Computer Graphics

Graphics refers to picture or image.

"Graphical representation and manipulation of data through computer is called Computer Graphics"

Application of Computer graphics

1. CAD (**C**omputer **A**ided **D**esign)

2. Presentation Graphics

3. Entertainment

Animation and Morphing are used in this field.

Animation:- Animation means "moving picture"

Def^{n} : Display a sequence of frames(pictures) to create moving picture.

Morphing:- It is a special effect that change one picture to another.

4. Visualization

5. Image processing

6. GUI (Graphical User Interface)

GUI provides a user friendly system.

GUI contains window manager, menu, icon etc.

Window manager controls the placement and appearance of windows.

Menu contains number of submenus.

Icon is a graphical symbol that represents a processing option.

Advantages:

- Take less space
- Understand more quickly than textual data.

Video Display Units

The primary output device in a graphics system is a video monitor.

It is based on the technique of **CRT (Cathode Ray Tube)**

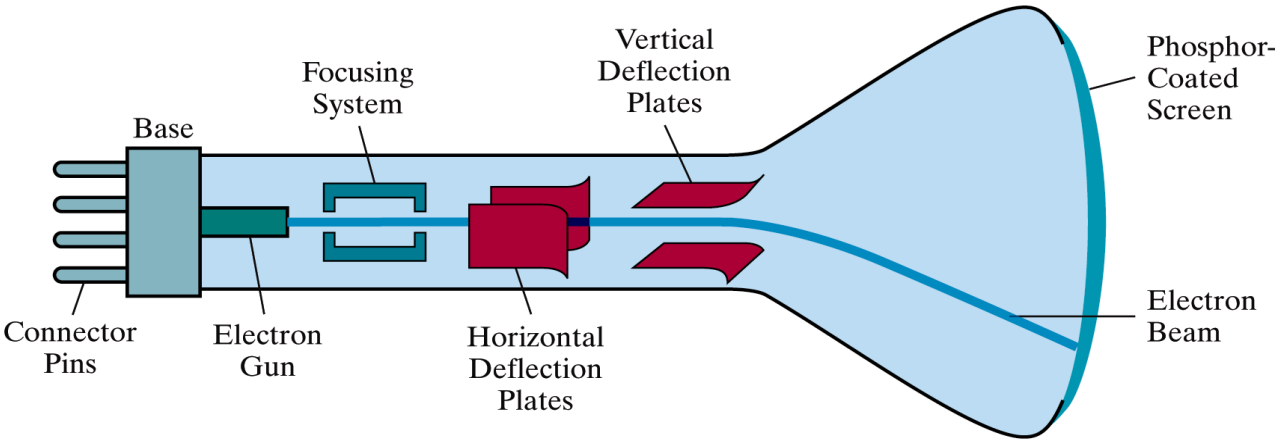


Fig: Design of CRT

Electron gun

Components of Electron gun are **Cathode** and **Control grid**.

Cathode

- Heat is supplied to cathode by passing current through it.
- Electrons (–ve charge) are emitted from cathode.

Control grid

- Control grid controls Intensity (brightness).
- It is done by passing appropriate voltage.
- A smaller –ve voltage decreases the no of electrons.
- A high –ve voltage highly decreases the no of electrons.
- A very high –ve voltage will stop the flow of electrons due to repulsion.

Focusing system

- It is used to focus the electron beam to a point on the screen.
- A point on the screen is called Pixel. Pixel means “Picture element”.

Deflection system

- Deflection of electron beam is controlled by the deflection plates.
- Horizontal deflection plates deflect the beam horizontally over the screen.
- Vertical deflection plates deflect the beam vertically over the screen.
- Proper deflection is achieved by adjusting the current through the filament.

Electrons strike the phosphorous screen with kinetic energy. This **kinetic energy** is converted to **heat energy** and the rest energy is converted to **light energy**. So we can see a light or picture in that position of screen.

Representation of graphics

There are two types of representation for graphics (or pictures)

a) Raster scan system

b) Random scan system

Raster scan system Vs Random scan system

✓ Picture is represented by set of pixels Example: .bmp files of MS Paint ✓ Realistic picture can be created ✓ Color capability is more ✓ Not suitable for animation ✓ Memory requirement is more ✓ Transformation results loss of information ✓ Zig-jag line are produced	✓ Picture is represented by set of lines Example: .jpeg files ✓ Realistic picture cannot be created. ✓ Color capability is less ✓ Suitable for animation ✓ Memory requirement is less ✓ Transformation results no loss of information ✓ No Zig-jag line

Interlacing

- It is a procedure in which there are two phases.
- In 1st phase all the odd lines are scanned, such as scan line 1,3,5,7……. etc.
- In 2nd phase the remaining even lines are scanned, such as scan line 2,4,6,8…….. etc.
- It increases the refresh rate.

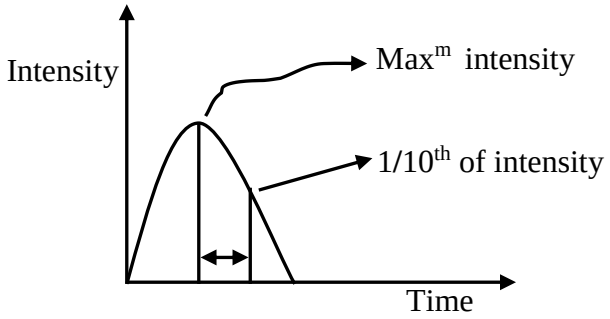
Flickering

- The refresh rate and persistence of phosphor must match with each other, Otherwise the screen will flash rapidly ON and OFF. This is called flickering.

Characteristics of CRT

1. Persistence:

- It means how long the phosphorous will give light.
- Def^{n} : Time taken by light to decay by $\frac{1}{10}th$ of original intensity.



Phosphor is of two types

- High persistence phosphor gives light for long time.
- Low persistence phosphor gives light for short time.

2. Resolution

- Resolution = Total number of pixels present on the screen.
- Resolution is expressed as X x Y pixels.

  Example: 800 x 600 pixels.
- Increase in resolution increases the quality of picture.
- High resolution system is called **H**igh **D**efinition (**HD**) system.

3. Aspect Ratio

- Aspect Ratio = Width to Height ratio.
- Example: Consider a monitor where width =12 inch, height = 9inch

  $\Rightarrow$ Aspect ratio = $\frac{12}{9} = 4:3$

  This means, in order to draw two lines of same length; you need 4 pixels in x direction. where as 3 pixels in y direction.

DDA Line Algorithm

- DDA stands for **D**igital **D**ifferential **A**nalyzer.
- It is an incremental technique.

Algorithm:

Step1:- Enter two end points of line.

Step2:- Calculate Δx , Δy

Step3:- If abs(Δx) > abs(Δy) then

STEPS = abs (Δx) // abs( ) means +ve value

Else

STEPS = abs (Δy)

Step4:- $$X_{increment} = \frac{\Delta x}{STEPS}$$

$$Y_{increment} = \frac{\Delta y}{STEPS}$$

Step5:- Calculate the new point :

$$X_{k+1} = X_k + X_{increment}$$

$$Y_{k+1} = Y_k + Y_{increment}$$

Step6:- Repeat Step5 '*STEPS*' number of times

Q: Find points over the line having end points (20, 10) and (30,18) using DDA Algorithm.

Ans:

$\Delta x = 30 - 20 = 10$

$\Delta y = 18 - 10 = 8$

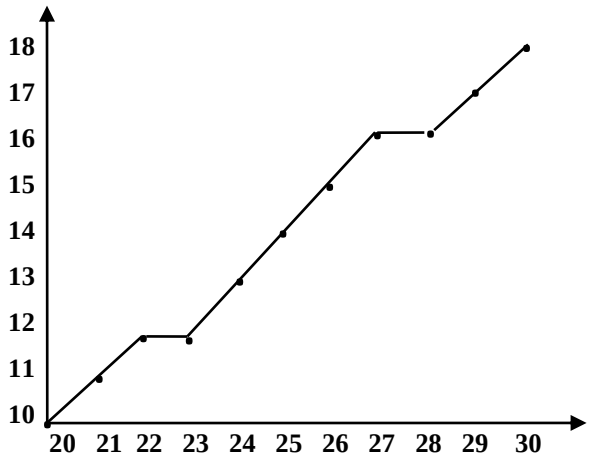
$10 > 8$ then $\Rightarrow STEPS = 10$

$$X_{increment} = \frac{\Delta x}{STEPS} = \frac{10}{10} = 1$$

$$Y_{increment} = \frac{\Delta y}{STEPS} = \frac{8}{10} = 0.8$$

(20 , 10)

STEPS	**(X , Y)**	**Rounding-up**
1	(21, 10.8)	(21, 11)
2	(22, 11.6)	(22, 12)
3	(23, 12.4)	(23, 12)
4	(24, 13.2)	(24, 13)
5	(25, 14)	(25, 14)
6	(26, 14.8)	(26, 15)
7	(27, 15.6)	(27, 16)
8	(28, 16.4)	(28, 16)
9	(29, 17.2)	(29, 17)
10	(30, 18)	(30, 18)

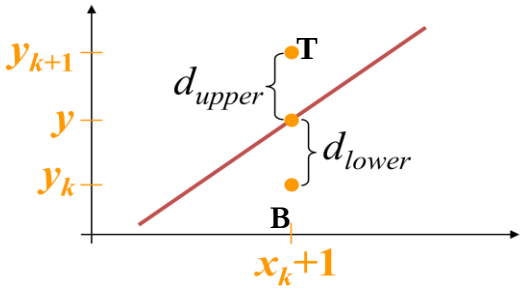


Bresenham's Line Algorithm

- ➔ It is an efficient line drawing algorithm.
- ➔ It calculates point (or pixel) according to the decision parameter P_k

Sign of P_k decides the pixel to be selected

P_k is –ve then Select B
P_k is +ve then Select T



B , **T** are the **B**ottom and **T**op pixels.

Algorithm : (for m ≤ 1)

Step1:- Input the two end points of line.

Step2:- Plot the starting point(X_0,Y_0) by scan conversion.

Step3:- Calculate Δx, Δy, 2Δy, 2Δy−2Δx

Calculate $P_0 = 2\Delta Y - \Delta X$

Step4:- For each X_k :

if $P_k < 0$ next pixel is (X_k+1, Y_k) and $P_{K+1} = P_K + 2\Delta y$

Else next pixel is (X_k+1, Y_k+1) and $P_{K+1} = P_K + 2\Delta y - 2\Delta x$

Step5:- Repeat Step4 Δx times

Q: Find points over the line having end points (20,10) and (30,18) using Bresenham's Algorithm

Ans: Δy = 18–10 = 8 and Δx = 30−20 = 10

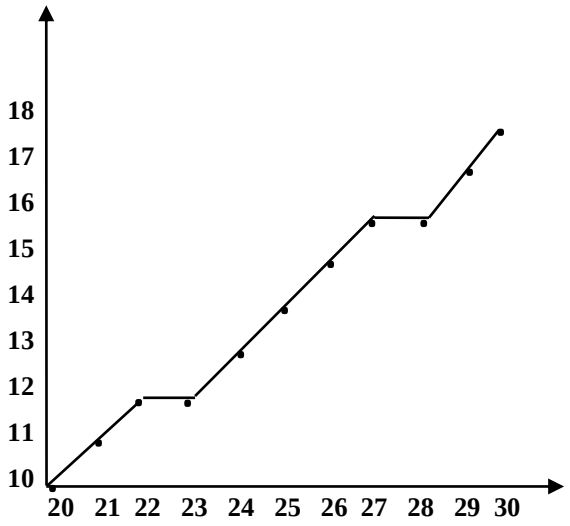
$2\Delta y = 16$ and $2\Delta y - 2\Delta x = -4$

if $P_k < 0$ next pt $(x_k + 1, y_k)$ and $P_{k+1} = P_k + 2\Delta y = P_k + 16$

Else next pt $(x_k + 1, y_k + 1)$ and $P_{k+1} = P_k + (2\Delta y - 2\Delta x) = P_k + (-4)$

(20 ,10)

k	P_k	(x , y)
0	$2\Delta Y - \Delta X = 6$	(21, 11)
1	6 + (– 4) = 2	(22, 12)
2	2 + (– 4) = –2	(23, 12)
3	–2 + 16 = 14	(24, 13)
4	14 + (–4) = 10	(25, 14)
5	10 + (–4) = 6	(26, 15)
6	6 + (–4) = 2	(27, 16)
7	2 + (– 4) = –2	(28, 16)
8	–2 + 16 = 14	(29, 17)
9	14 + (–4) = 10	(30, 18)



Mid-Point Circle Algorithm

Algorithm:

Step1:– Input radius (r) and center of circle (X_c, Y_c)

Plot the first point over the circle $(X_0, Y_0) = (0, r)$

Step2:– Calculate $P_0 = 1 - r$

Step3:– For each X_k

If $p_k < 0$ then next point is $(X_k + 1, Y_k)$ and $P_{k+1} = P_k + 2 \cdot X_{k+1} + 1$

Else next point is $(X_k + 1, Y_k - 1)$ and $P_{k+1} = P_k + 2 \cdot X_{k+1} + 1 - 2Y_{k+1}$

Step4:– Determine the symmetric points in other 7 octants.

Step5:– Move each pixel position (X, Y) as

$X = X + X_c$

$Y = Y + Y_c$

Step6:– Repeat Step3 to Step5 until $X \leq Y$

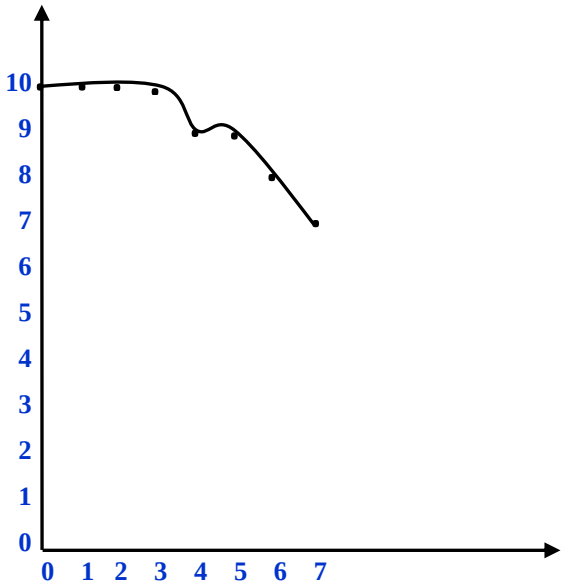
Q: Determine the points over the circle center at origin (0, 0) and radius r =10 using mid-point circle algorithm.

Ans :

Given, r = 10

$(0, 10)$

P_k	(X_{k+1}, Y_{k+1})
$1-10 = -9$	(1 , 10)
$-9 + 2 \times 1 + 1 = -6$	(2 , 10)
$-6 + 2 \times 2 + 1 = -1$	(3 , 10)
$-1 + 2 \times 3 + 1 = 6$	(4 , 9)
$6 + 2 \times 4 + 1 - 2 \times 9 = -3$	(5 , 9)
$-3 + 2 \times 5 + 1 = 8$	(6 , 8)
$8 + 2 \times 6 + 1 - 2 \times 8 = 5$	(7 , 7)



Two Dimensional(2D) Transformation

Sometimes we need to change the shape, size and orientation of object. These changes can be accomplished by transformation.

Basic Transformations

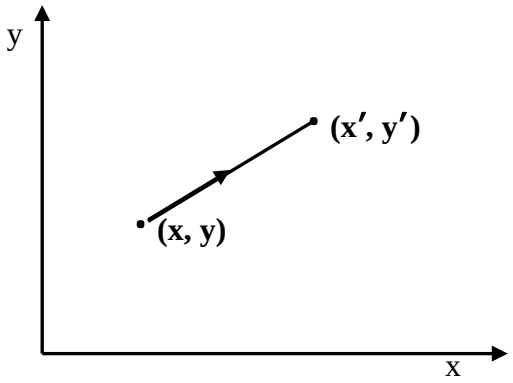
Basic transformations are Translation, Rotation and Scaling.

1. Translation

– Moving or shifting an object from one position to another is called Translation.

The equation of translation about origin is :

$$x' = x + t_x$$
$$y' = y + t_y$$



where (x', y') is translation point
(x , y) is original point
t_x , t_y are translation parameters

In matrix form :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = T + P$$

2. Rotation

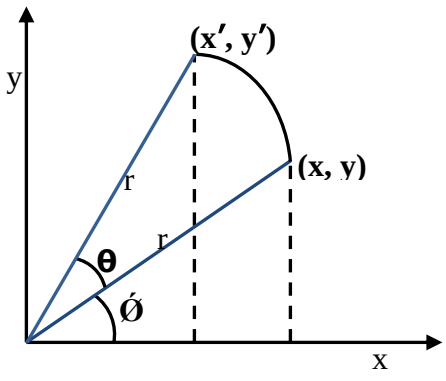
The object is rotated θ degree about origin.

If direction of rotation is anti-clock wise θ = +ve

If direction of rotation is clock wise θ = −ve

The equation of 2D rotation about origin is :

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$



where (x', y') is the rotation point
(x, y) is the original point
θ is angle of rotation

In matrix form:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$P' = R_\theta \cdot P$$

3. Scaling

Scaling means increasing or decreasing the size of an object or picture.

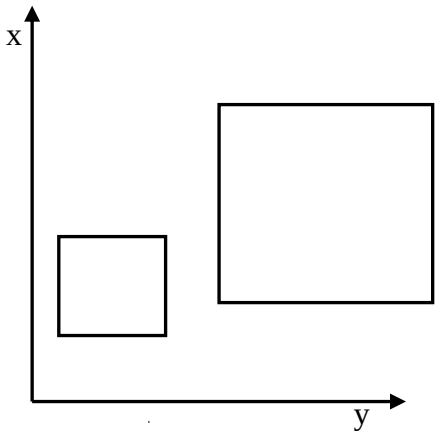
The equation of 2D scaling about origin is :

$$x' = S_x \cdot x$$
$$y' = S_y \cdot y$$

where (x', y') is the scaling point

(x, y) is the original point

S_x and S_y are scaling parameters in x and y direction.



In matrix form

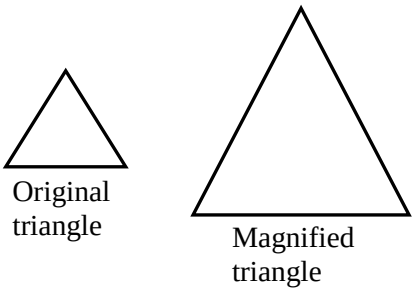
$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$P' = S \cdot P$$

Scaling parameters > 1 indicates increase in size

< 1 indicates decrease in size

= 1 indicates no change in size



If $S_x = S_y$ then it is called uniform or homogeneous scaling

If $S_x \neq S_y$ then it is called non-uniform or heterogeneous scaling

In uniform scaling, the original shape of the picture does not change.

In non-uniform scaling, the original shape of the picture changes.

Other transformations

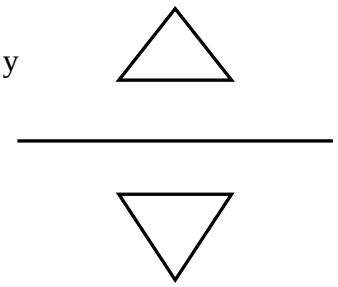
Reflection

- Reflection means mirror image.
- Reflection is 180^0 rotation of an object.

Reflection about x–axis

$x' = x$

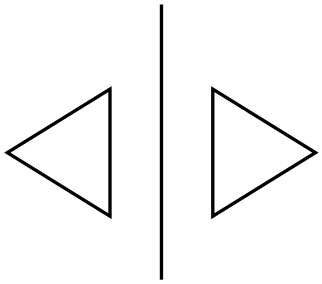
$y' = -y$



Reflection about y–axis

$x' = -x$

$y' = y$

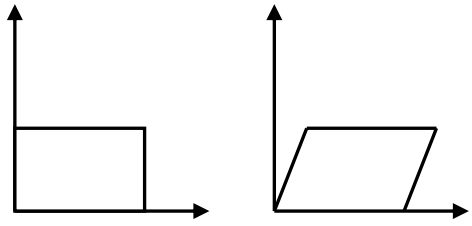


Shearing

- Shearing means stretching of object

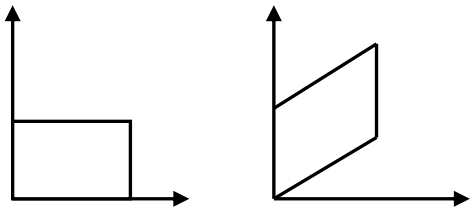
Shearing in X–direction:

$$x' = x + Sh_x \cdot y$$
$$y' = y$$



Shearing in Y–direction:

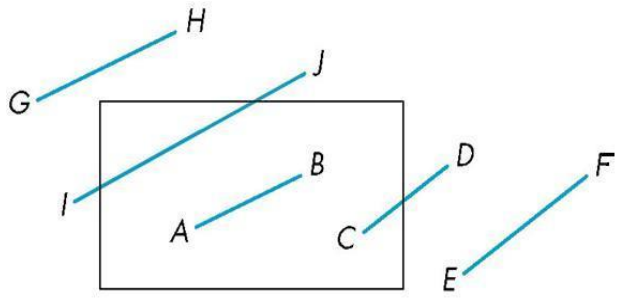
$$x' = x$$
$$y' = y + Sh_y \cdot x$$



Clipping

- Clipping means to Cut (or Crop) a portion of a picture.
- Clip that portion which is outside the window.

Line Clipping



There are 3 categories of line:

1. Visible Line : Line is inside the window. Example: Line AB
2. Invisible Line : Line is outside the window. Example: Line GH, EF
3. Partially Visible Line : Line is inside as well as outside the window. Example: Line CD, IJ

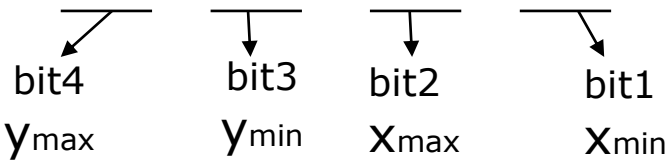
Cohen Sutherland Algorithm

- It is an efficient line clipping algorithm.
- It was developed by Dr. Ivan E. Sutherland.
- Window has 4 boundary lines. They are x_{min} , x_{max} , y_{min} , y_{max}

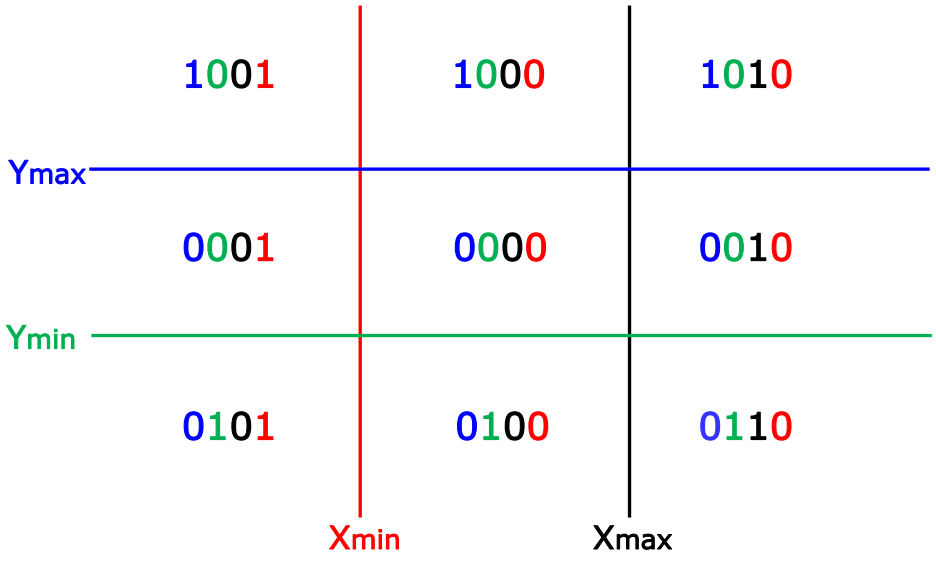
Region code:

- Total area is divided into 9 regions or locations.
- Each region has a region code.

A region code has 4 bits.



If the point is inside then bit = 0.
If the point is outside then bit = 1



ALGORITHM (Cohen Sutherland)

Step 1: Enter the end-points of line

Step 2: Find region code for end points.

Step 3: If both the region codes = 0000 then Line is visible.

Step 4: If AND operation of region codes = 0000 then Line is partially visible.

Step 5: If AND operation of region codes ≠ 0000 then Line is invisible.

Step 6: Calculate the intersection points as follows:

$$x_{in} = x_1 + \frac{y - y_1}{m}$$
$$y_{in} = y_1 + (x - x_1) \cdot m$$

where x = x_{min} or x_{max}

y = y_{min} or y_{max}

Step 7: Replace the end point with the intersection point.

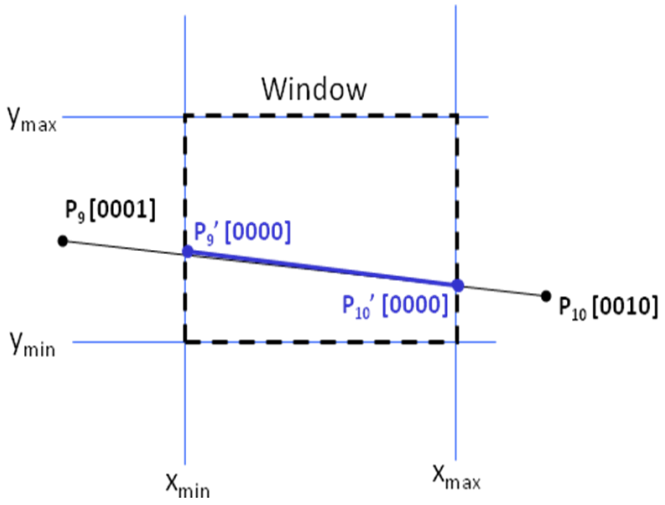
Step 8: Repeat above steps for all the lines.

Example: Consider the line P_9 to P_{10}

- Start at P_9
- From two region codes of end-points; we know that line crosses the left boundary so calculate intersection point P_9'

Consider the line P_9' to P_{10}

- Start at P_{10}
- Calculate the intersection point P_{10}'
- P_9' to P_{10}' is inside the window



Polygon Clipping

- Polygon clipping is also called Area clipping.
- Polygon is clipped against left, right, top and bottom boundary.

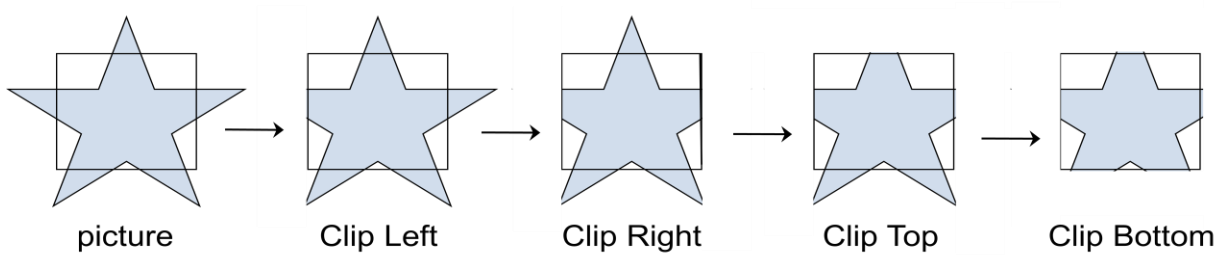


Fig : Clipping polygon against 4 boundary clipper (Left, Right, Top, Bottom)

Sutherland-Hodgeman Algorithm

There are four cases

Case 1: Out→In Case 2: In→In Case 3: In→Out Case 4: Out→ Out

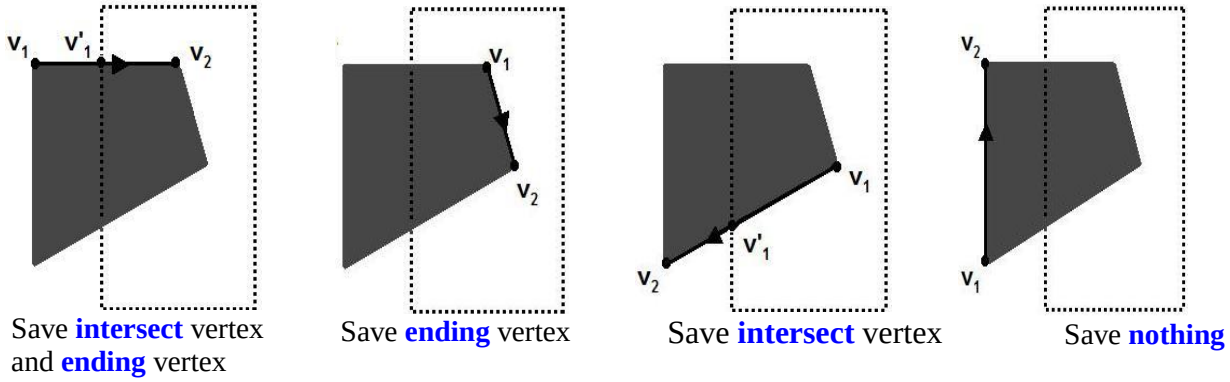
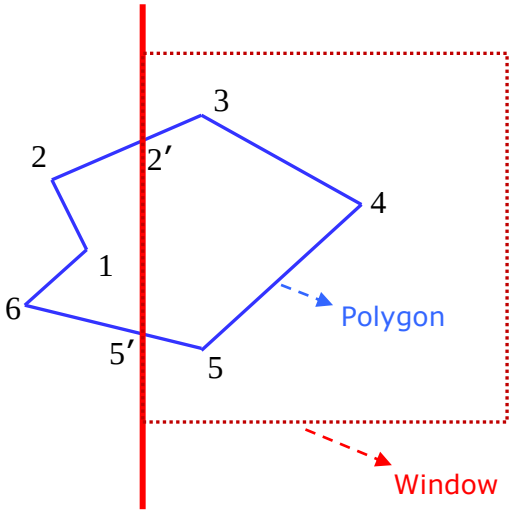


Fig: Clipping a polygon against left boundary

Example: Processing polygon in figure against the left window boundary.

Pairs of vertices	Case	Output
1 , 2	out→ out	
2 , 3	out → in	2′ , 3
3 , 4	in → in	4
4 , 5	in → in	5
5 , 6	in → out	5′
6 , 1	out→ out	



Output Polygon : 2′ 3 4 5 5′

Polygon Filling

- Polygon Filling is the process of filling (or coloring) a particular polygon / region / area.

Seed Fill Algorithm

- Seed means starting point. An interior point of polygon is taken as seed.
- Coloring begins from the seed.
- Coloring proceeds through the neighboring pixels.
- Coloring stops when the region is completely filled with color.

Seed fill algorithm is of two types

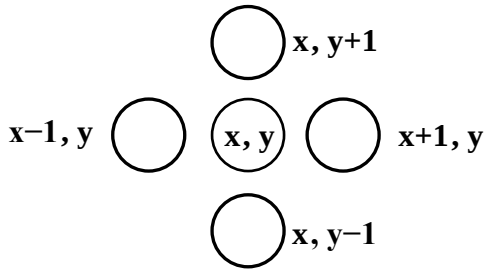
i) Boundary-Fill algorithm

ii) Flood-Fill algorithm

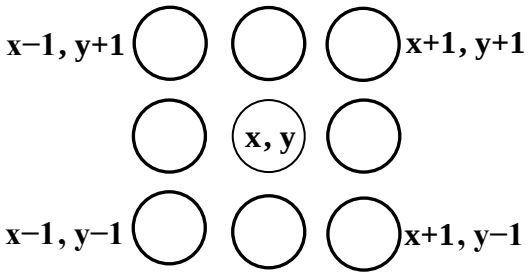
Boundary-Fill algorithm

- Filling starts from a seed point and continue until the boundary color is found.
- Algorithm uses either 4-conneected or 8-connected method for filling the pixels

4-conneected method: 4 neighboring pixels (left, right, top, bottom pixels)



8-connected method: 4 neighboring pixels + 4 diagonal pixels



The procedure for 4-connected is given below:

```
void boundaryfill (int x, int y, int boundarycolor, int fillcolor)
{
        int color;
        color = getpixel(x, y)   ; //find color at (x,y) position
        if ( color != boundarycolor  &&   color != fillcolor)
        {
                putpixel(x, y, fillcolor); //give color to (x,y) position
                boundaryfill(x+1, y, boundarycolor, fillcolor)
                boundaryfill(x−1, y, boundarycolor, fillcolor)
                boundaryfill(x, y+1, boundarycolor, fillcolor)
                boundaryfill(x, y−1, boundarycolor, fillcolor)
        }
}
```

Flood-Fill algorithm

- Filling starts from a seed point and continue until old color is replaced with new color.
- Interior-color(or old-color) is replaced with the fill-color(new-color).

The procedure for 4-connected is given below:

```
void floodfill (int x, int y, int oldcolor, int newcolor)
{
        int color;
        color = getpixel(x, y) ;  //find color at (x,y) position
        if (color = = oldcolor)
        {
                putpixel(x, y, newcolor);  //give color to (x,y) position
                floodfill(x+1, y, oldcolor, newcolor) ;
                floodfill(x−1, y, oldcolor, newcolor) ;
                floodfill(x, y+1, oldcolor, newcolor) ;
                floodfill(x, y−1, oldcolor, newcolor) ;
        }
}
```

Boundary-fill algorithm	Flood-fill algorithm
1. It is used for coloring a polygon. 2. Cannot be applied to a polygon with more than one boundary color. 3. It can be applied when interior region contains more than one color. 4. Take more memory and time due to recursion.	1. It is used for re-coloring a polygon. 2. Can be applied to a polygon with more than one boundary color. 3. It cannot be applied when interior region contains more than one color. 4. Take more memory and time due to recursion.

Visible Surface Detection Methods

From a viewing position, we will determine what is visible and what is invisible. This is called "visible surface detection" or "hidden surface elimination".

Z-Buffer Algorithm

- Z means Depth. It is also called Depth-Buffer Algorithm
- Compares depths of surfaces and select the surface nearest to view plane.

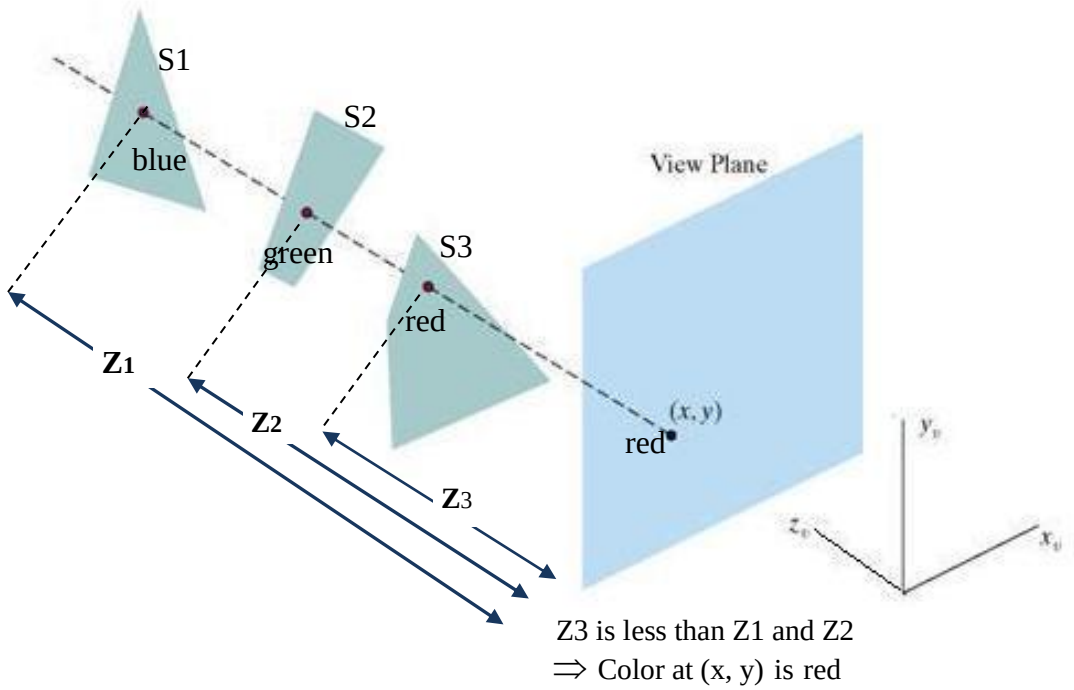


Fig : Three surfaces(S1, S2, S3) with Z values Z1, Z2, Z3

Algorithm:

// Depth-buffer(x, y) stores the z-values at (x, y) position.

// Frame-buffer(x, y) stores the color-values at (x, y) position

Step 1: For each position (x, y), Initialize :

Depth-buffer(x, y) = ∞ , Frame-buffer(x, y) = background-color

Step 2: Process each surface as follows:

(i) Calculate z at (x, y) position

(ii) if z < Depth-buffer(x, y) then

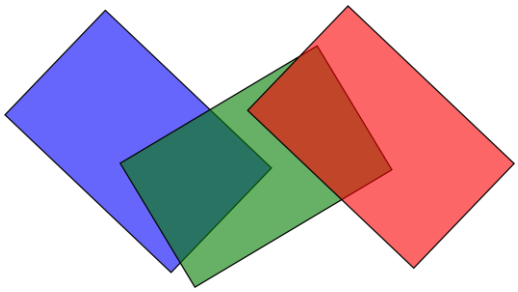
Depth-buffer(x, y) = z

Frame-buffer(x, y) = Surfacecolor(x, y)

Disadvantage: Z-buffer can't be used for transparent surfaces.

A-Buffer Algorithm

- A-buffer refers to Accumulation Buffer.
- It is an extension of the depth-buffer Algorithm.
- It operates just like depth-buffer Algorithm.
- Used for transparent surfaces.



Linked list:

depth	surface data

Fig : Transparent surfaces in 3D (one surface is present behind other surface)

The depth and opacity parameters are used to determine the colour of a pixel.
Surface information in the A-buffer includes:

- RGB intensity
- Opacity parameter
- Depth
- Other surface rendering parameters

Scan-Line Algorithm

- Two important tables are maintained:

  Edge table contains:
  - endpoints of each Edge
  - pointers to surface table

  Surface table contains:
  - surface material properties
  - pointers to edge table

- An active list is used for each scan-line. It stores those edges that intersect the scan-line.
- A flag variable is used for each surface. flag is on when scanline is inside surface. flag is off when scanline is outside surface.
- For each scan-line, calculate intersection points with polygon edges. Sort intersection points by increasing order of x. Then put respective color between two intersection points.
- Depth comparison is done when more than one surface overlap each other. [if more than one flag is on at same time then it indicates overlap]

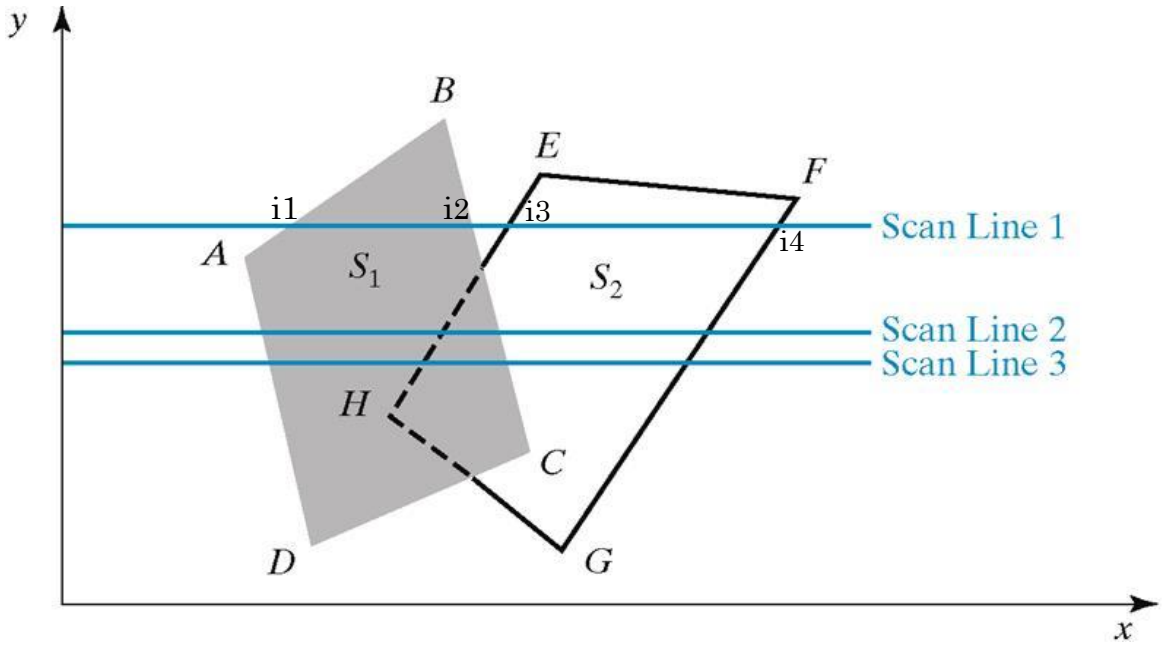


Fig: Surface S1 hides some portion of Surface S2. Because S2 is present behind S1 (Dashed line indicates hidden line)

– For scanline1, Intersection points are i1, i2, i3, i4

Between (i1, i2) we put color of S1.

Between (i3, i4) we put color of S2.

Painter's Algorithm

- This algorithm is similar to the idea of a painter. Hence it is called Painter's algorithm.
- It is also called depth-sorting algorithm.
- A Painter initially paints the distant part of the picture. After that he paints the nearer part of the picture. Similarly, Painter-algorithm paints all the polygons in decreasing order of z-value.

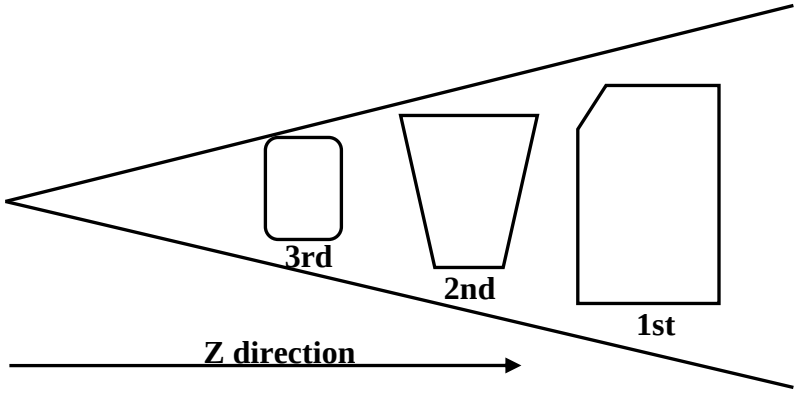


Fig : Objects (or Pictures) in 3D

- Objects are sorted by decreasing order to Z-values (farthest to nearest). Objects are designed (colored), one by one in this sorted order.

Bezier Curves

- Bezier Curve was developed by the French engineer Pierre Bézier.
- Bezier curve is an approximation spline.

Properties of Bezier Curve

(i) It passes through first and last control point.

(ii) Slope at the beginning of the curve is along the first two control points. The slope at the end of the curve is along the last two control points.

(iii) It lies within the convex-hull of control points.

Local control property: if we apply transformation to one of the control point then other control points are affected.

Global Control property: Bezier curve will pass closer to a coordinate position by assigning multiple control points to that position.

Control points decide the degree of polynomial equation of Bezier curve.
If number of control points = 4 then degree of equation = 3 (cubic)
If number of control points = 3 then degree of equation = 2 (quadratic)
If number of control points = n then degree of equation = n−1

A point P(u) on the curve can be written as:

$P(u) = M_{Bez} \cdot M_{Geom}$

where M_{Bez} = Martix of Bezier polynomial
M_{Geom} = Martix of control points

The polynomial equation of Bezier curve are:

$$P(u) = \sum_{k=0}^{n} p_k \; BEZ_{k,n}(u) \qquad where \quad 0 \le k \le 1$$

$$BEZ_{k,n}(u) = C_{n,k}(u) \cdot u^k \cdot (1-u)^{n-k}$$

$$where \quad C(n,k) = \frac{n!}{k!\;(n-k)!} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Bezier Curve with 4 control points (Cubic Bezier curve)

Put the following values in equation(1)

putting k = 0, n = 3 $\quad BEZ_{0,3}(u) = C_{3,0}(u)\, u^0 (1-u)^{3-0}$

$= -u^3 + 3u^2 - 3u + 1$

putting k =1, n = 3 $\quad BEZ_{1,3}(u) = C_{3,1}(u)\, u^1 (1-u)^{3-1} = 3u(1-u)^2$

$= 3u^3 - 6u^2 + 3u$

putting k = 2, n = 3 $\quad BEZ_{2,3}(u) = C_{3,2}(u)\, u^2 (1-u)^{3-2}$

$= -3u^3 + 3u^2$

putting k = 3, n = 3 $\quad BEZ_{3,3}(u) = C_{3,3}(u)\, u^3 (1-u)^{3-3}$

$= u^3$

$$M_{BEZ} = \begin{bmatrix} -u^3+3u^2-3u+1 \\ 3u^3-6u^2+3u \\ -3u^3+3u^2 \\ u^3 \end{bmatrix}$$

Let (X1, Y1) , (X2, Y2) , (X3, Y3) , (X4, Y4) are 4 control points.

$X(u) = X1(-u^3 + 3u^2 - 3u + 1) + X2(3u^3 - 6u^2 + 3u) + X3(-3u^3 + 3u^2) + X4(u^3)$

$Y(u) = Y1(-u^3 + 3u^2 - 3u + 1) + Y2(3u^3 - 6u^2 + 3u) + Y3(-3u^3 + 3u^2) + Y4(u^3)$

Q: Plot Bezier curve with 4 control points (0, 1) , (2, 5) , (5, 5) , (8,0)

$X(u) = X1(-u^3 + 3u^2 - 3u + 1) + X2(3u^3 - 6u^2 + 3u) + X3(-3u^3 + 3u^2) + X4(u^3)$

$= 0(-u^3 + 3u^2 - 3u + 1) + 2(3u^3 - 6u^2 + 3u) + 5(-3u^3 + 3u^2) + 8(u^3)$

$= -u^3 + 3u^2 + 6u$

$Y(u) = Y1(-u^3 + 3u^2 - 3u + 1) + Y2(3u^3 - 6u^2 + 3u) + Y3(-3u^3 + 3u^2) + Y4(u^3)$

$= 1(-u^3 + 3u^2 - 3u + 1) + 5(3u^3 - 6u^2 + 3u) + 5(-3u^3 + 3u^2) + 0(u^3)$

$= -u^3 - 12u^2 + 12u + 1$

Putting u values from 0 to 1 (i.e. 0.1, 0.2, ... 0.9, 1). We find the following graph.

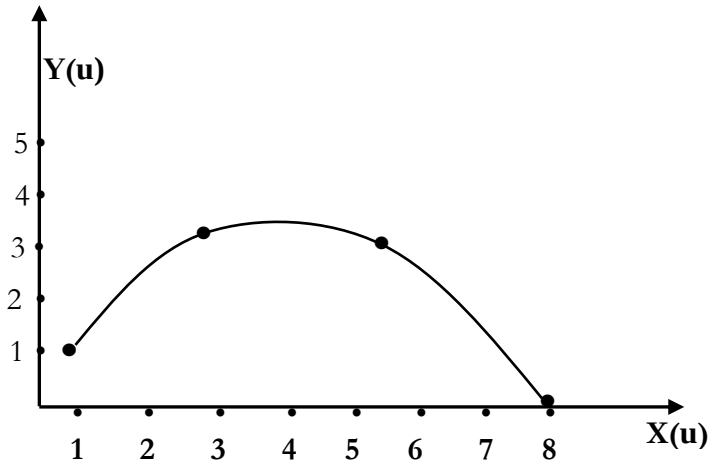
Let us find x(u), y(u) for u=0.1, u=0.4, u=0.7, u=1.0

u=0.1 x(u) = 0.629 , y(u) = 1.079

u=0.4 x(u) = 2.816 , y(u) = 3.816

u=0.7 x(u) = 5.327 , y(u) = 3.177

u=1.0 x(u) = 8 , y(u) = 0



Bezier Curve with 3 control points (Quadratic Bezier curve)

Put the following in equation(1)

putting k = 0, n = 2 $BEZ_{0,2}(u) = C_{2,0}(u)\, u^0 (1-u)^{2-0} = u^2 - 2u + 1$

putting k =1, n = 2 $BEZ_{1,2}(u) = C_{2,1}(u)\, u^1 (1-u)^{2-1} = -2u^2 + 2u$

putting k = 2, n = 2 $BEZ_{2,2}(u) = C_{2,2}(u)\, u^2 (1-u)^{2-2} = u^2$

Projection

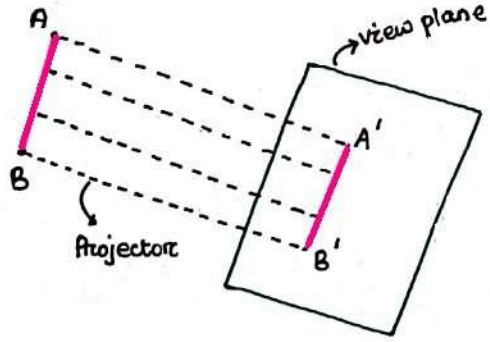
→ Projection is mapping of a 3D picture on a 2D plane.

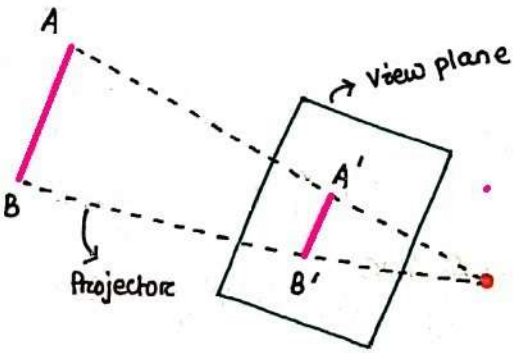
i.e. "Projection = 3D to 2D"

→ Projection is of two types,

1. Parallel Projection
2. Perspective Projection

Parallel Projection	Perspective Projection
→ Projectors are parallel to each other.	→ Projectors are not parallel to each other.
→ Non-realistic picture is produced.	→ Realistic picture is produced.
→ Relative proportion of size is maintained.	→ Relative proportion of size is not maintained
→ used in engineering drawing.	→ used in camera, computer etc.





Parallel Projection

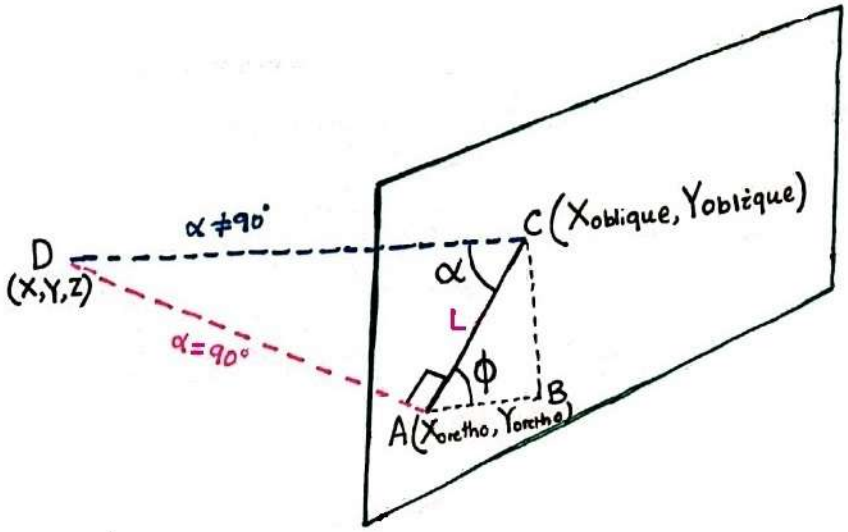
- If the projector is perpendicular (=90°) then it is called as, Orthographic projection.
- Otherwise, it is called as, Oblique Projection (≠90°).

Orthographic Projection: –

The equation of Orthographic projection is given by,

$$X_{ortho} = X$$
$$Y_{ortho} = Y$$

Oblique Projection:-



L = distance between orthographic and oblique projection.

α = angle between line L and projector.

ϕ = angle between line L and X-axis

In ΔABC,

$$\cos\phi = \frac{AB}{AC} = \frac{X_{oblique} - X_{ortho}}{L}$$

$$\Rightarrow X_{oblique} = X_{ortho} + L\cos\phi \quad \text{--------(1)}$$

Similarly, $Y_{oblique} = Y_{ortho} + L\sin\phi$ --------(2)

In ΔADC, $\tan\alpha = \frac{AD}{AC} = \frac{Z}{L}$

$$\Rightarrow L = \frac{Z}{\tan\alpha} = Z\cot\alpha$$

Putting the value of L in equation (1) and (2), we find,

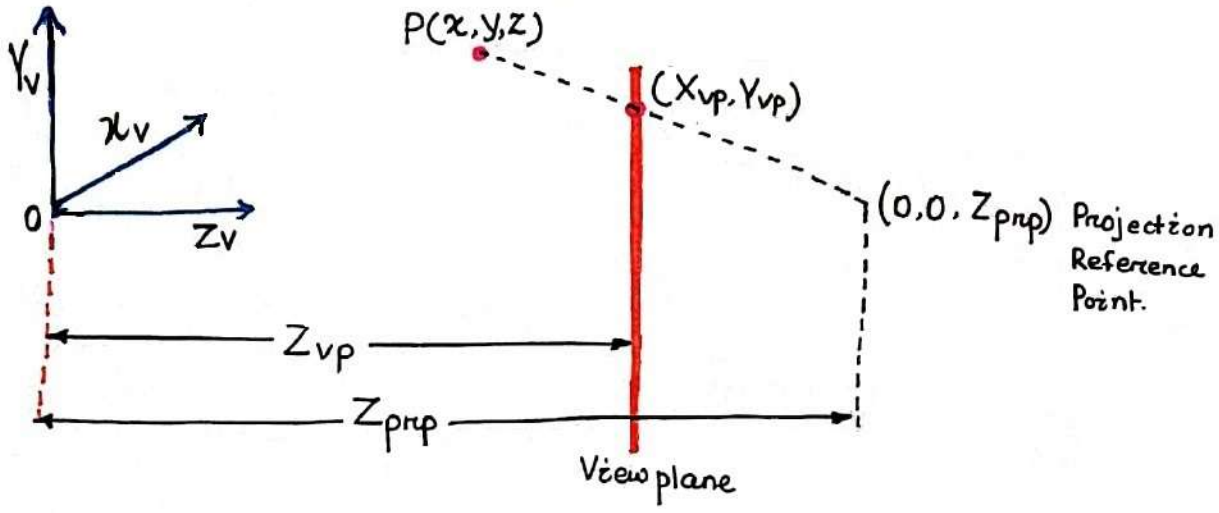
$$X_{oblique} = X_{ortho} + Z\cot\alpha\cos\phi$$

$$Y_{oblique} = Y_{ortho} + Z\cot\alpha\sin\phi$$

Putting $\alpha = 90^\circ$ in above equation, We find, $X_{oblique} = X_{ortho}$

$Y_{oblique} = Y_{ortho}$

Perspective Projection



Z_{vp} = distance from Origin to View-plane (or camera position)

Z_{prp} = distance from Origin to projection reference Point (or focus Point)

Any Point (x', y', z') over the projector is given by,

$$x' = x - x \cdot u$$
$$y' = y - y \cdot u$$
$$z' = z - (z - Z_{prp})u \quad \text{----------(1)}$$

where, $0 \leq u \leq 1$

Putting $u=0$ in eq^n (1)

$$x' = x$$
$$y' = y$$
$$z' = z$$

{ When $u=0$, we are at $P(x,y,z)$

Putting $u=1$ in eq^n (1)

$$x' = 0$$
$$y' = 0$$
$$z' = Z_{prp}$$

{ When $u=1$, we are at the PRP (Projection Reference Point)

At the view plane $z' = z_{vp}$

Putting $z' = z_{vp}$ in eq^n (1), we can find "u" as follows:

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

Putting $x' = x_{vp}$, $y' = y_{vp}$, $u = \frac{z_{vp} - z}{z_{prp} - z}$ in eq^n (1)

we got:

$$x_{vp} = x - x\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

$$y_{vp} = y - y\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

Above eq. shows that, x_{vp} and y_{vp} depend on z_{vp} and z_{prp} (i.e. camera position and focus point)

Why Lighting?

If we don't have lighting effects then nothing looks three dimensional!

Diffuse Reflection

Surfaces that are rough tend to reflect light in all directions. This scattered light is called diffuse reflection.

Specular Reflection

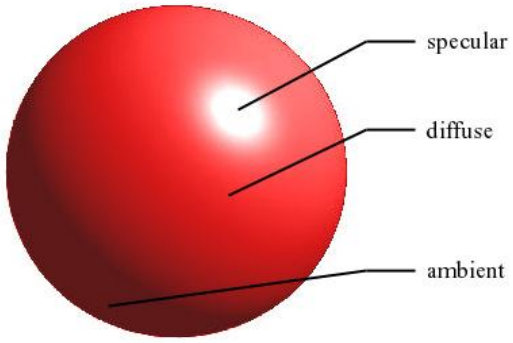
Some of the reflected light is concentrated into a highlight or bright spot. This is called specular reflection.

Ambient Light

A surface also gets light by reflections from nearby light sources. These lights are called ambient light.

Total reflected light from a surface = contributions from light sources + Reflected light**.**

Example:



Basic Models

Every surface is fully illuminated by the ambient light, Ia

Diffuse Reflection Model

Assume that surfaces reflect incident-light with equal intensity in all directions. Such surfaces are called ideal diffuse reflectors or Lambertian reflectors.

kd = diffuse-reflection coefficient (kd is assigned a value between 0.0 and 1.0)

= fraction of incident light that is to be scattered as diffuse reflection from that surface

kd = 0.0 $\Rightarrow$ dull surface [absorbs almost all light]

kd = 1.0 $\Rightarrow$ shiny surface [reflects almost all light]

Ambient contribution to the diffuse reflection is given as: $I_{ambdiff} = k_d I_a$

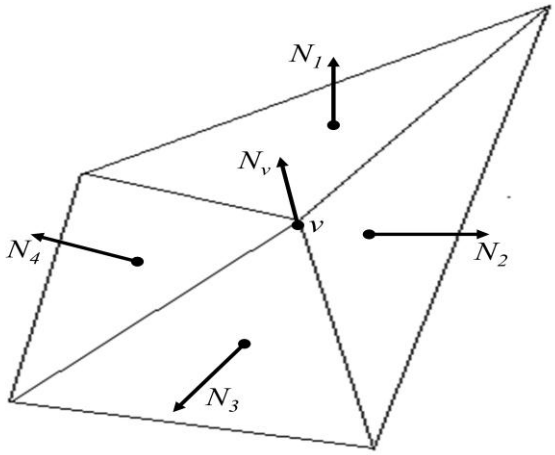
Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.

GOURAUD SHADING

- Developed in 1970s by Henri Gouraud
- Also called intensity-interpolation surface rendering
- Intensity levels are calculated at each vertex and interpolated across the surface

Steps:

1. Determine the average unit normal vector at each vertex of polygon
2. Apply an illumination model at each polygon vertex to obtain the light intensity at that position
3. Linearly interpolate the vertex intensities over the projected area of the polygon



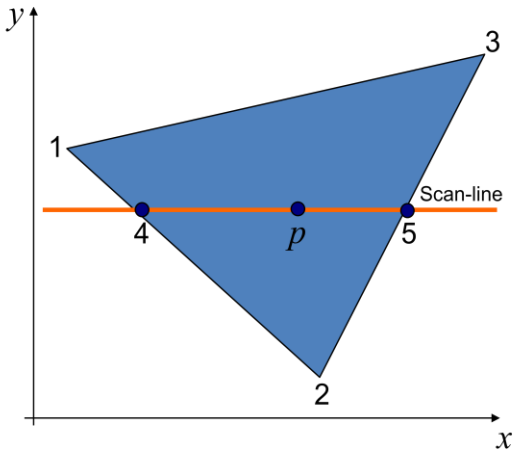
Average unit normal vector at v = N_V

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

$$More\ generally, \quad N_v = \frac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$$

Fig : Normal vector Nv is the average of normal to surfaces connected to v

Illumination values are linearly interpolated across each scan-line



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

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

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

Fig: Intensity at point 4 is interpolated from point 1and 2.
Similarly, point 5 is interpolated from point 2 and 3.
point p is interpolated from point 4 and 5.

Mach bands: A psychological phenomenon for which we see bright bands where two blocks of solid colour meet.

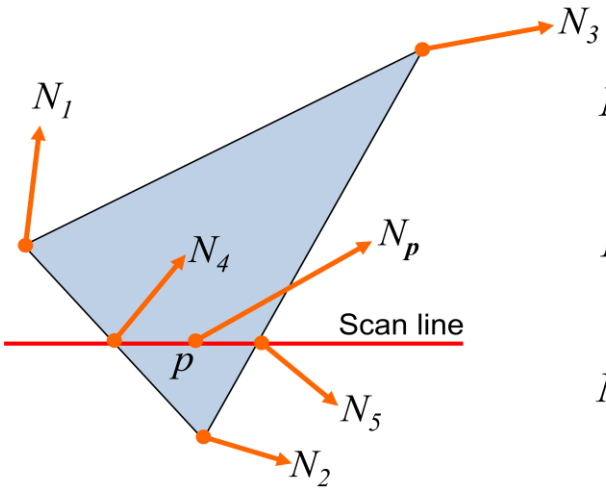
Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.

PHONG SHADING

- Developed by Phong Bui Tuong
- Used for rendering a polygon
- A more accurate interpolation based approach
- Also called normal-vector interpolation rendering (interpolates normal vectors instead of intensity values)

Steps:

1. Determine the average unit normal vector at each vertex of polygon
2. Linearly interpolate the vertex normals over the projected area of polygon
3. Apply an illumination model at positions along scan lines to calculate pixel intensities using the interpolated normal vectors



$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

Fig: Interpolation of surface normals

Note: Phong shading is slower than Gouraud shading as the lighting model is revaluated many times.

Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.

COMPUTER ANIMATION

Animation refers to moving picture.

- Animation is defined as **"**Quick display of sequence of pictures to create motion**"**.
- A picture is called frame. Frames are created by animator.

Design of Animation sequence

In general, animation sequence is designed with the following steps**:**

1. Storyboard layout
2. Object definition
3. Key-frame specification
4. Generation of in-between frames.

1. Storyboard

It defines the motion sequence as a set of basic events.

2. Object definition

The action of each object needs to be defined. Objects can be defined in terms of basic shapes such as polygons, curves etc.

3. Key frame

It is the detailed drawing of the animation-movie at a certain time.

4. In between frames

These are intermediate frames between the key frames. Usually, there are 3 to 5 in between frames for each pair of key-frame.

Types of Animation

Keyframe Animation

- Keyframe Animation is created through key-frames at different time interval.
- The animator creates only the start and end key-frames of the sequence. The in-between frames are automatically created by the computer by interpolating (according to) the key-frames.

Procedural Animation

- Procedural Animation is created by programs(or procedures).
- The animator does not create the individual frames. Animator defines the objects, their transformation and the sequence of animation through a program.
- Procedural Animation is used to generate realistic effect which is not possible in traditional animation. Example: simulation of particle systems (smoke, fire, water etc.), character (or behavior) animation.

Keyframe Vs Procedural Animation

- Key Frame animation defines starting and ending frames. Then "in between" frames are generated. Procedural animation uses programs to generate animation effects (special effects).
- Key frame animation has control over the motion parameters where as procedural animation does not have any control.
- Key frame animation is very labour intensive as compared to procedural animation.

Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.

METHODS OF CONTROLLING ANIMATION

1. Explicitly Declared Control

- All events in an animation are declared. This can be done at the frame level by specifying key frames and methods for interpolating between them.

2. Procedural Control

- Each object obtains knowledge about the static or dynamic properties of other objects.

3. Constraint-Based Control

- Many objects movements are determined by other objects which they come in contact. e.g. presence of wind.
- Constraints based on the environment can be used to control object's motion.

4. Analyzing Live Action-Based Control

- Control is achieved by examining the motions of objects in the real world. For example data glove measures the position and orientation of the user's hand.
- Rotoscoping is a technique where animators trace live action movement, frame by frame. A film is made in which people act out the parts of the characters in the animation, then animators draw over the film, replacing the human actors with their animated equivalents.

5. Kinematic and Dynamic Control

- Kinematics refer to the position and velocity of points.
- Dynamics takes the physical laws that govern kinematics.
  - Newton laws for the movement of large objects
  - A particle moves with an acceleration proportional to the forces acting on it.

Morphing

- Morphing is the process of transforming (converting) one picture into another.
- Morphing is used to produce special effects in entertainment industry.
  Example: Michael Jackson's music video, Terminator Movie etc.
- How morphing is done?
  - The first picture is slowly changed. [First picture disappears slowly.]
  - The second picture starts from this changed picture toward the original. [Second picture appears slowly.]

Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.

'Virtual Reality' basically means 'near to reality'.

Virtual reality describes a Three-Dimensional (3D) computer generated environment**.** Designers are using computers to create a virtual reality in order to see how their designs or products might work in the real world.

Applications of Virtual Reality

1. **Architecture**: With virtual reality, designers can interactively test a building before construction begins.
2. **Training**: These allow participants to safely practice skills that are dangerous and expensive to develop in the real world. For example: perform surgery, learn flying aircrafts, operate a plant control room, etc.
3. **Realization**: Virtual reality allows the participant to reach in the virtual world and manipulate data representations as if they are real objects.

Types of Virtual Reality systems

1. Non-Immersive (Desktop) System: It is the least immersive VR technique. Virtual environment (VE) is viewed through a high resolution monitor.

2. Semi-Immersive Projection System: It consists of a high performance graphics system with a large screen projector. Example: Flight simulator.

3. Fully Immersive Head-Mounted Display System: It give a sense of presence that cannot be done by the above two systems. The sense of immersion depends on several parameters such as view of HMD, resolution, update rate, and illumination of display.

Input and Output Virtual Reality devices

1. Data Glove

Data glove is a virtual reality input device. It has many sensors to find the exact position and orientation of the hand. It provides hand-gesture based input to the system rather than pressing the buttons on the keyboard.

2. Head-Mounted Display (HMD)

HMD come with earphones and goggles. HMD is used with data glove. Sensors are present in goggles which track the movement of the user's eyes and adjust the picture on the screen.

Imagine you are in the dining room at home. Once you put on a HMD connected to a computer, you may feel that you have been teleported to the surface of Mars. When you take a step forward, your point of view (3D) moves forward in this virtual space. You find yourself no longer in the dining room. Everything is so real that you cannot deny you are exploring on that planet. These are all due to the 3D Computer Generated Images (CGI) produced by the sophisticated computer.

Note is prepared by : Shekharesh Barik
Asso. Professor, DRIEMS University, Odisha.