COMPUTER GRAPHICS

Disclaimer

This document is part of teaching materials for Computer Graphics and does not claim any originality. This document does not cover all aspect of learning Computer Graphics, nor are these be taken as primary source of information. As the core textbooks and reference books for learning the subject has already been specified and provided to the students, students are encouraged to learn from the original sources because this document cannot be used as a substitute for prescribed textbooks.

Various text books as well as freely available material from internet were consulted for preparing this document. Contents in This document are ***copyrighted*** *to the instructor and authors of original texts where applicable.*

Unit: 5 Visual Realism

5.1 Hidden Surfaces and Hidden Surface Removal Approaches

- When a picture that contains the Opaque (non-transparent) objects and surfaces are viewed, the objects that are behind the objects that are closer cannot be viewed.
- Surfaces which are obscured by other opaque surfaces along the line of sight (projection) are invisible to the viewer.
- To obtain a realistic screen image, these hidden surfaces need to be removed.
- This process of identification and removal of these surfaces is known as Hidden-surface problem.
- 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 chosen 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.
- Deciding upon a method for a particular application can depend on such factors as the complexity of the scene, type of objects to be displayed, available equipment, and whether static or animated displays are to be generated.
- Visible surface detection methods are broadly classified according to whether they deal with objects or with their projected images.
    - **Object-Space method and**
    - **Image-space method**
- In physical coordinate system, object-space method is implemented and in case of screen coordinate system, image-space method is implemented.
- When a 3D object needs to be displayed on the 2D screen, the parts of the screen that are visible from the chosen viewing position is identified.

Object Space Method (OSM)

- 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.
- Method proceeds by determination of parts of an object whose view is obstructed by other object and draws these parts in the same color
- An object-space method compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.
- ***E.g. Back-face detection method***

Image Space Method (ISM)

- In this method, 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.
- In an image-space algorithm, visibility is decided point by point at each pixel position on the projection plane. Most visible-surface algorithms use image-space methods.
- **E.g. Depth-buffer method, Scan-line method, Area-subdivision method**

List Priority Algorithms

- This is a hybrid model that combines both object and image precision operations.
- Here, depth comparison & object splitting are done with object precision and scan conversion (which relies on ability of graphics device to overwrite pixels of previously drawn objects) is done with image precision.
- **E.g. Depth-Sorting method, BSP-tree method**

Difference between OSM and ISM

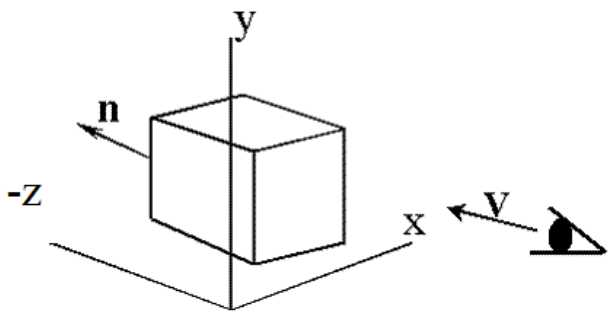
Object Space	Image Space
1. Object space is object based. It concentrates on geometrical relation among objects in the scene.	1. It is a pixel-based method. It is concerned with the final image, what is visible within each raster pixel.
2. Here surface visibility is determined.	2. Here line visibility or point visibility is determined.
3. It is performed at the precision with which each object is defined, no resolution is considered.	3. It is performed using the resolution of the display device.
4. Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4. Calculations are resolution base, so the change is difficult to adjust.
5. These were developed for vector graphics system.	5. These are developed for raster devices.
6. Object-based algorithms operate on continuous object data.	6. These operate on object data.
7. Vector display used for object method has large address space.	7. Raster systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. There are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.

10. If the number of objects in the scene increases, computation time also increases.	10. In this method complexity increase with the complexity of visible parts.

5.1.1 Back Face Detection Method

- When we project 3-D objects on a 2-D screen, we need to detect the faces that are hidden on 2D.
- Back-Face detection, also known as **Plane Equation method**
- It is an object space method in which objects and parts of objects are compared to find out the visible surfaces.
- Let us consider a triangular surface that whose visibility needs to decide. The idea is to check if the triangle will be facing away from the viewer or not. If it does so, discard it for the current frame and move onto the next one.
- Each surface has a normal vector. If this normal vector is pointing in the direction of the center of projection, then it is a front face and can be seen by the viewer.
- If this normal vector is pointing away from the center of projection, then it is a back face and cannot be seen by the viewer.
- A fast and simple object-space method for locating back faces
- A point (x,y,z) is "inside" a polygon surface with plane parameters A, B, C, D if

$$Ax + By + Cz + D < 0$$



- When an inside point is along the line of sight to the surface, the polygon must be a back face and so cannot be seen.

HOW TO DETECT?

- BY A SIMPLE TEST

If the z component of the normal vector is positive, then, it is a back face. If the z component of the vector is negative, it is a front face.

Principle:

i. Remove all surfaces pointing away from the viewer

ii. Eliminate the surface if it is completely obscured by other surfaces in front of it Render only the visible surfaces facing the viewer

iii. Back facing and front facing faces can be identified using the sign of V •N where V is the view vector and N is normal vector (N= Ai+Bj+Ck)

- We can simplify this test by considering the normal vector N to a polygon surface which has Cartesian components (A, B, C).
- There are two types of algorithm system to calculate V.N (V DOT N), i.e. ***Right-handed system and left-handed system.*** You can choose any one of them.

Algorithm For Left-Handed system

1. Start
2. Compute N for every face of the polygon
3. Check following

If (V • N > 0) then back face

else if (V • N < 0) then front face

else if (V • N= 0) then on line of view

else (Undefined)

4. End

Conclusion: if the Z component of the normal vector is positive, then it is a back face. If the Z component of the vector is negative, then it is a front face.

Algorithm for right-handed system

1. Start
2. Compute N for every face of the polygon
3. Check following condition

If (V • N < 0) then back face

else if (V • N >0) then front face

else if (V • N= 0) then on line of view

else (Undefined)

4. End

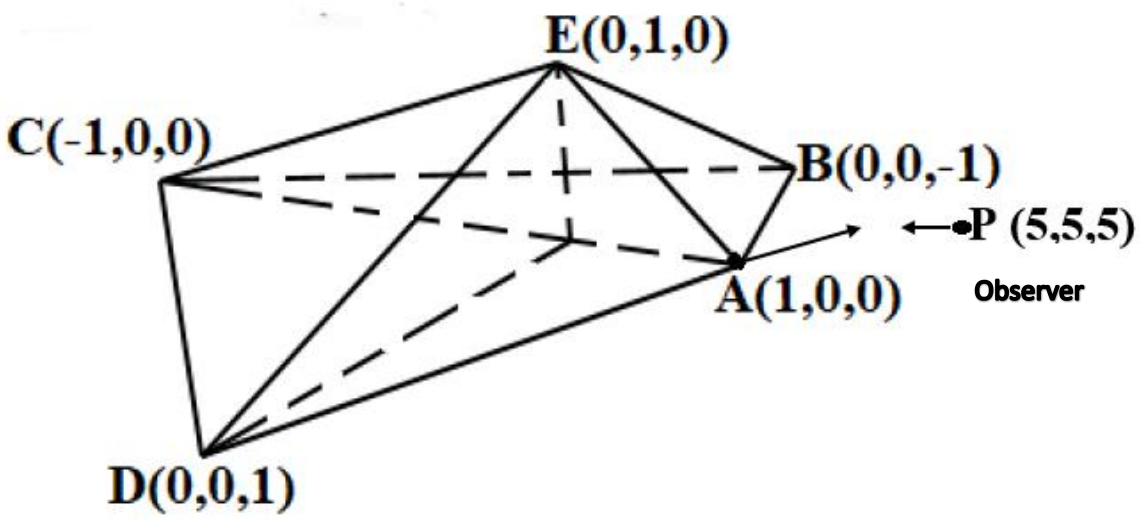
Conclusion: if the Z component of the normal vector is negative, then it is a back face. If the Z component of the vector is positive, then it is a front face.

Limitations of BDM

- This method works fine for convex polyhedral, but not necessarily for concave polyhedral.
- This method can only be used on solid objects modeled as a polygon mesh.

Numerical:

Q. Find the visibility for the surface AED where an observer is at P (5, 5, and 5).



Solution:

Here,

AE = (0-1) i + (1-0) j + (0-0) k = - i + j

AD= (0-1) i + (0-0) j + (1-0) k = - i + k

Step-1: Normal vector N for AED

Thus, N = AE x AD is calculated as

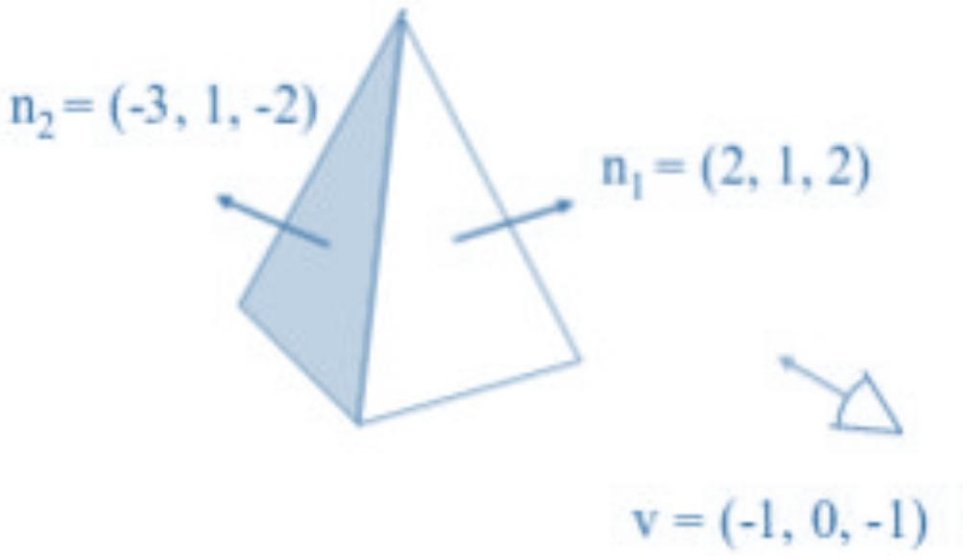
$$\begin{pmatrix} i & j & k \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = i\,(1-0) - j\,(-1+0) + k\,(0+1) = i + j + k$$

Step-2: If observer at P (5, 5, 5) so we can construct the view vector V from view point A (1, 0, 0) as: V = PA = (1-5) i + (0-5) j + (0-5) k = -4i - 5j - 5k

Step-3: To find the visibility of the object, we use dot product of view vector V and normal vector N as: V • N = (-4i - 5j – 5k) • (i + j + k) = -4-5-5 = -14< 0

This shows that the surface is visible for the observer.

Q. Find the visibility of n1 and n2 from V.



Solution

$N_1 \bullet v = (2, 1, 2) \cdot (-1, 0, -1) = -2 - 2 = -4,$

i.e $N_1 \bullet v < 0$

So, N_1 front facing polygon

$N_2 \bullet v = (-3, 1, -2) \cdot (-1, 0, -1) = 3 + 2 = 5$

i.e $N_2 \bullet v > 0$

So N_2 back facing polygon

5.1.2 Depth-Buffer Method

- It is also known as **z-buffer method**.
- Commonly used image-space method for detecting visible surface
- Developed by Edwin Catmull
- It is an image-space approach.
- The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface.
- 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.

OR

In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.

How this method works?

- ✓ 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 position (x,y) on the view plane, object depth is compared by z-values.

- ✓ It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers are used.

  1. **Depth buffer / (z- Buffer)** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq$ depth ≤ 1).

  2. **The frame buffer/ (refresh / image buffer)** is used to store the intensity value for each position (x, y).

The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.

Process:

i. Initially all the positions in depth buffer are set to 0, and refresh buffer is initialized to background color.
ii. Each surface listed in polygon table are processed one scan line at a time by calculating the depth (z-value) for each position (x, y).
iii. The calculated depth is compared to the value previously stored in depth buffer at that position.
iv. 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:

Start

1. Initialize depth buffer and refresh buffer so that for all buffer position (x, y)

i.e. Depthbuffer (x,y) =0, Refreshbuffer (x, y) = $I_{background}$.

2. For each position on each polygon surface, compare depth values to previously stored value in depth buffer to determine visibility.

2.1 Calculate the depth ***Z*** for each (x, y) position on polygon

2.2 If *Z* > depth (x, y) then set depth (x, y) = Z, refresh (x, y) = $I_{surface(x, y)}$

Where,$I_{background}$ = Intensity value for background color

$I_{surface\ (x, y)}$ = Intensity value for surface color at pixel position (x, y) on projected plane.

3. Repeat step 2 for all surfaces

4. After all surfaces are processed, the depth buffer contains the depth value of the visible surface and refresh buffer contains the corresponding intensity values for those surfaces.

5. End

How to calculate depth value / Z- value?

The depth values of the surface position (x, y) are calculated by plane equation

(Ax + By + Cz + D = 0) of surface then,

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

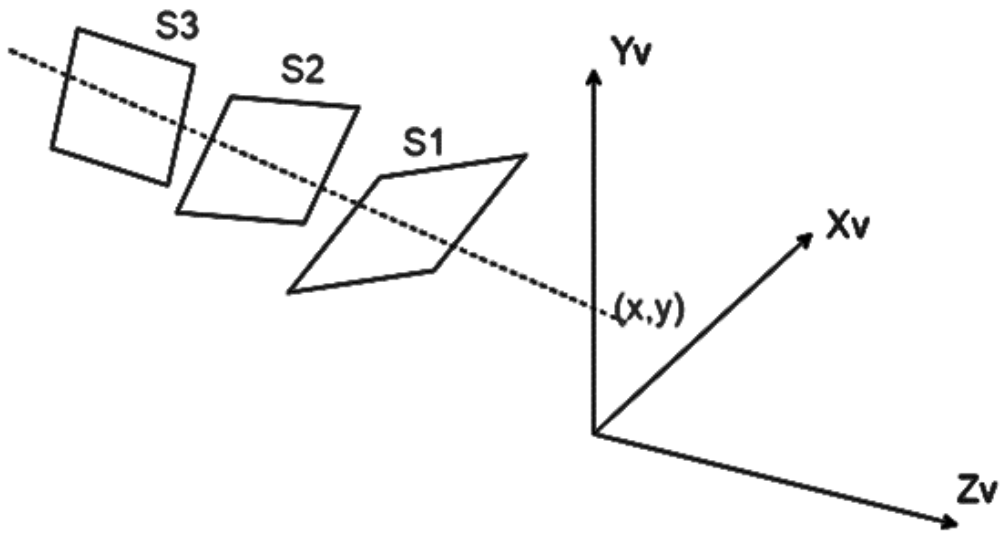
Let Depth Z' at position (x+1, y)

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

$$\Rightarrow Z' = Z - A/C$$

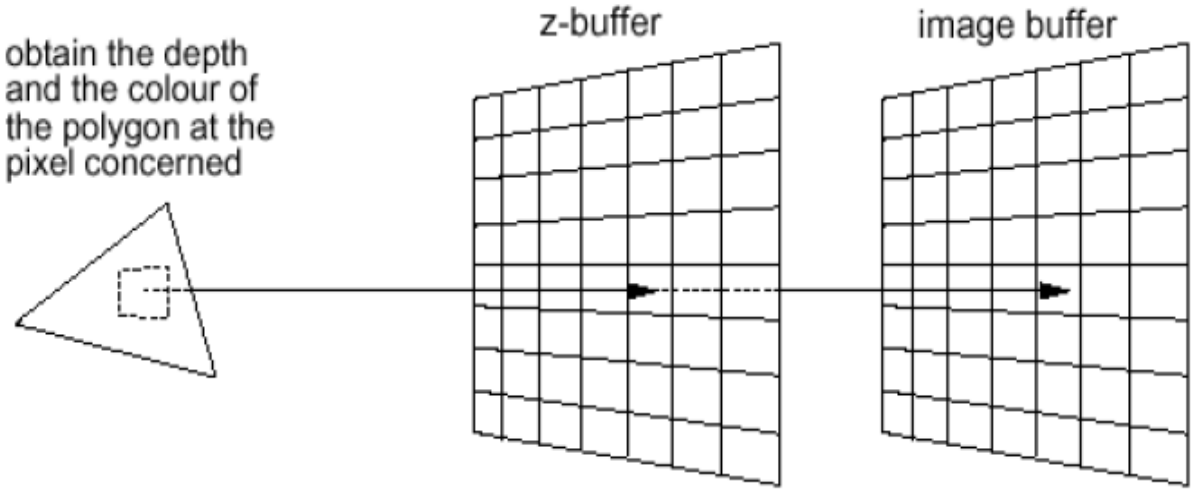
Where, A/C is constant for each surface so succeeding depth value across a scan line are obtained from preceding values by simple calculation.

Example.



- In above figure, at view plane position (x, y), the surface S_1 has the smallest depth from the view plane so is visible at that position.

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



Advantages of Z-buffer method

- It is easy to implement.
- It reduces the speed problem if implemented in hardware.
- It processes one object at a time.

Disadvantages of Z-buffer method

- It requires large memory.
- It is time consuming process

5.1.3 Scan Line Method

- It is an image space algorithm.
- It processes one line at a time rather than one pixel at a time.
- This algorithm records **edge list, active edge list**. So accurate bookkeeping is necessary.
- The edge list or edge table contains the coordinate of two endpoints.
- Active Edge List (AEL) contain edges a given scan line intersects during its sweep.
- The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.
- It is an extension of the scan-line algorithm for filling polygon interiors where, we deal with multiple surfaces rather than one.
- Each scan line is processed with calculating the depth for nearest view for determining the visible surface of intersecting polygon.
- When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.
- To facilitate the search for surfaces crossing a given scan line, we can set up an **active list** of edges from information in the edge table that contain only edges that cross the current scan line, sorted in order of increasing x.
- In addition, we define a **flag** for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface.
- Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned on; and at the rightmost boundary, it is turned off.

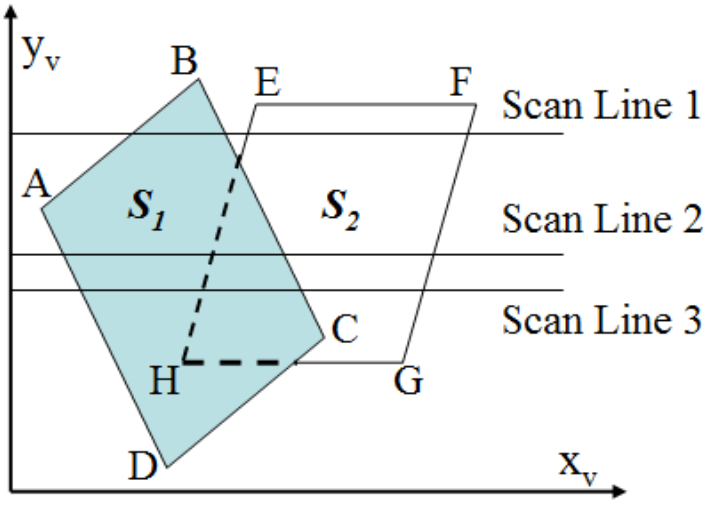


Figure: Scan Line crossing the projection of two surfaces, S1 and S2 in the view plane. Dashed lines indicate the boundaries of hidden surfaces.

- **The active list for scan line -1** contains information from the edge table for edges AB, BC, EH, and FG.
- For positions along this scan line between edges AB and BC, only the flag for surface S1 is on. Therefore, no depth calculations are necessary, and intensity information for surface S1, is entered from the polygon table into the refresh buffer.
- Similarly, between edges EH and FG, only the flag for surface S2 is on. NO other positions along scan line-1 intersect surfaces, so the intensity values in the other areas are set to the background intensity.

For scan lines 2 and 3 (line 2 and 3 have same intersection point and plane so no need to proceed 3 if 2 is done)

- The active edge list contains edges AD, EH, BC, and FG.
- Along scan line 2 from edge AD to edge EH, only the flag for surface S1, is on. But between edges EH and BC, the flags for both surfaces are on. In this interval, depth calculations must be made using the plane coefficients for the two surfaces.

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

5.1.4 A-Buffer Method

- Also known as **anti-aliased** or **area-averaged** or **accumulation buffer**
- Hidden face detection method suited to medium scale virtual memory computers.
- Extension of **depth-buffer (or Z Buffer)** method.
- Developed at Lucas film Studios for the rendering system "***Renders Everything You Ever Saw***" (REYES).
- As the depth buffer method can only be used for opaque object but not for transparent object, the A-buffer method provides advantage in this scenario.

- A buffer method requires more memory, but different surface colors can be correctly composed using it.
- A buffer method is slightly costly than Z-buffer method because it requires more memory in comparison to the Z-buffer method.
- It proceeds just like the depth buffer algorithm.
- Being a descendent of the Z-buffer algorithm, each position in the buffer can reference a linked list of surfaces.
- In A-buffer method, each pixel is made up of a group of sub-pixels. The final color of a pixel is computed by summing up all of its sub-pixels. Due to this accumulation taking place at sub-pixel level, it is called **accumulation buffer**.
- The key data structure in the A buffer is the **accumulation buffer**.

Each position in the A buffer has 2 fields

1. ***Depth field***

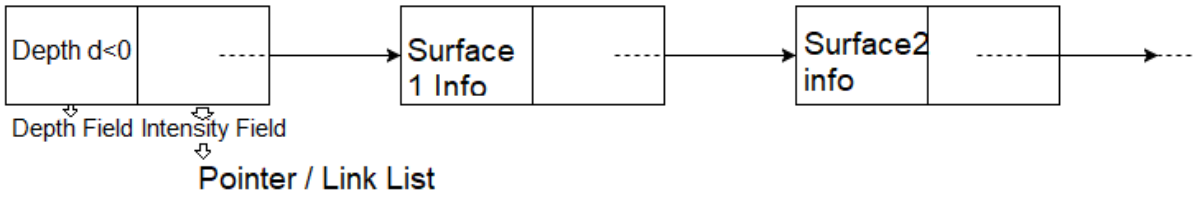
   - stores a positive or negative real number

2. ***Surface data field or Intensity field***

   - Stores surface intensity information or a pointer value to a linked list of surfaces that contribute to that pixel position.

- **If depth >= 0,** single surfaces exist i.e. the number stored at that position is the depth of a single surface overlapping the corresponding pixel area.

- **If depth < 0,** multiple surfaces exist i.e. multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data



- As shown in the above figure, **multiple-surface contributions** to the pixel intensity is indicated by depth < 0. The 2nd field, i.e., the intensity field then stores a pointer to a linked list of surface data

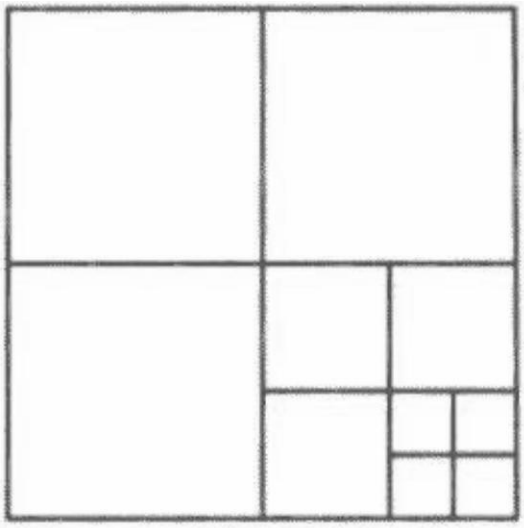
The surface buffer in the A buffer method includes:

- Depth
- Surface Identifier
- Opacity Parameter
- Percent of area coverage
- RGB intensity components
- Pointer to the next surface

5.1.5 Area Subdivision Method *(not in new course)*

- Uses image space method
- 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.

How it works?



1. Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.

2. Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.

Process

- Successively divide the area into four equal parts at each step.
- There are **four possible relationships** that a surface can have with a specified area boundary.

1. **Surrounding surface** − One that completely encloses the area.
2. **Overlapping surface** − One that is partly inside and partly outside the area.
3. **Inside surface** − One that is completely inside the area.
4. **Outside surface** − One that is completely outside the area.

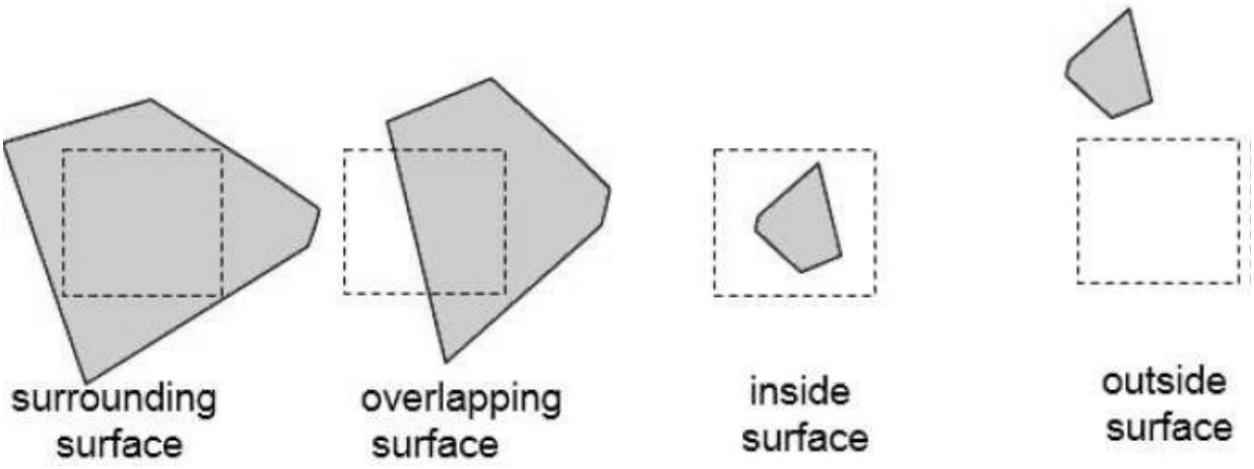


Figure: Area boundary relationship in Area Subdivision Method

The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true:

1. All surfaces are outside surfaces with respect to the area.
2. Only one inside, overlapping or surrounding surface is in the area.
3. A surrounding surface obscures all other surfaces within the area boundaries

5.1.6 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.
- 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.

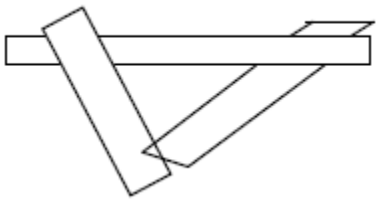
The conceptual steps that performed in depth-sort algorithm are

1. Surfaces are sorted in order of decreasing depth (z-coordinate).

2. Resolve any ambiguity this may cause when the polygons z-extents overlap, splitting polygons if necessary.

3. Surfaces are scan converted in order, starting with the surface of greatest depth.

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

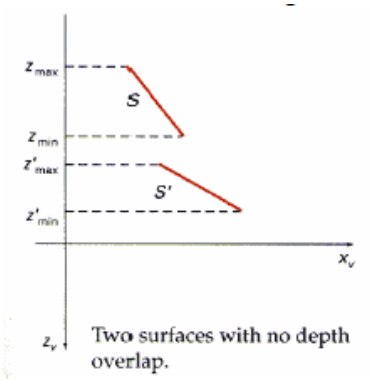
Example (view details in the book of Hearn and Baker)

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.

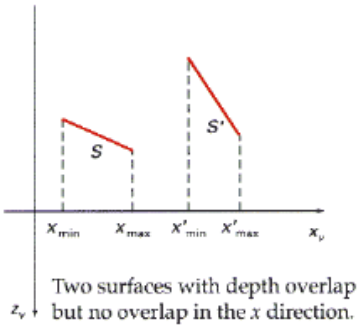
We make the following tests for each surface that overlaps with S. If any one of these tests is true, no reordering is necessary for that surface. The tests are listed in order of increasing difficulty.

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.

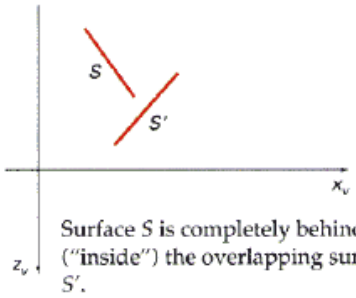
We perform these tests in the order listed and proceed to the next overlapping surface as soon as we find one of the tests is true. If all the overlapping surfaces pass at least one of these tests, none of them is behind S. No reordering is then necessary and S is scan converted.



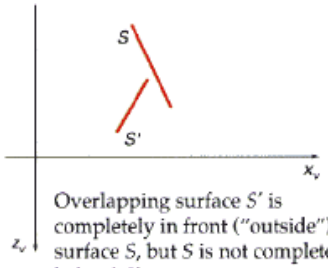
Two surfaces with no depth overlap.



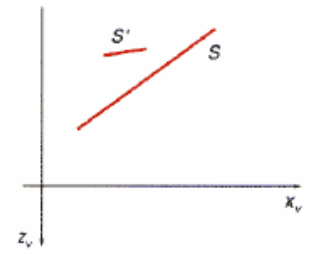
Two surfaces with depth overlap but no overlap in the x direction.



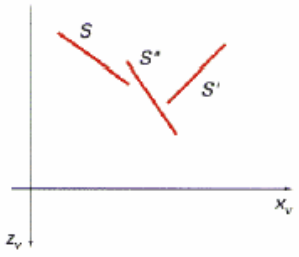
Surface S is completely behind ("inside") the overlapping surface S'.



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.

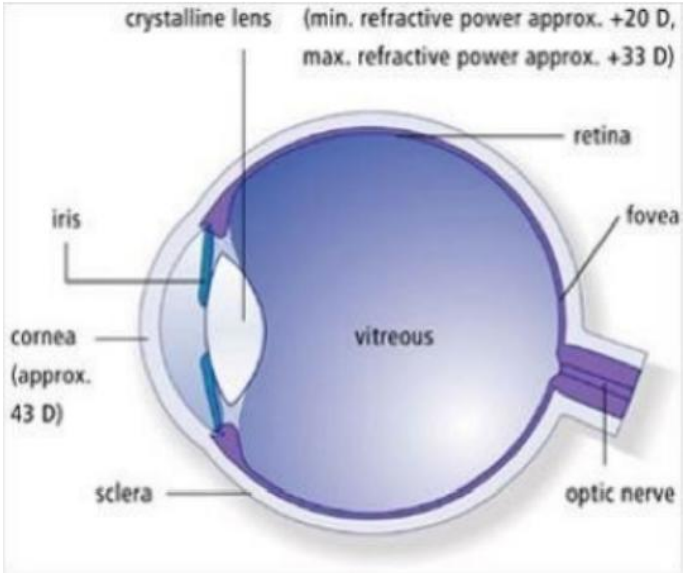
5.2 Illumination and Shading Methods

Before we start,

1. What the human eye (or virtual camera) sees is a result of light coming off of an object or other light source and striking receptors in the eye. In order to understand and model this process, it is necessary to understand different light sources and the ways that different materials reflect those light sources.
2. Trying to recreate reality is difficult.
3. Lighting calculations can take a VERY long time.
4. The techniques described here are heuristics which produce appropriate results, but they do not work in the same way reality works - because that would take too long to compute, at least for interactive graphics.
5. Instead of just specifying a single color for a polygon we will instead specify the properties of the material that the polygon is supposed to be made out of, (i.e. how the material responds to different kinds of light), and the properties of the light or lights shining onto that material.

Let's Create Background of content…

Human Eye…



- ✓ The eye works like a camera
- ✓ Sensors at the back of eye
- ✓ Sense the amount of light coming from different directions
- ✓ Similar to CMOS and CCDs

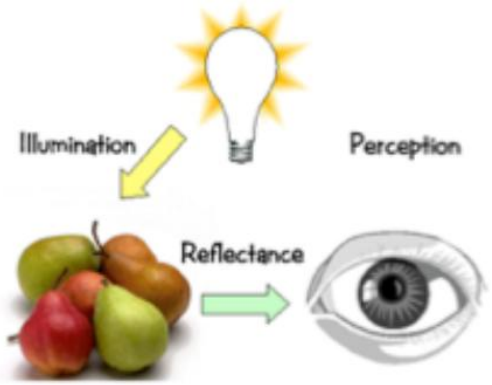
Bonus Knowledge…

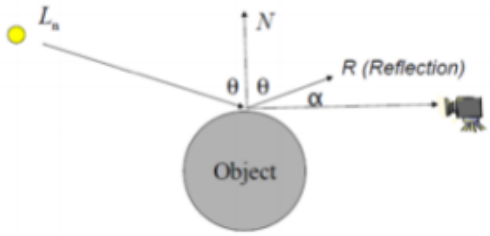
CCD (charge coupled device) and CMOS (complementary metal oxide semiconductor) image sensors are two different technologies for capturing images digitally. Each has unique strengths and weaknesses giving advantages in different applications.

Both types of imagers convert light into electric charge and process it into electronic signals.

Or, hey have to ***convert light into electrons****.*

Light coming into the eye…





- ✓ Position of the point the eye is looking at
- ✓ Position of the light source
- ✓ Color and intensity of light
- ✓ Vector from eye to point
- ✓ Normal vector of surface at point
- ✓ Physical properties of the object

Let's start the syllabus….

5.2.1 Illumination Theory and Model

Illumination:

- ✓ Act of illuminating, or suppling with light
- ✓ The state of being illuminated
- ✓ **CHAMAK (चमक)**

An observable property and effect of light is **Illumination**. The illumination of the subject of a drawing or painting is a key element in creating an artistic piece, and the interplay of light and shadow is a valuable method.

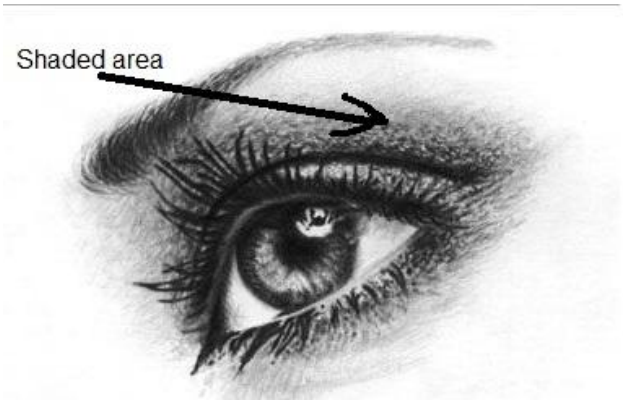
The placement of the light sources can make a considerable difference in the type of message that is being presented.

For Example:

Multiple light sources can wash out any wrinkles in a person's face, for instance, and give a more youthful appearance. In contrast, a single light source, such as harsh daylight, can serve to highlight any texture or interesting features.

Shading

- ✓ **D**arkness where light, particularly sunlight, is blocked
- ✓ A variety of color, in particular one, obtained by adding black.
- ✓ Shading is the implementation of the illumination model at the pixel points or polygon surfaces of the graphics objects.
- ✓ Shading model is used to compute the intensities and colors to display the surface.



***Note**: Figure of eye just indicates the random picture obtained by adding black. Picture may not have perfect meaning in this topic but it may help to get concept of shading.*

Intensity

- ✓ Degree of strength
- ✓ Quantity of illumination

Illumination model

- ❖ Also called a lighting model / shading model
- ❖ It is used to calculate the intensity of light that we should see at a given point on the surface of an object.

Surface rendering

- ❖ It is a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.
- ❖ A surface-rendering algorithm uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.
- ❖ Surface rendering can be performed by applying the illumination model to every visible surface point.

Light Source

- ❖ When we view an opaque non-luminous object, we see reflected light from the surfaces of the object. The total reflected light is the sum of the contributions from a single light source or illuminated nearby objects.
- ❖ When light is incident on an opaque surface part of it is reflected and part of it is absorbed.
- ❖ Sometimes light sources are referred as light emitting object and light reflectors. Generally light source is used to mean an object that is emitting radiant energy e.g. Sun.

1. **Point source:**

    - ❖ Point source is the simplest light emitter e.g. light bulb
    - ❖ All light rays originate at a point and radially diverge

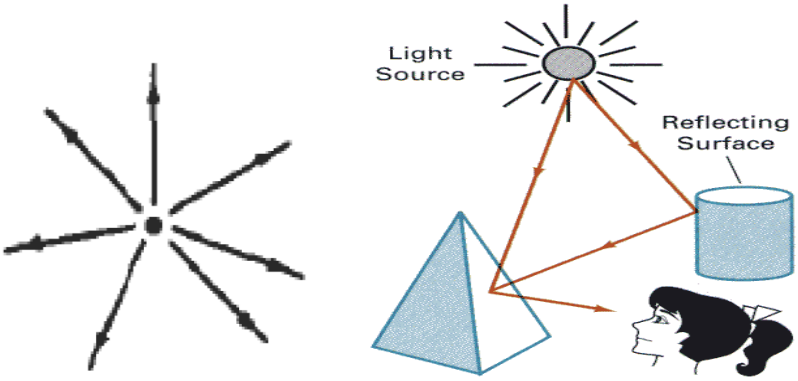


Figure: diverging ray path from the point light source

2. **Parallel source**

- ❖ Light rays are all parallel. May be modeled as a point source at infinite distance (the sun)

3. **Distributed source**

- ❖ All light rays originate at a finite area in space.
- ❖ It models a nearby source, such as a fluorescent light.

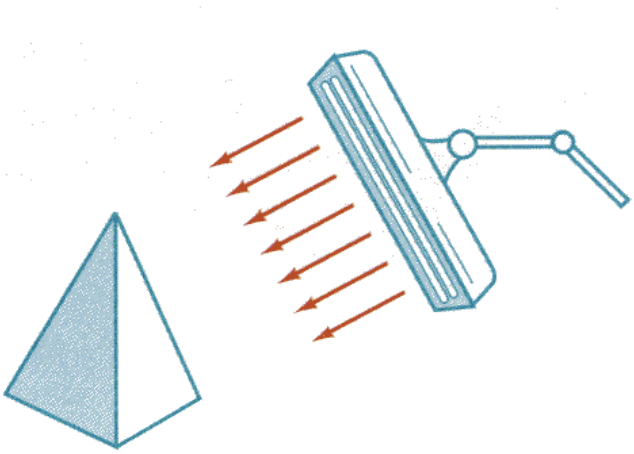


Figure*:* An object illuminated with a distributed light source

Illumination models:

- Illumination models are used to calculate light intensities that we should see at a given point on the surface of an object.
- Lighting calculations are based on the
  - optical properties of surfaces,
  - The background lighting conditions and
  - The light source specifications.
- All light sources are considered to be point sources, specified with a co-ordinate position and an intensity value (color).

Different illumination models are:

5.2.1.1 Ambient Light / Background light

- This is a simplest illumination model.
- We can think of this model, which has no external light source-self-luminous objects.
- A surface that is not exposed directly to light source still will be visible if nearby objects are illuminated.
- The combinations of light reflections form various surfaces to produce a uniform illumination called ambient light or background light.

OR

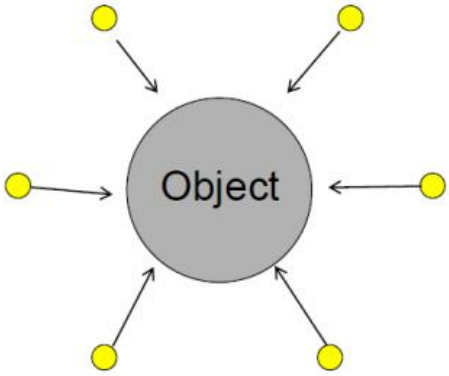
Light that is already present in a scene, before any additional lighting is added is called Ambient light.

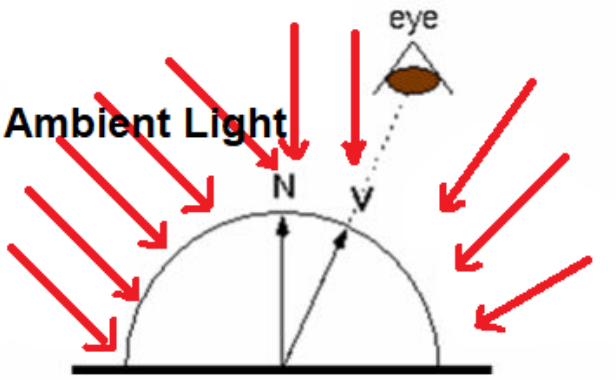
- It usually refers to natural light, either outdoors or coming through windows etc.

- It can also mean artificial lights such as normal room lights.

- Ambient light has no spatial or directional characteristics and amount on each object is a constant for all surfaces and all directions.

- Ambient light has a uniform intensity in all directions. It causes a surface to be uniformly illuminated at any viewing position.





- In this model, illumination can be expressed by an illumination equation in variables associated with the point on the object being shaded

The equation expressing Ambient Light model is given by

$$I = K_a$$

Where, I is the resulting intensity and is the object's intrinsic intensity.

And, K_a → is the ambient-reflection coefficient which determines the amount of light reflected from an object's surface. Value of K_a ranges from 0 to 1

If we assume that ambient light impinges equally on all surfaces from all direction, then

$$I = I_a K_a$$

Where, Ia → is the intensity of ambient light.

Reflection: Diffuse and specular

When light is reflected (or refracted) from a surface, it does not reflect all light that enters the surface. Part of the energy will be absorbed by the surface. The reflection coefficient represents the amount of light reflected from an object's surface.

5.2.1.2 Diffuse Reflection:

- Lambertian Reflection
- When light hits an object with a rough surface, it is reflected in all directions and Objects illuminated by ambient light are uniformly illuminated across their surfaces even though light are more or less bright in direct proportion of ambient intensity.
- Amount of light hitting the surface depends on the angle between the normal vector and the incident vector of the incoming light.

  OR

  The object's brightness varies from one part to another, depending on the direction of and distance to the light source.

- The larger the angle (up to 90 degrees), the larger the area the incident light is spread over.
- **Diffuse reflections** are constant over each surface in a scene, independent of the viewing direction

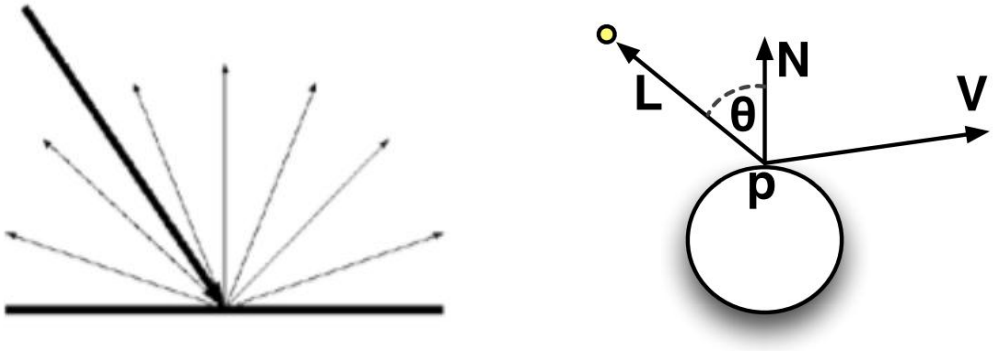


Figure: diffuse Reflection

- Diffuse lighting of an object approximates how photons will bounce off of the object's surface in many different directions.

- In diffuse reflection, we consider the surfaces (surface called. "Ideal diffuse reflectors / Lambertian reflectors), which reflect light with equal intensity in all directions (i.e. independent of the viewing direction). The amount of light energy that falls on a given area and hence the intensity of the reflection is proportional to the **cosine** of the angle of incident (i.e. cos θ). This kind of surfaces are normally dull or matt. **This mechanism is governed by Lambert's cosine law**

$$I_{diff} = K_d\, I_l \,\text{Cos}\,\theta$$

Where I_l is the intensity of the point light source.

If N is unit vector normal to the surface & L is unit vector in the direction to the point slight source then

$$I_l,\text{diff} = K_d\, I_l\, (N.L)$$

In addition, many graphics packages introduce an ambient reflection coefficient $K\alpha$ to modify the ambient light intensity I_α

$$I_{diff} = K\alpha\, I\alpha + K_d\, I_l\, (N.L)$$ **is Required Expression.**

OR

Above derivation can be done in this way too...

$I = I_p k_d \cos\theta$

Where, I_p Light intensity

θ Angle between normal vector and direction to light source

k_d is Diffuse reflectivity

Note: there is no dependence on the angle between the direction to the camera and the surface normal.

5.2.1.3 Specular Reflection (phong Model)

- Also called regular reflection.
- Mirror-like reflection of waves such as light from a surface.
- In this process, each incident ray is reflected at the same angle to the surface normal as the incident ray, but on the opposing side of the surface normal in the plane formed by incident and reflected rays.
- Result of this reflection is that an image reflected by the surface is reproduced in mirror-like (*specular*) fashion
- When we look at an illuminated shiny surface, such as polished metal, a person's forehead, we see a highlight or bright spot, at certain viewing direction. Such phenomenon is called specular reflection.

- It is the result of total or near total reflection of the incident light in a concentrated region around the specular reflection angle.

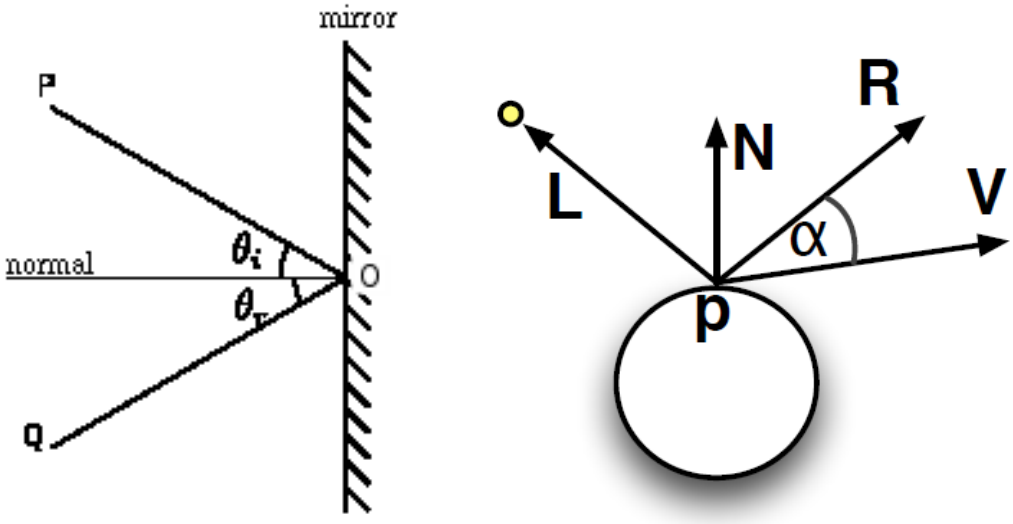


Figure: Specular-reflection equals the angle of incidence θ

Where, N - Unit vector normal to surface at incidence point

R - Unit vector in the direction of ideal specular reflection.

L - Unit vector directed to words point light source.

V - Unit vector pointing to the viewer from surface.

α - Viewing angle relative to the specular reflection direction.

Intensity Calculation

$$I = I_p \, k_s \cos^n \alpha$$

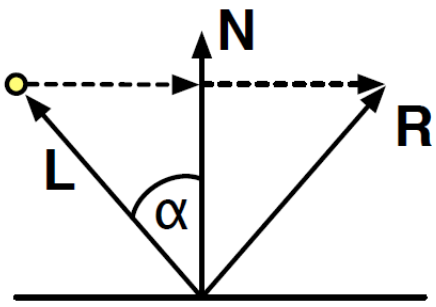
Where,

I_p → Light intensity

α → Angle between reflection vector and direction to camera

k_s → Specular reflectivity and

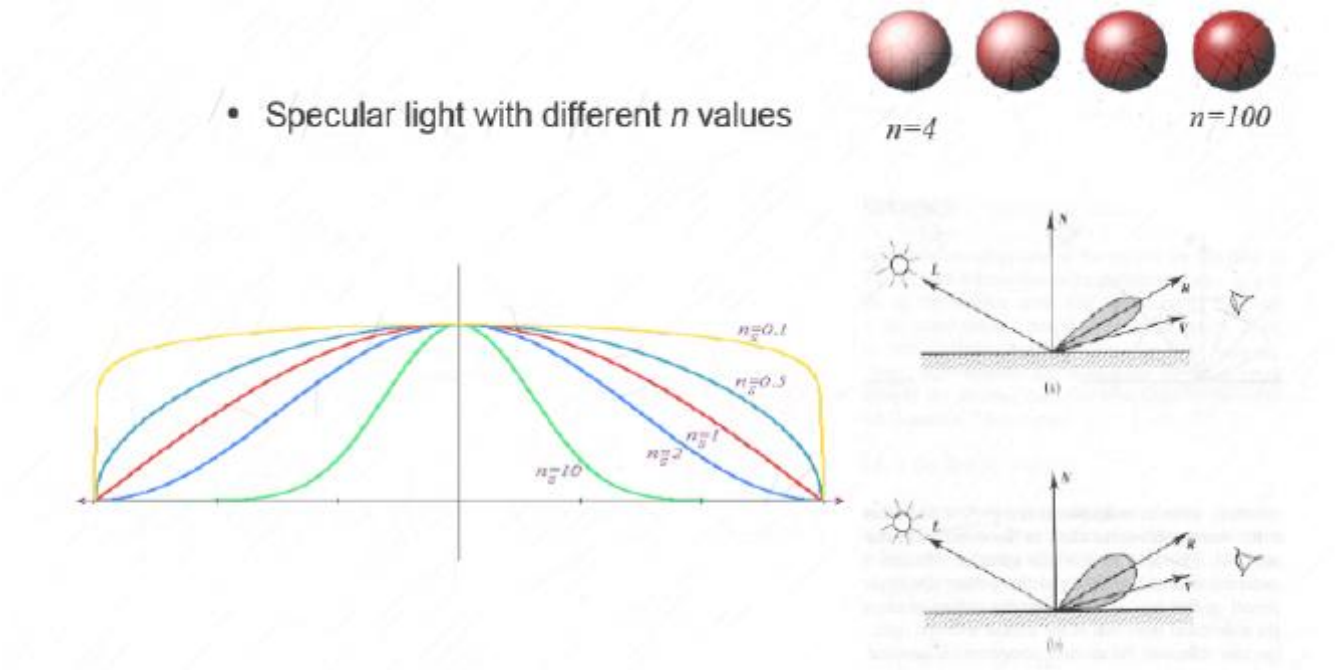
n → Specular intensity



Specular Reflection $R = 2N\,(N \cdot L) - L$

Note: *how this reflection (R) is obtained will described in phong modle below. No need to describe in specular diffusion.*

- For ideal reflector (perfect mirror), incident light is reflected only in the specular reflection direction i.e. V and R coincides (α = 0).
- Shiny surfaces have a narrow specular-reflection range (narrow α), and dull surfaces have a wider reflection (wider α).

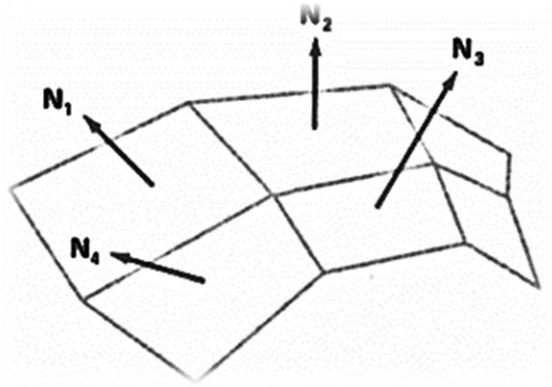


5.2.2 Polygon Surface Shading Method

- **Shading model** is used to compute the intensities and colors to display the surface.
- **A shading model** is used in computer graphics to simulate the effects of light shining on a surface.
- The intensity that we see on a surface is dependent upon
  - Properties of the surface (i.e. The surface characteristics e.g. Shining, matte, dull, and opaque or transparent)
  - The type of light sources (i.e. Properties of the illumination falling on it)
- Shading refers to the process of altering the color of an object / surface / polygon in the 3D scene, based on following facts (may more facts) to create a photo realistic effect.
  - the surface's angle to lights, its distance from lights
  - Its angle to the camera and material properties (e.g. bidirectional reflectance distribution function).
- The principal surface property is its reflectance, which determines how much of the incident light is reflected.
- If a surface has different reflectance for the light of different wavelengths, it will appear to be colored.
- Shading models determine the shade of a point on the surface of an object in terms of a number of attributes.
- Shading is performed during the rendering process by a program called a shader.

In order to produce a realistic image with various kinds of reflection, there are 3 common shading methods which are mainly applied to polygons:

5.2.2.1. Constant Shading



- Also called **Flat shading / Faceted 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. **(i.e. one lighting calculation per polygon)**
- Fast and simple but cannot model specular reflection.
- Problem of intensity discontinuity between polygons (The light intensity is computed based on the normal of each polygon surface. Since different polygons have different normal, the light intensity computed may be quite different).
- This approach is valid if several assumptions are true:
    1. The light source is sufficiently far so that **N.L** is constant across the polygon face.

    2. The viewing position is sufficiently far from the surface so that **V.R** is constant over the surface

    3. The object is a polyhedron and is not an approximation of an object with a curved surface.

Remarks: *Even if all of these 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.*

Interpolated Shading:

Remarks: *This is not in syllabus of PU but we have to know for the understanding of Gouraud and phong shading*)

- An alternative to evaluating the illumination equation at each point on the polygon, we can use the interpolated shading, in which shading information is linearly interpolated across a triangle from the values determined for its vertices. Gouraud generalized this technique for arbitrary polygons.

- This is particularly easy for a scan line algorithm that already interpolates the z-value across a span from interpolated z-values computed for the span's endpoints.

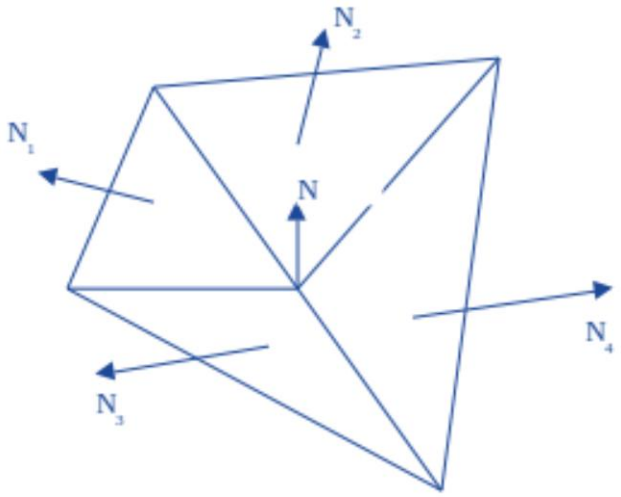
5.2.2.2 Gouraud Shading

- Henri Gouraud, French computer scientist
- Also called intensity interpolating shading or color interpolating shading eliminates intensity discontinuities that occur in flat shading
- Method of simulating the effects of light and color across the surface of an object, based on estimates of the surface normal of each vertex in a polygonal 3D model.

- **one lighting calculation per vertex (i.e.** The intensity value is calculated once for each vertex of a polygon**)**

- Each polygon has one normal vector per vertex; the color of each vertex is computed and then interpolated across the surface of the polygon.
- The interpolation of color values can cause bright or dark intensity streaks, called the **Mach-bands (***An optical illusion, discovered by Ernst Mach***)**, to appear on the surface
- This method eliminates the intensity discontinuity problem, but still not model the specular reflection correctly
- 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

Consider a polygon as shown in figure below,



Process:

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

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

From the above polygon,

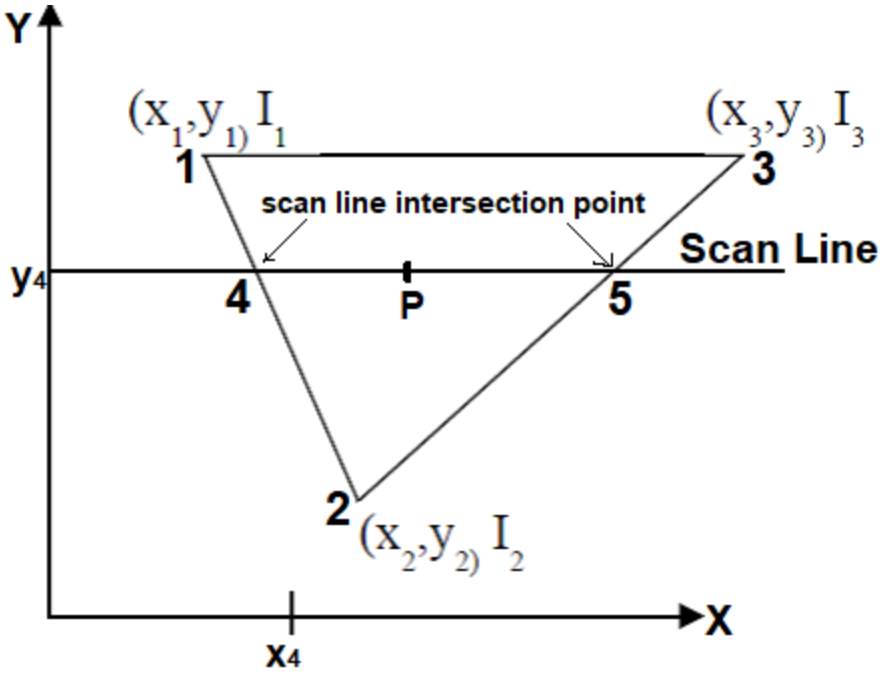
$$N_v = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

Where,

N_v is normal vector at a vertex sharing four surfaces as in above figure.

Step 2: Gouraud model is one lightening calculation per vertex therefore, **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 consider following figure.



In This figure, the intensity of vertices 1, 2, 3 are I_1, I_2, I_3 respectively, are obtained by averaging normal of each surface sharing the vertices and applying an 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 **<u>at point 4</u>** 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 **<u>at point 5</u>** 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,

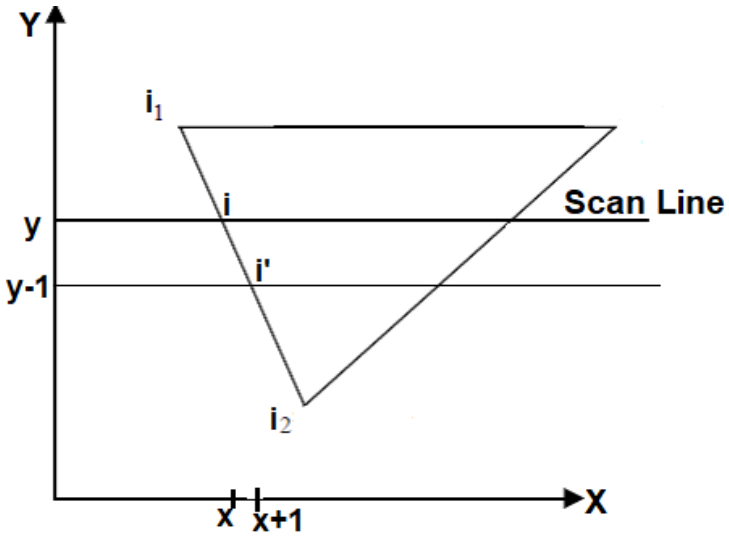
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

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:

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

Which is required Expression

Furthermore, for next scan line



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.

Problems with Gouraud shading

In specular reflection the highlight can be sharp, depending on the shape of $\cos^n \alpha$

- Gouraud shading interpolates linearly and so can make the highlight much bigger
- Gouraud shading can miss highlights that occur in the middle of a polygon
- Highlights on the surface are sometimes displayed with anomalous shape and linear intensity interpolation can cause bright or dark intensity streak called mach-bands.

Advantages: Removes intensity discontinuities at the edge as compared to constant shading.

5.2.2.3 Phong Shading

- **Also called normal vector interpolation shading**
- Lighting computation is performed at each pixel
- More accurate method for rendering a polygon surface is to interpolate normal vector and then apply illumination model to each surface point.
- It displays more realistic highlights and greatly reduces the mach-band effect
- Each rendered polygon has one normal vector per vertex; shading is

Performed by interpolating the vectors across the surface and computing the color for each point of interest.

- Interpolating the normal vectors gives a reasonable approximation to a smoothly-curved surface while using a limited number of polygons

- This method greatly reduces the Mach-band problem but it requires more computational time

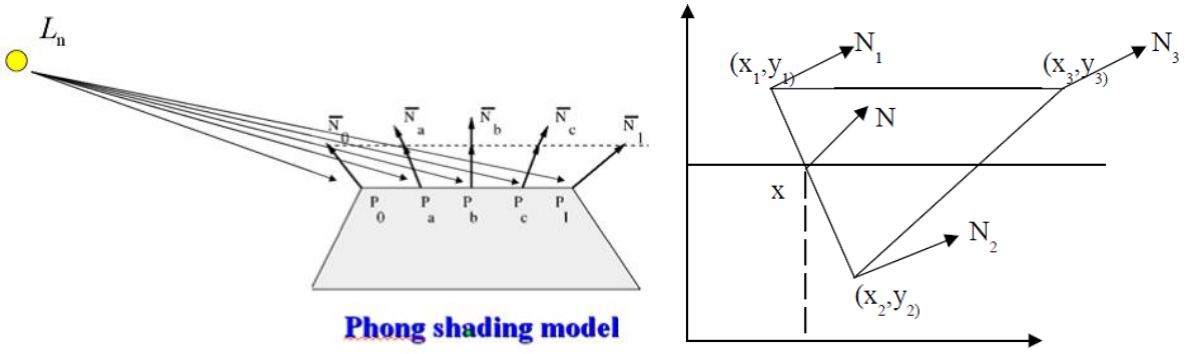


Figure: a) Phong Sheding Model **b**) Polygon surface having normal vector on Every vertex

- Polygon surface is rendered with Phong 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 the pixel intensities for the surface point.

In figure b), N1, N2, N3 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.

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

5.2.2.4 Fast Phong Method

- 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 normal, 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.

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

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

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

Re writing this expression in generic form,

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

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 (i) can be expressed as Taylor series expansions and retains terms up to second degree in x and y. These yields

$I_{diff}(x,y) = T_5 x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0$

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

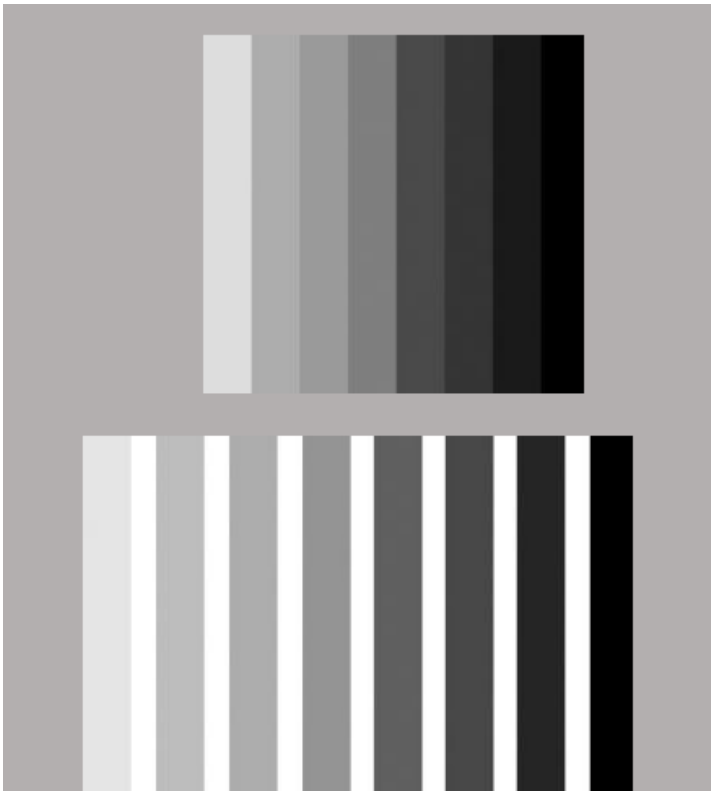
- This method still takes twice as long as in Gouraud shading.
- Normal Phong shading takes six to seven times that of Gouraud shading.

Mach Bands:

- Physicist Ernst Mach (1838–1916).
- Mach noticed that when two bars, one dark and one bright, are next to each other, you see little bands of extra dark at the edge of the dark band and extra light at the edge of the light bar.
- Mach bands is often based on the concept of lateral inhibition. Lateral inhibition is a process by which the collection of light by retinal cells in the eye is affected by the light collection of neighboring cells. In nature, this helps boost the perception of object edges, making it easier to see the edge of an object that might otherwise not be noticed.
- Mach Bands are an optical illusion that leads us to perceive densities that do not exist in reality.
- **Mach bands** are an optical illusion consisting of an image of two wide bands, one light and one dark, separated by a narrow strip with a light-to-dark gradient. The human eye perceives two narrow bands of different *brightnesses* either side of the gradient that are not present in the original image. (see below image)

Figure: Mach Bands Effect

If you look closely at this grey scale, it appears that each band of grey is not a consistent density. The left side of each band appears slightly darker than the right. This is another example of the Mach band optical illusion. →



These are the same grey blocks separated by a white strip. They now appear as solid shades of grey. →

Depth Cueing

- Depth cueing aims to improve the 3D effect by displaying objects at the front more brightly than those at the back. The extent to which this improves 3D perception varies according to what is on display and it may not always be effective.
- Depth cueing is implemented by having objects blend into the background color with increasing distance from the viewer.
- It is the process of rendering distant objects at a lower intensity than near ones, hence giving them the appearance of depth**.**
- Depth cuing indicates the process in which the lines closest to the viewing position are displayed with the highest intensities, and lines farther away are displayed with decreasing intensities.
- Depth cueing is applied by choosing maximum and minimum intensity (or color) values and a range of distances over which the intensities are to vary.

5.3 Color Models

- A color model is an abstract numerical system for describing color, using three or four main numbers to represent primary colors.
- A color model is an orderly system for creating a whole range of colors from a small set of primary colors.
- There are two types of color models, those that are subtractive and those that are additive.
- Additive color models use light to display color while subtractive models use printing inks.
- Colors perceived in additive models are the result of transmitted light. Colors perceived in subtractive models are the result of reflected light.
- There are several established color models used in computer graphics, but the two most common are

1. **RGB model (Red-Green-Blue)** for computer display and
2. **CMYK model (Cyan-Magenta-Yellow- Key / black)** for printing

5.3.1 RGB model (Red-Green-Blue)

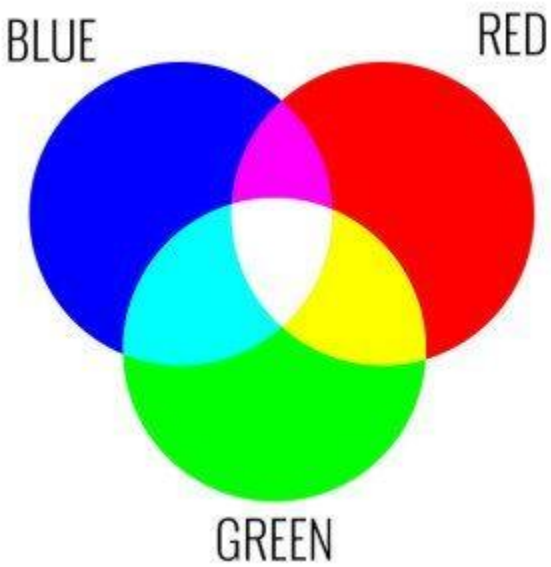


Figure: RGB color model

- RGB stands for Red, Green, Blue
- RGB is primarily used for designing elements that will be displayed on a television set, a computer monitor, a mobile phone, or any other kind of light source.
- RGB (Red, Green and Blue) is the color space for digital images. Use the RGB color mode if your design is supposed to be displayed on any kind of screen.
- It is a device dependent color mode.
- A light source within a device creates any color you need by mixing red, green and blue and varying their intensity. This is known as additive mixing or additive color model, means all colors begin as black darkness and then red, green and blue light is *added* on top of each other to brighten it and create the perfect pigment. When red, green and blue light is mixed together at equal intensity, they create pure white.
- Designers can control aspects like saturation, vibrancy and shading by modifying any of the three source colors. Because it's done digitally, the designer manipulates how the light on the screen manifests to create the color they want.
- If the end destination of your design project is a digital screen, use the RGB color mode. This would go for anything that involves computers, smartphones, tablets, TVs, cameras, etc.
- Following design project involves RGB pattern:

  - **web & app design**
    - icons
    - buttons
    - graphics
  - **Branding**
    - online logos
    - online ads
  - **Social media**
    - images for posts

- profile pictures
- profile backgrounds

- **Visual content**

  - video
  - digital graphics
  - infographics
  - photographs for website, social media, or apps

Summary:

- ✓ Additive color model
- ✓ For computer displays
- ✓ Uses light to display color
- ✓ Colors result from transmitted light
- ✓ Red + Green + Blue = White

5.3.2 CMYK model (Cyan-Magenta-Yellow-black)

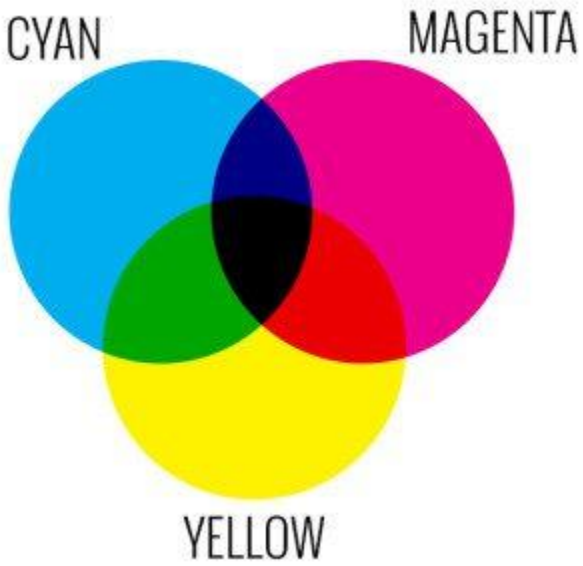


Figure: CMYK color Model

- CMYK (Cyan, Magenta, Yellow, Key / Black) is the color space for printed materials.
- CMYK is also a device dependent color mode
- It is most frequently used to this day as the main color mode used in printing, whether it's posters, brochures, business cards, books, or magazines.
- A printing machine creates images by combining CMYK colors to varying degrees with physical ink. This is known as subtractive mixing. All colors start as blank white, and each layer of ink *reduces* the initial brightness to create the preferred color. When all colors are mixed together, they create pure black.
- Use CMYK for any project design that will be physically printed, not viewed on a screen. If you need to recreate your design with ink or paint, the CMYK color mode will give you more accurate results.
- Following project involves CMYK pattern:

- **Branding**
  - business cards
  - stationary
  - stickers
  - signs & storefronts
- **Advertising**
  - Billboards
  - Posters
  - Flyers
  - vehicle wraps
  - brochures
- **Merchandise**
  - T-shirts, hats and other branded clothing
  - Promotional swag (pens, mugs, etc.)

- **Essential materials**

  - product packaging
  - restaurant menus

Summary:

- ✓ Subtractive color model
- ✓ For printed material
- ✓ Uses ink to display color
- ✓ Colors result from reflected light
- ✓ Cyan + Magenta + Yellow = Black

*** End of Unit 5 ***