\[ I_\lambda = k_a I_{\lambda a} O_{\lambda a} + \sum_{i=1}^{k} S_i f_{\text{att}} I_{L_i \lambda} [k_d O_{d \lambda} (\mathbf{N} \cdot \mathbf{L}_i) + k_s (\mathbf{R} \cdot \mathbf{V})^n] \]

\[ S_i = \begin{cases} 
0 & \text{if light } i \text{ is blocked,} \\
1 & \text{if light } i \text{ is not blocked.} 
\end{cases} \]

**Scan-line method**

★ Use light source as the center of projection.
★ Project the edges of polygons that cast shadows on the polygons intersecting the scan line.
★ Whenever the scan line visits one of the projected points, change the intensity.
Two-Pass Object Precision Algorithm:

★ Find the portion of each polygon visible from the light source.

★ Decompose each polygon into subpolygons, each being either completely lit or completely under dark.

★ Render each polygon as follows:
  • If the polygon is in dark, set intensity to the ambient light
    \[ I_\lambda = k_a I_a \lambda O_a \lambda. \]
  • If the polygon is lit, then use ambient, diffuse, and specular reflection.

★ Repeat the first two steps for each light source.
Shadow Generation

Two-Pass Z-Buffer Algorithm:

★ Two passes
  • W.r.t light source
  • W.r.t. view point

★ Compute depth information w.r.t. the light source.
  *light buffer* (LB) or *shadow mask*

★ Compute the value of the frame buffer at each pixel \( \pi \) w.r.t. the viewpoint as follows:
  
  • Suppose the point \( p \) in the world coordinate is drawn at pixel \( \pi \).
  
  • Determine if \( p \) is under shadow.
  
  • If under shadow, use ambient light.

  Otherwise compute the lighting information at \( \pi \).

★ For multiple light sources, maintain a shadow mask for each light source.
Compute the pixel $(a, b)$ in the shadow mask corresponding to point $p$.

Compute the distance $c$ of $p$ from the light source.

Compare $c$ with $z_L = LB[a, b]$.

If $z_L < c$, $p$ is under shadow; otherwise $p$ is lit.