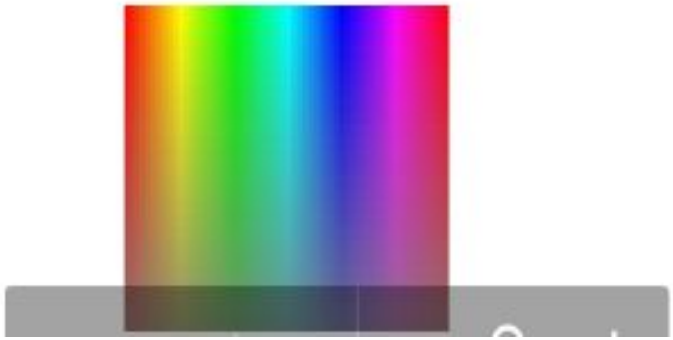
UNIT 4 Light, Color, shading and Hidden surfaces

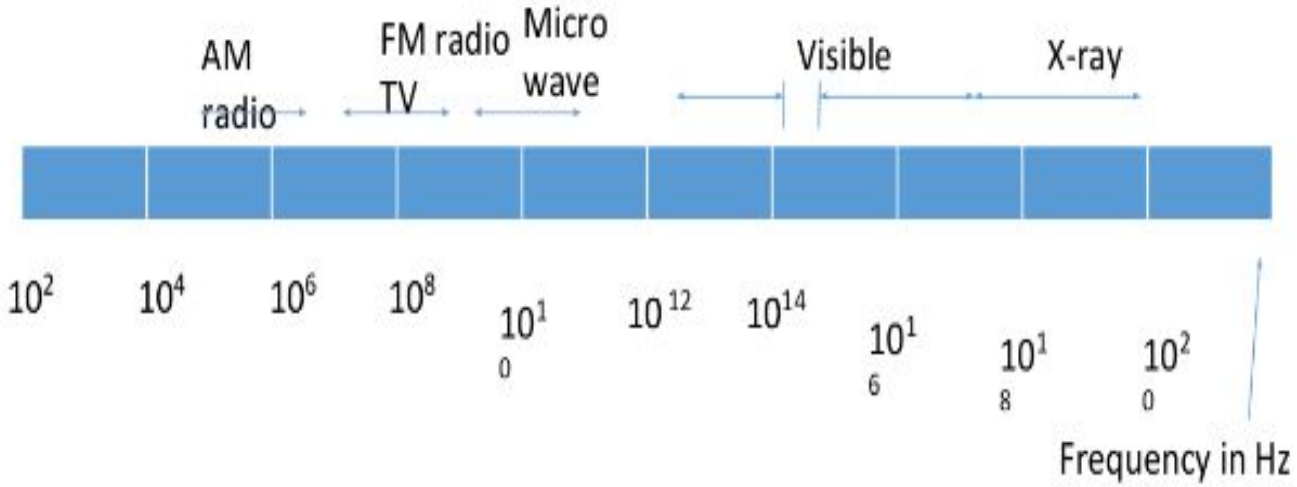
Introduction

- Mechanism for generating **color displays with combination of red , green and blue light.**
- **Color model** will help to understand how color is represented on video monitor, printers and plotters.
- **Color model** is a method for **explaining properties /behavior of color within some particular context.**
- **Each color model explains different properties**

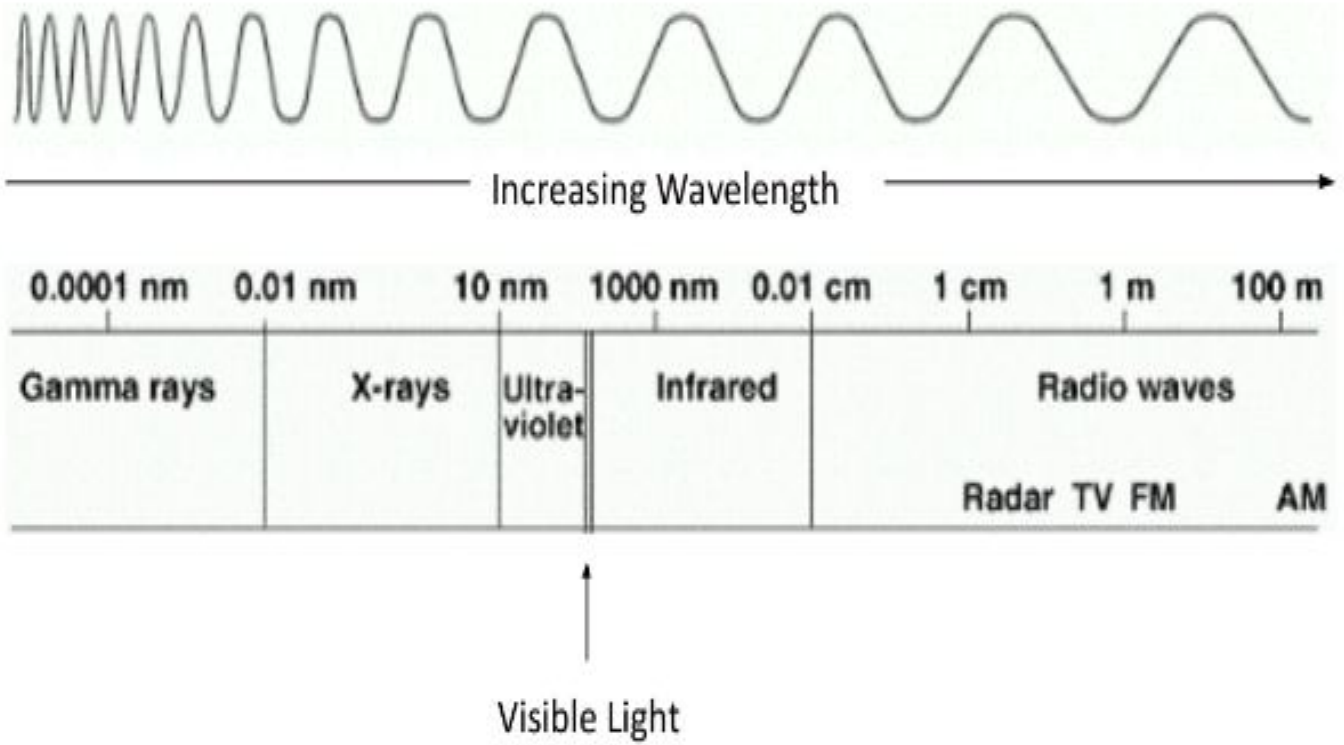


Properties of Light

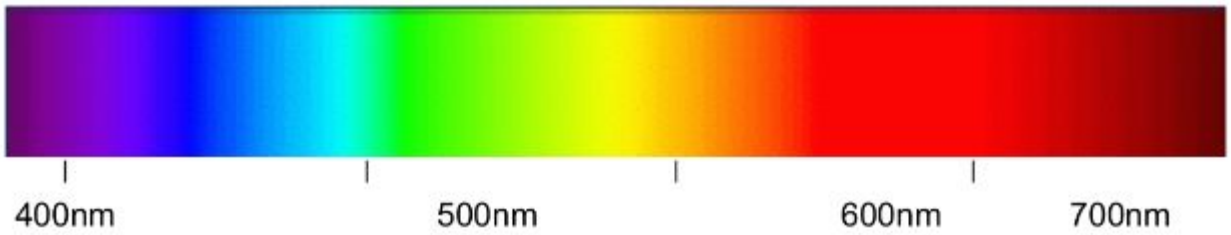
- **Light**: light is **narrow frequency band within electromagnetic spectrum.**
- A few of other frequency bands within this spectrum called **radio waves ,micro waves and X-ray.**



Electromagnetic Spectrum



The Visible Spectrum



Wavelength means COLOR

Properties of Light

1. **Light sources**-Sun or Bulb which emits all frequencies within visible range to produce white color.
2. When light incident on object some freq. are **reflected** and some are **absorbed by object.**
3. **Combination of freq. present in reflected light determine color of object .**
4. e.g.**If low freq**. are predominant in reflected light ,obejct color is red

- **The dominant freq. is also called hue/color of light**
- Other than frequency there **two characteristics of light**

1. **Brightness**-> intensity of light
2. **Purity->** how much pure color of light appears

- **The term chromaticity** is used to refer collectively **to two properties describing color characteristic purity and dominant freq.**

Properties of Light(Terms)(2M each)

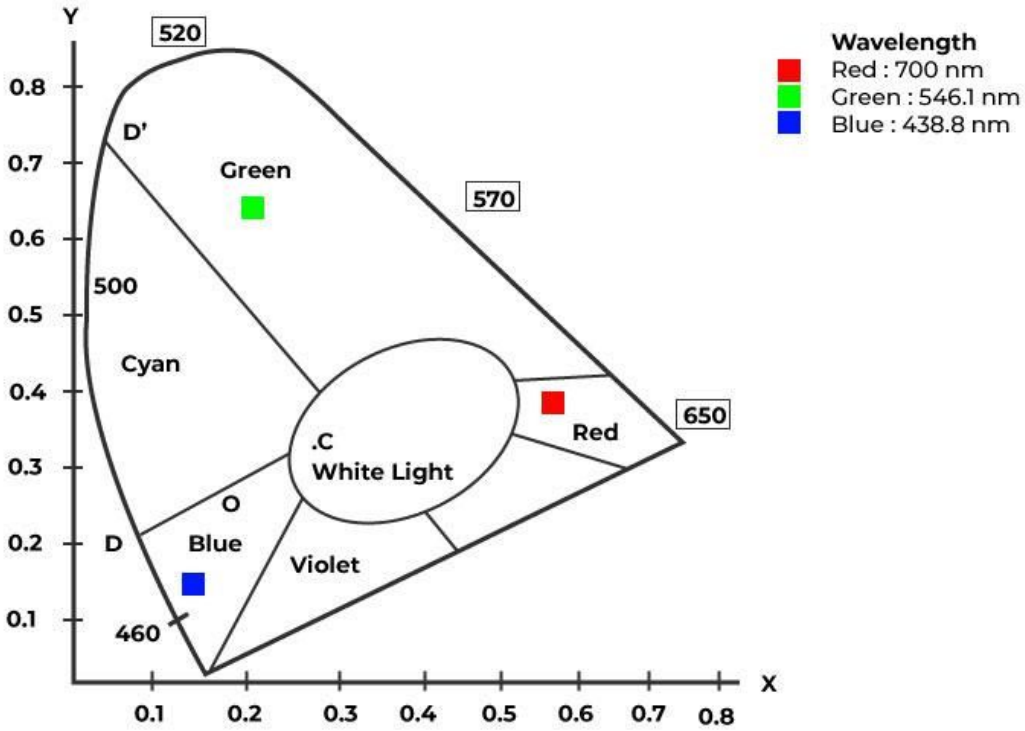
1. **Complementary color:** if **two color sources combine to produce white light**. e.g. Red and cyan ,Green and magenta , blue and yellow.

1. **Primary color:** if two or three colors are **used to produce other colors** then they are called primary color of that model e.g.Red,Green,Blue

3. **Color Gamut:** Color model uses combination of three colors to produce **wide range of colors** called **color gamut.**

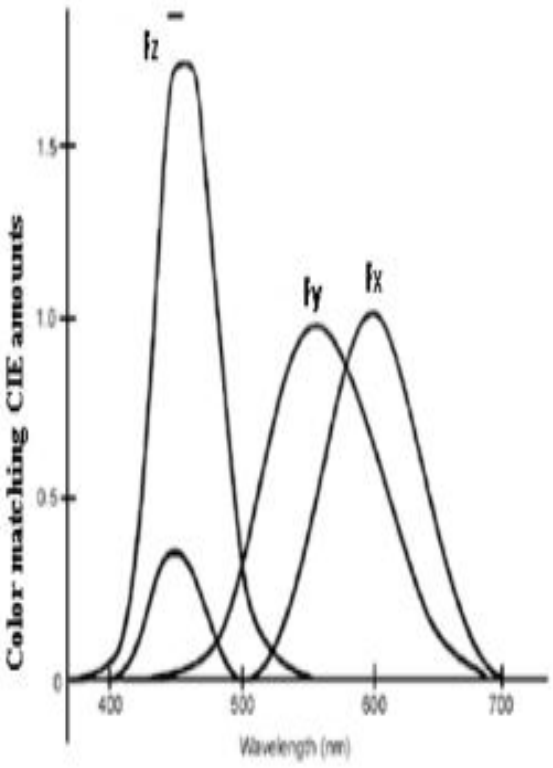
CIE Chromaticity

1. The chromaticity diagram represents the spectral colours and their mixtures based on the values of the primary colours(i.e Red, Green, Blue) contained by them.
2. Chromaticity contains two parameters i.e, hue and saturation. When we put hue and saturation together then it is known as Chrominance.
3. The various saturated pure spectral colours are represented along the perimeter of the curve representing the three primary colours – red, green and blue. Point C marked in chromaticity diagram represents a particular white light formed



Standard primaries

- Since **no finite set of color light** sources can be combined to **display all possible colors three standard primaries** defined in 1931 by **CIE** (Commission International Eclargae).
- Three standard primaries are imaginary colors
- They are **defined mathematically with +ve color matching functions** that specify the **amount of each primary color needed to describe any spectral color**
- This provides international standard **definition for all colors**



CIE standard XYZ color-matching functions $\bar{x}\,\bar{y}\,\bar{z}$

- It elliminate **negative value color matching and other problem to select a set of real primaries**

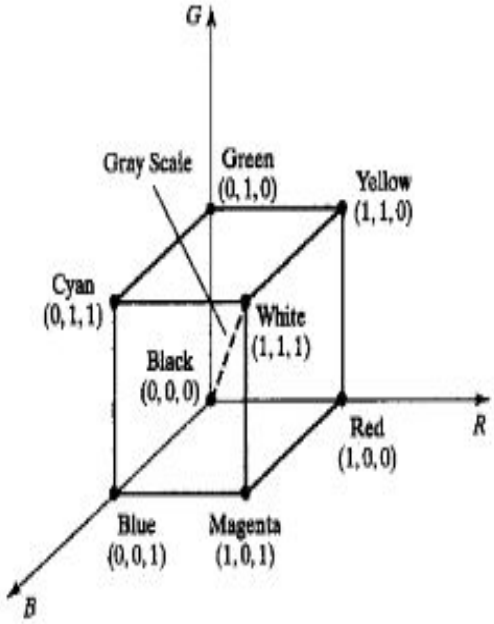
Color Models (4-6Marks each)

- **Color Models:**

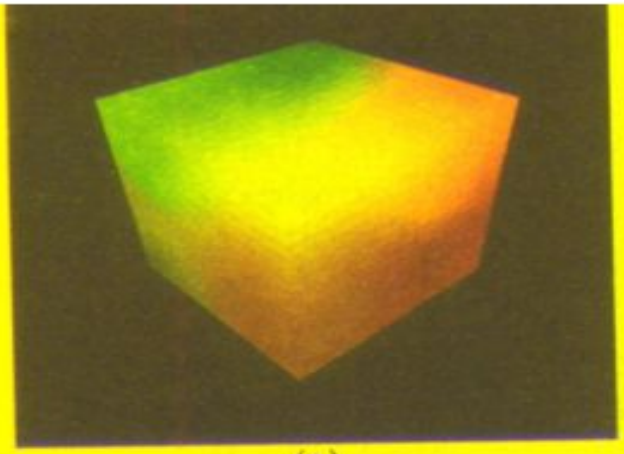
1. RGB,
2. HSV,
3. HLS,
4. CMY(K),
5. YIQ

RGB Color Model

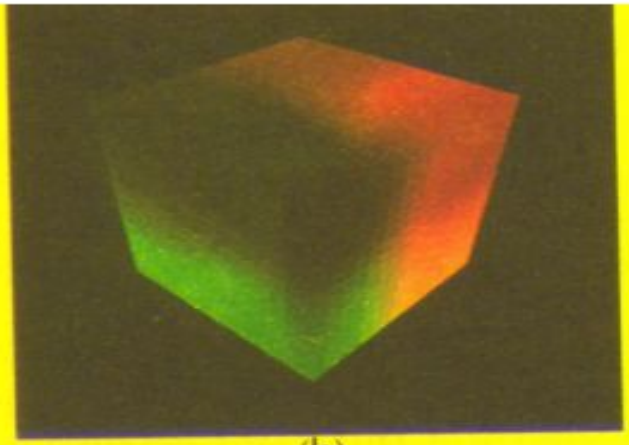
- Red-Green-Blue(RGB) generally used in CG
- We can represent this with unit cube defined on R,G and B axes
- Origin represents black (0,0,0)
- Co-ordinates (1,1,1) is white
- Vertices of the cube on axes represent the **primary colors**
- Other colors are generated by adding intensities of primary colors
- Yellow(1,1,0)=red+green
- Each color is represented by triple(R,G,B)
- C λ is expressed in RGB components as
- C λ =RR+GG+BB
- Each point along this diagonal has equal contribution from each primary color so that a gray shade halfway **between black and white is represented as (0.5,0.5,0.5)**



RGB Color Model

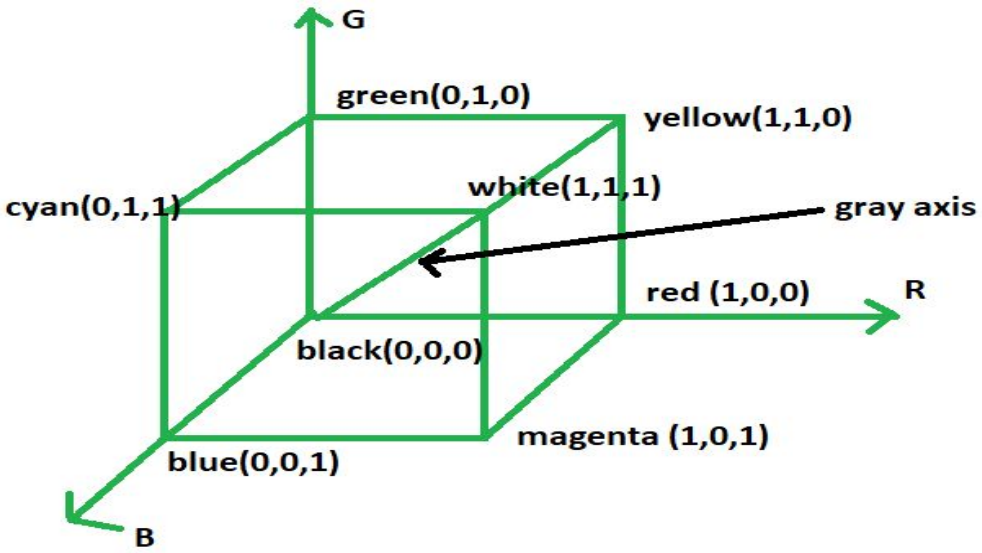


(a)



(b)

Figure 15-12
Two views of the RGB color cube: (a) along the grayscale diagonal from white to black and (b) along the grayscale diagonal from black to white.



CMYk Color Model

- If we assume **cyan**, magenta and yellow colors are **subtract complements of red**
- green and blue then we can represents this relationship as
- **C=1-R M=1-G Y=1-B**
- **CMY representation In matrix form**

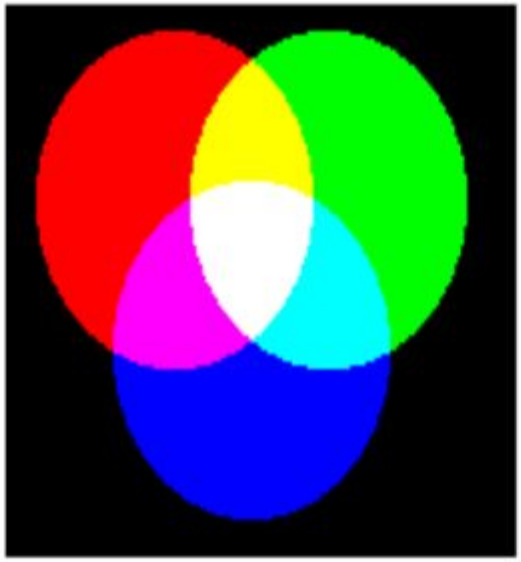
$$\begin{pmatrix} c \\ m \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} r \\ g \\ b \end{pmatrix}$$

Where black is represented as unit column vector in CMY

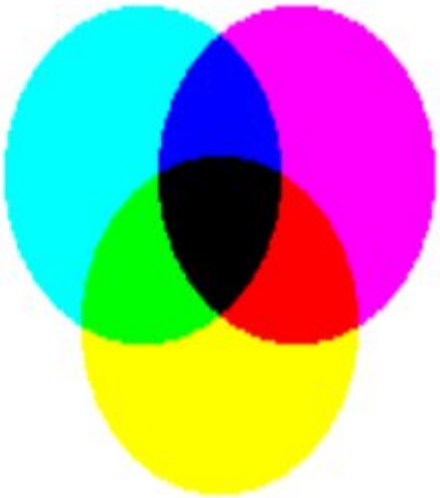
- **RGB representation with matrix form**

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} c \\ m \\ y \end{pmatrix}$$

Where white is represented as unit column vector in RGB



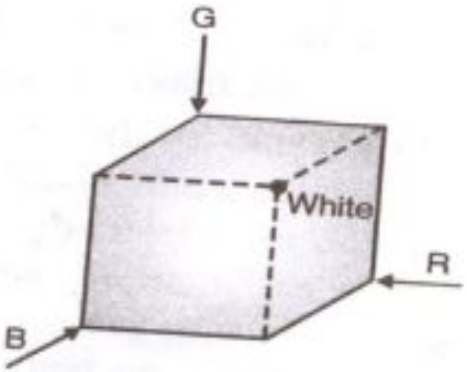
RGB



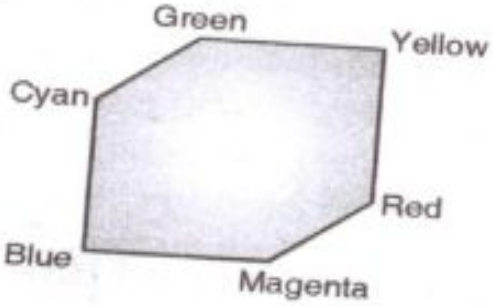
CMY

HSV Color model

- It uses color description instead of a set color primaries.
- **H(Hue),S(Saturation),V(Value)** are used to indicate color
- **H(Hue),** ->**pure color of light**
- **S(Saturation)**-> **how much white light is mixed with color**
- **V(Value)** ->**value**
- HSV color model representation is **derived from the RGB color model.**
- **If we imagine viewing the cube along the diagonal from the white vertex to the origin(black) we see outline of cube has hexagon shape.**



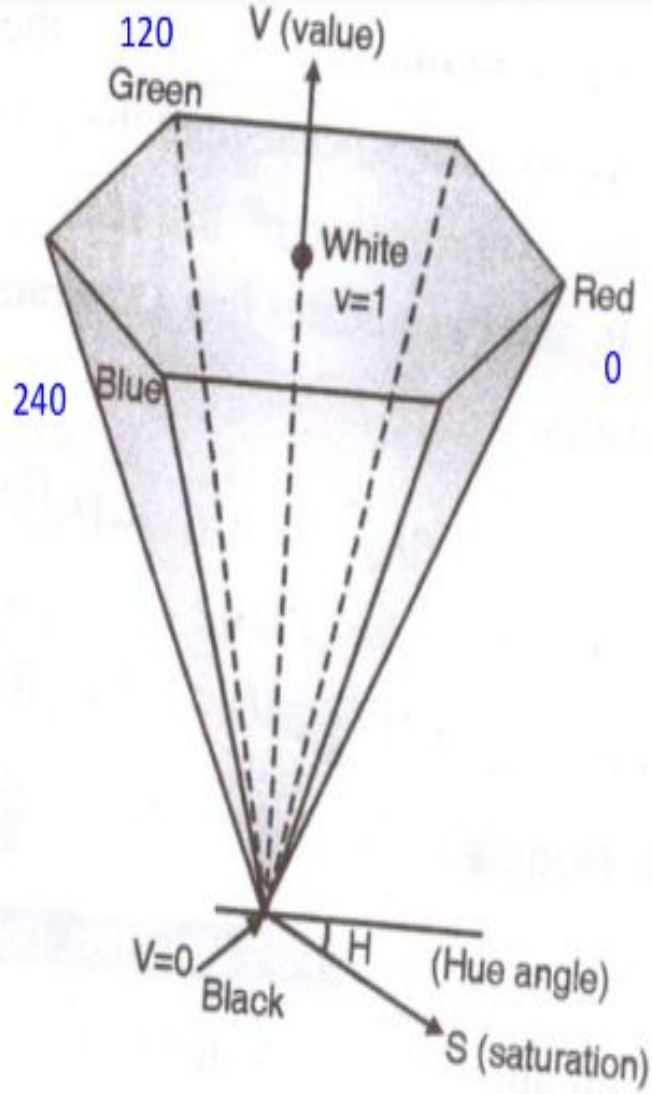
RGB color cube



Color Hexagon

HSV Color model

- In the hexagon the **saturation** is **measured along horizontal axis** and **value** is measured along **vertical axis** through centre of hexagon
- **Hue** is represented as an **angle about vertical axis from 0 degree at red through 360 degree**
- Vertices are separated at 60 degree intervals
- Yellow is at 60 degree, green is at 120 degree, cyan is at 180 degree

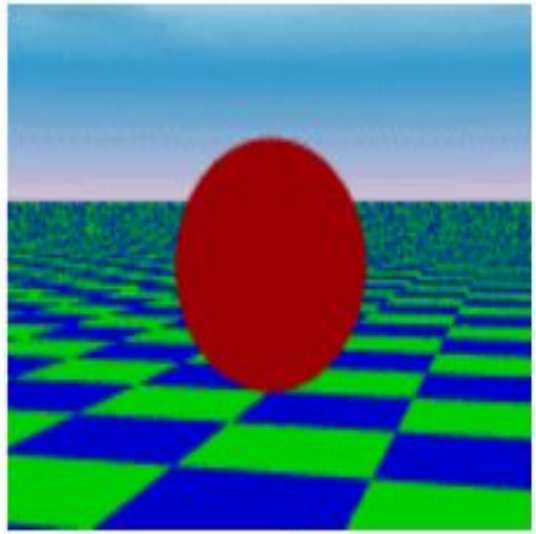


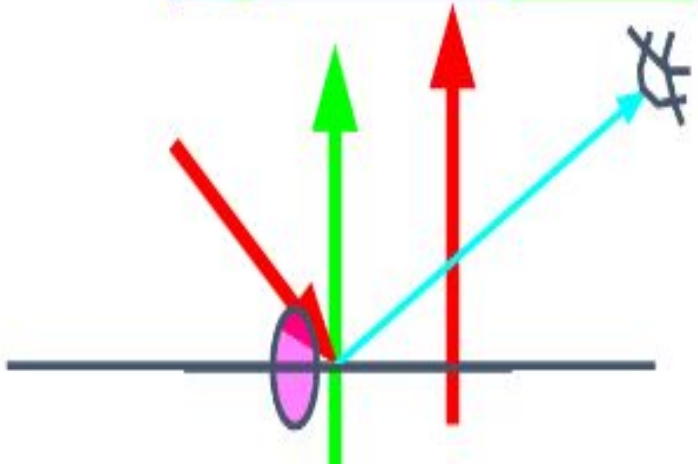
Light sources - Illumination models

- Illumination and Rendering
- An illumination model in computer graphics also called a lighting model or a shading model
- used to calculate the color of an illuminated position on the surface of an object
- Approximations of the physical laws Surface rendering is a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.

Ambient Term - Background Light

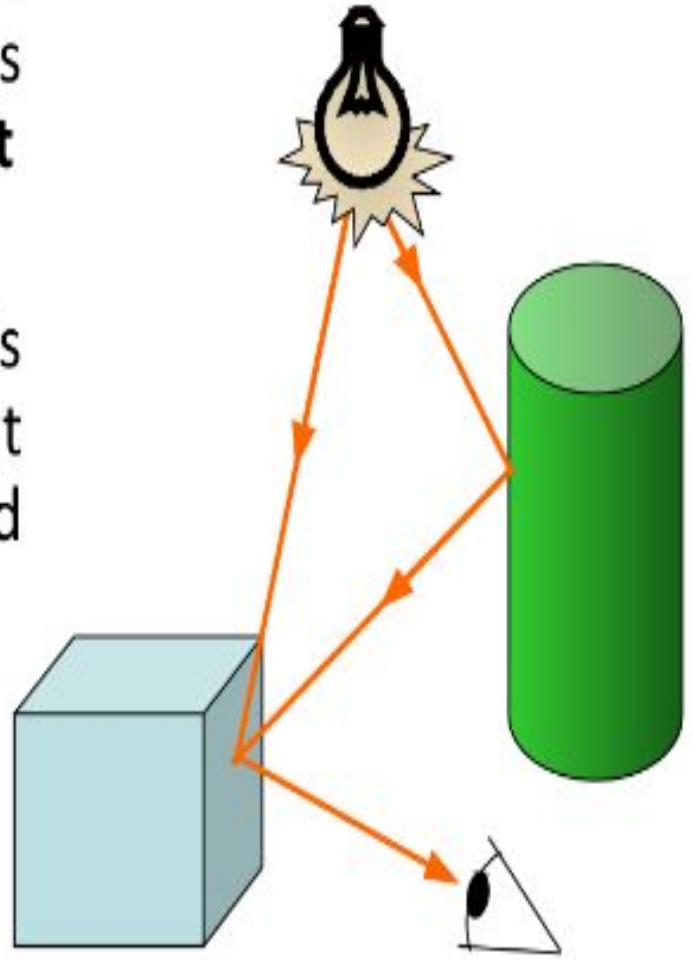
- The ambient term is a HACK!
- It represents the approximate contribution of the light to the general scene, regardless of location of light and object
- Indirect reflections that are too complex to completely and accurately compute
- $I_{ambient}$ = color





Ambient Light

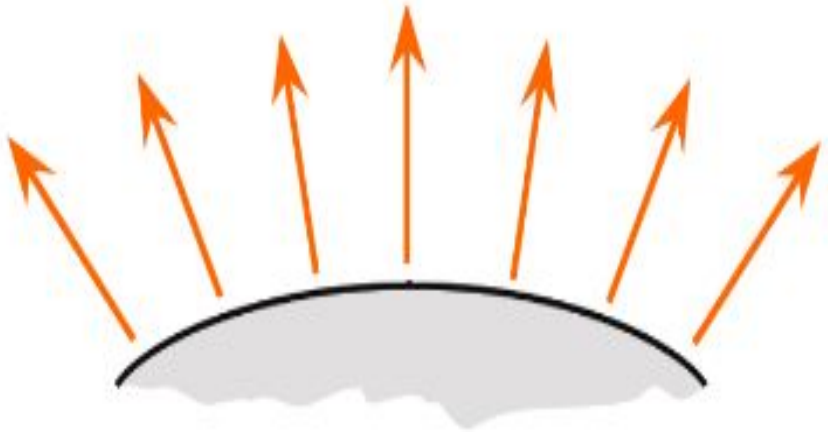
- A surface that is not exposed to direct light may still be lit up by reflections from other nearby objects – **ambient light**
- The total reflected light from a surface is the sum of the contributions from light sources and reflected light



Diffuse Reflection

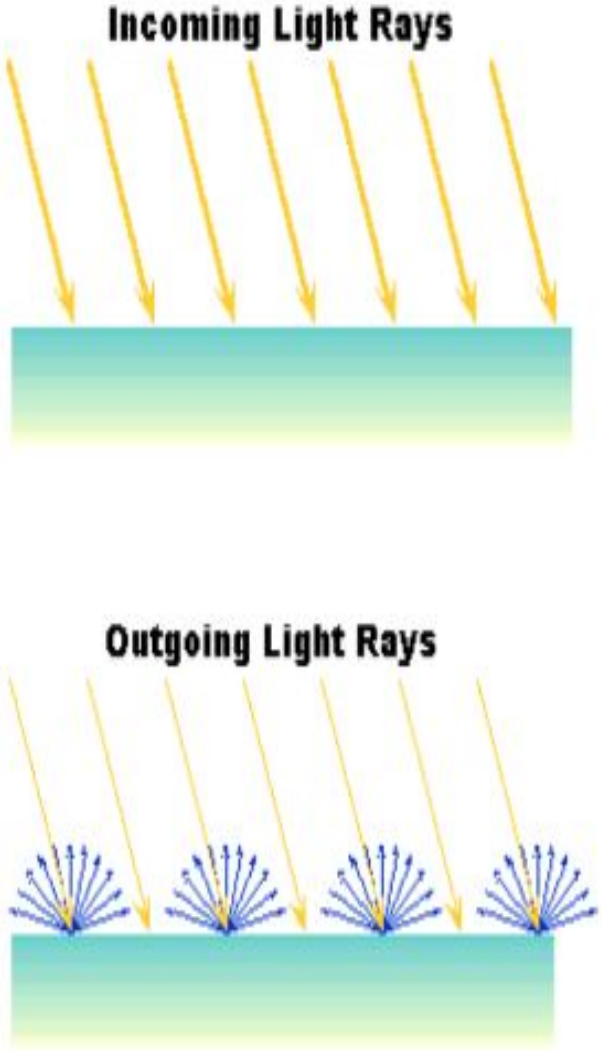
Surfaces that are rough or grainy tend to reflect light in all directions

This scattered light is called **diffuse reflection**



Diffuse Term

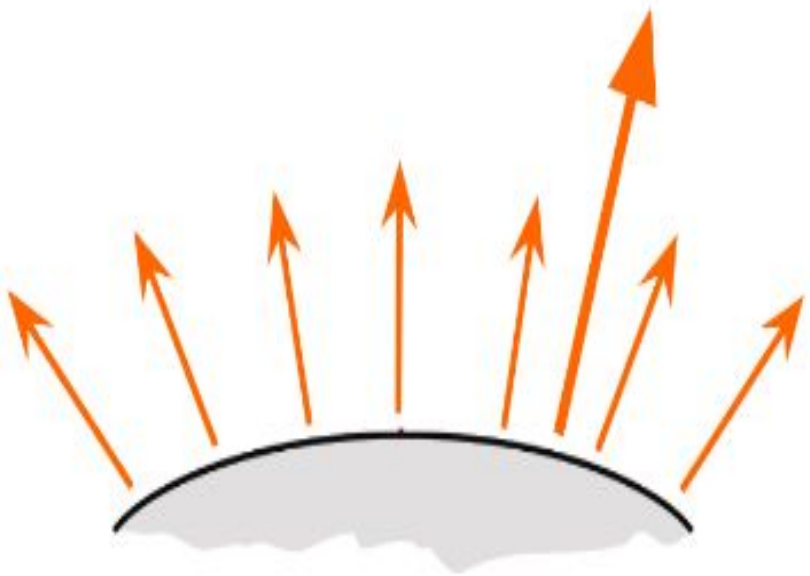
- Contribution that a light has on the surface, ***regardless of viewing direction***.
- Diffuse surfaces, on a microscopic level, are very rough. This means that a ray of light coming in has an equal chance of being reflected in *any* direction



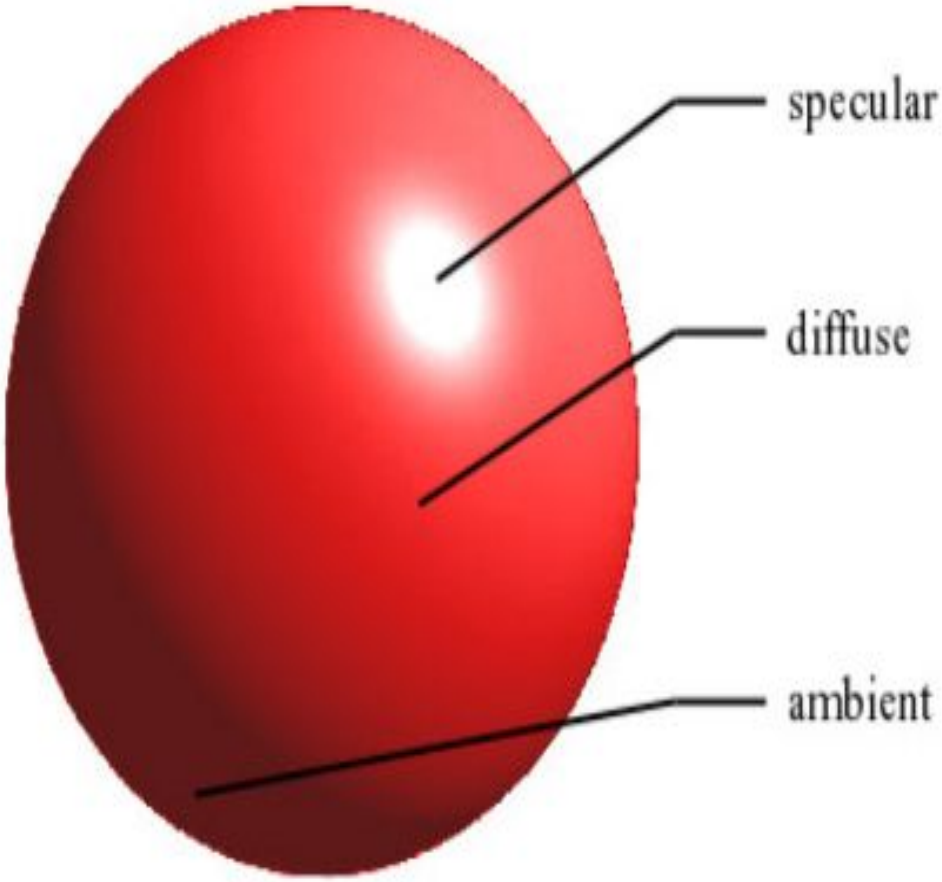
Specular Reflection

Additionally to diffuse reflection some of the reflected light is concentrated into a highlight or bright spot

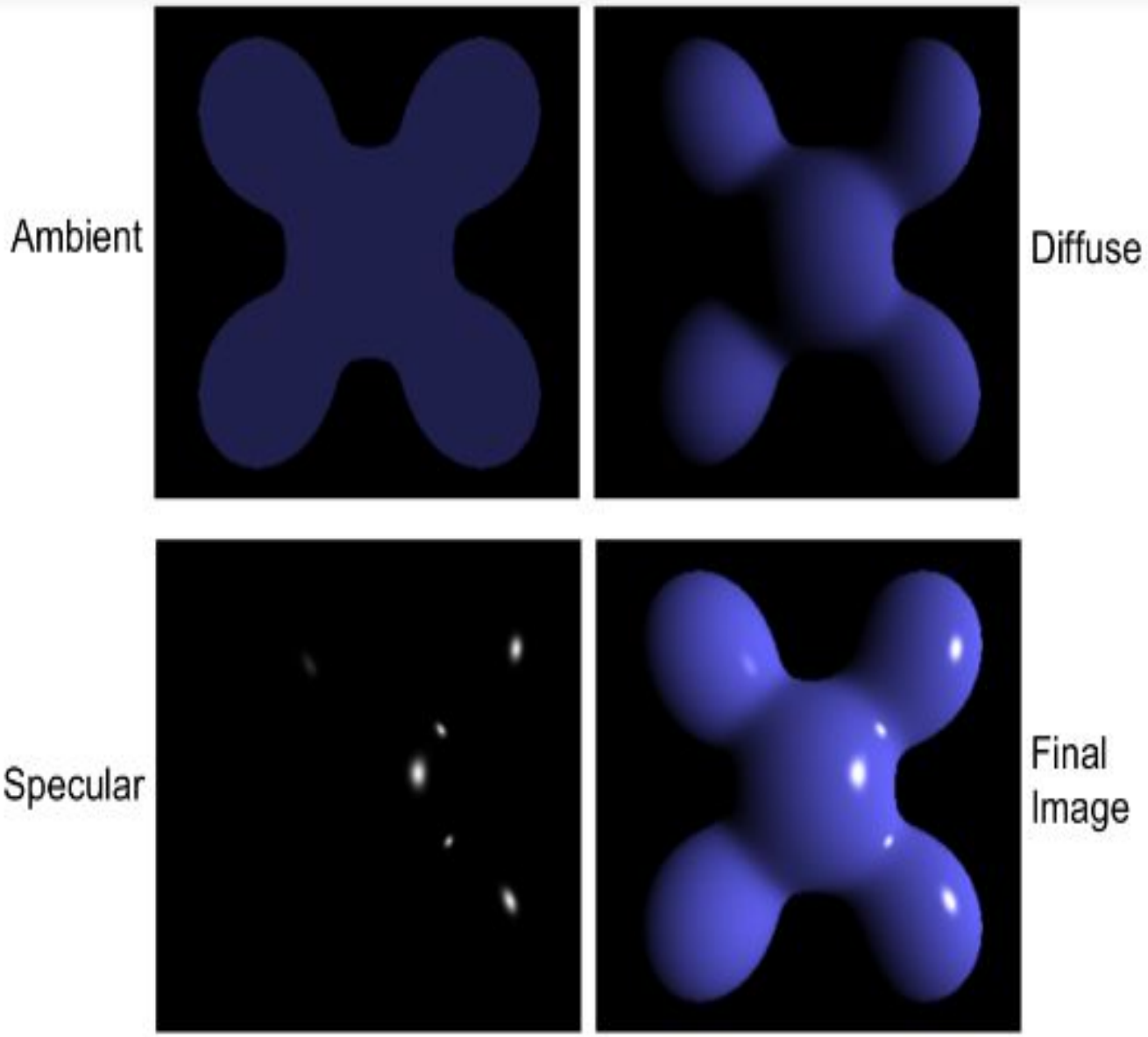
This is called **specular reflection**



Example



Example



Ambient Light

- This is a simple way to model combination of light reflection from various surfaces to produce a uniform illumination called **ambient light,** or **background light.**
- Ambient light has no directional properties.
- The amount of ambient light incident on all the surfaces and object are constant in all direction.
- If consider that ambient light of intensity I_a and each surface is illuminate with I_a intensity then resulting reflected light is constant for all the surfaces.

Diffuse Reflection

- When some intensity of light is falls on object surface and that surface reflect light in all the direction in equal amount then the resulting reflection is called **diffuse reflection.**
- Ambient light reflection is approximation of global diffuse lighting effects.
- Diffuse reflections are constant over each surface independent of our viewing direction.
- Amount of reflected light is depend on the parameter K_d, the **diffuse reflection coefficient** or **diffuse reflectivity.**
- K_d is assign value in between 0 and 1 depending on reflecting property.

Contd.

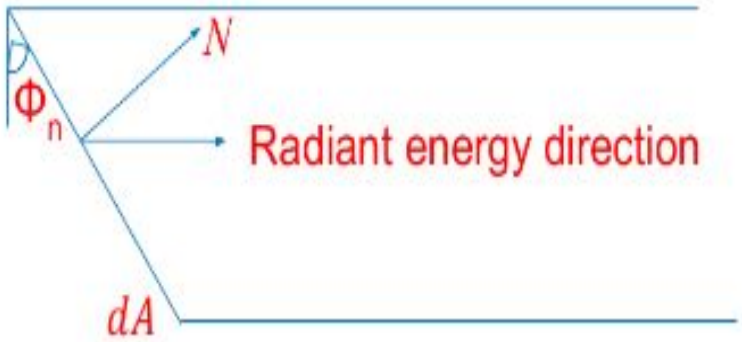
- Shiny surface reflect more light so K_d is assign larger value while dull surface assign small value.
- If surface is exposed to only ambient light we calculate ambient diffuse reflection as,

  $I_{ambdiff} = K_d I_a$

  where I_a the ambient light is falls on the surface.
- Practically most of times each object is illuminated by one light source so now we discuss diffuse reflection intensity for point source.

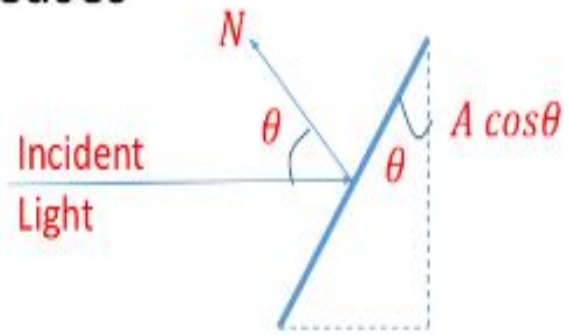
Contd.

- We assume that the diffuse reflection from source are scattered with equal intensity in all directions, independent of the viewing direction.
- Such a surface are sometimes referred as **ideal diffuse reflector** or **lambertian reflector.**
- This is modelled by **lambert's cosine law.**
- Law states that the radiant energy from any small surface area dA in any direction $\emptyset_n$ relative to surface normal is proportional to $cos\emptyset_n$.



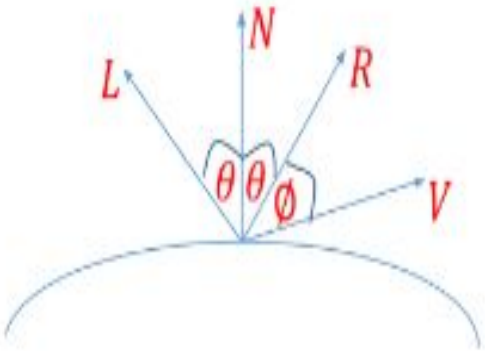
Contd.

- Reflected light intensity is does not depends on viewing direction.
- For lambertian reflection, the intensity of light is same in all viewing direction.
- Even though light distribution is equal in all direction for perfect reflector the brightness of a surface does depend on the orientation of the surface relative to light source.
- As the angle between surface normal and incidence light direction increases light falls on the surface is decreases



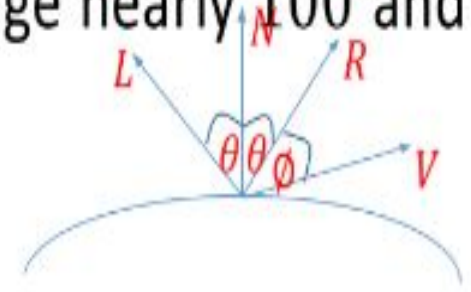
Specular Reflection

- While looking at an illuminated shiny surface, such as polished metal, we see a highlight, or bright spot, at certain viewing directions.
- This phenomenon is called **specular reflection,** is the result of total, or near total reflection of the incident light in a concentrated region around the **specular reflection angle.**
- The specular reflection angle equals the angle of the incident light.
- R is unit vector in direction of reflection
- L is unit vector point towards light source
- N is unit normal vector
- V is unit vector in viewing direction.

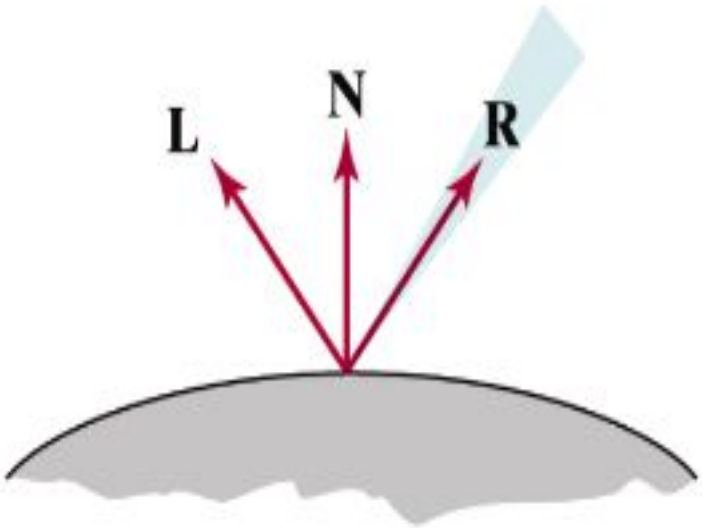


Contd.

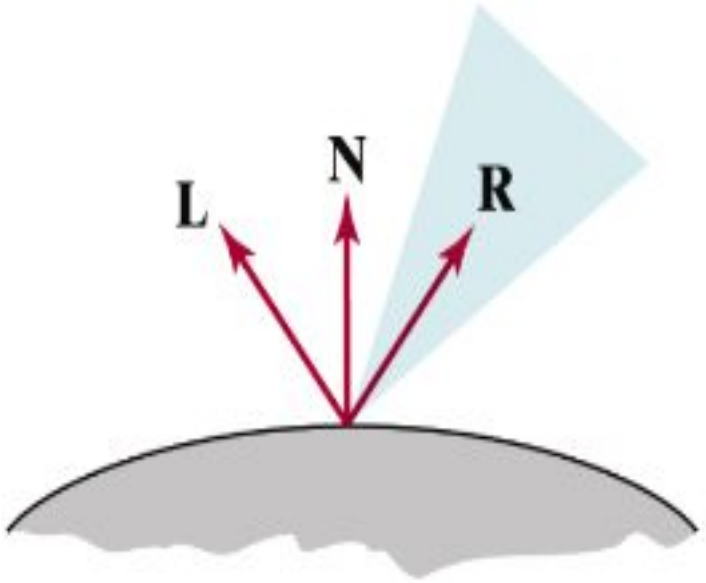
- Objects other than ideal reflectors exhibits specular reflection over a finite range of viewing positions around vector R.
- Shiny surface have a narrow specular reflection range and dull surface have wide specular reflection range.
- **Phong model** states that the intensity of specular reflection proportional to $cos^{ns}\emptyset$. Angle $\emptyset$ varies in between 0^0 to 90^0.
- Values assigned to **specular reflection parameter** $'ns'$ is determined by the type of surface that we want to display.
- A shiny surface assigned $'ns'$ values large nearly 100 and dull surface assigned small nearly 1.



Contd.



Shiny Surface
(Large n_s)



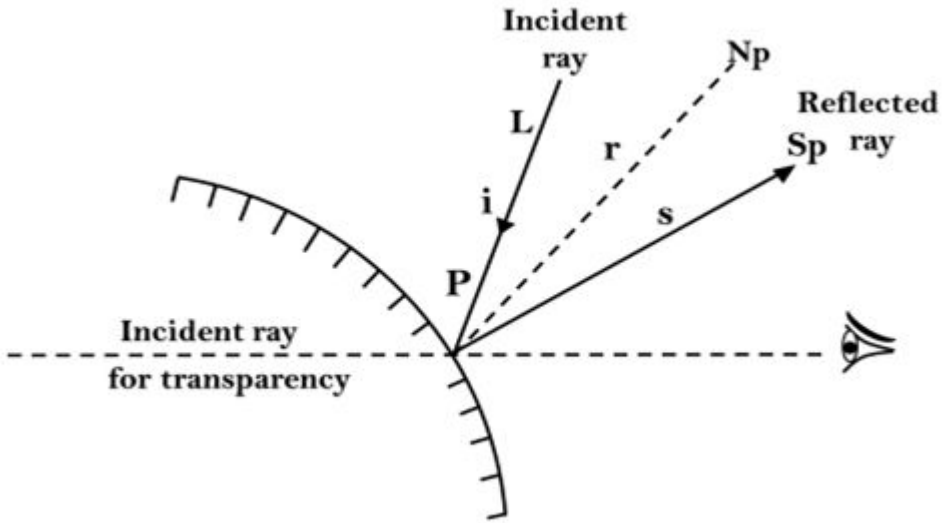
Dull Surface
(Small n_s)

Warn Model

- The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.
- Light sources are modeled as points on a reflecting surface, using the Phong model for the surface points.
- Then the intensity in different directions is controlled by selecting values for the Phong exponent
- In addition, light controls and spotlighting, used by studio photographers can be simulated in the Warn model.
- Flaps are used to control the amount of light emitted by a source In various directions

Shading Algorithms:

1. Shading is referred to as the implementation of the illumination model at the pixel points or polygon surfaces of the graphics objects.
2. Shading model is used to compute the intensities and colors to display the surface.
3. The shading model has two primary ingredients: properties of the surface and properties of the illumination falling on it.
4. The principal surface property is its reflectance, which determines how much of the incident light is reflected.
5. If a surface has different reflectance for the light of different wavelengths,



The simplest form of shading considers only diffuse illumination:

$$E_{pd} = R_p I_d$$

where Epd is the energy coming from point P due to diffuse illumination. Id is the diffuse illumination falling on the entire scene, and Rp is the reflectance coefficient at P

Constant Intensity Shading

1. A fast and straightforward method for rendering an object with polygon surfaces is constant intensity shading, also called Flat Shading.
2. 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.
3. Constant Shading can be useful for quickly displaying the general appearances of the curved surface as shown in fig:

Fig: A Polygon mesh approximation of an object
(a) is rendered with flat shading (b) and With Gouraud shading (c)

Gouraud shading

1. This Intensity-Interpolation scheme, developed by Gouraud and usually referred to as Gouraud Shading, renders a polygon surface by linear interpolating intensity value across the surface.
2. Intensity values for each polygon are coordinate with the value of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.
3. Each polygon surface is rendered with Gouraud Shading by performing the following calculations:
   a. Determining the average unit normal vector at each polygon vertex
   b. Apply an illumination model to each vertex to determine the vertex intensity.
   c. Linear interpolate the vertex intensities over the surface of the polygon.
4. At each polygon vertex, we obtain a normal vector by averaging the surface normals of all polygons staring that vertex as shown in fig:

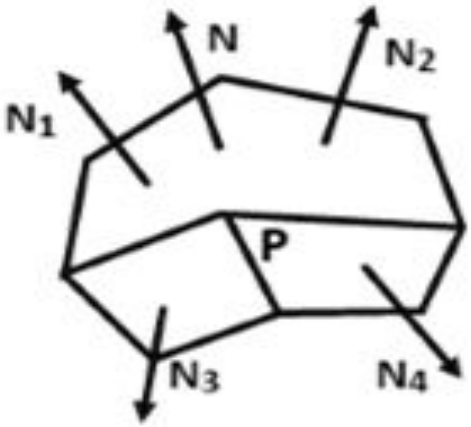


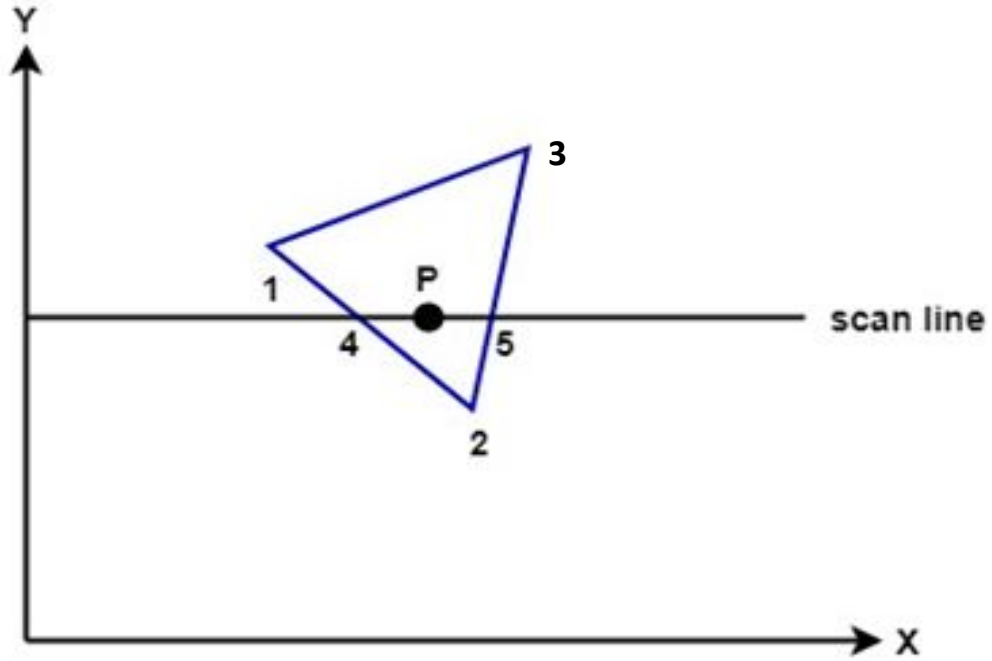
Fig: The normal vertex at vertex V is calculated as the average of surface normal for each polygon sharing the vertex.

5. Thus, for any vertex position V, we acquire the unit vertex normal with the calculation

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

- **Apply an illumination model to each vertex to determine the vertex intensity.**

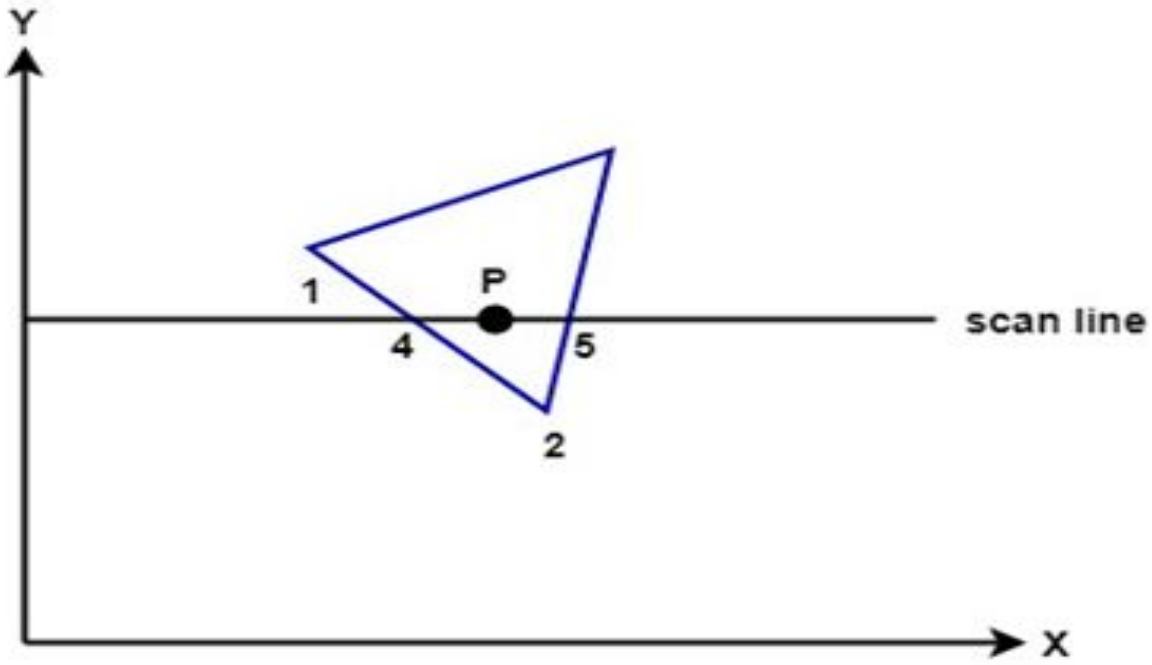
1. Interpolating intensities along the polygon edges.
2. For each scan line, the intensities at the intersection of the scan line with a polygon edge are linearly interpolated from the intensities at the edge endpoints.
3. **For example:** In fig, the polygon edge with endpoint vertices at position 1 and 2 is intersected by the scanline at point 4. A fast method for obtaining the intensities at point 4 is to interpolate between intensities I_1 and I_2 using only the vertical displacement of the scan line.



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

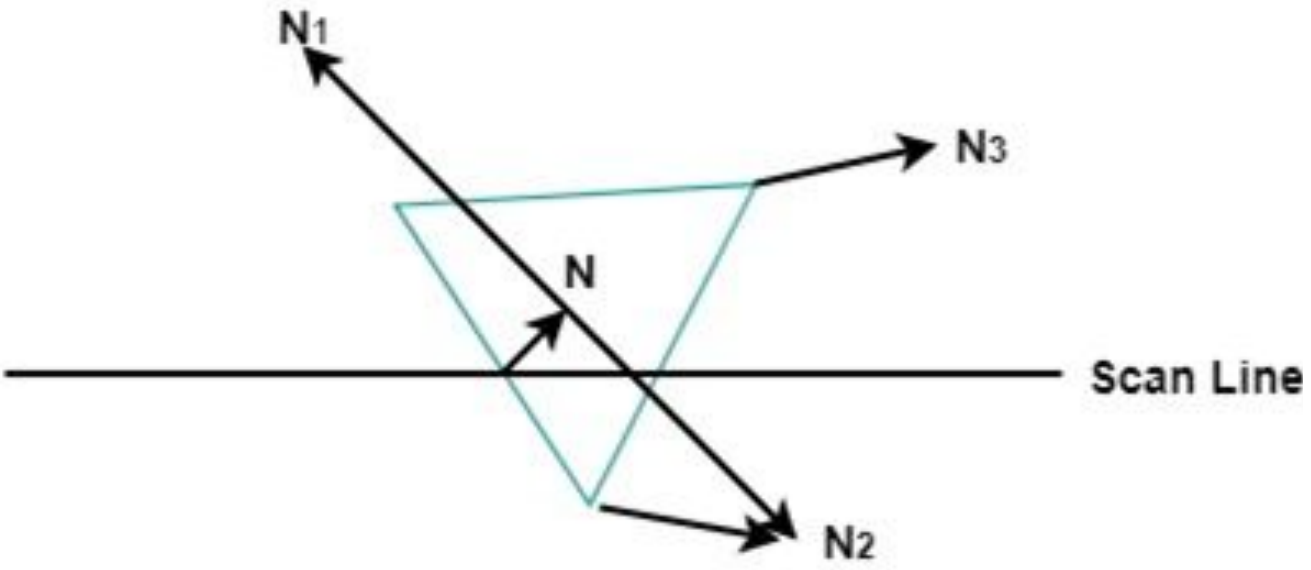
4. Similarly, the intensity at the right intersection of this scan line (point 5) is interpolated from the intensity values at vertices 2 and 3. Once these bounding intensities are established for a scan line, an interior point (such as point P in the previous fig) is interpolated from the bounding intensities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

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



Phong Shading

1. A more accurate method for rendering a polygon surface is to interpolate the normal vector and then apply the illumination model to each surface point.
2. This method developed by Phong Bui Tuong is called Phong Shading or normal vector Interpolation Shading.
3. It displays more realistic highlights on a surface and greatly reduces the Match-band effect
4. A polygon surface is rendered using Phong shading by carrying out the following steps:
   a. Determine the average unit normal vector at each polygon vertex.
   b. Linearly & interpolate the vertex normals over the surface of the polygon.
   c. Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.
5. Interpolation of the surface normal along a polynomial edge between two vertices as shown in fig:



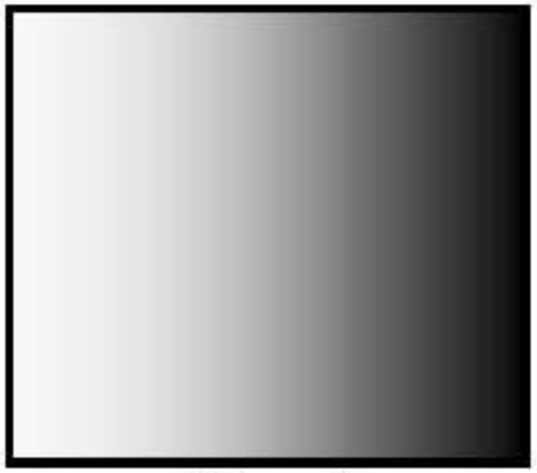
$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Fig: Interpolation of surface normals along a polygon edge.

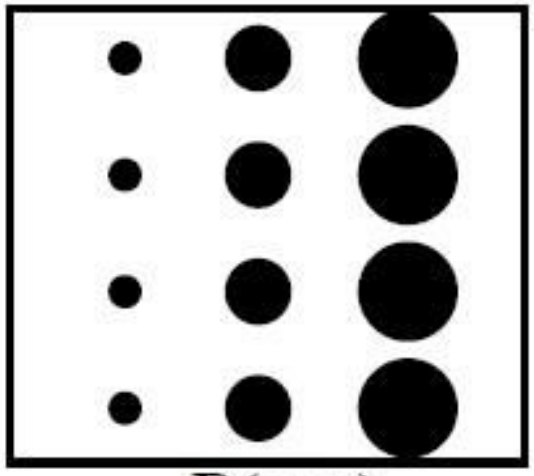
6. Incremental methods are used to evaluate normals between scan lines and along each scan line. At each pixel position along a scan line, the illumination model is applied to determine the surface intensity at that point.
7. Intensity calculations using an approximated normal vector at each point along the scan line produce more accurate results than the direct interpolation of intensities, as in Gouraud Shading.

Halftone Shading

1. Halftone is the reprographic technique that simulates continuous tone imagery through the use of dots, varying either in size, in shape or in spacing.
2. "Halftone" can also be used to refer specifically to the image that is produced by this process.
3. The human visual system has a tendency to average brightness over small areas, so the black dots and their white background merge and are perceived as an intermediate shade of grey.
4. The process of generating a binary pattern of black and white dots from an image is termed halftoning.
5. Use dots of varying size to represent intensities
6. Area of dots proportional to intensity in image

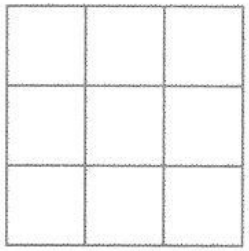


I(x,y)

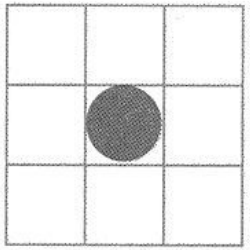


P(x,y)

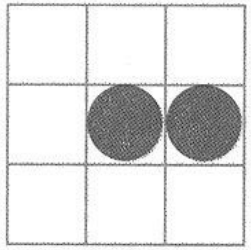
intensities in a n x n cluster



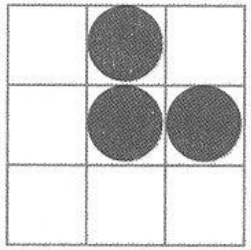
0
$0 \leq I < 0.1$



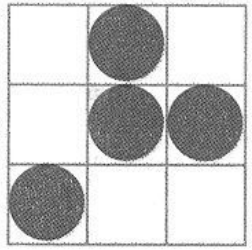
1
$0.1 \leq I < 0.2$



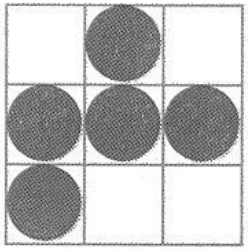
2
$0.2 \leq I < 0.3$



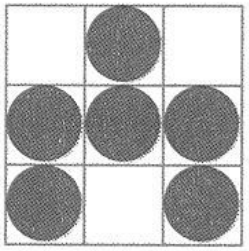
3
$0.3 \leq I < 0.4$



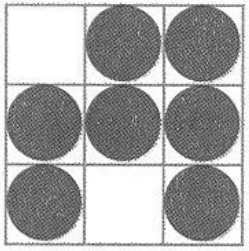
4
$0.4 \leq I < 0.5$



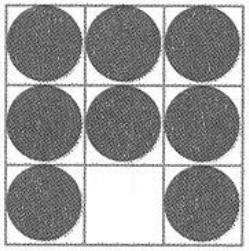
5
$0.5 \leq I < 0.6$



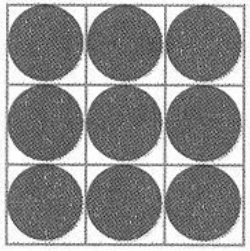
6
$0.6 \leq I < 0.7$



7
$0.7 \leq I < 0.8$

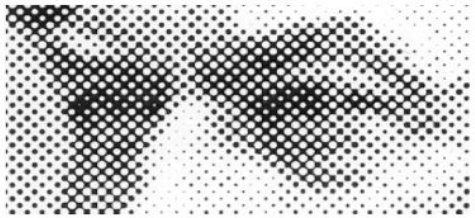


8
$0.8 \leq I < 0.9$

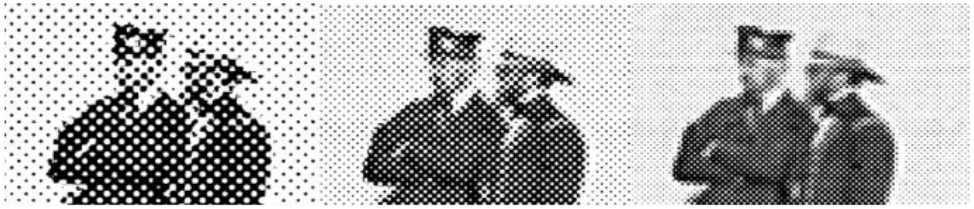


9
$0.9 \leq I \leq 1.0$

Halftoning



Halftoning – dot size



Hidden Surface Removal

1. One of the most challenging problems in computer graphics is the removal of hidden parts from images of solid objects.
2. In real life, the opaque material of these objects obstructs the light rays from hidden parts and prevents us from seeing them.
3. In the computer generation, no such automatic elimination takes place when objects are projected onto the screen coordinate system.
4. Instead, all parts of every object, including many parts that should be invisible are displayed.
5. To remove these parts to create a more realistic image, we must apply a hidden line or hidden surface algorithm to set of objects.
6. Hidden line and Hidden surface algorithms capitalize on various forms of coherence to reduce the computing required to generate an image.
7. Different types of coherence are related to different forms of order or regularity in the image.
8. Scan line coherence arises because the display of a scan line in a raster image is usually very similar to the display of the preceding scan line.
9. **Frame coherence** in a sequence of images designed to show motion recognizes that successive frames are very similar.
10. **Object coherence** results from relationships between different objects or between separate parts of the same objects.

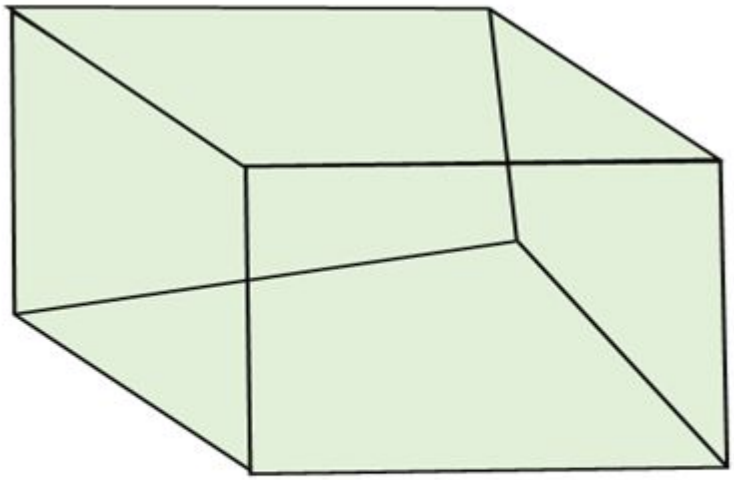
Types of hidden surface detection algorithms

Object space methods:

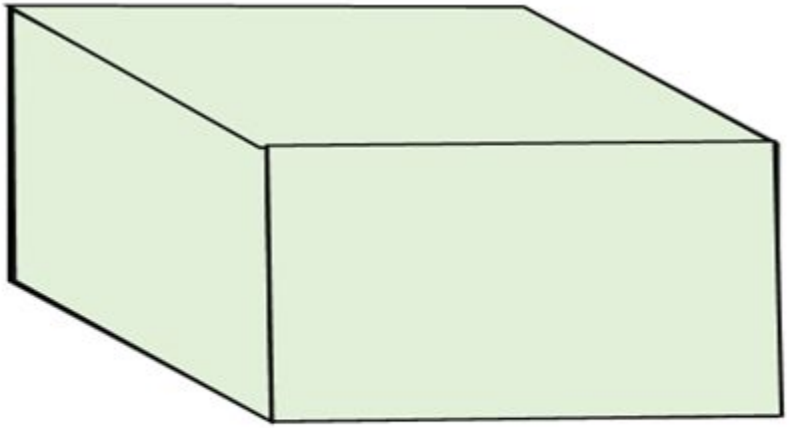
- In this method, various parts of objects are compared. After comparison visible, invisible or hardly visible surface is determined.
- These methods generally decide visible surface. In the wireframe model, these are used to determine a visible line. So these algorithms are line based instead of surface based.

Image space methods:

- Here positions of various pixels are determined. It is used to locate the visible surface instead of a visible line.
- Each point is detected for its visibility. If a point is visible, then the pixel is on, otherwise off.
- So the object close to the viewer that is pierced by a projector through a pixel is determined. That pixel is drawn is appropriate color.



Object with hidden line



Object when hidden lines removed

Algorithms used for hidden line surface detection

1. Back Face Removal Algorithm
2. Z-Buffer Algorithm
3. Painter Algorithm
4. Subdivision Algorithm

Back Face Removal Algorithm

1. It is used to plot only surfaces which will face the camera. The objects on the back side are not visible.
2. This method will remove 50% of polygons from the scene if the parallel projection is used. If the perspective projection is used then more than 50% of the invisible area will be removed.
3. The object is nearer to the center of projection, number of polygons from the back will be removed.
4. It applies to individual objects. It does not consider the interaction between various objects. Many polygons are obscured by front faces, although they are closer to the viewer, so for removing such faces back face removal algorithm is used.
5. When the projection is taken, any projector ray from the center of projection through viewing screen to object pieces object at two points, one is visible front surfaces, and another is not visible back surface.
6. This algorithm acts a preprocessing step for another algorithm. The back face algorithm can be represented geometrically. Each polygon has several vertices.
7. All vertices are numbered in clockwise. The normal M_1 is generated a cross product of any two successive edge vectors.
8. M_1represent vector perpendicular to face and point outward from polyhedron surface

$$N_1=(v_2-v_1)(v_3-v_2)$$

If $N_1.P \geq 0$ visible

$N_1.P < 0$ invisible

Back Face Removed Algorithm

Repeat for all polygons in the scene.

1. Do numbering of all polygons in clockwise direction i.e.

   $v_1\ v_2\ v_3 \ldots\ldots v_z$

2. Calculate normal vector i.e. N_1

   $N_1=(v_2-v_1)*(v_3-v_2)$

3. Consider projector P, it is projection from any vertex

   Calculate dot product

   Dot=N.P

4. Test and plot whether the surface is visible or not.

   If Dot ≥ 0 then

   surface is visible

   else

   Not visible

Z-Buffer Algorithm

1. It is also called a **Depth Buffer Algorithm**. Depth buffer algorithm is simplest image space algorithm.
2. For each pixel on the display screen, we keep a record of the depth of an object within the pixel that lies closest to the observer.
3. In addition to depth, we also record the intensity that should be displayed to show the object.
4. Depth buffer is an extension of the frame buffer. Depth buffer algorithm requires 2 arrays, intensity and depth each of which is indexed by pixel coordinates (x, y).

Algorithm

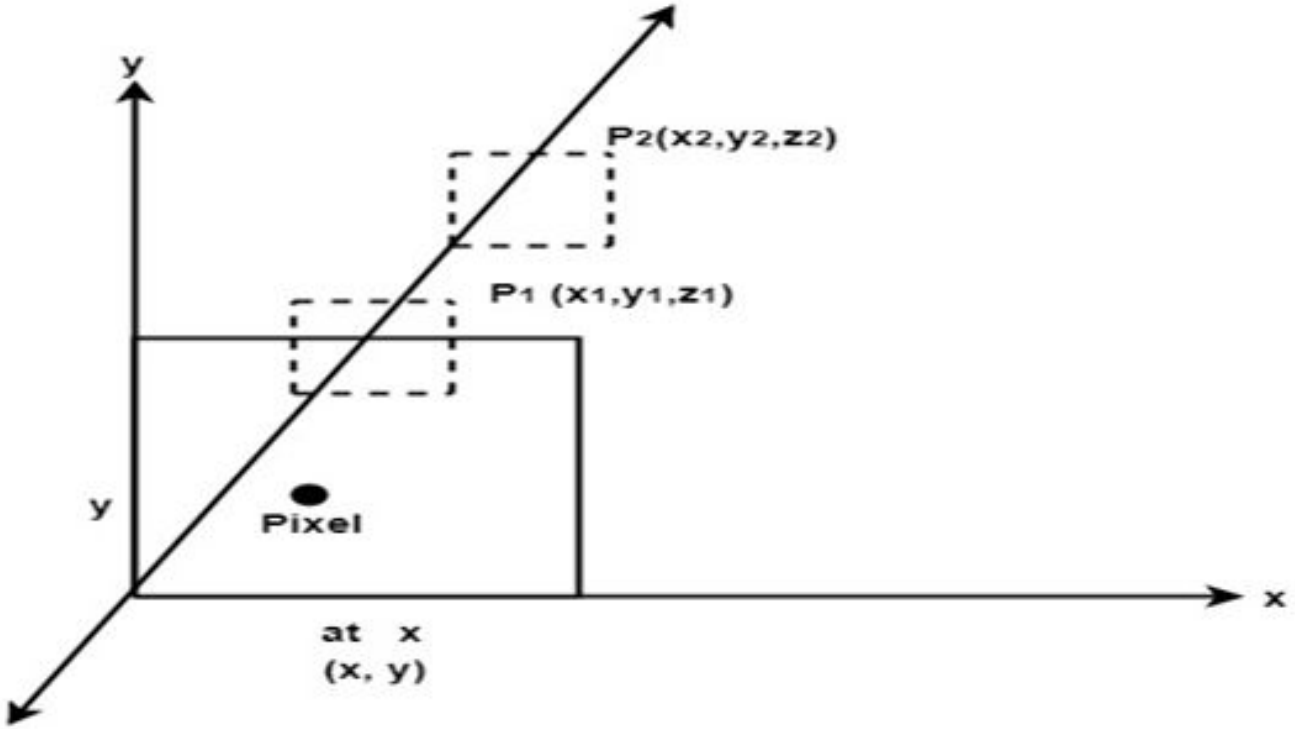
For all pixels on the screen, set depth [x, y] to 1.0 and intensity [x, y] to a background value.

For each polygon in the scene, find all pixels (x, y) that lie within the boundaries of a polygon when projected onto the screen. For each of these pixels:

(a) Calculate the depth z of the polygon at (x, y)

(b) If z < depth [x, y], this polygon is closer to the observer than others already recorded for this pixel. In this case, set depth [x, y] to z and intensity [x, y] to a value corresponding to polygon's shading. If instead z > depth [x, y], the polygon already recorded at (x, y) lies closer to the observer than does this new polygon, and no action is taken.

3. After all, polygons have been processed; the intensity array will contain the solution.



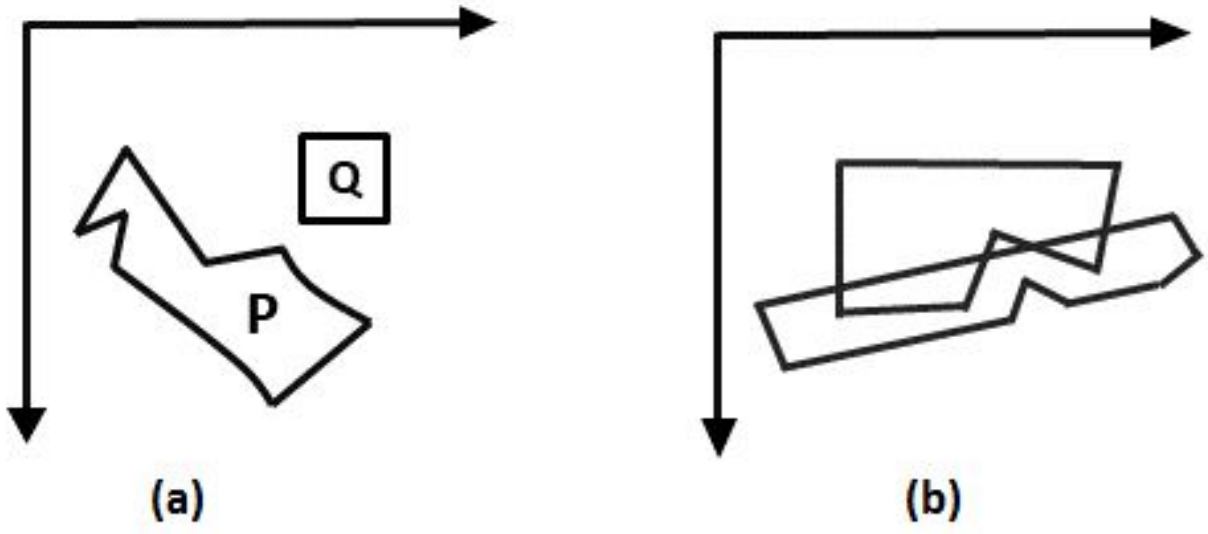
4. **The depth buffer algorithm illustrates several features common to all hidden surface algorithms.**

5. First, it requires a representation of all opaque surface in scene polygon in this case.

6. These polygons may be faces of polyhedral recorded in the model of scene or may simply represent thin opaque 'sheets' in the scene.

Painter Algorithm

- It came under the category of list priority algorithm. It is also called a **depth-sort algorithm**. In this algorithm ordering of visibility of an object is done. If objects are reversed in a particular order, then correct picture results.
- Objects are arranged in increasing order to z coordinate. Rendering is done in order of z coordinate. Further objects will obscure near one. Pixels of rear one will overwrite pixels of farther objects. If z values of two overlap, we can determine the correct order from Z value as shown in fig (a).
- If z objects overlap each other as in fig (b) this correct order can be maintained by splitting of objects.



- Depth sort algorithm or painter algorithm was developed by Newell, sancha. It is called the painter algorithm because the painting of frame buffer is done in decreasing order of distance. The distance is from view plane. The polygons at more distance are painted firstly.
- The concept has taken color from a painter or artist. When the painter makes a painting, first of all, he will paint the entire canvas with the background color. Then more distance objects like mountains, trees are added. Then rear or foreground objects are added to picture. Similar approach we will use. We will sort surfaces according to z values. The z values are stored in the refresh buffer.

Steps performed in-depth sort

1. Sort all polygons according to z coordinate.
2. Find ambiguities of any, find whether z coordinate overlap, split polygon if necessary.
3. Scan convert each polygon in increasing order of z coordinate.

Painter Algorithm

Step1: Start Algorithm

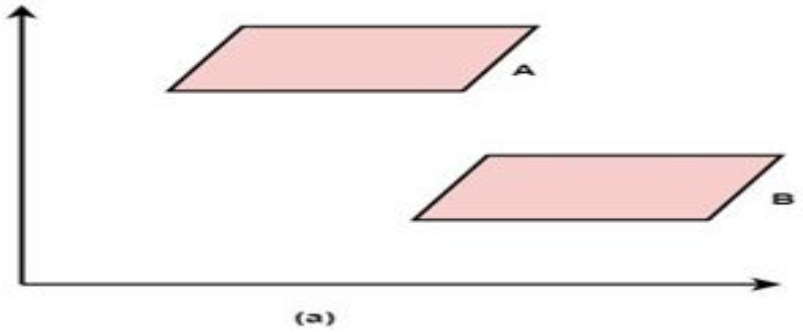
Step2: Sort all polygons by z value keep the largest value of z first.

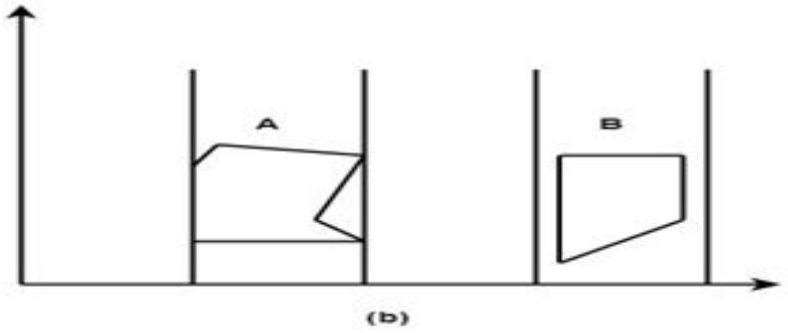
Step3: Scan converts polygons in this order.

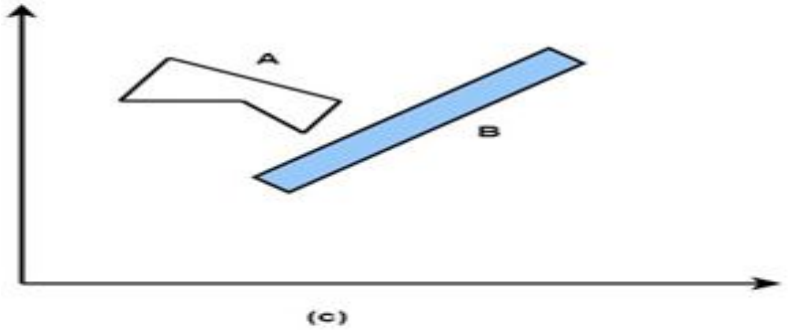
Test is applied

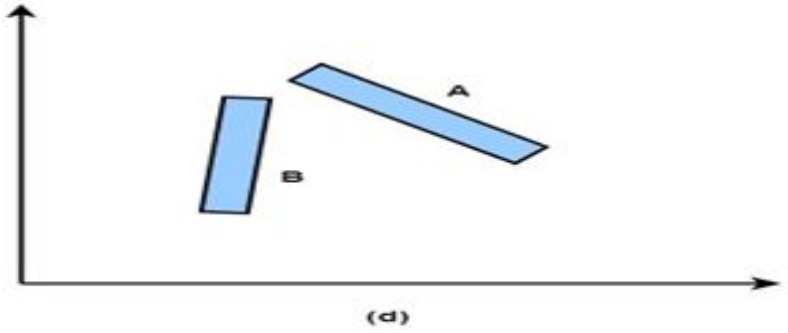
1. Does A is behind and non-overlapping B in the dimension of Z as shown in fig (a)
2. Does A is behind B in z and no overlapping in x or y as shown in fig (b)
3. If A is behind B in Z and totally outside B with respect to view plane as shown in fig (c)
4. If A is behind B in Z and B is totally inside A with respect to view plane as shown in fig (d)

The success of any test with single overlapping polygon allows F to be painted.



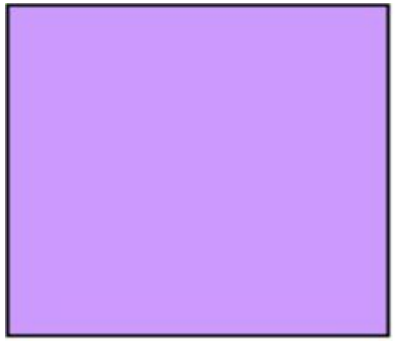




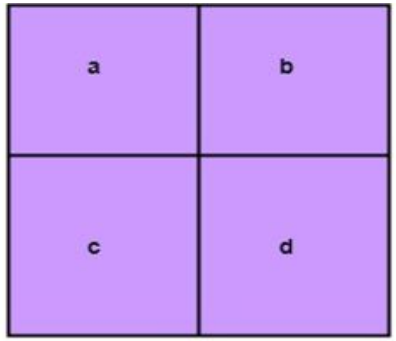


Area Subdivision Algorithm

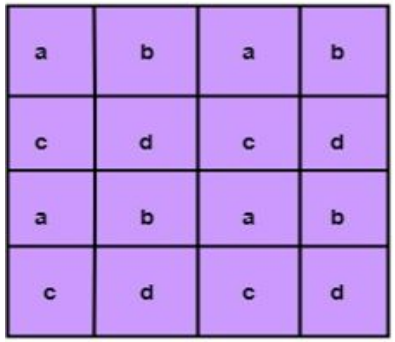
- It was invented by John Warnock and also called a Warnock Algorithm.
- It is based on a divide & conquer method. It uses fundamental of area coherence.
- It is used to resolve the visibility of algorithms. It classifies polygons in two cases i.e. trivial and non-trivial.
- Trivial cases are easily handled.
- Non trivial cases are divided into four equal subwindows.
- The windows are again further subdivided using recursion until all polygons classified trivial and non trivial.



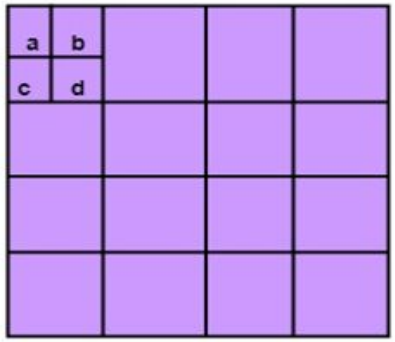
Original area
(a)



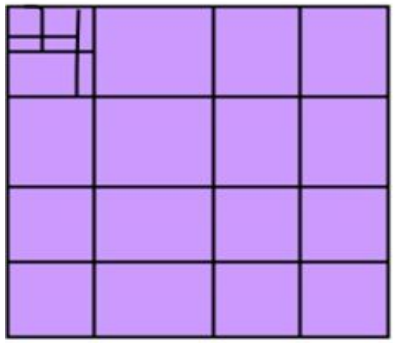
First division or subdivision of area
(b)



Second division
(c)



Third subdivision
(d)



Fourth subdivision
(e)

Classification of Scheme

It divides or classifies polygons in four categories:

1. Inside surface: It is surface which is completely inside the surrounding window or specified boundary as shown in fig(b).

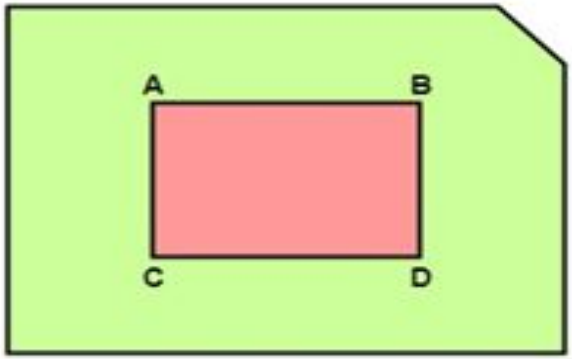
2. Outside surface: The polygon surface completely outside the surrounding window as shown in fig (d)

3. Surrounding surface: It is polygon surface which completely encloses the surrounding window as shown in fig (e)

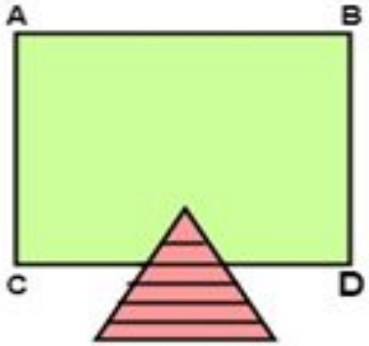
4. Overlapping surface: It is surface partially inside or partially outside the surface area as shown in fig (c)

(a)

ABCD is current window against which particular window is determined to be of either of four categories

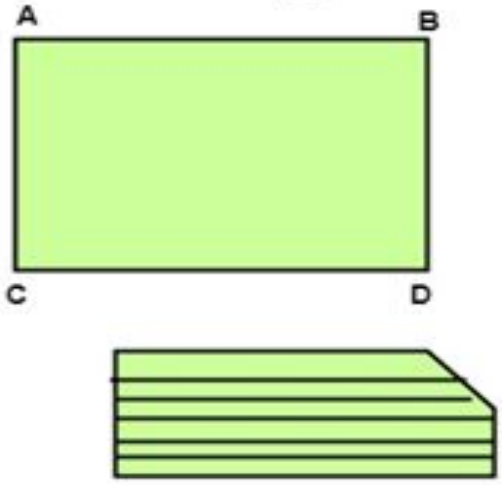


(b)



(c)

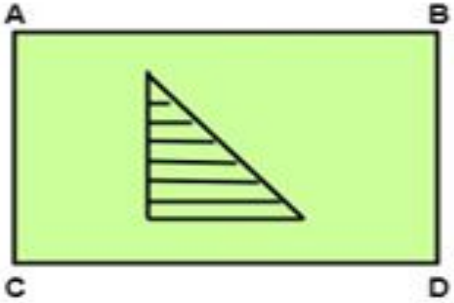
Surface of object intersecting desired window



Outside

(d)

Surface is outside specified window



(e)

Surface is inside specified window