelProduct.sceneColor;  
 vec3 N = normalize(normal);  
 int i;  
 for (i=0; i<NUM\_LIGHTS; ++i)  
 {  
 vec3 L = normalize(lightDir[i]);  
 vec3 D = normalize(gl\_LightSource[i].spotDirection);  
 float lambertTerm = dot(N,L);  
 if (dot(-L, D) > gl\_LightSource[i].spotCosCutoff)   
 {  
 if (lambertTerm > 0.0)  
 {  
 final\_color +=  
 gl\_LightSource[i].diffuse \*  
 gl\_FrontMaterial.diffuse \*  
 lambertTerm;  
 vec3 E = normalize(eyeVec);  
 vec3 R = reflect(-L, N);  
 float specular = pow(max(dot(R, E), 0.0),  
 gl\_FrontMaterial.shininess);  
 final\_color +=  
 gl\_LightSource[i].specular \*  
 gl\_FrontMaterial.specular \*  
 specular;  
 }  
 }  
 }  
 gl\_FragColor = final\_color;  
}*