Classical Phong Local Lighting Model at a Point, Q, on a Surface

$$I_Q = k_a * L_a + \sum_i f_i(Q) L_i \left\{ k_d * \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_i \right) + k_s * \left(\hat{\mathbf{r}}_i \cdot \hat{\mathbf{v}} \right)^m \right\}$$

Term	Type	Variable in shader	Notes (All points, vectors, and computations are in EC)
Q	*point*	`pvaIn.ecPosition`	Point at which lighting model is to be evaluated.
k_a	*rgb*	`uniform vec3 ka`	Fraction of incident ambient light that is reflected; oftentimes $k_a = k_d$
k_d	*rgb*	`uniform vec3 kd`	Fraction of incident light that is diffusely reflected
k_s	*rgb*	`uniform vec3 ks`	Fraction of incident light that is specularly reflected
L_a	*rgb*	`uniform vec3 La`	Amount of ambient light in the environment
L_i	*rgb*	`uniform vec3 lightStrength[max]`	Strength of i^{th} light source
$f_i(Q)$	*float*	`float atten(i, Q)`	Your shader function that computes the attenuation for light source i at point Q.
$\hat{\mathbf{n}}$	*vector*	`vec3 ec_nHat`	Local variable computed from `pvaIn.ecUnitNormal`. (It is conditionally negated.)
$L_{i,xyzw}$	*xyzw*	`uniform vec4 p_ecLightPos[max]`	Projective space description of light source placement.
$\hat{\mathbf{l}}_i$	*vector*	`vec3 liHat`	Computed unit normal to source i: $\hat{\mathbf{l}}_i = \text{normalize}\left(L_{i,xyz} - Q\right)$ ‡ or $\hat{\mathbf{l}}_i = \text{normalize}\left(L_{i,xyz}\right)$ §
$\hat{\mathbf{r}}_i$	*vector*	`vec3 riHat`	Computed unit vector in primary reflection direction for light source i
$\hat{\mathbf{v}}$	*vector*	`vec3 vHat`	Computed unit vector towards the eye; if perspective, $\hat{\mathbf{v}} = \text{normalize}(O - Q)$, where O=(0,0,0); else $\hat{\mathbf{v}} = \text{normalize}(-\mathbf{M}_{02}/\mathbf{M}_{00}, -\mathbf{M}_{12}/\mathbf{M}_{11}, 1)$, where $\mathbf{M}$ is `ec_lds`.
m	*scalar*	`uniform float m`	Specular coefficient (m>0; m<10 ➔ only slightly glossy; m>25 ➔ fairly glossy)

‡ if $L_{i,xyzw}$=(x, y, z, 1); § if $L_{i,xyzw}$=(x, y, z, 0); Alpha (translucency) appended, if applicable, to fragColor in fragment shader.