4

VISIBLE SURFACE DETERMINATION AND ILLUMINATION [12HR]

4.1 Visible-surface determination

Visible surface detection or Hidden surface removal is major concern for realistic graphics for identifying those parts of a scene that are visible from a choosen viewing position. Several algorithms have been developed. Some require more memory, some require more processing time and some apply only to special types of objects.

Visible surface detection methods are broadly classified according to whether they deal with objects or with their projected images. These two approaches are :

1. Object Space Methods: Compares objects and parts of objects to each other within the scene definition to determine which surface as a whole we should label as visible.

2. Image-space methods: Visibility is decided point by point at each pixel position on the projection plane.

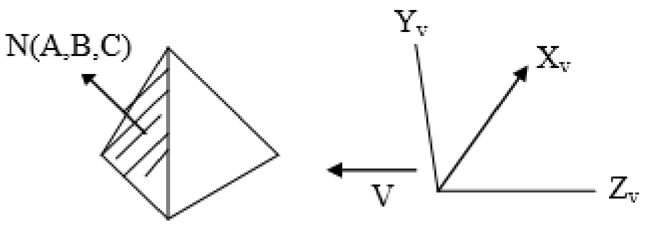
Most visible surface detection algorithm use image-space method but in some cases object space methods are also used for it.

Back Face Detection (Plane Equation Method):

A fast and simple object space method used to remove hidden surface from a 3D object drawing is known as "Plane equation method" and applied to each side after any rotation of the object takes place. It is commonly known as back-face detection of a polyhedron is based on the "inside-outside" tests.
A point (x,y,z) is inside a polygon surface if, $Ax+By+Cz+D < 0$

We can simplify this test by considering the normal vector N to a polygon surface which has Cartesian components (A, B, C).

If V is the vector in viewing direction from the eye position then this polygon is a back face if, $V.N > 0$.



In the equation $Ax+By+Cz+D = 0$, if A, B, C remains constant, then varying value of D results in a whole family of parallel planes. One of which (D=0) contains the origin of the co-ordinates system and,
If D>0, plane is behind the origin (Away from observer)
If D<0, plane is in front of origin (towards the observer)

If we clearly defined out object to have centered at origin, the all those surface that are viewable will have negative D and unviewable surface have positive D.

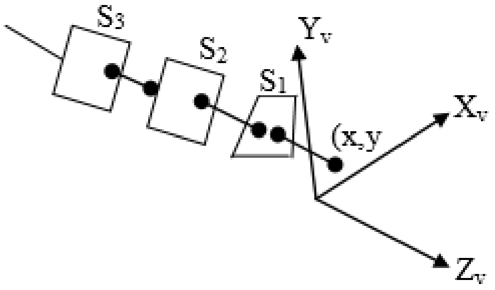
So, simply our hidden surface removal routine defines the plane corresponding to one of 3D surface from the co-ordinate of 3 points on it and computing D, visible surface are detected.

Depth-Buffer Method:
Depth Buffer Method is the commonly used image-space method for detecting visible surface. It is also know as z-buffer method. It compares surface depths at each pixel position on the projection plane. It is called z-buffer method since object depth is usually measured from the view plane along the z-axis of a viewing system.

Each surface of scene is processed separately, one point at a time across the surface. The method is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and method is easy to implement. This method can be apply to non planer surfaces.

With object description converted to projection co-ordinates, each (x,y,z) position on polygon surface corresponds to the orthographic projection point (x,y) on the view plane. Therefore for each pixel postion (x,y) on the view plane, object depth is compared by z values.

With object description converted to projection co-ordinates, each (X,Y,Z) positon on polygon surface correspond to the orthographic projection point (X,Y) on the view plane. The object depth is compared by Z-values.



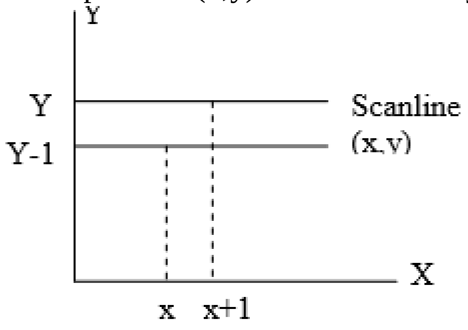
In figure, three surface at varying distance from view plane XvYv, the projection along (x,y) surface S1 is closest to the view-plane so surface intensity value of S1 at (x,y) is saved.

In Z-buffer method, two buffers area are required. A depth buffer is used to store the depth value for each (x,y) position or surface are processed, and a refresh buffer stores the intensity value for each position. Initially all the position in depth buffer are set to 0, and refresh buffer is initialize to background color. Each surface listed in polygon table are processed one scan line at a time, calculating the depth (z-val) for each position (x,y). The calculated depth is compared to the value previously stored in depth buffer at that position. If calculated depth is greater than stored depth value in depth buffer, new depth value is stored and the surface intensity at that position is determined and placed in refresh buffer.

Algorithm:

```
1. Initialize depth buffer and refresh buffer so that for all buffer position (x,y), depth(x,y) = 0, refresh
   (x,y) = Ibackground
2. For each position on each polygon surface, compare depth values to previously stored value in
   depth buffer to determine visibility.
     a. Calculate the depth Zfor each (x,y) position on polygon.
     b. If Z > depth (x,y) then depth (x,y) = Z
        refresh (x,y) = Isurface(x,y)
```

Where, Ibackground is the intensity value for background and Isurface (x,y) is intensity value for surface at pixel position (x,y) on projected plane. After all surface are processed, the depth buffer contains the depth value of the visible surface and refresh buffer contains the corresponding intensity values for those surface. The depth value of the surface position (x,y) are calculated by plane equation of surface.



$$Z = \frac{-Ax - By - D}{C}$$

Let Depth Z' at position (x+1,y)

$$Z' = \frac{-A(x+1) - By - D}{C}$$

$$\Rightarrow Z' = Z - A/C \quad \text{———} \quad (1)$$

- A/C is constant for each surface so succeeding depth value across a scan line are obtained from preceding values by simple calculation.

Advantages:

- No presorting and object-object comparisons are required .
- Time taken by visible surface calculation is constant .
- It is simple and easy implementation hardware.
- If memory is at permission scan converted in strips.
- Good for animation.

Disadvantages:

- Requires amount of memory for z and frame buffer.
- It is subject to aliasing.

Scan Line Method:

- The scan line method solve the hidden surface problem one scan line at a time.
- Processing of the scan line start from the top to the bottom of the display.
- This method calculates, the z-value for only the overlapping surface which is tested by scan line.
- It request edge table, polygon table/surface table active edge list and flag.

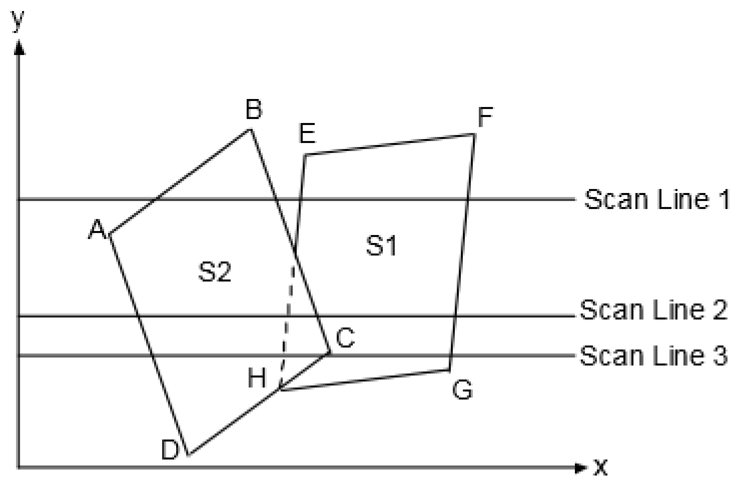


Fig: Scan lines crossing the projection of two surfaces S_1 and S_2 in the view plane. Dashed line indicate the boundaries of hidden surfaces.

Edge table contents:

i) Coordinate end points for each scan line.
ii) Pointers in the polygon table to identify the surfaces bounded by each line.
iii) Inverse slope of each line.

Polygon table contents:

i) Coefficients of plane equations for each surfaces.
ii) Pointers into edge table.
iii) Intensity information of the surfaces.

Active edge list contains edges that cross the current scan line, shorted in order of increasing x.
Flag is defined for each surface that is set 'ON' or 'OFF' to indicate if a position alone a scan line is inside or outside of the surface at left most boundary surface flag is 'ON' and at right most flag is 'OFF'.

Above figure illustrate the scan line method for locating visible portion of surfaces for pixel position along the line. The active list for scan line one contains information form the edge table for edges AB , BC, EH and FG.

For position along this scan line between edges AB and BC, flag (S1) = ON and flag (S2) = OFF.

Therefore no depth calculation are necessary and intensity information for surface S1 is entered from the polygon table into the refresh buffer.

Similarly , between edge EH and FG Flag (S1) = OFF and flag (S2) = ON

On other position along scan line 1intersect surfaces, so the intensity values in the other areas are set to the background intensity. The background intensity can be loaded throughout the buffer in an initialization routine.

For scan line 2 and 3 figure, the active edge list contains edges AD, EH, BC and FG.

Along the scan line 2, from edge AD to edge EH, flag (S1)= ON and flag (S2)= OFF Between edges EH and BC, Flag (S1)= ON and flag (S2) = ON

Therefore depth calculation must be made using the plane coefficient for the two surfaces.

For this example the depth of surface S1 is assumed to be less than that of S2, So intensity for surface S1 are loaded into the refresh buffer until boundary BC is encountered. Then the flag for the surface S1 goes off. And intensity for surface S2 are stored until edge FG is passed.

For the scan line 3 the active edge list contains the same edge as scan line 2.

Since no changes have occurred in line intersection, it is unnecessary again to make depth calculation between edges EH and BC. The surfaces must be in the same orientation as determined on scan line 2. So the intensity for the surface S1 can be entered without further calculation.

Any number of overlapping polygon surfaces can be processed with this scan line method. Flag for the surfaces are set to indicate whether a position is inside or outside, and depth calculation are performed when surfaces overlap.

<u>Algorithm:</u>

1. Established data structure.
   a. Edge table with line endpoints, inverse slope to polygon pointers.
   b. Polygon table with plane coefficient, colour and pointer to edge table.
   c. Active edge list sorted in increasing order of x.
   d. A flag for each surface.
2. Repeat for each scan line.
   a. Update AEL for scan line y
   b. Scan line across light using background color until flag goes ON.
   c. When one flag is ON , enter intensity of that surface to refresh buffer.
   d. When two or more flags are ON, do depth sorting and stored the intensity of pixel nearest to the view plane.

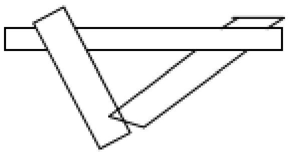
Depth Sorting Method:
This method uses both object space and image space methods. In this method the surface representation of 3D object are sorted in of decreasing depth from viewer. Then sorted surface are scan converted in order starting with surface of greatest depth for the viewer.

The conceptual steps that performed in depth-sort algorithm are:

- Surfaces are sorted in order of decreasing depth (z-coordinate).
- Resolve any ambiguity this may cause when the polygons z-extents overlap, splitting polygons if necessary.
- Surfaces are scan converted in order, starting with the surface of greatest depth.

This algorithm is also called "Painter's Algorithm" as it simulates how a painter typically produces his painting by starting with the background and then progressively adding new (nearer) objects to the canvas.

Problem:
One of the major problems in this algorithm is intersecting polygon surfaces. As shown in fig. below.



- Different polygons may have same depth.

- The nearest polygon could also be farthest.
- We cannot use simple depth-sorting to remove the hidden-surfaces in the images.

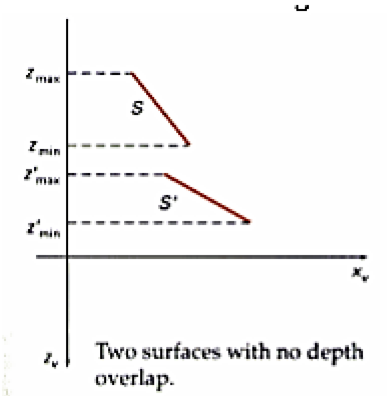
Solution:
For intersecting polygons, we can split one polygon into two or more polygons which can then be painted from back to front. This needs more time to compute intersection between polygons. So it becomes complex algorithm for such surface existence.

Example:
Assuming we are viewing along the z axis. Surface S with the greatest depth is then compared to other surfaces in the list to determine whether there are any overlaps in depth. If no depth overlaps occur, S can be scan converted. This process is repeated for the next surface in the list. However, if depth overlap is detected, we need to make some additional comparisons to determine whether any of the surfaces should be reordered.

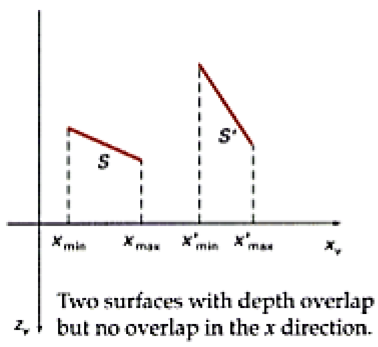
We make the following tests for each surface that overlaps with S. If any one of these tests is true, no reordering is necessary for that surface. The tests are listed in order of increasing difficulty.

1. The bounding rectangles in the xy plane for the two surfaces do not overlap
2. Surface S is completely behind the overlapping surface relative to the viewing position.
3. The overlapping surface is completely in front of S relative to the viewing position.
4. The projections of the two surfaces onto the view plane do not overlap.

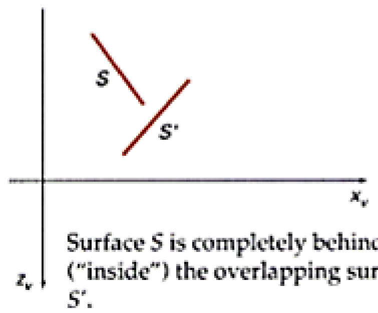
We perform these tests in the order listed and proceed to the next overlapping surface as soon as we find one of the tests is true. If all the overlapping surfaces pass at least one of these tests, none of them is behind S. No reordering is then necessary and S is scan converted.



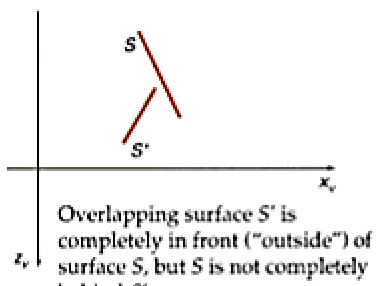
Two surfaces with no depth overlap.



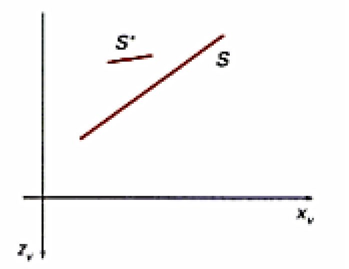
Two surfaces with depth overlap but no overlap in the x direction.



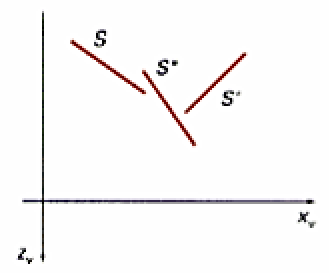
Surface S is completely behind ("inside") the overlapping surface S'.



Overlapping surface S' is completely in front ("outside") of surface S, but S is not completely behind S'.



Surface S has greater depth but obscures surface S'.



Three surfaces entered into the sorted surface list in the order S, S', S" should be reordered S', S", S.

BSP Tree Method:
A binary space partitioning (BSP) tree is an efficient method for determining object visibility by painting surfaces onto the screen from back to front as in the painter's algorithm. The BSP tree is particularly useful when the view reference point changes, but object in a scene are at fixed position.

Applying a BSP tree to visibility testing involves identifying surfaces that are "inside" or "outside" the partitioning plane at each step of space subdivision relative to viewing direction. It is useful and efficient for calculating visibility among a static group of 3D polygons as seen from an arbitrary viewpoint.

In the following figure,

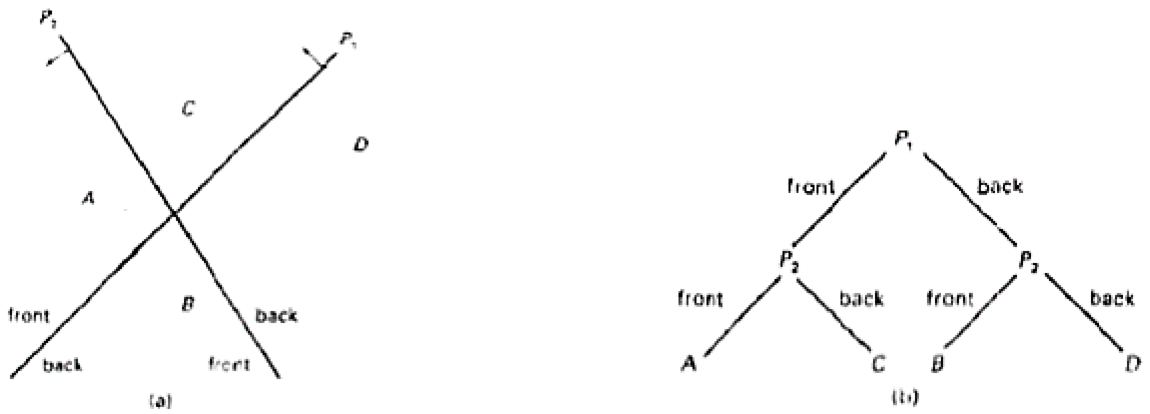


Fig: A region of space (a) Partitioned with two planes P_1 and P_2 to form the BSP tree representation in (b)

- Here plane P1 partitions the space into two sets of objects, one set of object is back and another set is in front of partitioning plane relative to viewing direction. Since one object is intersected by plane P1, we divide that object into two separate objects labeled A and B. Now object A & C are in front of P1, B and D are back of P1.
- We next partition the space with plane P2 and construct the binary free as fig (b). In this tree, the objects are represented as terminal nodes, with front object as left branches and behind object as right branches.
- When BSP tree is complete, we process the tree by selecting surface for displaying in order back to front. So foreground object are painted over back ground objects.

Octree Method:
When an octree representation is used for viewing volume, hidden surface elimination is accomplished by projecting octree nodes into viewing surface in a front to back order. Following figure is the front face of a region space is formed with octants 0, 1, 2, 3. Surface in the front of these octants are visible to the viewer. The back octants 4, 5, 6, 7 are not visible. After octant sub-division and construction of octree, entire region is traversed by depth first traversal.

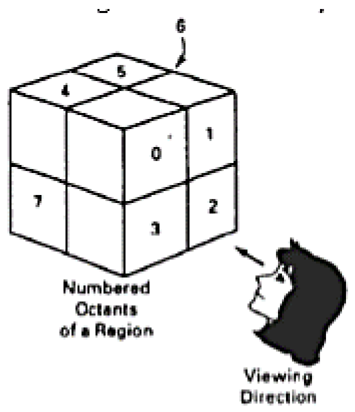


Fig1: Objects in octants 0, 1, 2 and 3 obscure objects in the back octants (4,5,6,7) when the viewing direction is as shown.

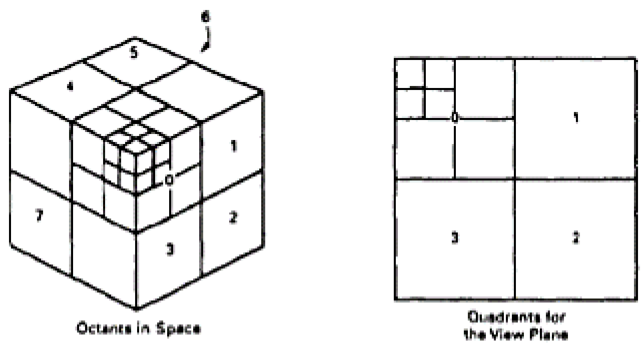


Fig2: Octant divisions for a region of space and the corresponding quadrant plane.

Different views of objects represented as octrees can be obtained by applying transformations to the octree representation that reorient the object according to the view selected.

Fig2 depicts the octants in a region of space and the corresponding quadrants on the view plane. Contributions to quadrant 0 come from octants 0 and 4. Color values in quadrant 1 are obtained from surfaces in octants 1 and 5, and values in each of the other two quadrants are generated from the pair of octants aligned with each of these quadrants.

Ray Casting (Ray Tracing):

Ray tracing also known as ray casting is efficient method for visibility detection in the objects. It can be used effectively with the object with curved surface. But it is also used for polygon surfaces.

- Trace the path of an imaginary ray from the viewing position (eye) through viewing plane t object in the scene.
- Identify the visible surface by determining which surface is intersected first by the ray.
- Can be easily combined with lightning algorithms to generate shadow and reflection.
- It is good for curved surface but too slow for real time application.

Ray casting, as a visibility detection tool, is based on geometric optics methods, which trace the paths of light rays. Since there are an infinite number of light rays in a scene and we are interested only in those rays that pass through pixel positions, we can trace the light-ray paths backward from the pixels through the scene. The ray-casting approach is an effective visibility-detection method for scenes with curved surfaces, particularly spheres.

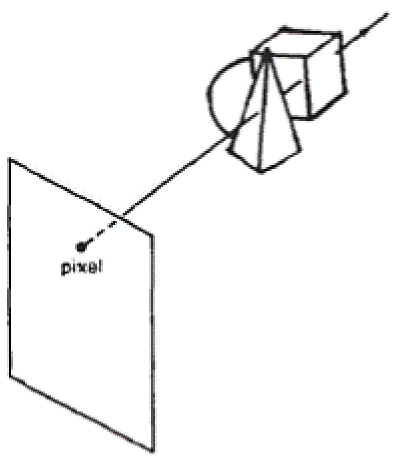


Fig: A ray along the line of sight from a pixel position through a scene.

- In ray casting, we process pixels one at a time and calculate depths for all surfaces along the projection path to that pixel.
- In fact, Ray casting is a special case of ray-tracing algorithms that trace multiple ray paths to pick up global reflection and refraction contributions from multiple objects in a scene. With ray casting, we only follow a ray out from each pixel to the nearest object.

4.2 Illumination and Shading

The realism of a raster scan image of a 3D scene depends upon the successful stimulation of shading effects.

Once visible surface has been identified by hidden surface algorithm , a shading model is used to compute the intensities and color to display for the surface.

The Shading model does not precisely stimulate the behaviour of light and surfaces in the real world but only approximation actual condition.

The shading model has two components.
1. Properties of surfaces
2. Properties of illumination falling on it.

The principle surface property is its reflectance which determine how much of the incident light is reflected.

- If a surface has different reactance for light of different wavelength, it will appear to be color.
- If a surface is texture or has a patterned printed on it, the reflectance will vary with position on the surface.
- Another surface property that play a role in shaded picture is transparency.

Light Source:

When we view a opaque non-luminous object, we see reflected light from the surface of the object. The total reflected light is the sum of the contribution from the light source and other reflective surfaces as shown in the figure:

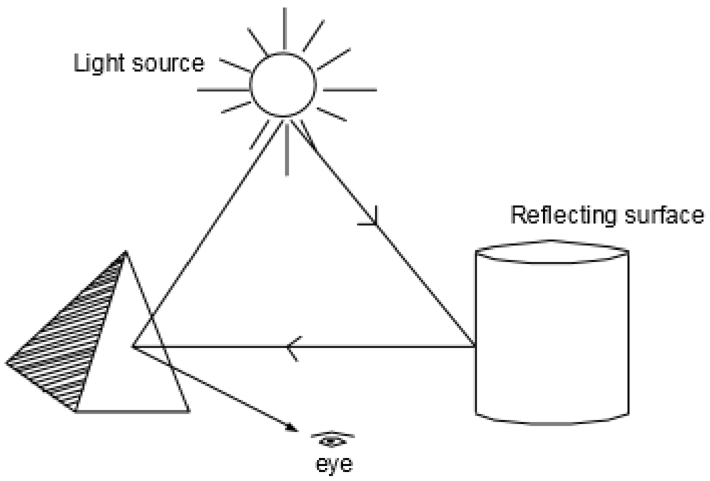
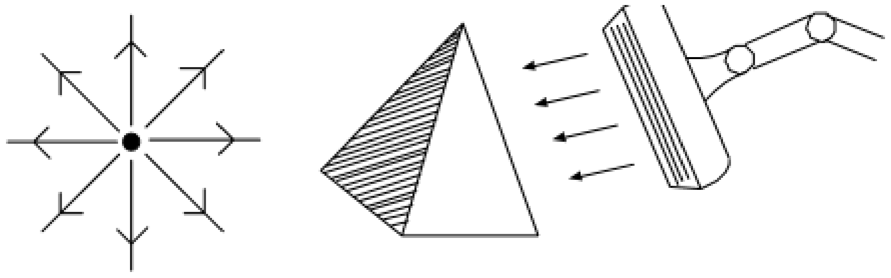


Fig: Light viewed from an opaque non luminous surface is in general a combination of reflected light from a light source and reflection of light reflection from other surfaces.

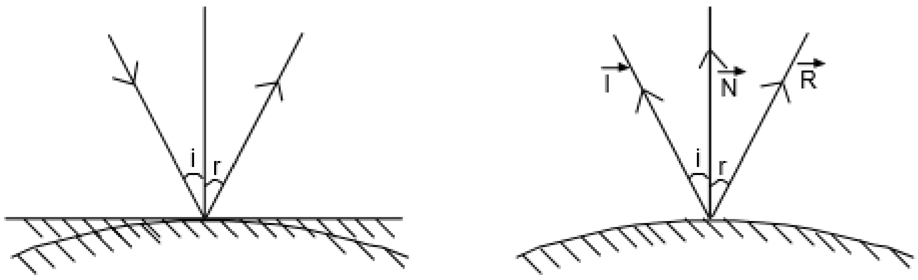
- Thus a surface that is not directly exposed to light sources may still be visible if near by objects are eliminated.
- Sometimes light sources are referred to as light emitting source and reflecting surface such as walls of a room are termed light reflecting source.
- A luminous object in general can be both light source and a light reflector. For example: A plastic globe with a light bulb inside both emits and reflects light from the surface of globe.
- A simplest model for a light emitter is a point source.
- A near by source such as long fluorescent light is more accurately modelled as a distributed light source.



Fig(a): Diverging ray paths from a point light source. Fig (b): An object illuminated with a distributed light source

- When light source is incident on a opaque surface, part of it is reflected and part of it is absorbed.
- The amount of incident light reflected by a surface depends on the type of material.
- Shining material reflects more incident light and dull surface absorbs more of the incident light.
- Surface to produce a uniform illumination is called the ambient light or background.

Reflection of light:



In ideal reflection I = r

Let $\vec{L}$ be the unit vector along incident ray. $\vec{N}$ be the unit vector along normal ray. $\vec{R}$ be the unit vector along reflected ray then,

i=r

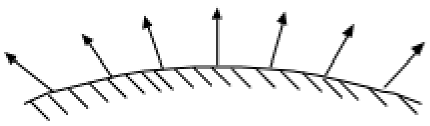
or, cos I = cos r

or $\vec{L}.\vec{N} = \vec{N}.\vec{R}$

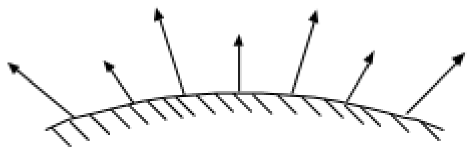
But, practically when light is incident on opaque surface part of it is reflected and part of it is absorbed.

Therefore, I = A + R

Shining material reflects more and dull surface absorb more of incident light. Rough surface tends to scatter the reflected light in all direction. The scattered light is called diffuse reflection. So surface appears equally bright from all viewing directions.

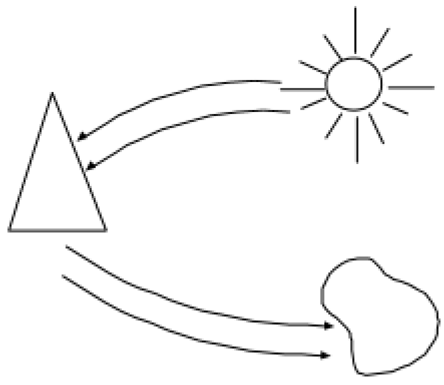


In addition to diffuse refection, light source creates high lights or bright spots called specular reflection. How ever this effect is seen more on shiny surface then dull surfaces. Examples: Persons forehead.



Basic Illumination models:

1. **Ambient Light:**
   Surface that is not exposed directly to a light source still will be visible if near by objects are illuminated. This light is called ambient light.



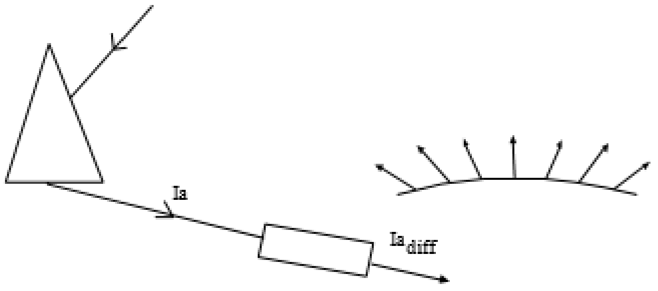
Characteristic of ambient light:

- It provide general level of brightness for all surfaces.
- It has no spatial or directional characteristics.
- It has same intensity for each surface regardless of viewing direction.

If intensity of ambient light = I_a
Then, $I_{amb} = I_a$
If we take surface reflectivity then,
$I_{amb} = I_a$
If we take surface reflectivity then,
$I_{amb} = K_a I_a$
Where K_a is coefficient of abnient reflection.

2. **Diffuse Reflection:**
   Diffuse reflection is constant over each surface in a scene independent of viewing direction.

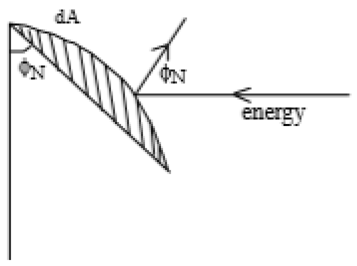
   The intensity of diffuse reflection due to ambient light is : $I_{adiff} = k_d . I_a$
   Where, K_d = diffuse reflection co-efficient For highly reflective surface k_d= 1
   For highly reflective surface k_d = 1
   For low reflective surface k_d = 0



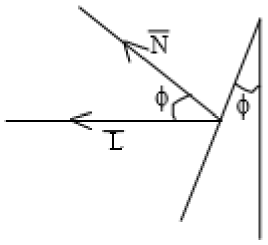
If surface is exposed to a point source we assume diffuse reflections from this surface are scattered with equal intensity in all directions. Such surfaces are also called ideal diffuse reflectors which follows Lambert's Cosine Law.

Lambert's Cosine Law: The radiant energy force from any small surface dA in and direction φ_N relative to surface normal is proportional to $\cos\varphi_N$.

$\therefore$ Light intensity α $\cos\varphi_N$



Although there is equal light scattering in all directions from an ideal diffuse reflection, the brightness of the surface depends on the orientation of the surface related to the light source. If the surface is perpendicular to source it appears brighter.



$\therefore$ Idiff = k_d I_l $\cos\varphi$

= $k_d I_l$ $(\vec{L}.\vec{N})$

$\therefore$ Total diffuse reflection = Diff (due to amb.)+Diff. due to pt.source.

= $k_a I_a$ $+k_d I_L(\vec{L}.\vec{N})$

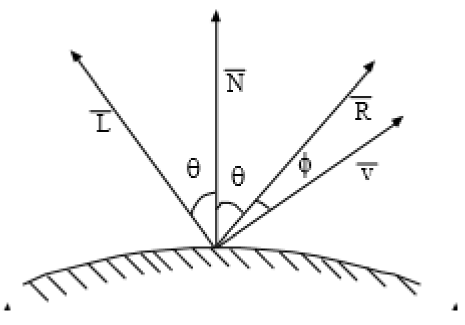
3. **Specular Reflection:**

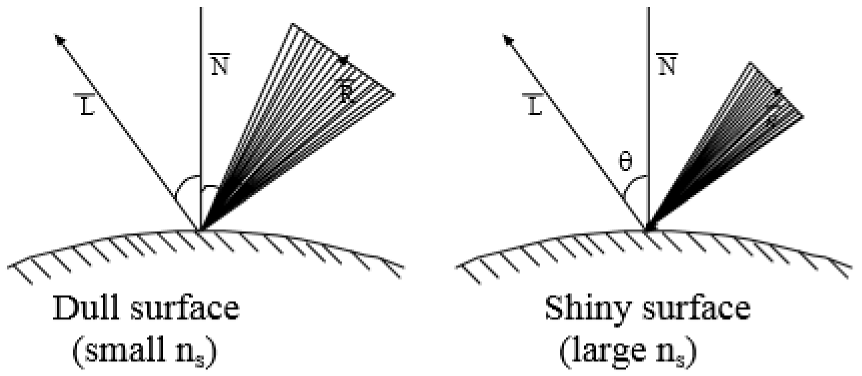
In shiny surface we see high light or bright spot from certain viewing directions called specular reflection. This is due to total or nearly total reflection of light.

For ideal reflector,
(perfect mirror) = $\varphi = 0$

i.e. incident light is reflected only in specular reflection direction. So $\vec{V}$ and $\vec{R}$ coincides.

Objects other than ideal reflection for a finite range of viewing positions around $\vec{R}$. Shiny surfaces have narrow specular reflection range and dull surfaces have a wider range.



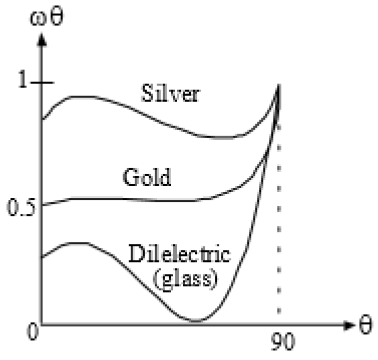


The empirical formula for calculating the specular reflection range is given by phong Model.

4. **Phong Model:** It sets s the intensity of the specular reflection proportional to $\cos^{\eta s}\varphi$, where η_s is specular reflection parameter and is determined by the type of surface that we want to display.

   For very shiny surface ηs is set 100 and for dull surface ηs is set 1. The intensity of specular reflection can be modeled using specular reflection coefficient ω (θ)

   $\therefore$Ispec = ω(θ) $I_L\cos^{\eta s}\varphi$



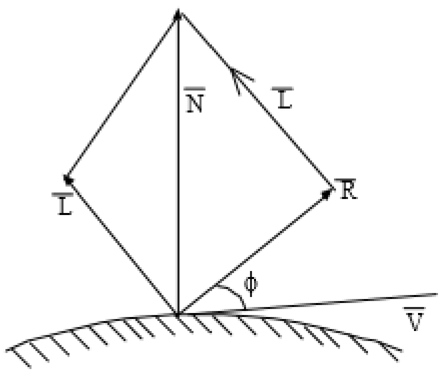
The glass exhibits appreciable specular reflection when. θ → 90°

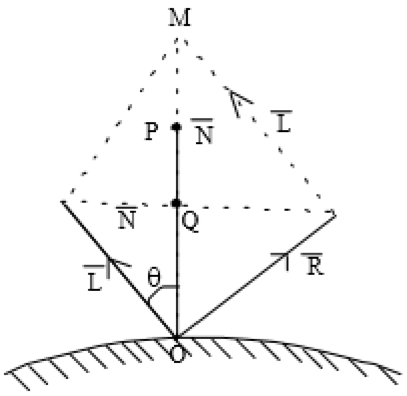
At θ = 0° only 4% of incident light is specularly reflected.

Similarly for gold at θ = 0° , almost 50% of light is specularly reflected and at θ = 90° almost 100% incident light is specularly reflected.
However, for many opaque materials specular reflection is nearly constant for all incident angles. So, we replace ω(θ) by a constant k_s (specular reflection co-efficient).

$\therefore I_{spec} = k_sI_L \cos^{\eta s}\varphi$

$= k_sI_L (\vec{V}\ \vec{R}\ )^{\eta s}$





Expanding R in terms of $\vec{L}$ and $\vec{N}$

From fig. aside,

$\vec{R} + \vec{L} = \overline{OM}$

$= 2\,\overline{OQ}$ (diagonal of ||gm bisect each other)

$= 2|\,\overline{OQ}\,|.\vec{N}$

$= 2$ ox $\cos\theta.\,\vec{N}$

$= 2(\vec{L}.\vec{N}).\,\vec{N}$

$\therefore \vec{R} = 2(\vec{L}.\vec{N}).\,\vec{N} - \vec{L}$

∴ Total intensity is,

$I = I_{amb} + I_{diff} + I_{spec}$

$= K_aI_a + k_dI_L\,(\vec{L}.\vec{N}) + K_sI_L(\vec{V}.\vec{R})^{\eta s}$

Where, $\vec{R} = 2(\vec{L}.\vec{N}) + \vec{N} - \vec{L}$

Polygon rendering Methods:

i) Constant intensity shading method
ii) Gouraud shading method (intensity interpolation)
iii) Phong shading method (Normal vector interpolation).

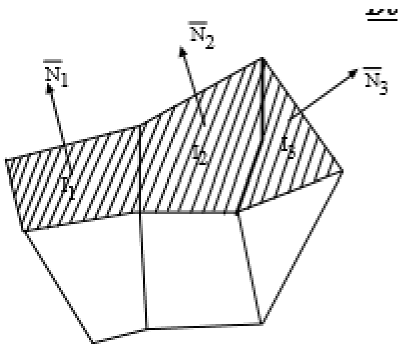
i) Constant Intensity Shading (Flat Shading) Method:

The simplest model of shading for a polygon is constant intensity shading also called as Faceted Shading or flat shading. This approach implies an illumination model once to determine a single intensity value that is then used to render an entire polygon. Constant shading is useful for quickly displaying the general appearance of a curved surface.

This approach is valid if several assumptions are true:

a) This light source is sufficiently for so that N.L is constant across the polygon face.
b) The viewing position is sufficiently far from the surface so that V.R is constant over the surface.
c) The object is a polyhedron and is not an approximation of an object with a curved surface.

Even if all of those conditions are not true, we can still reasonably approximate surface-lighting effects using small polygon facets with flat shading and calculate the intensity for each facet, say, at the center of the polygon.



The intensity is calculated for a point for each surface (usually at the center) and that intensity is applied to all the points of that surface. Hence, all the points over the surface of the polygon are displayed with same intensity value given by,

$I = k_a.I_a + k_a.I_a (\vec{N}.\vec{L}) + k_sI_L (\vec{V}.\vec{R})^{ns}$

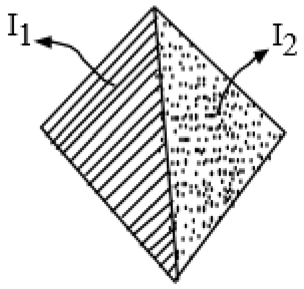
Where, $R = 2 (\vec{N}.\vec{L}). \vec{N} - \vec{L}$

This method is faster and simple but not realistic and holds following assumptions:

a) Object is polyhedron and is not an approximation of an object with a curved surface.
b) All light sources are sufficiently far from the object so that $\vec{N}.\vec{L}$ is constant.
c) The viewing position is sufficiently far so that $\vec{V}.\vec{R}$ is constant.

Disadvantage:

- The intensity discontinuity occurs at the border of the two surfaces.

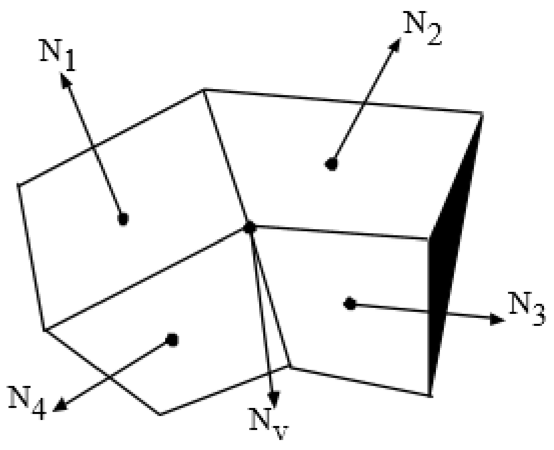


ii) Gouraud Shading:

Gouraud shading, also called intensity interpolating shading or color interpolating shading eliminates intensity discontinuities that occur in flat shading. Each polygon surface is rendered with Gouraud shading by performing 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

Step 1: At each polygon vertex, we obtain a normal vertex by averaging the surface normal of all polygons sharing the vertex as:



At each polygon vertex, we obtain a normal vertex by averaging the surface normals of all polygons sharing the vertex as:

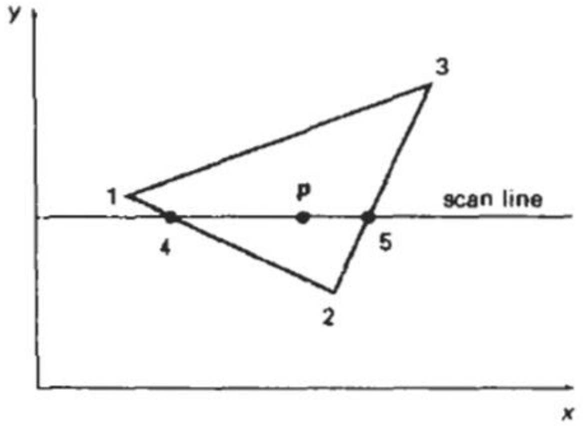
$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

Here in example, $N_v = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$

Where, N_v is normal vector at a vertex sharing 4 surfaces as in figure.

Step 2:
Once we have the vertex normal (N_v), we can determine the intensity at the vertices from a lighting model.

Step 3:
Now to interpolate intensities along the polygon edges, we consider following figure:



In figure, the intensity of vertices 1, 2, 3 are I_1, I_2, I_3, which are obtained by averaging normals of each surface sharing the vertices and applying a illumination model. For each scan line, intensity at intersection of line with Polygon edge is linearly interpolated from the intensities at the edge end point.

So intensity at point 4 is to interpolate between intensities I_1 and I_2 using only the vertical displacement of the scan line:

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

Similarly, the intensity at point 5 is obtained by linearly interpolating intensities at I_2 and I_3 as:

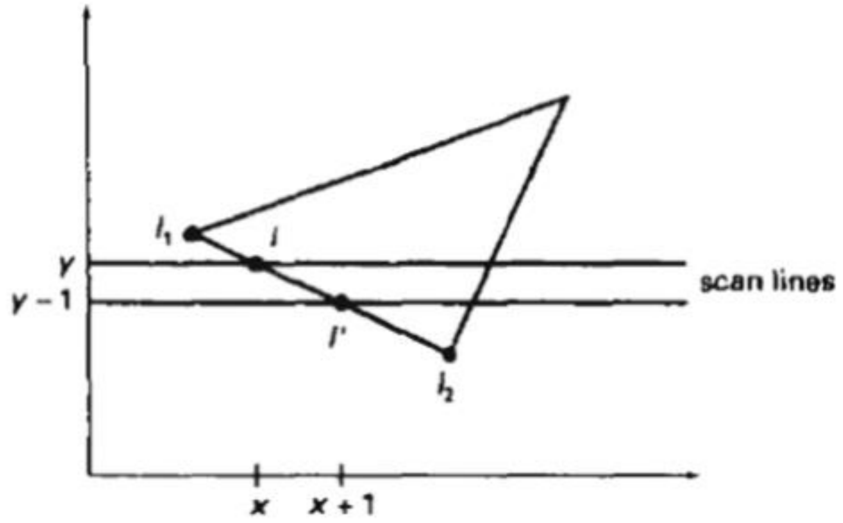
$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

The intensity of a point P in the polygon surface along scan-line is obtained by linearly interpolating intensities at I_4 and I_5 as,

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Then incremental calculations are used to obtain Successive edge intensity values between scan-lines as and to obtain successive intensities along a scan line. As shown in Fig. below, 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 next scan line at y-1 position as:

$$I' = \frac{y - 1 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - (y-1)}{y_1 - y_2} I_2 \quad = I + \frac{I_2 - I_1}{y_1 - y_2}$$

Similar calculations are made to obtain intensity successive horizontal pixel.

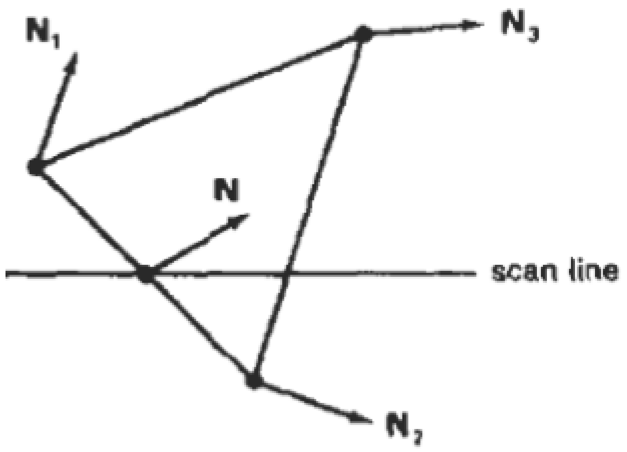
Advantages: Removes intensity discontinuities at the edge as compared to constant shading. **Disadvantages:** Highlights on the surface are sometimes displayed with anomalous shape and linear intensity interpolation can cause bright or dark intensity streak called mach-bands.

iii) Phong Shading:

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

A polygon surface is rendered with phone shading by carrying out following calculations.

- Determine the average normal unit vectors at each polygon vertex.
- Linearly interpolate vertex normal over the surface of polygon.
- Apply illumination model along each scan line to calculate projected pixel intensities for the surface points.



In figure, N_1, N_2, N_3 are the normal unit vectors at each vertex of polygon surface. For scan-line that intersect an edge, the normal vector N can be obtained by vertically interpolating normal vectors of the vertex on that edge as.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Incremental calculations are used to evaluate normals between scan lines and along each individual scan line as in Gouraud shading. Phong shading produces accurate results than the direct interpolation but it requires considerably more calculations.

iv) 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 A, B, C are determined from the three vertex equations.

$$N_k = Ax_k + By_k + C, \; k = 1, 2, 3, \text{ for } (x_k, y_k) \text{ vertex.}$$

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

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

Rewriting this,

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

Where parameters a, b, c, d, …. are used to represent the various dot products as $a = \frac{L.N.}{|L|}$ and so on.

Finally, denominator of equation (1) can be expressed as Taylor series expansions and retains terms up to second degree in x and y. This yields

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

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

This method still takes twice as long as in Gouraud shading. Normal phong shading takes six to seven times that of Gouraud shading.

Exercise

1. Explain in detail about plain equation method. Explain which algorithm is better for hidden surface removal.
2. Explain in detail about depth buffer method. Justify that is better than plane equation method.
3. Why shedding is required in the computer graphics? Explain in detail about constant intensity shading.
4. List the different types of shading models. Explain in detail about Gouraud shading model.
5. Justify that hidden surface removal is required in computer graphics. Explain in detail about depth buffer method.
6. Explain in detail about scan line method. Justify that it is better than depth buffer method.
7. Explain in detail about Phong shading. How can you modify Phong shading model?
8. Explain detail about Gouraud shading model. Compare it with Phong shading model.
9. What do you mean by solid modelling? Explain the process for solid modelling with example.
10. Hidden surface removal is required in computer graphics is very important, justify it. Explain in detail about scan line method.
11. Explain in detail about scan line method. Justify that it is better than plane equation method.
12. Differentiate between diffuse reflection and specular reflection. Why do we require shading model? Explain it.
13. Explain in detail about Phong shading model. Compare it with Gouraud Shading model.
14. Explain the z-buffer algorithm for removing hidden faces.
15. Explain the simple illumination model with example.
16. Explain the Gourand shading model with example.
17. What is solid modelling? Explain the basic procedures for solid modelling.
18. Explain the area subdivision method for visible surface detection.