# Lighting Appendix

COMP3421 Computer Graphics • KC Notes

## A1 Flat and Gouraud shading

|  |
| --- |
| // **vertex\_flat.glsl**  // Note: This shader assumes there is no non-uniform scale in either the view  // or the model transform.  in vec3 position; // Incoming vertex position  in vec3 normal; // Incoming normal  uniform mat4 model\_matrix;  uniform mat4 view\_matrix;  uniform mat4 proj\_matrix;  // Light properties  uniform vec3 lightPos;  uniform vec3 lightIntensity;  uniform float ambientIntensity;  // Material properties  uniform vec3 ambientCoeff;  uniform vec3 diffuseCoeff;  uniform vec3 specularCoeff;  uniform float phongExp;  **flat out vec3 intensity; // only in if gouraud**  void main() {  // The global position is in homogenous coordinates  vec4 globalPosition = model\_matrix \* vec4(position, 1);  // The position in camera coordinates  vec4 viewPosition = view\_matrix \* globalPosition;  // The position in CVV coordinates  gl\_Position = proj\_matrix \* viewPosition;  // Compute the normal in view coordinates  vec3 m = normalize(view\_matrix\*model\_matrix \* vec4(normal, 0)).xyz;  // Compute the s, v and r vectors  vec3 s = normalize(view\_matrix\*vec4(lightPos,1) - viewPosition).xyz;  vec3 v = normalize(-viewPosition.xyz);  vec3 r = normalize(reflect(-s,m));  vec3 ambient = ambientIntensity\*ambientCoeff;  vec3 diffuse = max(lightIntensity\*diffuseCoeff\*dot(m,s), 0.0);  vec3 specular;  // Only show specular reflections for the front face  if (dot(m,s) > 0)  specular = max(lightIntensity\*specularCoeff\*pow(dot(r,v),phongExp), 0.0);  else  specular = vec3(0);  intensity = ambient + diffuse + specular;  } |
| // fragment\_flat.glsl  out vec4 outputColor;  uniform vec4 input\_color;  **flat in vec3 intensity; // only in if gouraud**  void main() {  outputColor = vec4(intensity,1) \* input\_color;  } |

## A2 Gouraud shading