ADVANCED TOPICS

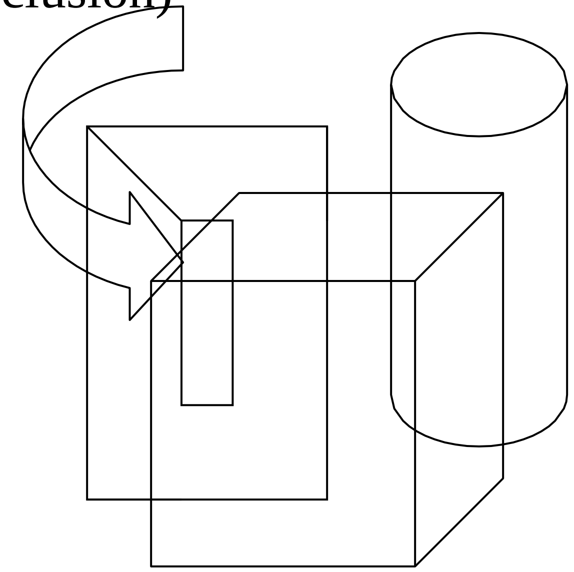
Unit-4: Advanced Topics

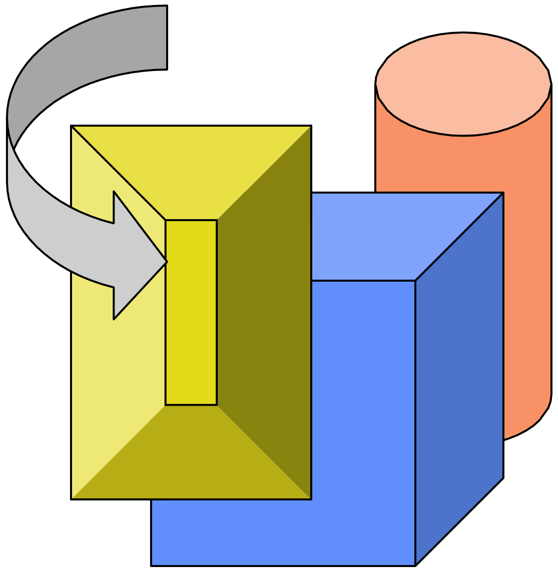
- Classification of Visible-Surface Detection Algorithms
- Back-Face Detection
- Depth Buffer Method/ Z Buffer Method
- Scan Line
- Depth Sorting Method
- Light source
- Basic Illumination Models/ Shading Model/ Lighting Model
- Ambient Light
- Diffuse Reflection
- Specular Reflection
- Properties of Light
- Color Model

Visible surface detection

Problem:

- Given a set of 3D objects and a viewing specifications, determine which lines or surfaces of the objects should be visible.
- A surface might be occluded by other objects or by the same object (self occlusion)

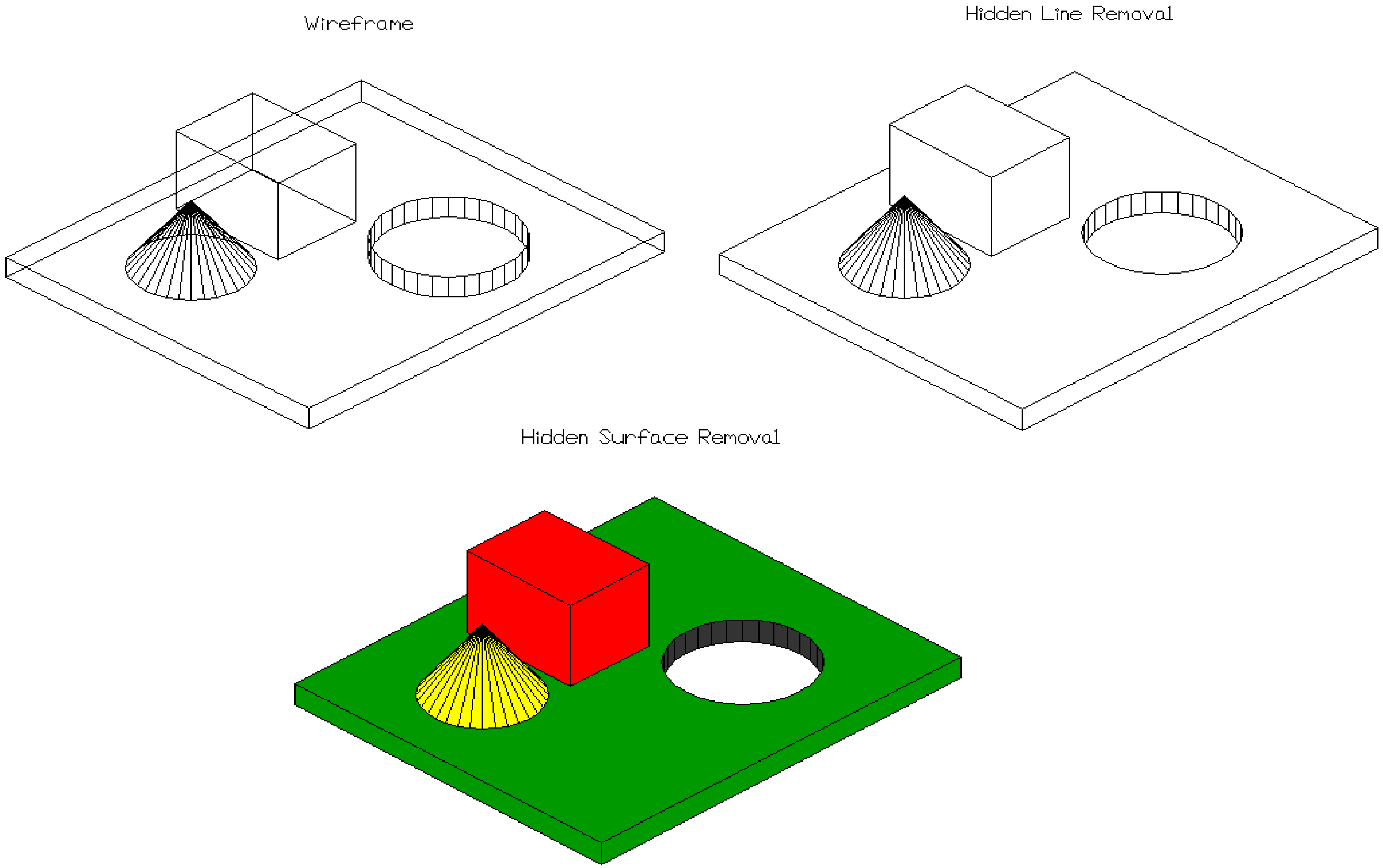




Classification of Visible-Surface Detection Algorithms

- It is broadly divided into two parts,
  1. Object-Space methods (e.g. Back-Face Detection)
     - which parts of each object are visible.
  2. Image-Space methods (e.g. Depth Buffer Method)
     - determine what is visible at each pixel.
- Object space method compares objects and parts of objects to each other within the scene definition to determine which surface is visible.
- In image space algorithm visibility is decided point by point at each pixel position on the projection plane.

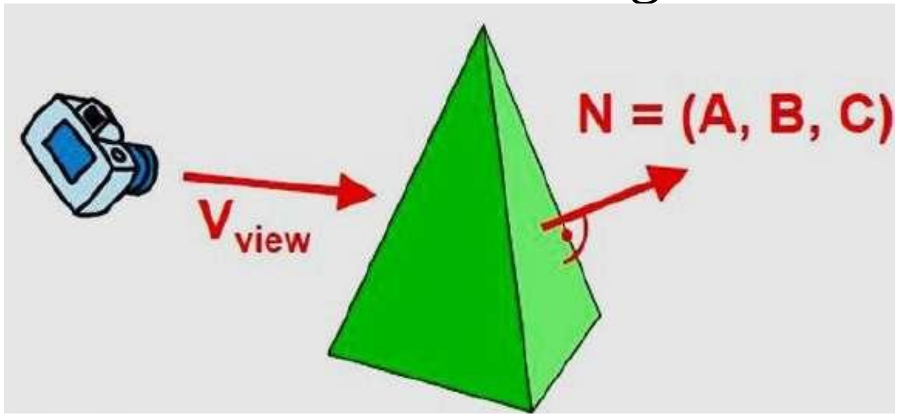
Hidden Surface Removal



Back-Face Detection

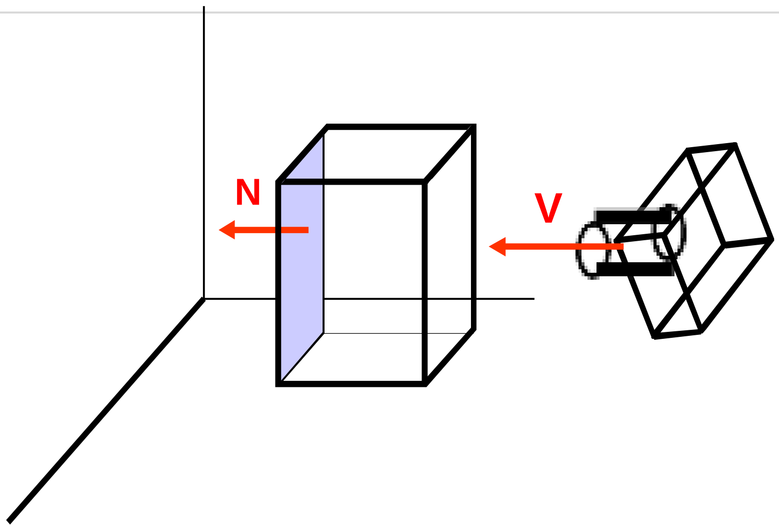
- Back-Face Detection is simple and fast **object space method.**
- Identifies back faces of polygon based on the inside-outside tests.
- A point (x, y, z) is inside if $Ax + By + Cz + D < 0$.

  where A, B, C, and D are constants and this equation is nothing but equation of polygon surface.
- We can simplify test by taking normal vector $N = (A, B, C)$ of polygon surface and vector V in viewing direction from eye.



Source: www.tutorialspoint.com

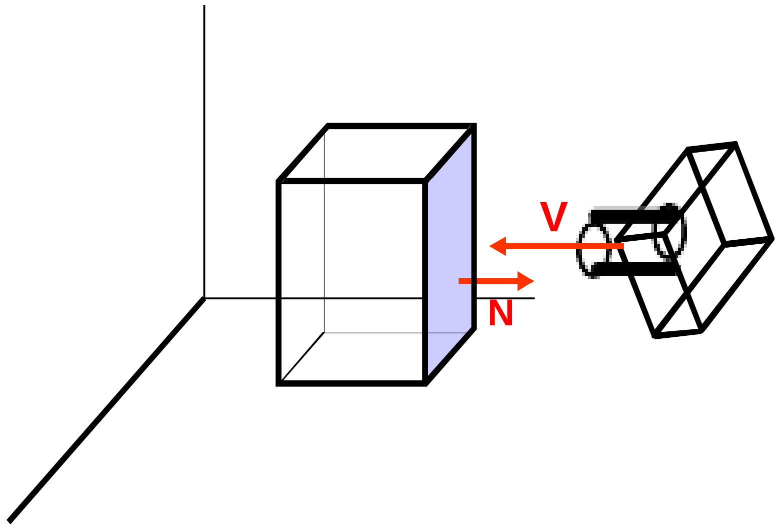
Back-Face Detection(Method 1)



$$\mathbf{V} \times \mathbf{N} > 0 : \text{back face}$$

Three possibilities:

V•N> 0	back face
V•N< 0	front face
V•N= 0	on line of view



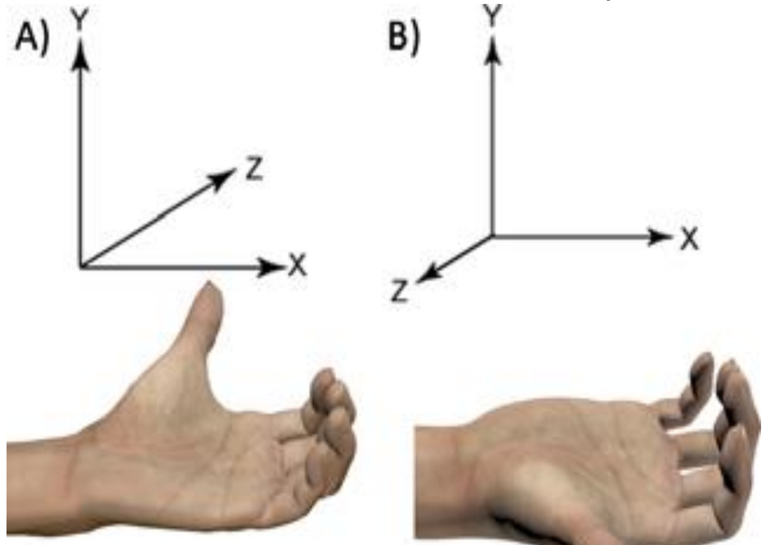
$$\mathbf{V} \cdot \mathbf{N} < 0 : \text{front face}$$

Back-Face Detection

Method 2:

- If we convert object description in projection coordinates and our viewing direction is parallel to z_v then $V = (0,0,V_z)$ and,

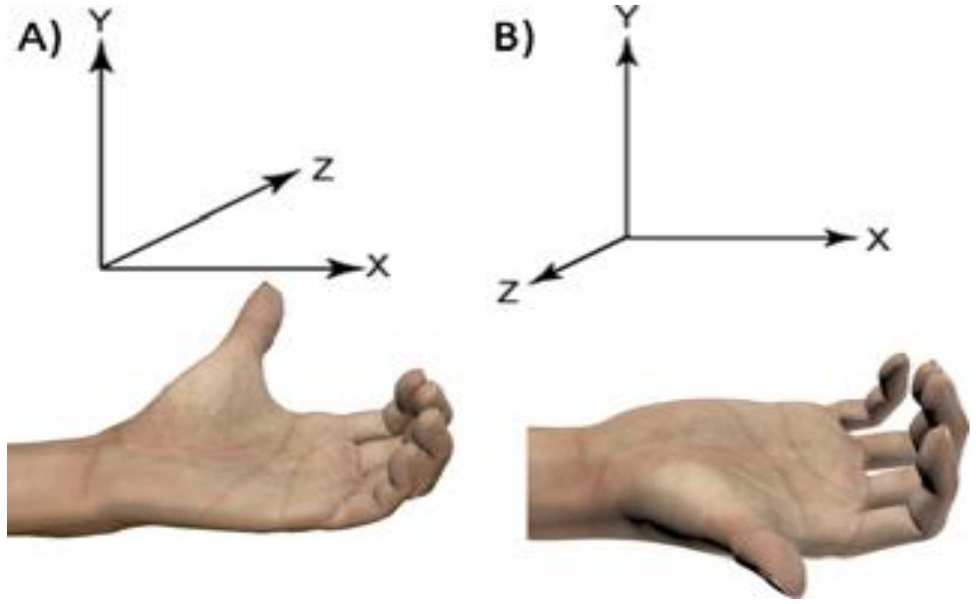
  $V \cdot N = V_z C.$
- So now we only need to check sign of C.
- In **Right handed viewing system** V is along negative z_v axis. And in that case
- **If $C \leq 0$ polygon is back face.**



A) Left Handed Viewing System(+ve Z)
B) Right Handed Viewing System (-ve Z)

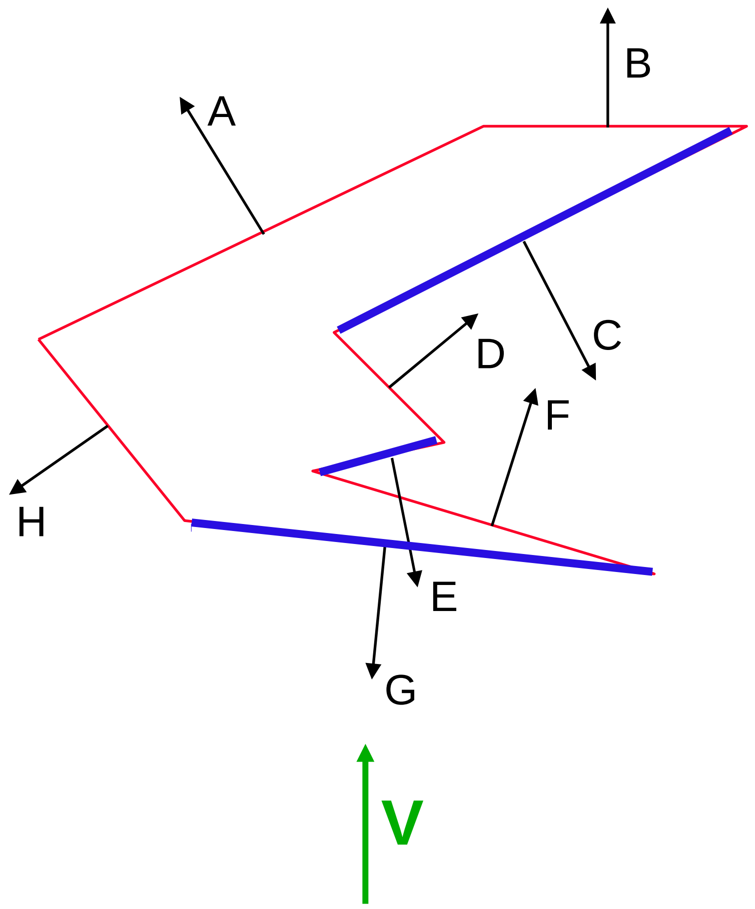
Back-Face Detection

- Also we cannot see any face for which $C = 0$.
- Similar method can be used for **left handed system.**
- In left handed system V is along the positive Z direction and **polygon is back face if $C \geq 0$**.



A) **Left Handed Viewing System(+ve Z)**
B) **Right Handed Viewing System (-ve Z)**

Back-Face Detection



Back Face Polygons:
A, B, D, F

Front Face Polygons:
C, E, G, H

- Convex objects
  - For convex objects, *back face detection* actually solves the *visible surfaces problem*.
  - Back face detection is easily applied to convex polyhedral objects.
- In a general object, a front face can be visible, invisible, or partially visible.

Back-Face Detection

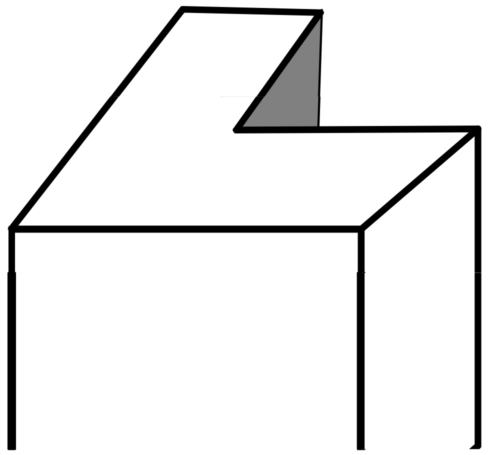
- For a single convex polyhedron such as the pyramid by examining parameter C for the different plane we identify back faces.
- So far the scene contains only non overlapping convex polyhedral, back face method works properly.
- For other object such as concave polyhedron we need to do more tests for determining back face.

Source: www.pinterest.com

Back-Face Detection

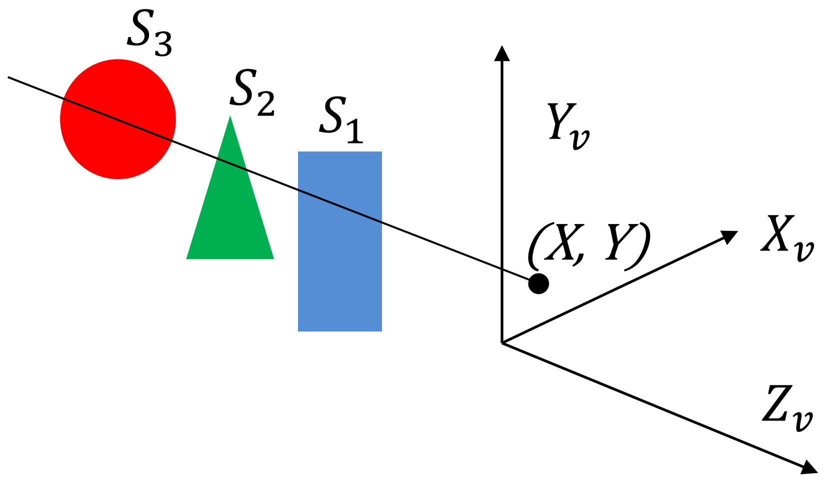
Disadvantages:

- **Partially hidden faces cannot be determined by this method**
- **Not useful for ray tracing, or photometry/ radiosity.**
- **However, this is still useful as a pre-processing step, as almost 50% of the surfaces are eliminated.**



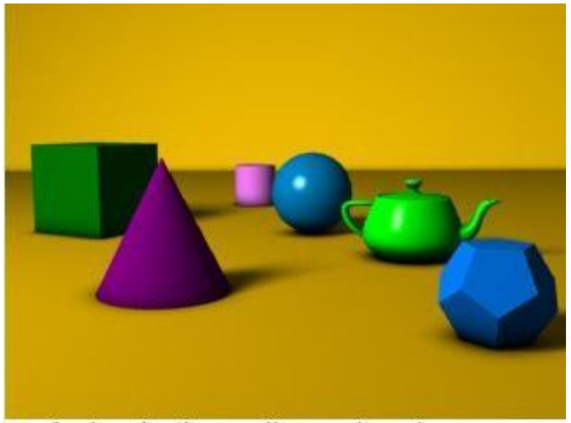
Depth Buffer Method/ Z Buffer Method

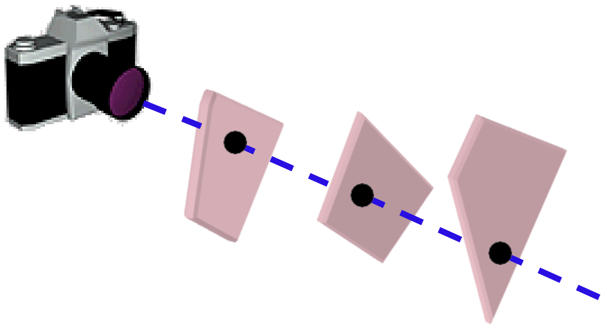
- It is image space approach.
- It compares surface depth at each pixel position on the projection plane.
- It is also referred to as *z*-buffer method since generally depth is measured in z-direction.
- Each surface of the scene is process separately one point at a time across the surface.

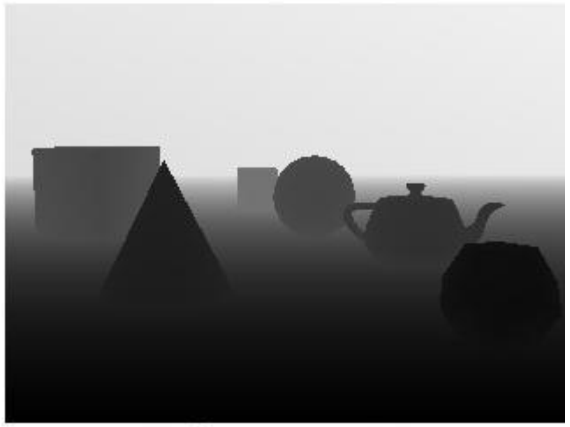


Depth Buffer Method/ Z Buffer Method

- In addition to the frame buffer (keeping the pixel values), keep a Z-buffer containing the depth value of each pixel.
- Surfaces are scan-converted in an arbitrary order.
- For each pixel (x,y), the Z-value is computed as well.
- The (x,y) pixel is overwritten only if its Z-values is closer to the viewing plane than the one already written at this location.


A simple three dimensional scene




Z-buffer representation

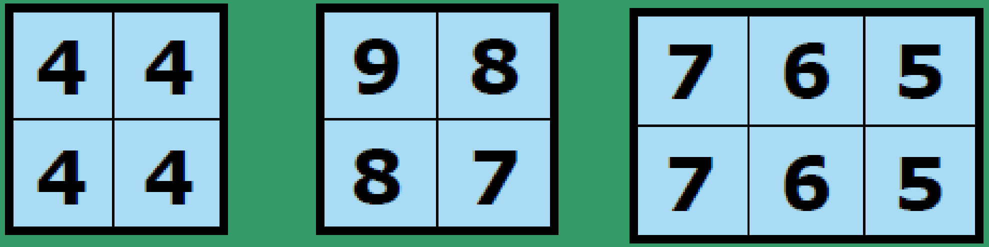
Depth Buffer Method/ Z Buffer Method

Algorithm(Right Handed viewing System, -ve Z axis)

1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x, y),
   - $depth(x, y) = MAX\_Z, \qquad refresh(x, y) = I_{backgnd}$
2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   - Calculate the depth z for each (x, y) position on the polygon.
   - If $z < depth(x, y)$, then set,

     $depth(x, y) = z, \qquad refresh(x, y) = I_{surf}(x, y)$

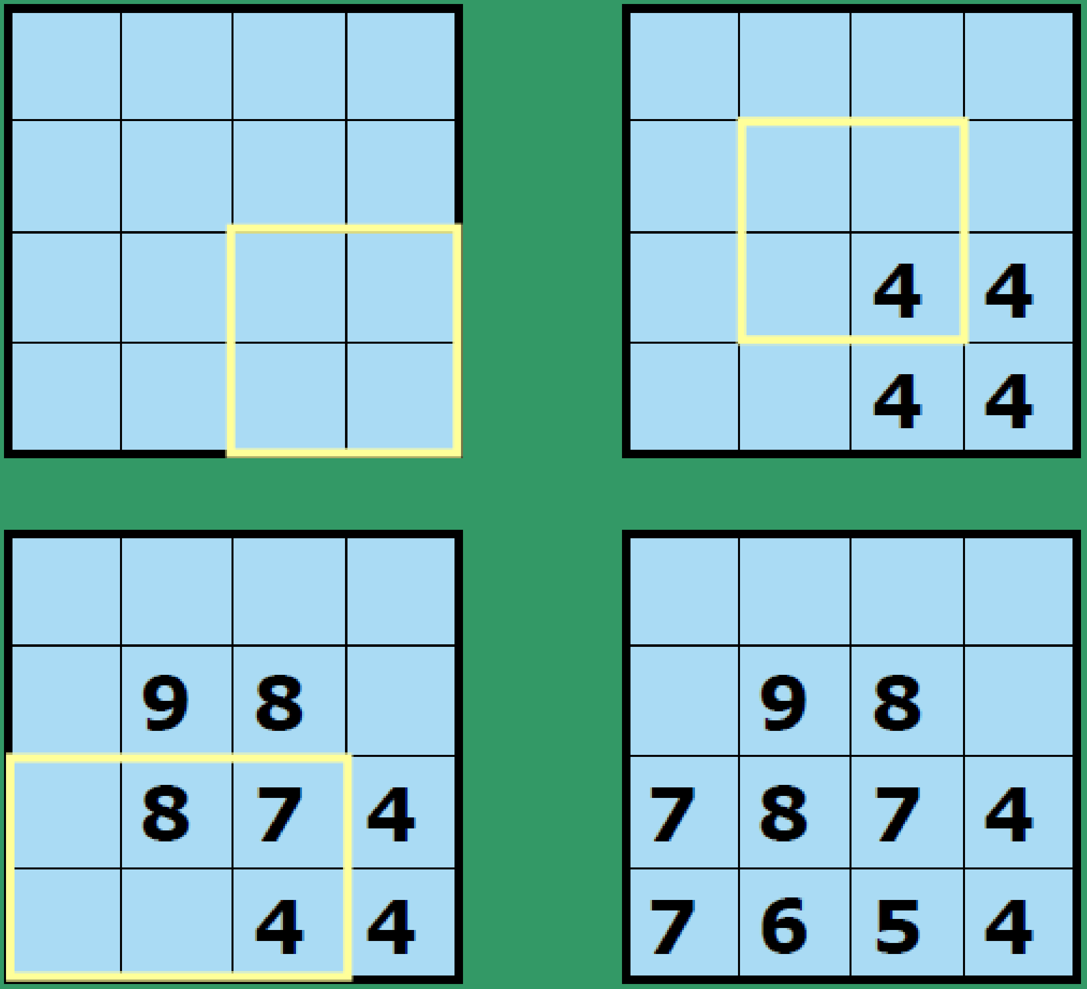
Z-values of the coordinates:



4	4
4	4

9	8
8	7

7	6	5
7	6	5



		4	4
		4	4

	9	8	
	8	7	4
		4	4

	9	8	
7	8	7	4
7	6	5	4

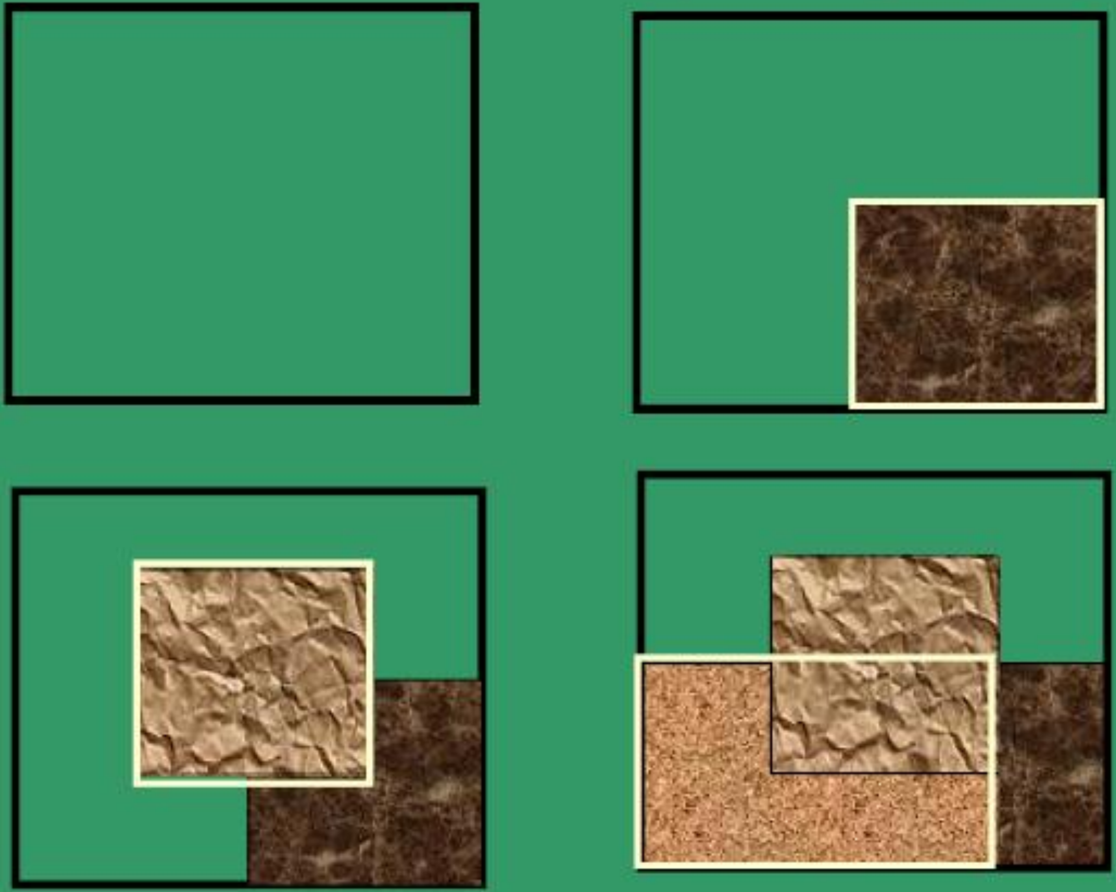
Z-Buffer values

Z-values of the coordinates:

4	4
4	4

9	8
8	7

7	6	5
7	6	5



	9	8	
7	8	7	4
7	6	5	4

Z-Buffer values

IMAGES (after pseudo-texture rendering)

Depth Buffer Method/ Z Buffer Method

Algorithm(Left Handed System ,+ve Z axis)

1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x, y),
   - $depth(x, y) = 0$, $\qquad refresh(x, y) = I_{backgnd}$
2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   - Calculate the depth z for each (x, y) position on the polygon.
   - If $z > depth(x, y)$, then set,

     $depth(x, y) = z, \qquad refresh(x, y) = I_{surf}(x, y)$

Calculation of Depth in Z buffer

- Depth values are calculated from plane equation,

$$Ax + By + Cz + D = 0$$

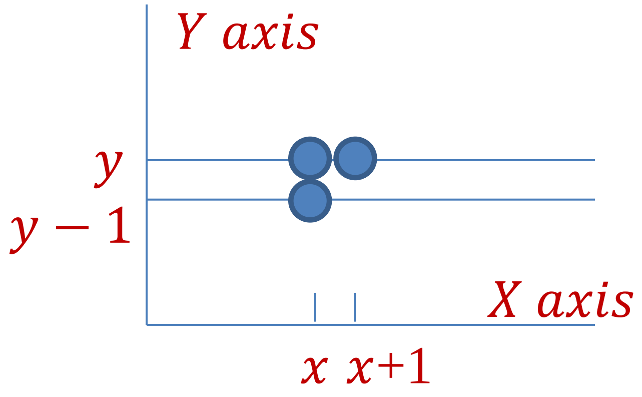
$$z = \frac{-Ax - By - D}{C}$$

- For horizontal line next pixel's z values can be calculated by putting $x' = x + 1$ in above equation,

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

$$z' = \frac{-Ax-By-D}{C} - \frac{A}{C}$$

$$z' = z - \frac{A}{C}$$



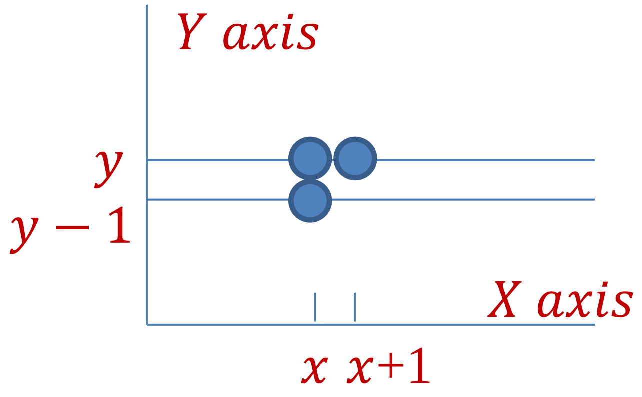
Contd.

- For vertical line pixel below the current pixel has $y' = y - 1$ so it's z values can be calculated as follows,

$$z' = \frac{-Ax - B(y-1) - D}{C}$$

$$z' = \frac{-Ax-By-D}{C} + \frac{B}{C}$$

$$z' = z + \frac{B}{C}$$



Contd.

- If we are moving along polygon boundary then it will improve performance by eliminating extra calculation.
- For this if we move top to bottom along polygon boundary we get $x' = x - 1/m$
- and $y' = y - 1$, so z value is obtain as follows,

$$\boldsymbol{z' = \frac{-A(x - 1/m) - B(y - 1) - D}{C}}$$

$$\boldsymbol{z' = z + \frac{{}^{A}/_{m} + B}{C}}$$

- Alternately we can use midpoint method to find the z values.

SCAN-LINE Algorithm (for V.S.D.)

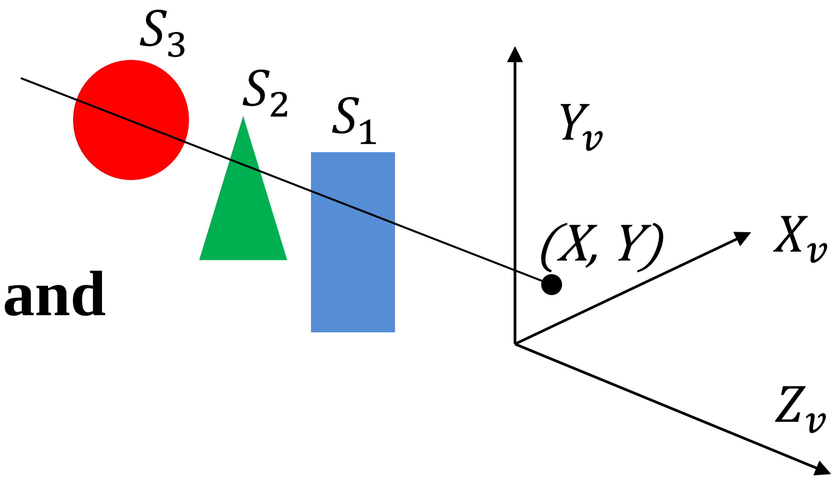
- **Extension of 2-D Scanline (polyfill) algorithm.**
- **Here we deal with a set of polygons.**

Data structure used:

ET (Edge Table),

AET (Active Edge Table) and

PT (Polygon Table).



- **Edge Table entries contain information about edges of the polygon, bucket sorted based on each edge's smaller Y coordinate.**
- **Entries within a bucket are ordered by increasing X-coordinate of their endpoint.**

Structure of each entry in ET:

- **X-Coordn. of the end point with the smaller Y-Coordn.**
- **Y-Coordn. of the edge's other end point**
- **⊗X = 1/m**
- **Polygon ID**

X	Y_{max}	⊗X	ID	●

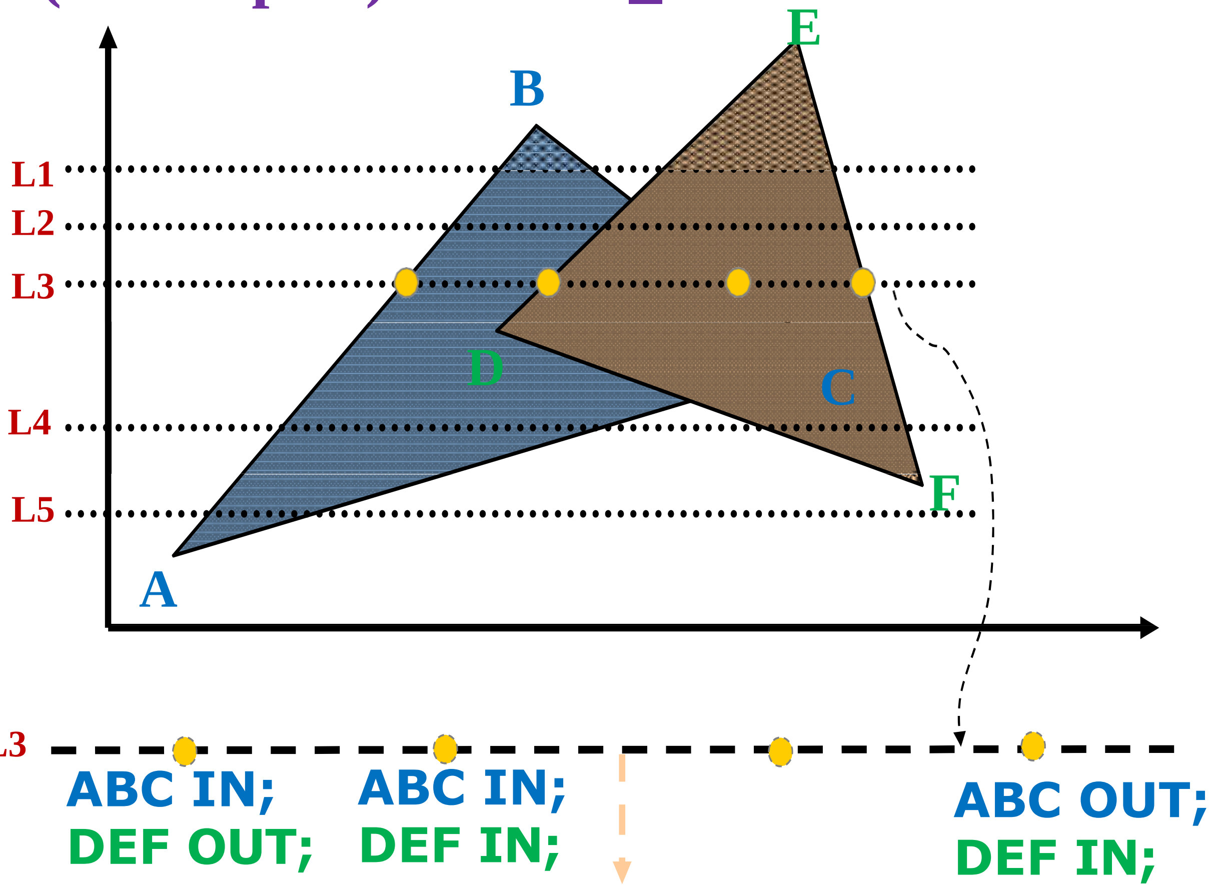
Structure of each entry in PT:

- **Polygon ID**
- **Coefficients of the plane equations**
- **Shading or color information of the polygon**
- **Flag (IN/OUT), initialized to '*false*'**

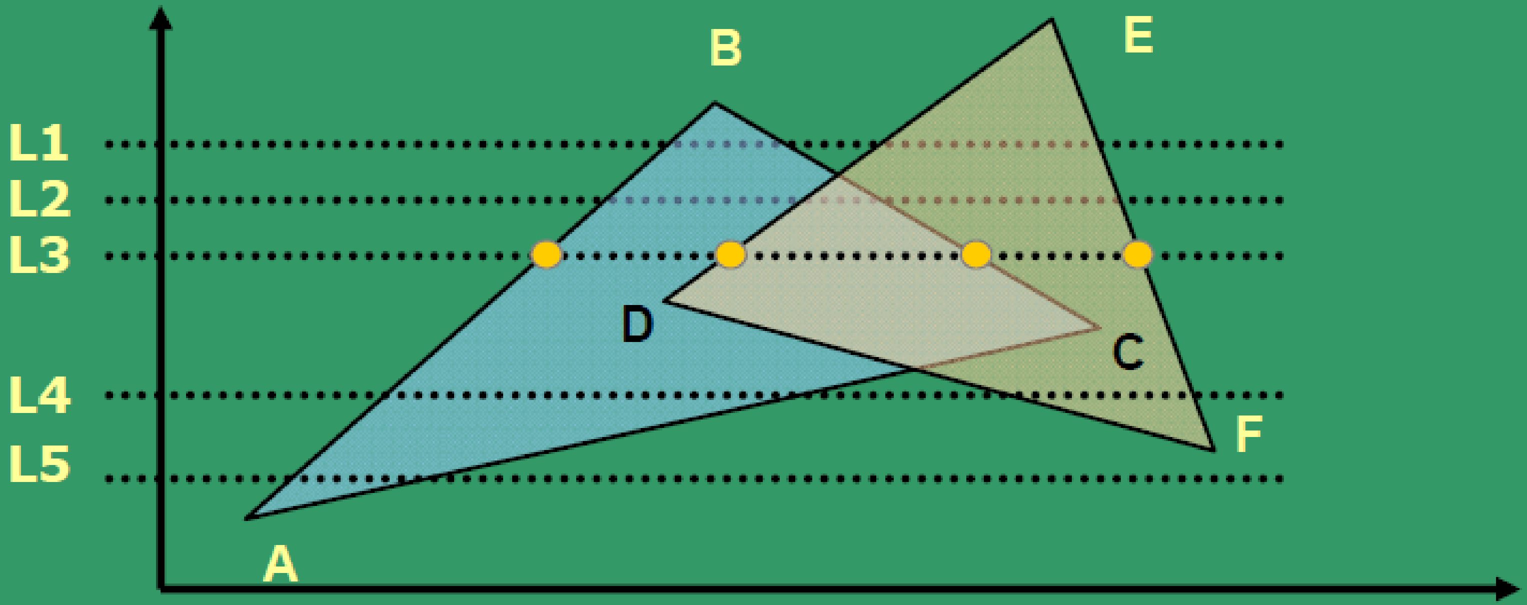
ID	Plane Coeffs.	Shading Info.	IN/OUT

Take Adjacent (not successive) pairs of intersections to fill;

FILL (within pair) if POLY_FLAG is set to IN.



Compare Z values in this region to find the visible Z value.



AET Contents				
Scan Line	Entries			
L5	AB	CA		
L4	AB	CA	FD	EF
L3, L2	AB	DE	BC	EF
L1	AB	BC	DE	EF

Scan Line Algorithm for V.S.D.

Step1: Start algorithm

Step2: Initialize the desired data structure

Create a polygon table(PT) having color, edge pointers, coefficients

Establish edge table(ET)contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.

Create Active edge list.(AET) This will be sorted in increasing order of x.

Create a flag F. It will have two values either IN or OUT(TRUE/FALSE). Initially F is OUT(FALSE)

Step3: Perform the following steps for all scan lines

Enter values in Active edge list table (AET) in sorted order using y as value

Scan until the flag, i.e. F is IN(TRUE) using a background color

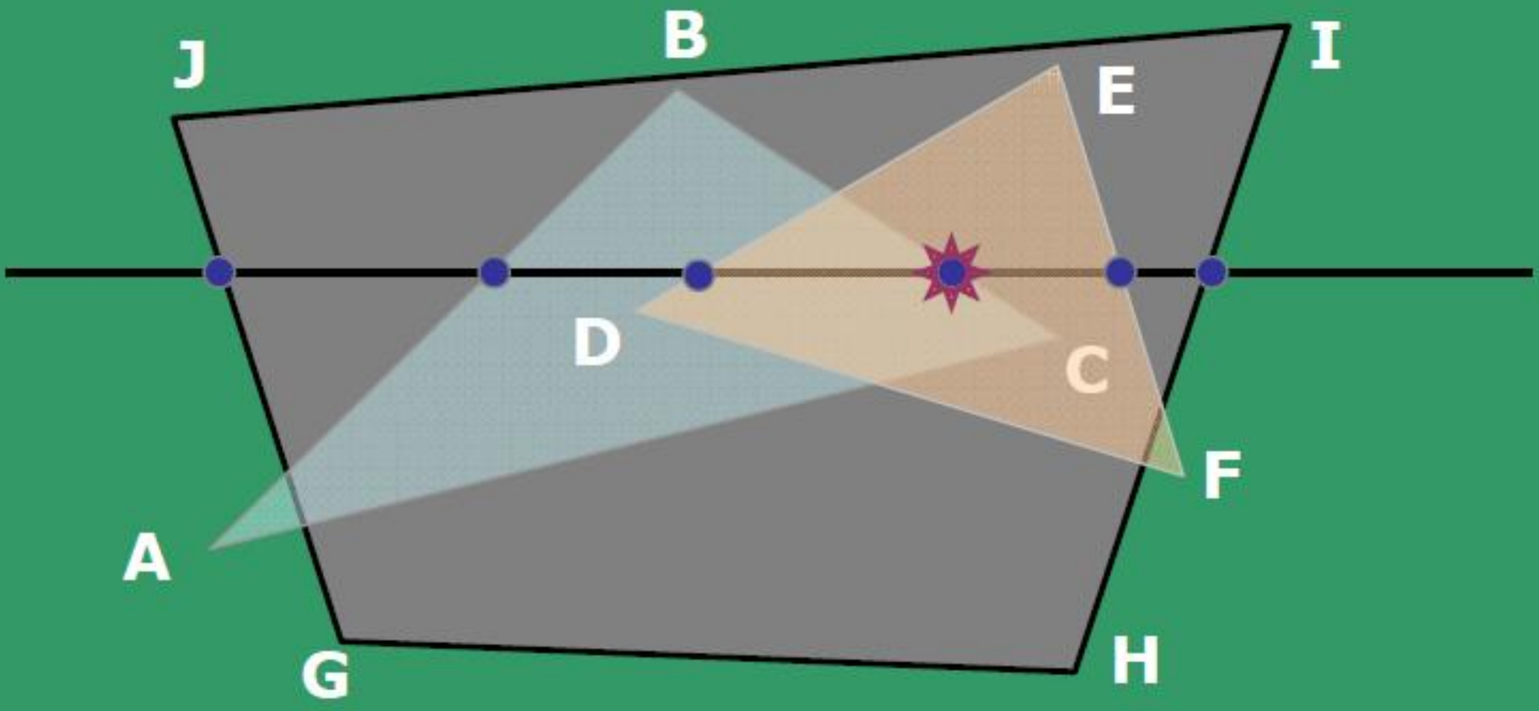
When one polygon flag F is IN(TRUE), and this is for surface $S_{1,}$enter color intensity as I_1 into refresh buffer

When two or image surface flag are IN, sort the surfaces according to depth and use intensity value S_n for the nth surface. This surface will have least z depth value

Use the concept of coherence for remaining planes.

Step4: Stop Algorithm

Three non-intersecting polygons



GHIJ is behind ABC and DEF.

SCAN LINE ALGORITHM

- **So, when scanline leaves edge BC, it is still inside polygons DEF and GHIJ. If polygons do not intersect, depth calculations and comparisons between GHIJ and DEF can be avoided.**

- **Thus depth computations are unnecessary when the scanline leaves an obscured polygon. It is required only when it leaves an obscuring polygon.**

- **Additional treatment is necessary for intersecting polygons.**

Depth-sorting or Painter's Algorithm

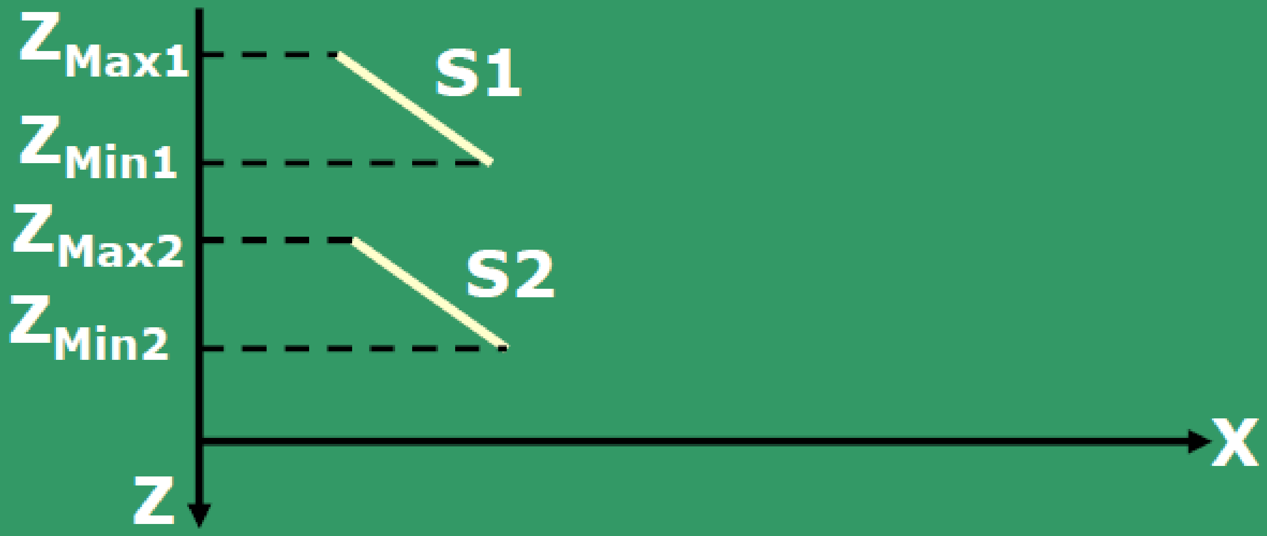
Paint the polygons in the frame buffer in order of decreasing distance from the viewpoint.

Broad Steps:

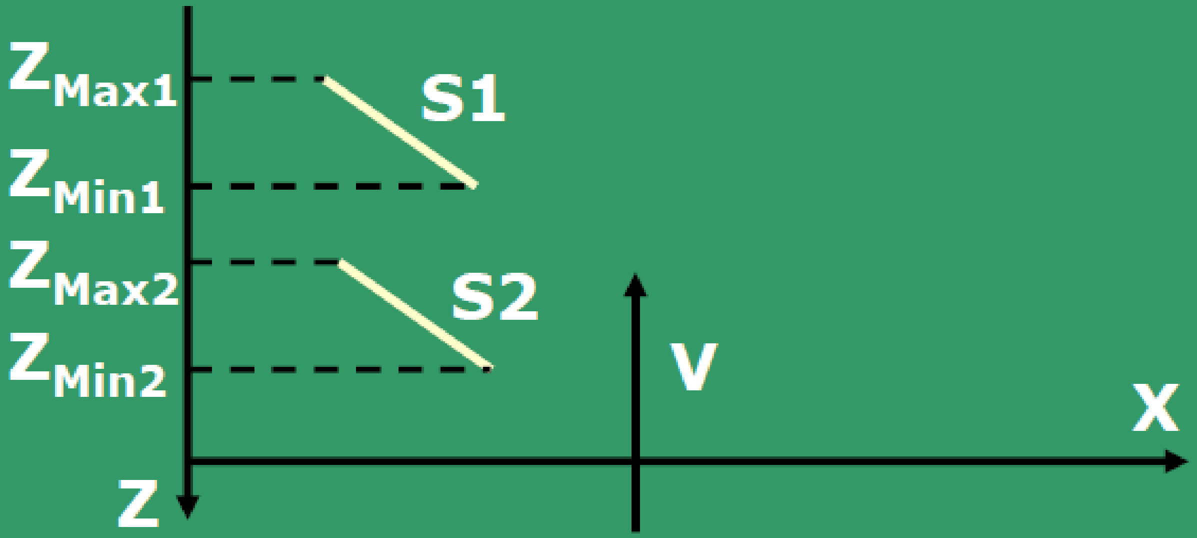
- Surfaces are sorted in increasing order of DEPTH.
- Resolve ambiguities when polygons overlap (in DEPTH), splitting polygons if necessary.
- Surfaces are scan converted in order, starting with the surface of greatest DEPTH.

Principle:

Each layer of paint (polygon surface of an object) covers up the previous layers while drawing.



Since, $Z_{Min1} > Z_{Max2}$ no overlap occurs.
S1 is first scan converted, and then S2.
This goes on, as long as no overlap occurs.



If depth overlap occurs, additional comparisons are necessary to reorder the surfaces.

The following set of <u>*tests*</u> are used to ensure that no re-ordering of surfaces is necessary:

Four(4) Tests in Painter’s algorithm

1. The boundary rectangles in the X-Y 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.

Tests must be performed in order, as specified.

Condition to RE-ORDER the surfaces:

If any one of the 4 tests is TRUE, we proceed to the next overlapping surface. i.e. If all overlapping surfaces pass at least one of the tests, none of them is behind S.

No re-ordering is required, and then S is scan converted.

RE-ORDERing is required if all the four tests fail.

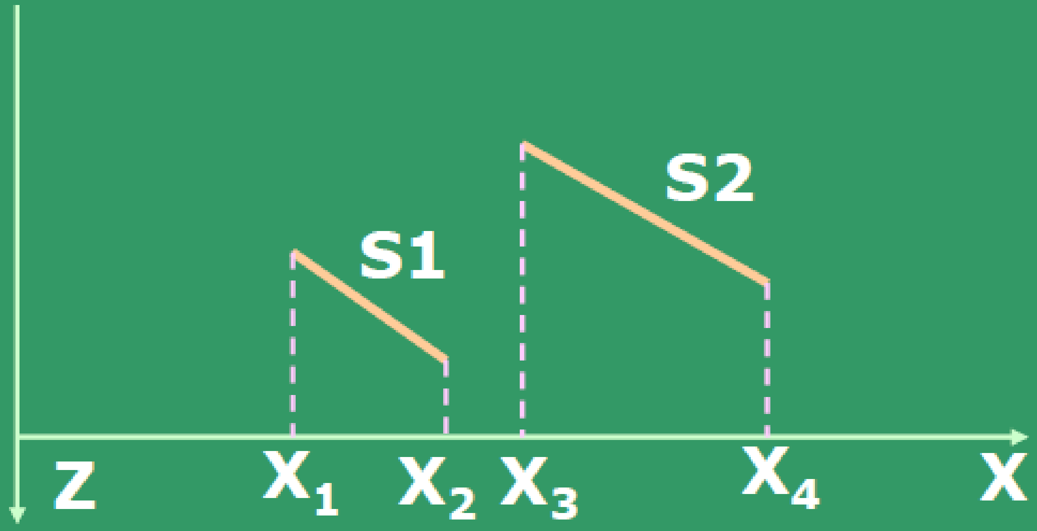
TEST #1:

The boundary rectangles in the X-Y plane for the two surfaces do not overlap.

We have depth overlap, but no overlap in X-direction.

Hence, Test #1 is passed, scan convert S2 and then S1.

If we have X overlap, check for the rest.



TEST #2:

Surface S is completely behind the overlapping surface relative to the viewing position.

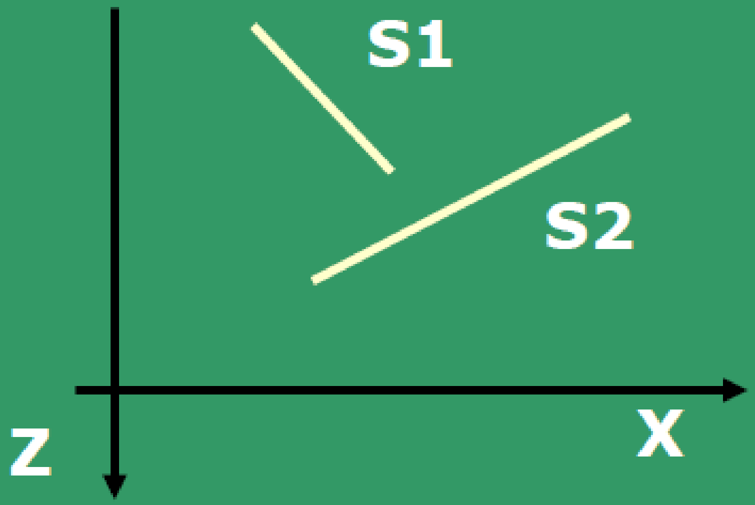


Fig. 1. S1 is completely behind/inside the overlapping surface S2

TEST #3:

The overlapping surface is completely in front of S relative to the viewing position.

S1 is not completely behind S2.
So, Test #2 fails.

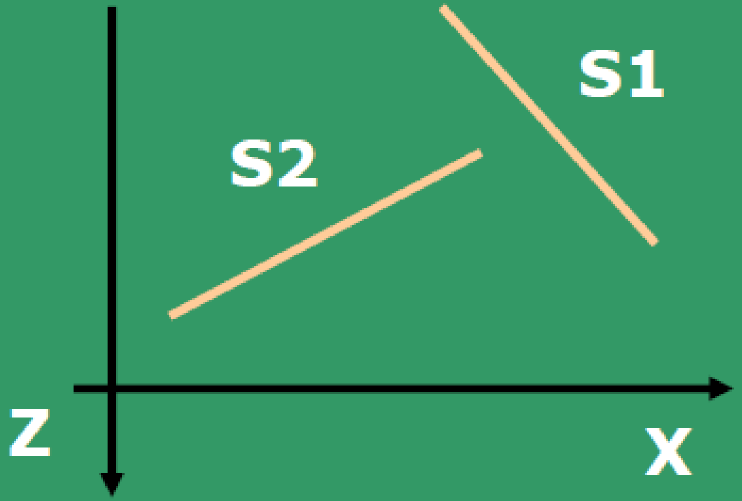


Fig. 2. Overlapping surface S2 is completely front/outside S1.

In Fig. 2, S2 is in front of S1, but S1 is not completely inside S2 – Test #2 it not TRUE or FAILS, although Test #3 is TRUE.

How to check these conditions?

Let plane equation for ,

S1 is $A_1X_1+B_1Y_1+C_1Z_1+D_1=0$ & S2 is $A_2X_2+B_2Y_2+C_2Z_2+D_2=0$

a. **Set the plane equation of S2, such that the surface S2 is towards the viewing position.**

b. **Substitute the coordinates of all vertices of S1 into the plane equation of S2 and check for the sign(as per Inside outside test).**

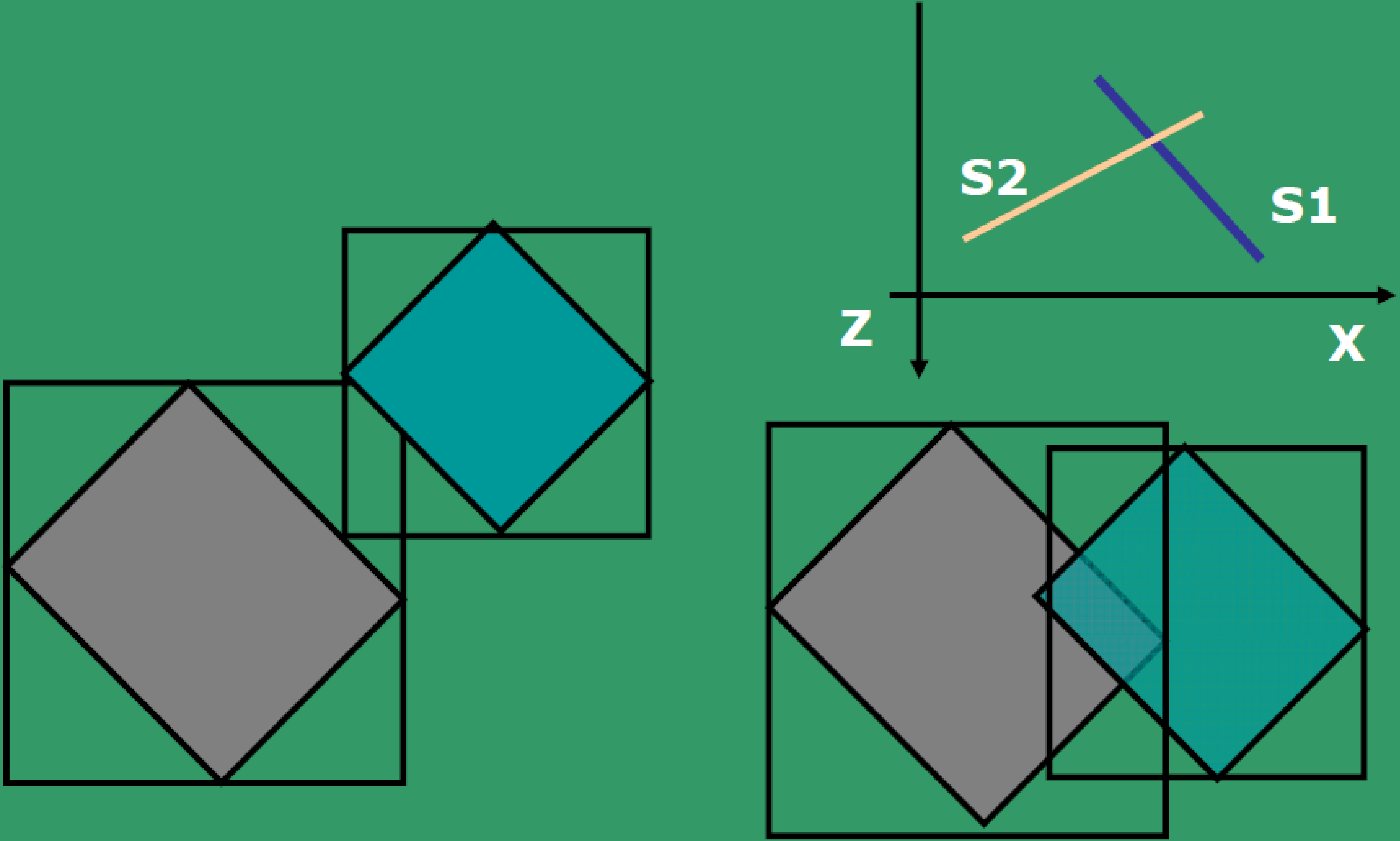
c. **If all vertices of S1 are inside S2,then S1 is behind S2.**

   i.e. for all vertices of S1, value of plane equation S1 is <=0, (Test -2 :Fig. 1)

d. **If all vertices of S1 are outside S2, S1 is in front of S2.**

   i.e. for all vertices of S1, value of plane equation S1 is >0,(Test-3: Fig. 2)

TEST #4:
The projections of the two surfaces onto the view plane do not overlap.



Four(4) Tests in Painter's algorithm - revisited

1. The boundary rectangles in the X-Y 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.

Tests must be performed in order, as specified.

Condition to RE-ORDER the surfaces:

If any one of the 4 tests is TRUE, we proceed to the next overlapping surface. i.e. If all overlapping surfaces pass at least one of the tests, none of them is behind S.

No re-ordering is required, and then S is scan converted.

RE-ORDERing is required if all the four tests fail.

Painter Algorithm/Depth sorting Algorithm

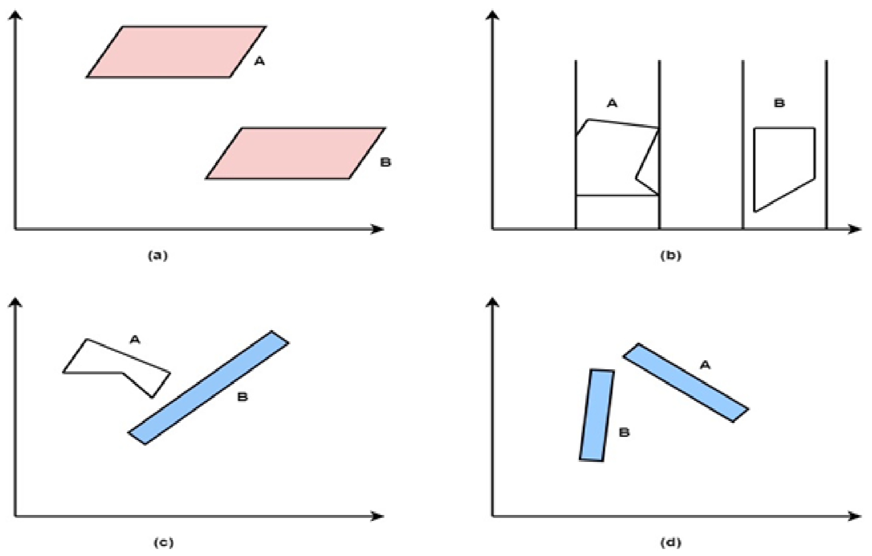
Step1: Start Algorithm

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.



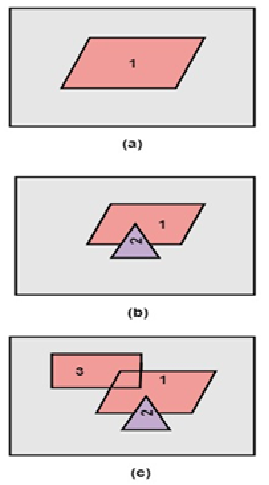
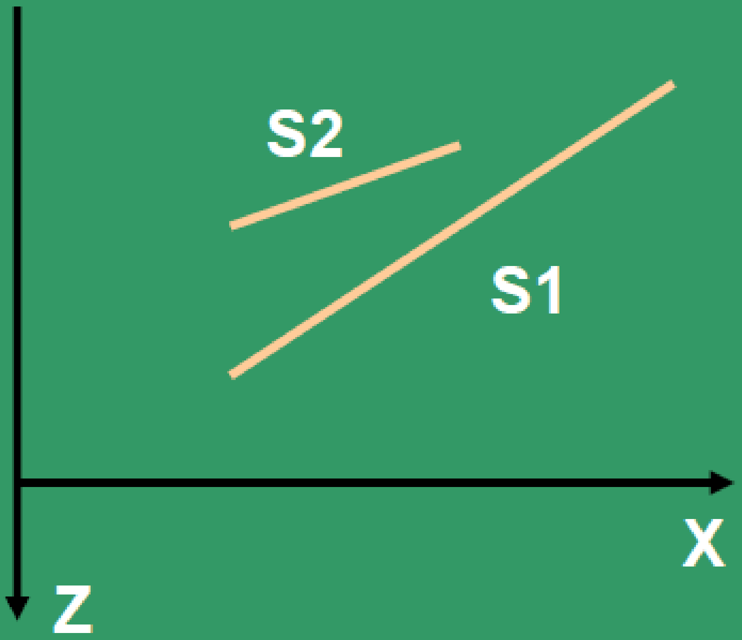


Figure showing addition of surface one by one and then painting done using painter algorithm

Case Studies – examples of

Painter’s Algorithm

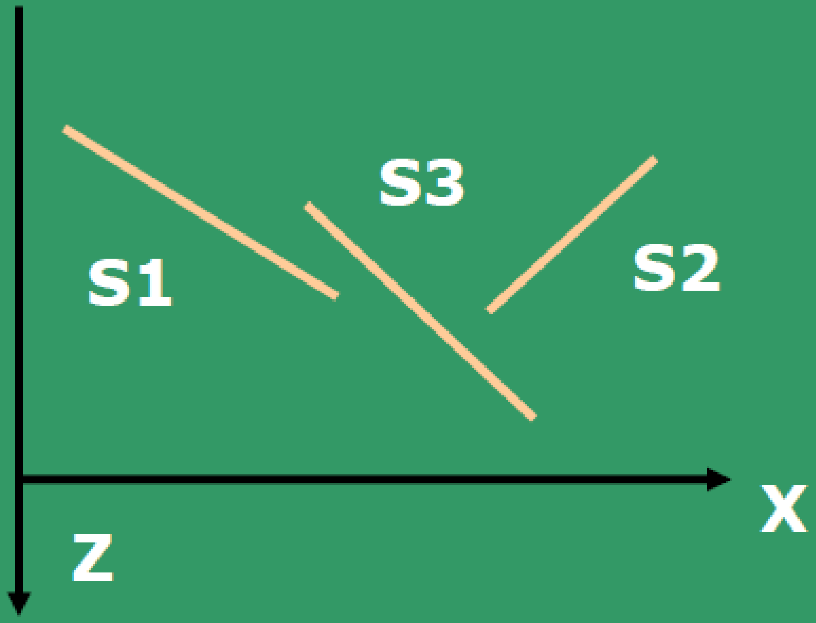
Case Study - I



Initial order:
S1 -> S2

Change the order:
S2 -> S1

Case Study - II



Initial Order:
S1 -> S2 -> S3.

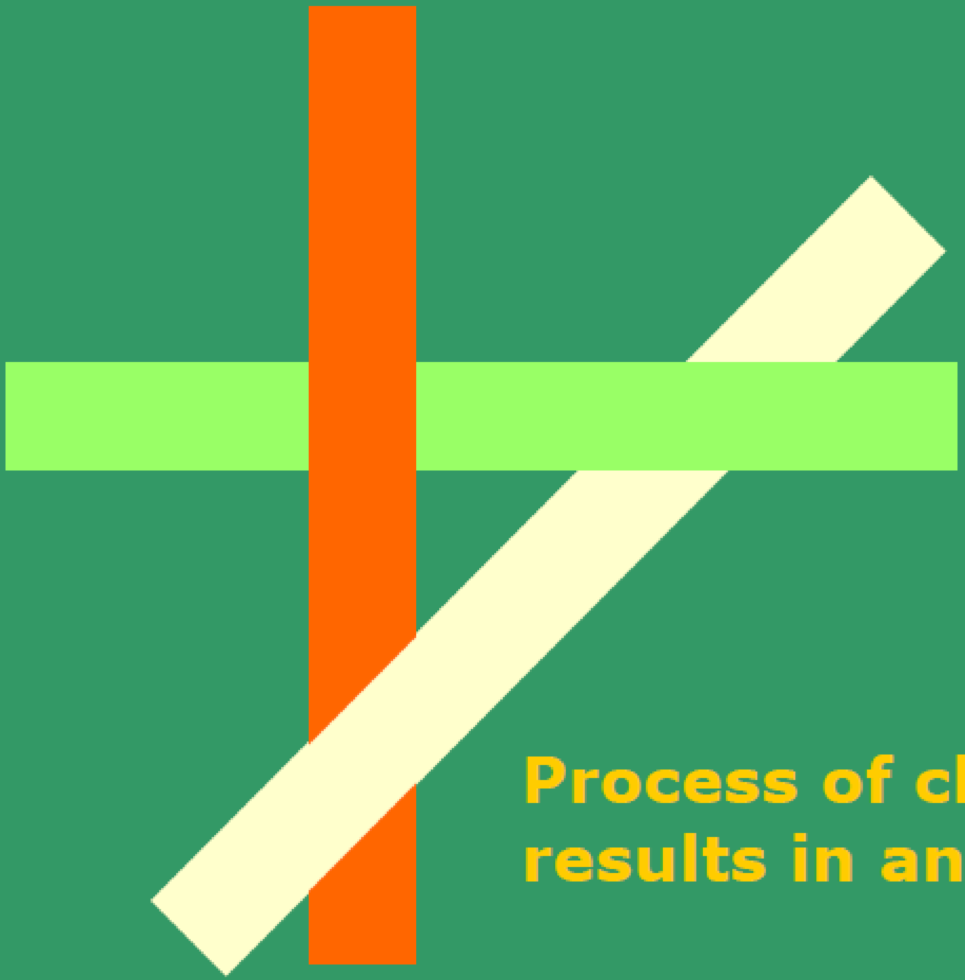
S1 -> S2,
Test 1 passed.

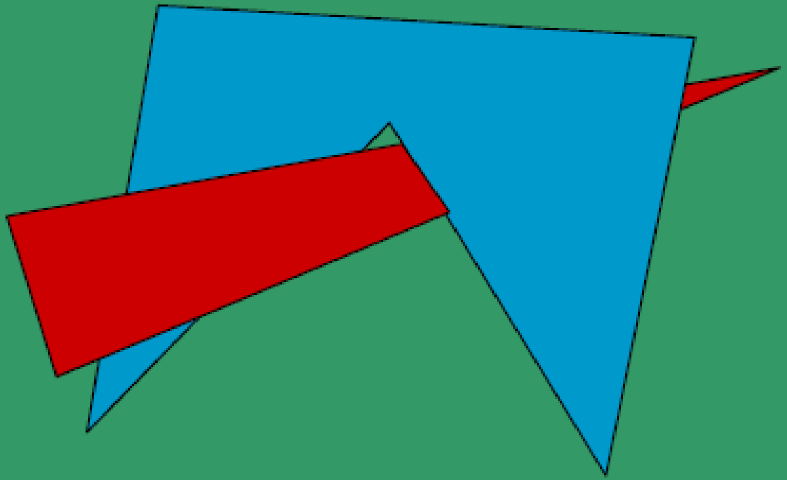
Check S1, S3. All tests fail.

Interchange. S3 -> S2 -> S1.

Check S2 and S3. All tests fail again.

Correct Order: S2 -> S3 -> S1.



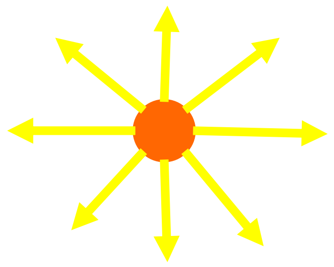


Process of checking the order, results in an infinite loop.

Avoided by setting a FLAG for a surface that has been re-ordered or shuffled. If it has to be altered again, split it into two parts.

Light Source

- When we see any object we see reflected light from that object.
- Total reflected light is the sum of contribution from all sources and reflected light from other object that falls on the object.
- So that the surface which is not directly exposed to light may also visible if nearby object is illuminated.
- The simplest model for light source is **point source.**
- Rays from the source then follows radial diverging paths from the source position.



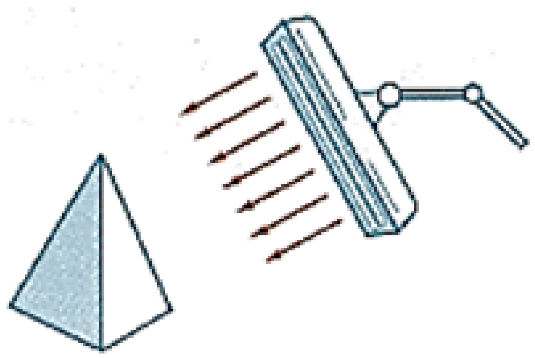
Contd.

- This light source model is reasonable approximation for source,

  - whose size is small compared to the size of object

  - may be at sufficient distance so that we can see it as point source.

- For example sun can be taken as point source on earth.
- A nearby source such as the long fluorescent light is more accurately modelled as a **distributed light source.**
- In this case the illumination effects cannot be approximated with point source because the area of the source is not small compare to the size of object.



Contd.

- When light is falls on the surface the part of the light is reflected and part of the light is absorbed.
- Amount of reflected and absorbed light is depends on the property of the object surface.
- For example shiny surface reflect more light while dull surface reflect less light.

Basic Illumination Models

- It is also known as **Shading Model** or **Lighting Model.**
- These models give simple and fast method for calculating the intensities of light for various reflections.
- Three models we discuss here for light intensity.
  1. Ambient Light
  2. Diffuse Reflection
  3. Specular Reflection

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.

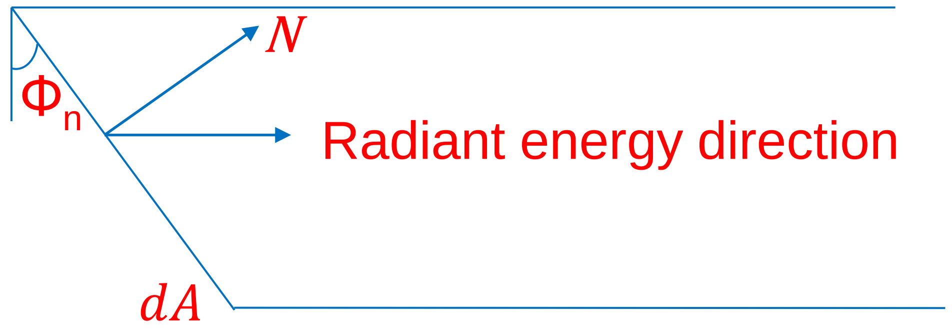
- Shiny surface reflect more light so K_d is assign larger value while dull surface assign small value.
- If surface is exposed to only ambient light we calculate ambient diffuse reflection as,

  $$I_{ambdiff} = K_d I_a$$

  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.

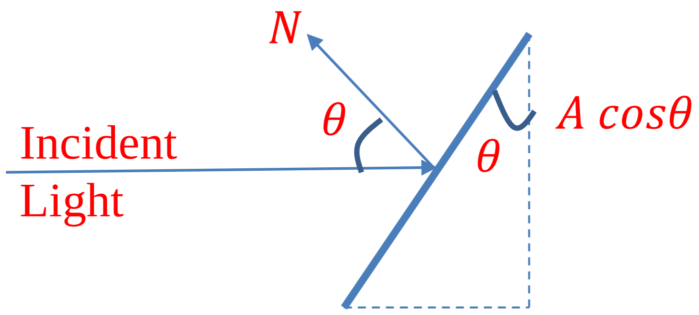
Contd.

- 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 $cos\emptyset_n$.



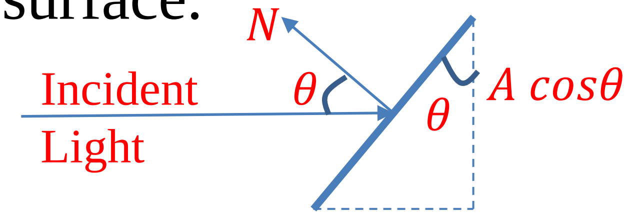
Contd.

- 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



Contd.

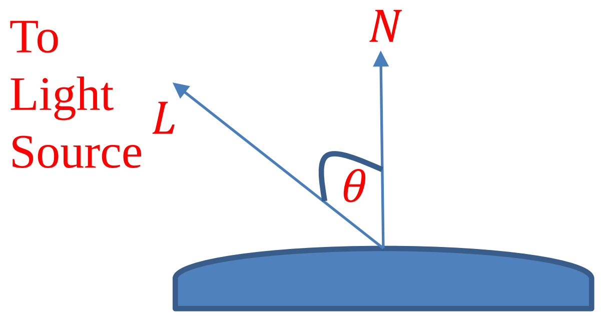
- **Angle of incidence** between the incoming light and surface normal is denoted as θ.
- Projected area of a surface patch perpendicular to the light direction is proportional to $cos\theta$.
- If I_l is the intensity of the point light source, then the diffuse reflection equation for a point on the surface can be written as
  $$I_{l,diff} = K_d I_l cos\theta$$
- Surface is illuminated by a point source only if the angle of incidence is in the range 0^0 to 90^0.
- Other value of θ light source is behind the surface.



Contd.

- As shown in figure N is the unit normal vector to surface.
- L is unit vector in direction of light source
- Dot product of this is,

  $N \cdot L = \cos\theta$



- Intensity equation is,

  $I_{l,diff} = K_d I_l (N \cdot L)$

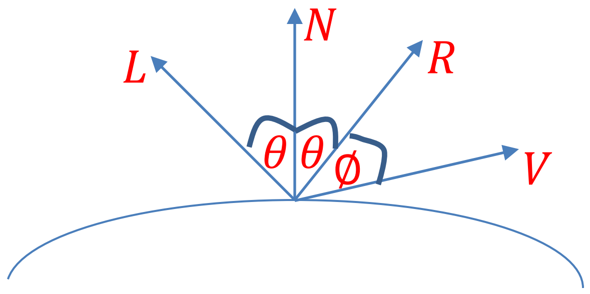
- Now in practical ambient light and light source both are present and so total diffuse reflection is given by,

  $I_{diff} = K_a I_a + K_d I_l (N \cdot L)$

- Here for ambient reflection coefficient K_a is used in many graphics package so here we use K_a instead of K_d.

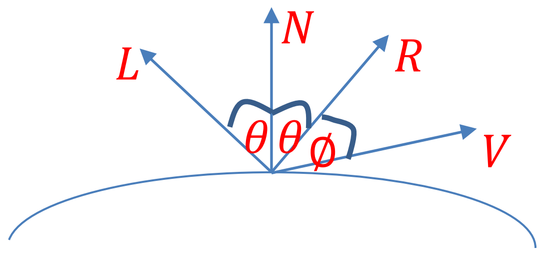
Specular Reflection and the Phong Model

- 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.



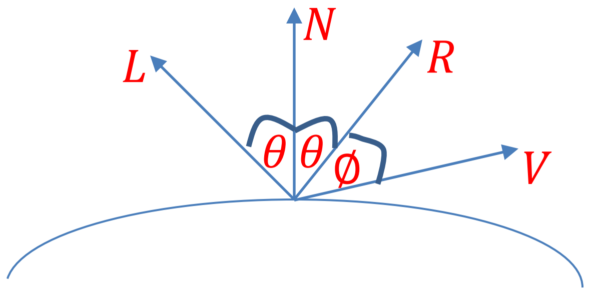
Contd.

- Intensity of specular reflection depends on the,
  - material properties of the surface
  - the angle of incidence
  - **specular reflection coefficient ($\boldsymbol{w(\theta)}$)** for each surfaces.
- Specular reflection is given by,

  $$I_{spec} = w(\theta) I_l cos^{ns}\emptyset$$

  where I_l is the intensity of light source and

  $\emptyset$ is angle between viewing direction V and specular reflection direction R.



- Since $\emptyset$ is angle between two unit vector V and R we can put,

  $$cos\emptyset = V \cdot R$$

Contd.

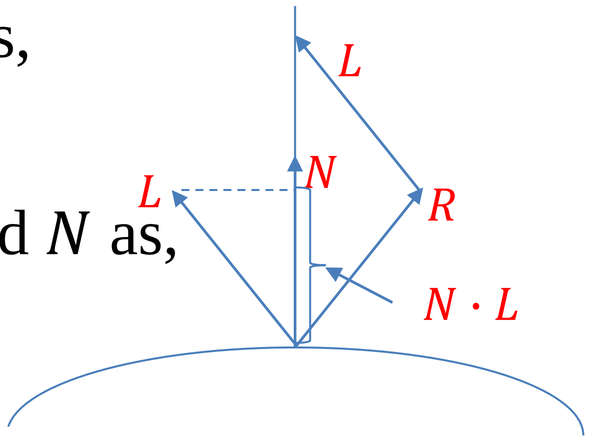
- And also for many surfaces $w(\theta)$ is constant so we take specular reflection constant as K_s so equation becomes,

  $$I_{spec} = K_s I_l (V \cdot R)^{ns}$$

- Vector R is calculated in terms of vector L and N as,

  $$R + L = (2N \cdot L)N$$

  $$R = (2N \cdot L)N - L$$



- Somewhat simplified phong model is to calculate between half way vectors H and use product of H and N instead of V and R.
- Here H is calculated as follow,

  $$H = \frac{L + V}{|L + V|}$$

Combined Diffuse and Specular Reflections with Multiple Light Sources

- For a single point light source we can combined both diffuse and specular reflection by adding intensity due to both reflection as,

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

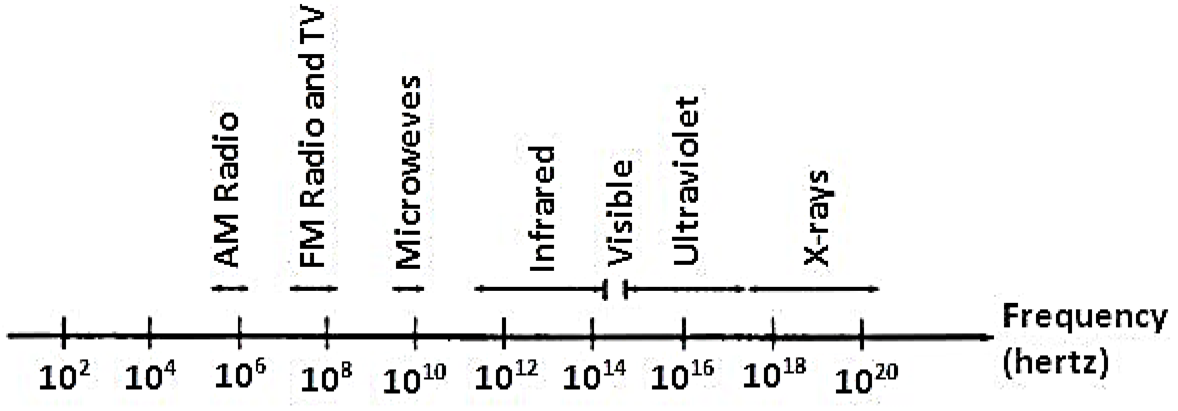
  $I = K_a I_a + K_d I_l (N \cdot L) + K_s I_l (N \cdot H)^{ns}$

- For multiple source we can extend this equation as,

$$I = K_a I_a + \sum_{i=1}^{n} I_l [K_d (N \cdot L) + K_s (N \cdot H)^{ns}]$$

Properties of Light

- Light is an electromagnetic wave.
- Visible light is have narrow band in electromagnetic spectrum only $400nm\ to\ 700nm$ light is visible by human eye.
- Electromagnetic spectrum shown in figure shows other waves are present in spectrum like microwave infrared etc.
- Frequency value from $4.3\ \times\ 10^{14}$ hertz (red) to $7.5 \times\ 10^{14}$ hertz (violet) is visible range.



Contd.

- We can specify different color by frequency f or by wavelength λ of the wave.
- We can find relation between f and λ as follows,
  $c = \lambda f$
- Frequency is constant for all the material but speed of the light and wavelength are material dependent.
- For white light, source emits light of all visible frequency.
- Reflected light have some frequency and some are absorbed by the surface.

Contd.

- Frequency reflected back is decide the color we see and this frequency is called as **dominant frequency (hue).**
- Corresponding reflected wavelength is called **dominant wavelength.**
- Other property are **brightness** and **purity.**
- Brightness is perceived intensity of light.
- Intensity is the radiant energy emitted per unit time, per unit solid angle and per unit projected area of the source.
- **Purity** or **saturation** of the light describes how washed out or how “pure” the color of the light appears.
- Dominant frequency and purity both collectively refers as **chromaticity.**

Contd.

- If two color source combined to produce white light they are called **complementary color** of each other.
- For example red and cyan are complementary color.
- Typical color models that are uses to describe combination of light in terms of dominant frequency.
- Color models use three colors to obtain reasonable wide range of colors, called the **color gamut** for that model.
- Two or three colors are used to obtain other colors in the range are called **primary colors.**

Color Models

- In practice we use variety of colors.
- For represent color, in display devices, or in hard copy different color models are used.
- Few color models we discuss are,
  - XYZ color model
  - RGB color model
  - YIQ color model
  - CMY color model

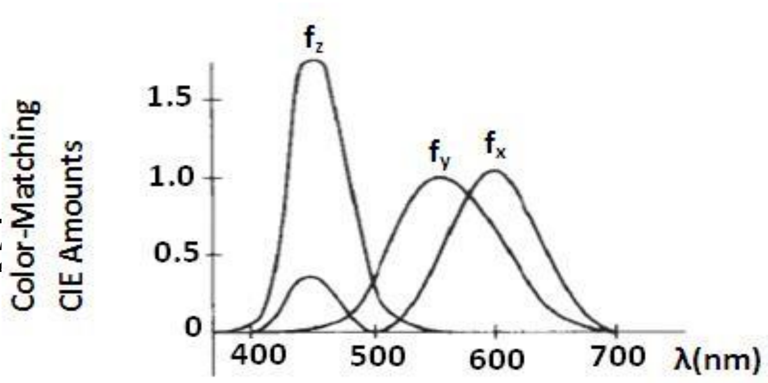
XYZ Color Model

- The set of CIE primaries is generally referred to as XYZ or (X, Y, Z) color model.
- X, Y, Z represents vectors in a three dimensional, additive color space.
- Any color C_λ is a combination of three primary colors as,

  $C_\lambda = XX + YY + ZZ$
- Where X, Y, Z is the amount of standard primary need to combine for obtaining color C_λ.
- If we normalize it then,

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+}$$

- With $x + y + z = 1$.



Contd.

- Now we can represent any color with x, y only as z we can find $z = 1 - x - y$.
- x and y are called chromaticity values because they depends only on hue and purity.
- Now if we specify colors with only x and y values we cannot find amount X, Y and Z.
- So we specify color with x, y and Y and rest CIE amount is calculated as,

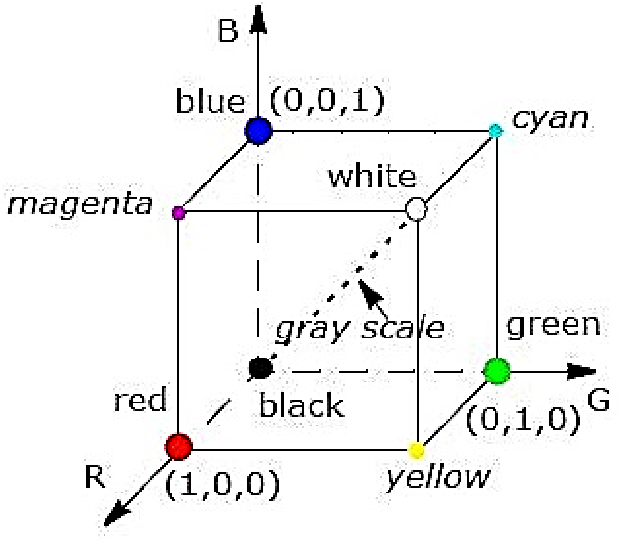
$$X = \frac{x}{y}Y \qquad Z = \frac{z}{y}Y$$

where $z = 1 - x - y$

RGB Color Model

- Based on tristimulus theory of vision our eye perceives color through stimulate one of three visual pigments in the cones of the retina.
- These visual pigments have peak sensitivity at red, green and blue color.
- So combining these three colors we can obtain wide range of color this concept is used in *RGB* color model.
- This model is represented as unit cube.
- Origin represent black color and vertex

  (1,1,1) is white.

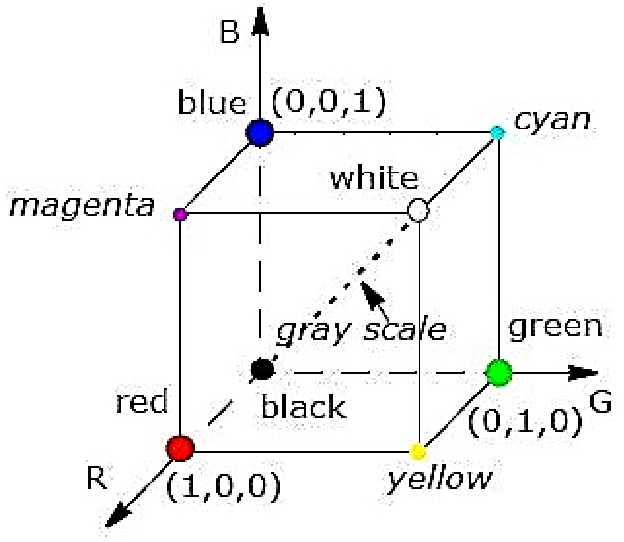


Contd.

- Vertex of the cube on the axis represents primary color R, G, B.
- In RGB color model any color intensity is obtained by addition of primary color,

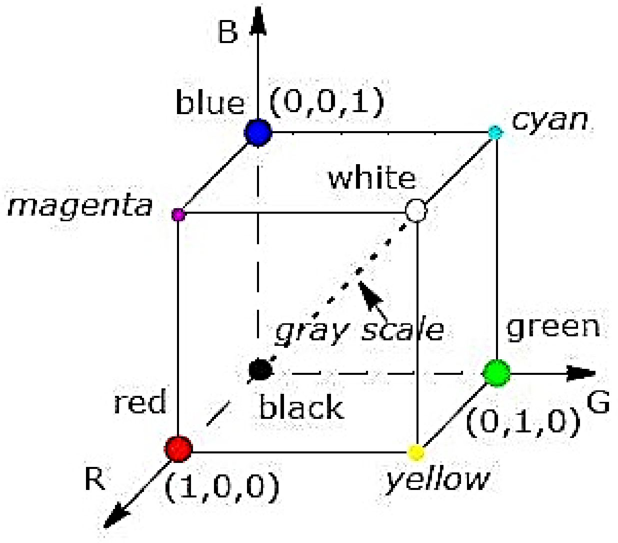
$$C_\lambda = RR + GG + BB$$

where R, G, B is amount of corresponding primary color.



Contd.

- Since it is bounded in between unit cube it's values is very in between 0 to 1 and represented as triplets (R, G, B).
- For example magenta color is represented with (1,0,1).
- Shades of gray are represented along the main diagonal of cube from black to white vertex.
- For half way gray scale we use triplets (0.5,0.5,0.5).



YIQ Color Model

- As we know RGB monitors requires separates signals for red, green, and blue component of an image.
- But television monitors uses single composite signals.
- For this composite signal NTSC use YIQ color model.
- Here parameter Y is represented as luminance (brightness).
- Chromaticity information (hue and purity) is specified into I and Q parameter.
- Combination of all red, green, and blue intensities are chosen for Y so black and white television monitors only use signal Y for brightness.
- So largest bandwidth (about 4 MHz) is assigned to Y information signal.

Contd.

- Parameter I contain orange-cyan hue information that provides the flash-tone shading, and occupies a bandwidth approximately 1.5 MHz.
- Parameter Q carries green-magenta hue information in a bandwidth of about 0.6 MHz.
- An RGB signal can be converted to a television signal using encoder which converts RGB to YIQ values.
- This conversion by transformation is given by,

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Contd.

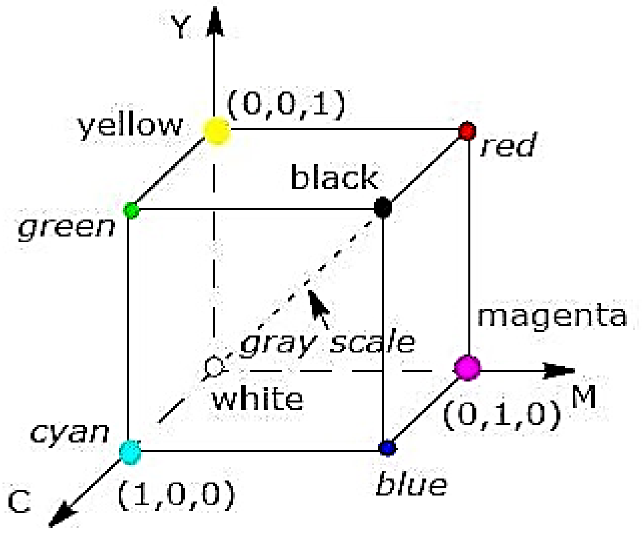
- Similarly reverse of this is performed by decoder and by transformation using inverse of above matrix as,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.108 & 1.705 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

CMY Color Model

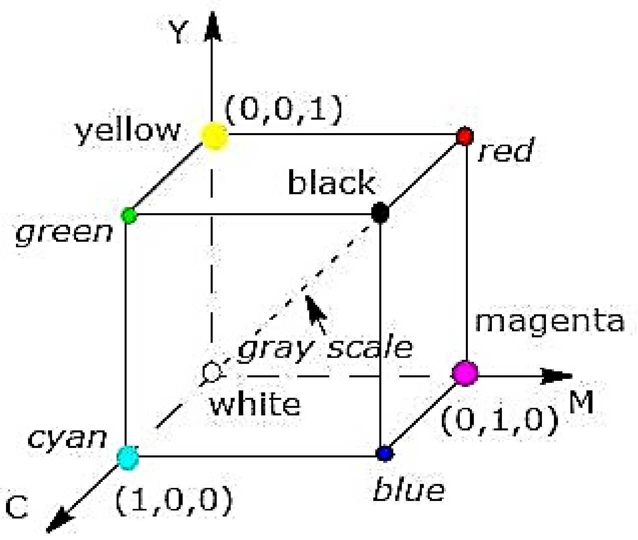
- A color model CMY is used for hardcopy devices.
- We produce picture by coating a paper with color pigments, we see the color by reflected light a subtractive process.
- When white light is reflected from cyan colored ink the reflected light must have no red component that is red light is absorbed or subtracted by the ink.
- Similarly magenta is subtracting

  green component.



Contd.

- Point (1,1,1) represents black because all components are subtracts and origin represents white light.
- Gray can be produce among main diagonal by using all three color in equal amount.
- Printing process often use CMY model generates a color points with a collection of four ink dots, one for each primary color C, M, and Y and one dot is black.



Contd.

- Conversion of RGB to CMY is done by,

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- And similarly reverse is done by,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

Thank You