9.4 Polygon-Mesh Rendering Methods

Rendering of graphics objects approximated by polygonal meshes

Constant-Intensity Shading (Flat Shading)

- a single intensity is computed for each polygon
- assumption: light source and viewer are both sufficiently far from the object
- works only if the polygons are small

Intensity Interpolation Shading (Gouraud Shading)

- Eliminate intensity discontinuities
- Basic idea:
  1. compute polygon (unit) normals
  2. compute the average unit normal vector at each polygon vertex
$N_v = \frac{(N_1+ N_2+N_3+N_4)}{4}$

3. compute vertex intensities

4. shade polygon by linear interpolation of vertex intensities along each edge and then between edges along each scan line

- May cause Mach band effect

(bright or dark intensity streaks, occurs when the light intensity on a surface changes abruptly)
Note: interpolation along edges can be integrated with the scan-line hidden surface elimination algorithm

- $I_{\text{min}}$ has to be updated for each new scan line
- If the span $(a, b)$ is visible, it is filled by interpolating the intensity values of $I_a$ and $I_b$
- Need to compute $\Delta I_x$
- The intensity $I_p$ at $p$ is $I_{p-1} + \Delta I_x$
Normal-Vector Interpolation Shading (Phong Shading)

- Interpolates the normal vector $N$ rather than intensity
- Basic idea:
  1. compute polygon normals
  2. compute vertex normals
  3. compute normals of start and end points of a visible span using linear interpolation
  4. compute normal at each pixel along a scan line, then compute intensity using a desired illumination model
Can also be integrated with scan-line method

- Highlights are more faithfully reproduced
- Reduces Mach band problems
9.5 Shadows

- Determines which faces or parts of faces are visible when viewed from the light sources
- Surfaces visible from both the viewpoint and the light source are not in shadow
- Surfaces visible from the viewpoint but not from the light source are in shadow

Basic idea:

1. Each edge of an object that is an outline of the object when viewed from the light source is extended to form a shadow polygon
If several light sources are being simulated then shadow polygons generated by different light sources will be tagged differently.

Shadow polygons are passed to the scan conversion process along with polygons in the scene.

2. As the scan conversion process proceeds, if an even number of shadow polygons with the same tags is encountered, the face is not shadowed by the object and the light source given in the tags. A set of tags encountered an odd times indicates that the face is shadowed by the light source in the tags.
How to determine if an edge is an outline when viewed from the light source?

Draw a plane through $e$ that is parallel to the DOP.

If all the other vertices of the object are on the same side of this plane then $e$ is an edge of the outline.
9.6 Light-Transmitting Surfaces

- specular transmittance: refracted & unrefracted
- diffuse transmittance: not easy to model

Law of refraction

$(n, n'): \text{indices of refraction}$
Using scan-line algorithm (ignoring refraction)

1. Within a **span**, sort the segments by **depth**

2. Find the **opaque** polygon closest to the observer and determine its **shade**

3. This **shade** is then modified by any **transparent segments** that lie between the observer and the **opaque surface**

\[
I = k \cdot I_1 + (1 - k) \cdot I_2
\]

\[0 \leq k \leq 1\]: degree of transparency of \(P_1\)
9.7 Ray-Tracing Method (Whitted & Kay)

- The tracing starts at the **viewpoint** and traces rays **backwards** through each **pixel** to their origin.
- Only rays from **specular reflection** and **refraction** are traced.
- A **tree** structure is needed.

\[ S_i: \text{specularly reflected ray} \]
\[ T_i: \text{specularly refracted ray} \]
Each node of the tree corresponds to a surface

After the tree is completely grown, the intensities at each leaf node are computed and then used to compute the intensity at their parent nodes, until the root node is reached.

Notes:

1. Only those rays that pass through the viewpoint and the corners of pixels are traced
2. Ray-object intersection test can be expedited by the use of 3D extents
3. VLSI implementations for ray tracing maybe expected.