```
export default `
  attribute vec4 aVertexPosition;
  attribute vec4 aVertexColor;
  attribute vec3 aVertexNormal;

  uniform mat4 uModelViewMatrix;
  uniform mat4 uProjectionMatrix;
  uniform vec4 uLightPos;

  varying lowp vec4 vColor;

  float computeDiffuseIntens(in vec4 position, in vec3 normal, in vec4 lightPos)
  {
    // since the lightPos is in world coordinates, need to transform the vertex
position and normal first
    position = uModelViewMatrix * position;
    normal = (uModelViewMatrix * vec4(normal, 0.0)).xyz;
    normal = normalize(normal);

    // compute the light direction
    vec3 vecToLight = normalize(lightPos.xyz - position.xyz);

    // the intensity is proportional to the angle between the surface and the
light direction
    float diffuseIntensity = dot(normal, vecToLight);
    diffuseIntensity = clamp(diffuseIntensity, 0.0, 1.0);

    return diffuseIntensity;
  }

  void main(void) {

    // transform the vertex position from object space to camera space
    gl_Position = uProjectionMatrix * uModelViewMatrix * aVertexPosition;

    // === very simplistic lighting model ===
    float ambientIntensity = 0.4; // controls how much ambient light there is in
the scene, some value between 0.0 and 1.0
    float diffuseIntensity = computeDiffuseIntens(aVertexPosition,
aVertexNormal, uLightPos); // accounts for direct light
    vColor = aVertexColor * (diffuseIntensity + ambientIntensity);

    // make sure we don't overshoot
    vColor = clamp(vColor, 0.0, 1.0);
    vColor[3] = 1.0; // alpha value
  }
  `;
```