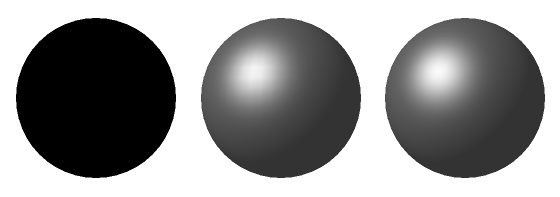
Writing shader lighting algorithms in OpenGL ES 2.0 is actually pretty simple, but If you’re writing them from scratch it may take some time to get everything right. Below you can find source code examples for a vertex lighting shader and a fragment lighting shader. You can also get them from[github.com/antonholmquist/opengl-es-2-0-shaders](http://github.com/antonholmquist/opengl-es-2-0-shaders)

Per-vertex lighting is the traditional way for OpenGL ES 1.1 to do lighting, and per-fragment lighting is new to OpenGL ES 2.0 made possible with shaders.



First is a sphere with no lighting. Then comes the same sphere with per-vertex-lighting, and finally with per-fragment lighting. You can clearly see some artifacts in the middle image, mainly in the specular part.



Per-vertex shading above. Per-fragment shading below.



Vertex Lighting Shader Example

Light is calculated once per vertex and sent to fragment shader.

**Vertex Shader:**

**struct** **DirectionalLight** {  
    vec3 direction;  
    vec3 halfplane;  
    vec4 ambientColor;  
    vec4 diffuseColor;  
    vec4 specularColor;  
};  
  
**struct** **Material** {  
    vec4 ambientFactor;  
    vec4 diffuseFactor;  
    vec4 specularFactor;  
    **float** shininess;  
};  
  
// Light  
uniform **DirectionalLight** u\_directionalLight;  
  
// Material  
uniform **Material** u\_material;  
  
// Matrices  
uniform mat4 u\_mvMatrix;  
uniform mat4 u\_mvpMatrix;  
  
// Attributes  
attribute vec4 a\_position;  
attribute vec3 a\_normal;  
  
// Varyings  
varying vec4 v\_light;  
  
**void** main() {  
  
    // Define position and normal in model coordinates  
    vec4 mcPosition = a\_position;  
    vec3 mcNormal = a\_normal;  
  
    // Calculate and normalize eye space normal  
    vec3 ecNormal = vec3(u\_mvMatrix \* vec4(mcNormal, 0.0));  
    ecNormal = ecNormal / length(ecNormal);  
  
    // Do light calculations  
    **float** ecNormalDotLightDirection = max(0.0, dot(ecNormal, u\_directionalLight.direction));  
    **float** ecNormalDotLightHalfplane = max(0.0, dot(ecNormal, u\_directionalLight.halfplane));  
  
    // Ambient light  
    vec4 ambientLight = u\_directionalLight.ambientColor \* u\_material.ambientFactor;  
  
    // Diffuse light  
    vec4 diffuseLight = ecNormalDotLightDirection \* u\_directionalLight.diffuseColor \* u\_material.diffuseFactor;  
  
    // Specular light  
    vec4 specularLight = vec4(0.0);  
    **if** (ecNormalDotLightHalfplane > 0.0) {  
        specularLight = pow(ecNormalDotLightHalfplane, u\_material.shininess) \* u\_directionalLight.specularColor \* u\_material.specularFactor;  
    }   
  
    v\_light = ambientLight + diffuseLight + specularLight;  
    gl\_Position = u\_mvpMatrix \* mcPosition;  
}

**Fragment Shader:**

precision highp **float**;  
  
varying vec4 v\_light;  
  
**void** main() {  
    gl\_FragColor = v\_light;  
}

Fragment Lighting Shader Example

Light is calculated in the fragment shader for each pixel.

**Vertex Shader:**

precision highp **float**;  
  
// Matrices  
uniform mat4 u\_mvMatrix;  
uniform mat4 u\_mvpMatrix;  
  
// Attributes  
attribute vec4 a\_position;  
attribute vec3 a\_normal;  
  
// Varyings  
varying vec3 v\_ecNormal;  
  
**void** main() {  
  
    // Define position and normal in model coordinates  
    vec4 mcPosition = a\_position;  
    vec3 mcNormal = a\_normal;  
  
    // Calculate and normalize eye space normal  
    vec3 ecNormal = vec3(u\_mvMatrix \* vec4(mcNormal, 0.0));  
    ecNormal = ecNormal / length(ecNormal);  
    v\_ecNormal = ecNormal;  
  
    gl\_Position = u\_mvpMatrix \* mcPosition;  
}

**Fragment Shader:**

precision highp **float**;  
  
**struct** **DirectionalLight** {  
    vec3 direction;  
    vec3 halfplane;  
    vec4 ambientColor;  
    vec4 diffuseColor;  
    vec4 specularColor;  
};  
  
**struct** **Material** {  
    vec4 ambientFactor;  
    vec4 diffuseFactor;  
    vec4 specularFactor;  
    **float** shininess;  
};  
  
// Light  
uniform **DirectionalLight** u\_directionalLight;  
  
// Material  
uniform **Material** u\_material;  
  
varying vec3 v\_ecNormal;  
  
**void** main() {   
  
    // Normalize v\_ecNormal  
    vec3 ecNormal = v\_ecNormal / length(v\_ecNormal);  
  
    **float** ecNormalDotLightDirection = max(0.0, dot(ecNormal, u\_directionalLight.direction));  
    **float** ecNormalDotLightHalfplane = max(0.0, dot(ecNormal, u\_directionalLight.halfplane));  
  
    // Calculate ambient light  
    vec4 ambientLight = u\_directionalLight.ambientColor \* u\_material.ambientFactor;  
  
    // Calculate diffuse light  
    vec4 diffuseLight = ecNormalDotLightDirection \* u\_directionalLight.diffuseColor \* u\_material.diffuseFactor;  
  
    // Calculate specular light  
    vec4 specularLight = vec4(0.0);  
    **if** (ecNormalDotLightHalfplane > 0.0) {  
        specularLight = pow(ecNormalDotLightHalfplane, u\_material.shininess) \* u\_directionalLight.specularColor \* u\_material.specularFactor;  
    }   
  
    vec4 light = ambientLight + diffuseLight + specularLight;  
  
    gl\_FragColor = light;  
}

http://antonholmquist.com/blog/opengl-es-2-0-shader-lighting-examples/