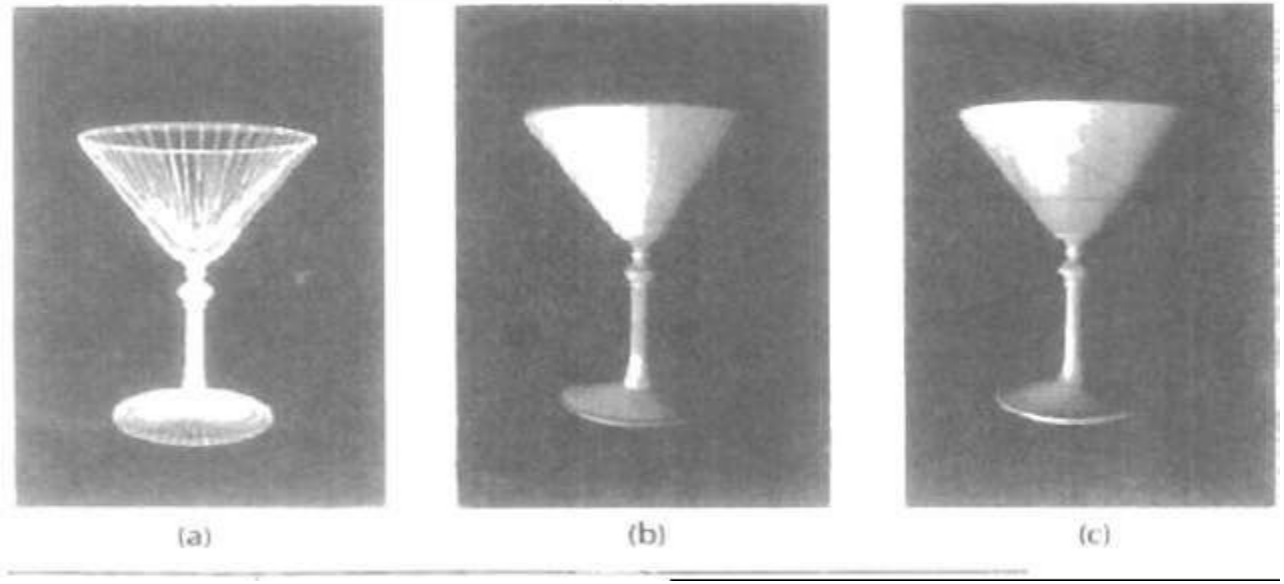
Shading

“ Shading is referred as the *implementation of illumination model* at the pixel points or polygon surfaces of the graphics objects.”

Constant-Intensity Shading

A fast and simple method for rendering an object with polygon surfaces is constant intensity shading, also called flat shading. In this method, a single intensity is calculated for each polygon. All points over the surface of the polygon are then displayed with the same intensity value. Constant shading can be useful for quickly displaying the general appearance of a curved surface, as in Fig(1).



Fig(1). A polygon mesh approximation of an object (a) is rendered with flat shading (b) and With Gouraud shading *(c).*

In general, flat shading of polygon facets provides an accurate rendering for an object if all of the following assumptions are valid: -

2. The object is a polyhedron and is not an approximation of an object with a curved surface.
3. All light sources illuminating the objects are sufficiently far from the surface so that N . L and the attenuation function are constant over the surface(where N is the unit normal to a surface and L is the unit direction vector to the point light source from a position on the surface).
4. The viewing position is sufficiently far from the surface so that V . R is constant over the surface.(where V is the unit vector pointing to the viewer from the surface position and R represent unit vector in the direction of ideal specular reflection).

Gouraud Shading

This intensity-interpolation scheme, developed by Gouraud and generally referred to as Gouraud shading, renders a polygon surface by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud shading by performing the following calculations:

1. Determine the average unit normal vector at each polygon vertex.
2. Apply an illumination model to each vertex to calculate the vertex intensity.
3. Linearly interpolate the vertex intensities over the surface of the polygon.

At each polygon vertex, we obtain a normal vector by averaging the surface normals of all polygons starting that vertex, as illustrated in Fig. (2).

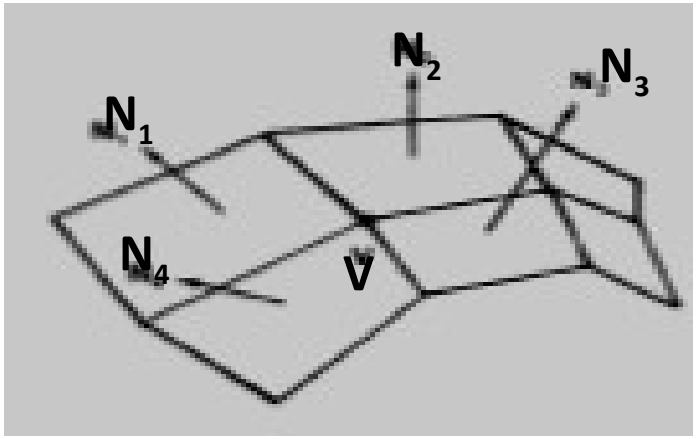


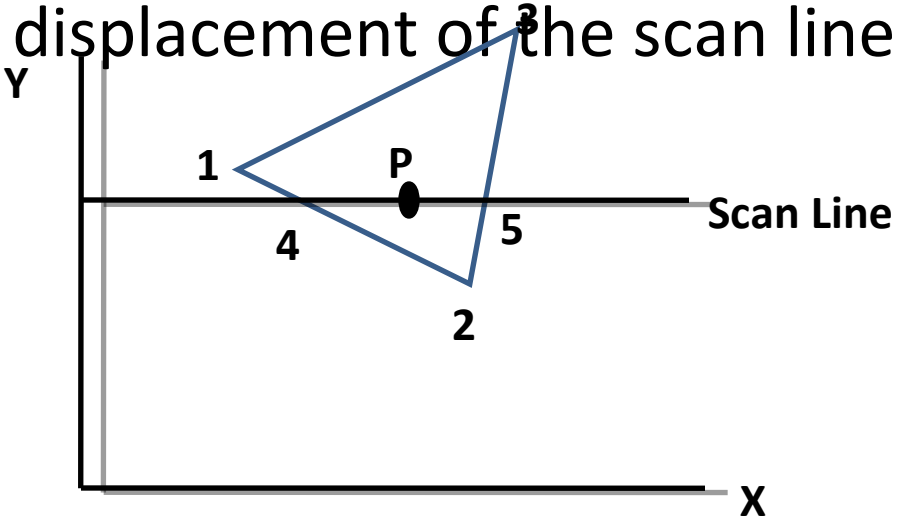
Figure (2).
The normal vector at vertex V is calculated as the average of the surface normals for each polygon sharing that vertex.

Thus, for any vertex position V, we obtain the unit vertex normal with the calculation

$$\mathbf{N}_V = \frac{\sum_{k=1}^{n} \mathbf{N}_k}{\left| \sum_{k=1}^{n} \mathbf{N}_k \right|}$$

Once we have the vertex normals, we can determine the intensity at the vertices from a lighting model.

Following Figure(3)demonstrates the next step: interpolating intensities along the polygon edges. For each scan line, the intensity at the intersection of the scan line with a polygon edge is linearly interpolated from the intensities at the edge endpoints. For the example in Fig.(3), the polygon edge with endpoint vertices at positions 1 and 2 is intersected by the scan line at point 4. A fast method for obtaining the intensity at point 4 is to interpolate between intensities I_1, and I_2 using only the vertical displacement of the scan line:



Fig(3). For Gouraud shading, the intensity at point 4 is linearly interpolated from the intensities at vertices 1 and2. The intensity at point 5 is linearly interpolated from intensities at vertices 2 and 3. An interior point p is then assigned an intensity value that is linearly interpolated from intensities at positions 4 and 5.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Similarly, intensity at the right intersection of this scan line (point 5) is interpolated from intensity values at vertices 2 and 3. Once these bounding intensities are established for a scan line, an interior point (such as point P in Previous Fig(3)) is interpolated from the bounding intensities at points 4 and 5 as

$$I_P = \frac{X_5 - X_P}{X_5 - X_4} I_4 + \frac{X_P - X_4}{X_5 - X_4} I_5$$

Incremental calculations are used to obtain successive edge intensity values between scan lines and to obtain successive intensities along a scan line. As shown in following Fig.(4),

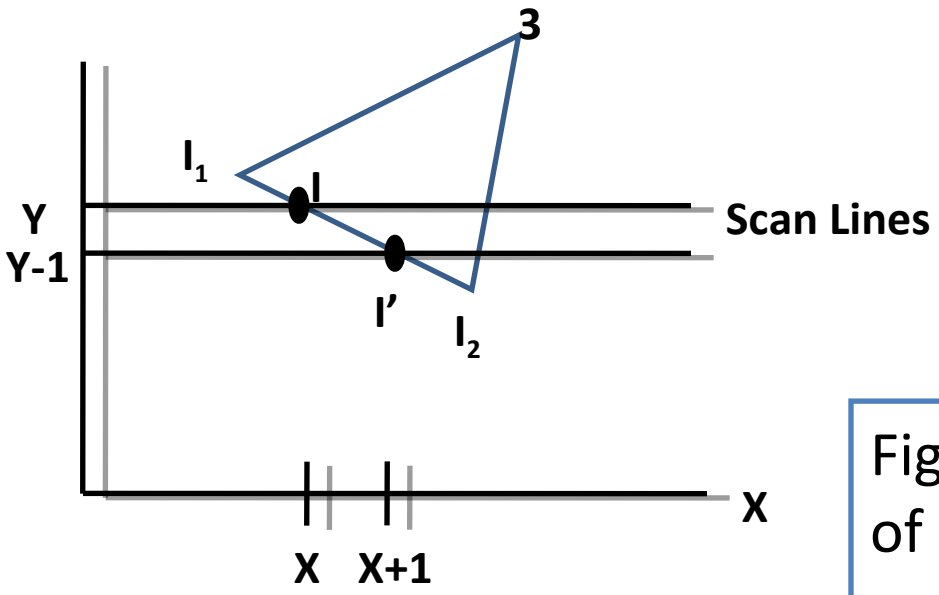


Fig.(4) Incremental interpolation of intensity values along a polygon edge for successive scan lines.

if the intensity at edge position (x,y) is interpolated as

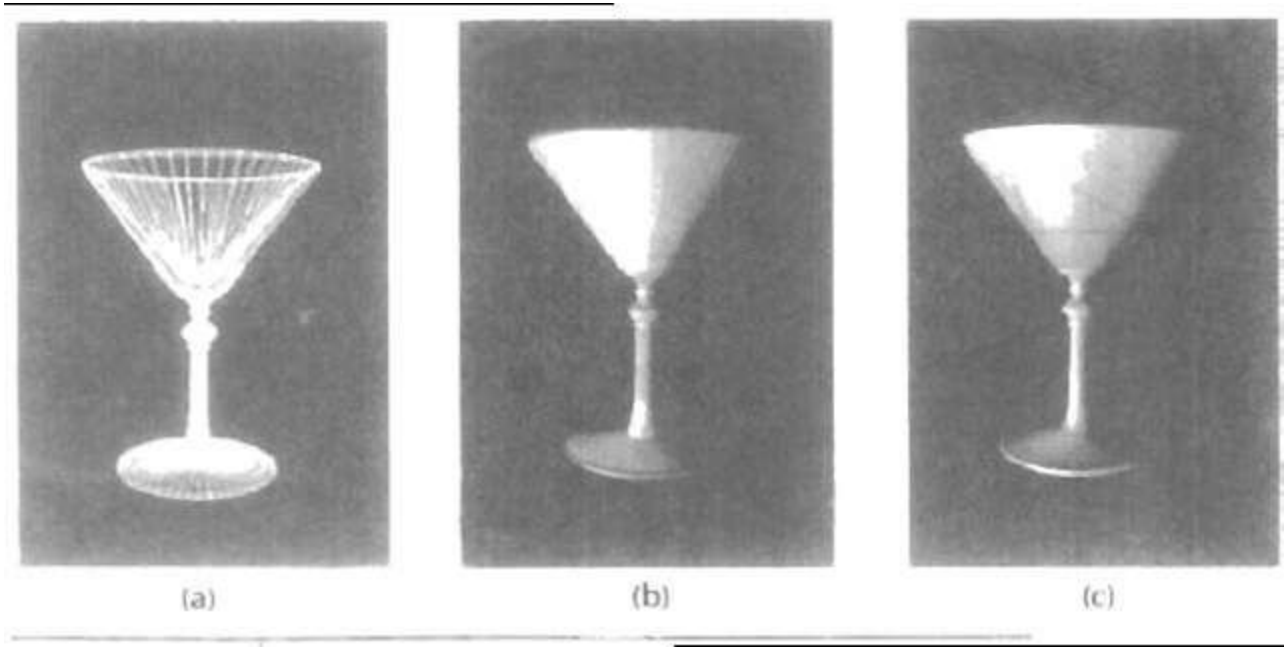
$$I = \frac{Y-Y_2}{Y_1-Y_2} I_1 + \frac{Y_1-Y}{Y_1-Y_2} I_2$$

then we can obtain the intensity along this edge for the next scan line, y - I, as

$$I' = I + \frac{I_2-I_1}{Y_1-Y_2}$$

Similar calculations are used to obtain intensities at successive horizontal pixel positions along each scan line.

When surfaces are to be rendered in color, the intensity of each color component is calculated at the vertices. Gouraud shading can be combined with a hidden-surface algorithm to fill in the visible polygons along each scan line. An example of an object shaded with the Gouraud method appears in following Fig.



A polygon mesh approximation of an object (a) is rendered with flat shading (b) and With Gouraud shading *(c).*

Gouraud shading removes the intensity discontinuities associated with the constant-shading model, but it has some other deficiencies. Highlights on the surface are sometimes displayed with anomalous shapes, and the linear intensity interpolation can cause bright or dark intensity streaks, called Mach bands, to appear

on the surface. These effects can be reduced by dividing the surface into a greater number of polygon faces or by using other methods, such as Phong shading, that require more calculations.

Phong Shading

A more accurate method for rendering a polygon surface is to interpolate normal vectors, and then apply the illumination model to each surface point. This method, developed by Phong Bui Tuong, is called Phong shading, or normal vector interpolation shading. It displays more realistic highlights on a surface and greatly reduces the Mach-band effect.

A polygon surface is rendered using Phong shading by carrying out the following steps:

1. Determine the average unit normal vector at each polygon vertex.
2. Linearly & interpolate the vertex normals over the surface of the polygon.
3. Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

Interpolation of surface normals along a polygon edge between two vertices is illustrated in Fig.(5). The normal vector N for the scan-line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals:

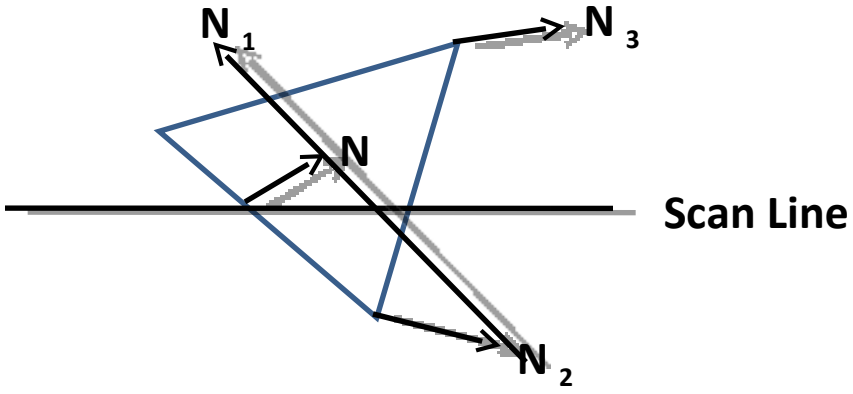


Fig.(5)
Interpolation of surface normals Along a polygon edge.

$$N = \frac{Y - Y_2}{Y_1 - Y_2} N_1 + \frac{Y_1 - Y}{Y_1 - Y_2} N_2$$

Incremental methods are used to evaluate normals between scan lines and along each individual scan line. At each pixel position along a scan line, the illumination model is applied to determine the surface intensity at that point.

Intensity calculations using an approximated normal vector at each point along the scan line produce more accurate results than the direct interpolation of intensities, as in Gouraud shading. The trade-off, however, is that Phong shading requires considerably more calculations.

Fast Phong Shading

Surface rendering with Phong shading can be speeded up by using approximations in the illumination-model calculations of normal vectors. Fast Phong shading approximates the intensity calculations using a Taylor-series expansion and triangular surface patches.

Since Phong shading interpolates normal vectors from vertex normals, we can express the surface normal N at any point (x, y) over a triangle as

$$N=Ax+By+C$$

where vectors A, B, and C are determined from the three vertex equations:

$$N_k=Ax_k+By_k+C, \qquad k=1,2,3$$

with (x_k,y_k) denoting a vector position.

Omitting the reflectivity and attenuation parameters, we can write the calculation for light-source diffuse reflection from a surface point (x, y) as

$$I_{diff}(x,y) = \frac{L.N}{|L||N|}$$

$$= \frac{L.(Ax+By+C)}{|L||Ax+By+C|}$$

$$= \frac{(L.A)x+(L.B)y+(L.C)}{|L||Ax+By+C|}$$

We can rewrite this expression in the form

$$I_{diff}(x,y)= \frac{ax+by+c}{(dx^2+exy+fy^2+gx+hy+i)^{1/2}} \quad \text{eq(1)}$$

where parameters such as a, b, c, and d are used to represent the various dot products. For example

$$a= \frac{L.A}{|L|}$$

Finally, we can express the denominator in Eq. (1) as a Taylor-series expansion and retain terms up to second degree in x and y. This yields

$$I_{diff}(x,y)= T_5x^2+T_4xy+T_3y^2+T_2x+T_1y+T_0 \quad \text{eq(2)}$$

where each T_k, is a function of parameters a, b, c, and so forth.

Using forward differences, we can evaluate Eq. (2) with only two additions for each pixel position (x, y) once the initial forward-difference parameters have been evaluated. Although fast Phong shading reduces the Phong-shading calculations, it still takes approximately twice as long to render a surface with fast Phong shading as it does with Gouraud shading. Normal Phong shading using forward differences takes about six to seven times longer than Gouraud shading.

Fast Phong shading for diffuse reflection can be extended to include specular reflections. Calculations similar to those for diffuse reflections are used to evaluate specular terms such as $(N.H)^{ns}$ in the basic illumination model. In addition, we can generalize the algorithm to include polygons other than triangles and finite viewing positions.