Unit: 1

Introduction of Computer Graphics.

Computer Graphics:

It is a field or art related to the generation of graphics using digital computers. It includes the creation, storage and manipulation of images of objects.

We use different input and display devices to generate and perform the modification on the digital images.

Some terminologies related to computer graphics:

(i) Modeling:

Creating and representing the geometry of objects in the 3D world.

(ii) Rendering:

Generating 2D images of the 3D objects.

(iii) Animation:

Describing how objects change in time.

Image processing:

It is the process of modifying or interpreting existing pictures. It analyzes picture to derive descriptions in mathematical or geometrical forms. It is a part of computer graphics that handles image manipulation.

Example: Making blurred image visible.

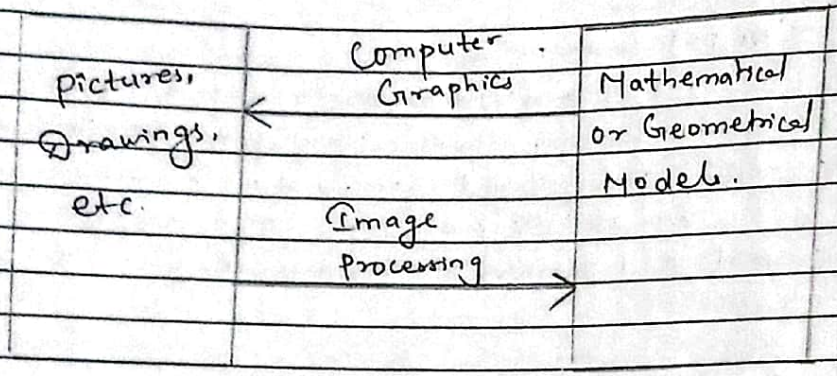


Fig: Image processing.

Application of CG:

(1) Graphical User Interface:

The GUI application on computer system provide the facility of selection of menu items, icons and object on screen, directly either by touching it (in case of touch panel) or just by clicking on it. This is more efficient mechanism rather than the assembly language coding.

(2) Plotting:

To represent meaningfully and concisely the trends and patterns of complex data, we use histogram bar, pie charts, task-scheduling charts, etc.

(3) Computer Aided Drafting & Design.

→ To design components and systems of mechanical, electrical, electromechanical and electronic devices including structures such as buildings, automobiles bodies, VLSI chips, etc.

→ Computer Aided Design (CAD) and computer Aided Manufacture (CAM) are important terms in the field of design and manufacture.

(i) CAD:

Involves the use of computer hardware and graphics software to generate design drawings.

(ii) CAM:

It is a system of automatically producing finished products by using computer controlled products machines.

(4) Simulation:

It is the limitation of the conditions encountered in real life. The simulation provides the virtual reality of any system and helps us to train people without accidental loss.

Examples: pilot training, military weapon training space training, etc.

(5) Entertainment:

Computer Graphics are used for cartooning, game playing, animation, etc.

(6) Cartography:

It is a subject which deals with the making of maps and charts of the geographical data in accurate and schematic way.

(7) Art and Commerce

Traffic lights, new car model, - - -

(8) Medical Imaging:

The range of application spans from tools for teaching and diagnosis, all the way to treatment. CG is tool in medical applications.

(9) Education and Training:

Different computer graphics and images are used in schools and many training center for the better understanding the subject of interest.

Other Applications:

Multimedia systems, Animation for entertainment, Robotics & virtual reality, et.

✓ Raster:

A rectangular array of points or dots.

✓ Pixel

→ One dot or picture element of the raster.

→ The pixel (a word invented from "picture element") is the basic unit of programmable color on computer display or in a computer image.

✓ Bitmap:

Ones and zeros representation of the rectangular array points on screen.

Black and white: - bitmap

Color : pixmap

✓ Vectors:

Vectors are based on mathematical formulas and can be scaled infinitely without any loss in quality. Every line and shape has a value that changes when the image expands.

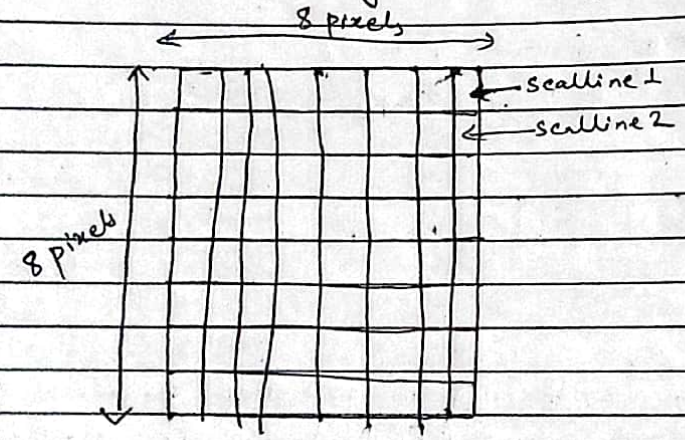
Raster Image / Pixmap or Bitmaps:

Bitmaps rely on a series of square blocks called pixels, arranged on a grid. The quality of the images depends on the amount of pixels per square inch.

Resolution:

⇒ The maximum number of points (pixel) that can be displayed without overlap on a CRT is referred to as the resolution.

⇒ It is also defined as the number of points per unit of measure (per centimeter or per inch) that can be plotted horizontally and vertically.



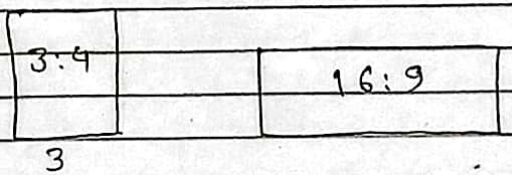
Resolution = 8 x 8 pixels (Standard way)
= 64 pixels (Total resolution)

4K video Resolution:

It was named so because it has 4000 pixels horizontal resolution approximately. The format can't have the change in horizontal resolution so changes in aspect are made through the vertical resolution.

Aspect ratio:

It describes the proportional relationship between its width and its height.



Q. Find out the aspect ratio of the raster system using 8 x 10 inches screen and 100 pixel/inch. Also find the resolution.

Solution

Here

Given: Width = 8 inches
Height = 10 inches

Then,

Aspect ratio = $\frac{\text{Width}}{\text{Height}} = \frac{8}{10} = 4:5$ //

Total resolution = $8 \times 100 \times 10 \times 100$

$= 8 \times 10^7$ pixels //

$= 800 \times 1000$ resolution.

Persistence:

=> It means how long they continue to emit light after the electron beam is removed.

=> The time it takes the emitted light from the screen to decay to one-tenth of its original intensity.

=> Lower persistence phosphors require higher refresh rates to maintain a picture on the screen.

=> Lower persistence phosphor is used for animation and high persistence phosphor is used to display complex static images.

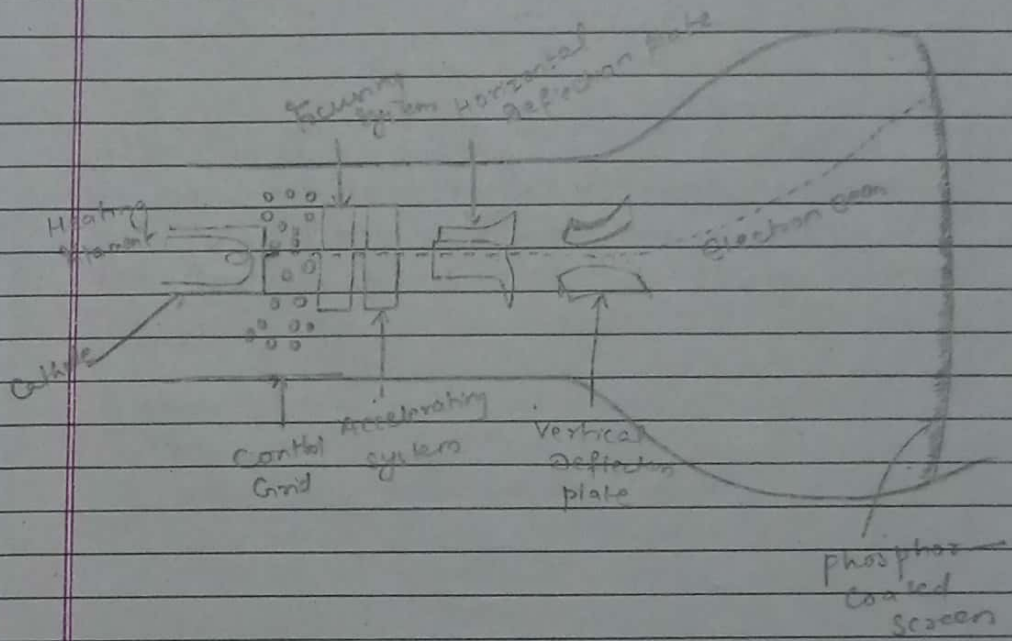
CRT:

-> A CRT is an evacuated glass tube, with a heating element on one end and a phosphor-coated screen on the other end.

-> When a current flows through this heating element (filament) the conductivity of metal is reduced due to high temperature. These cause electrons to pile up on the filament.

-> These electrons are attracted to a strong positive charge from the outer surface of the focusing anode cylinder

-> The forwarding last electron beam is known as cathode ray.



Main Components of CRT:

a) An electron gun
b) Control grid
c) focusing system and Accelerating Anode

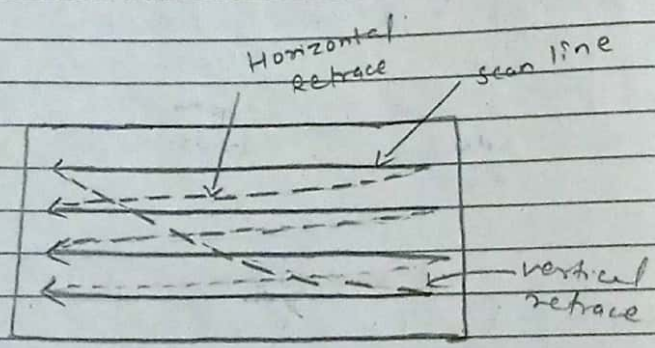
Types of Refresh CRT's

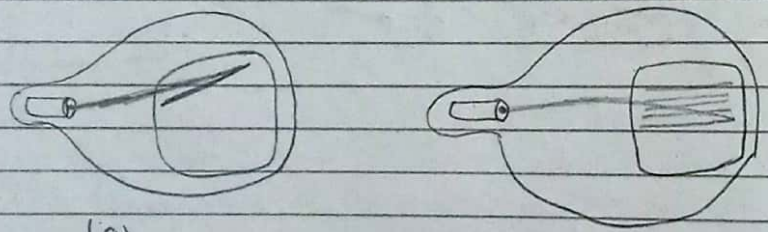
i) Raster-scan Displays.
ii) Random-scan Displays

Raster-scan Display:

=> The most common type of graphics monitor employing a CRT is the raster-scan display

=> The viewing screen is divided into a large number of discrete phosphor picture elements, called pixels.





(a)

Architecture of Raster Scan System

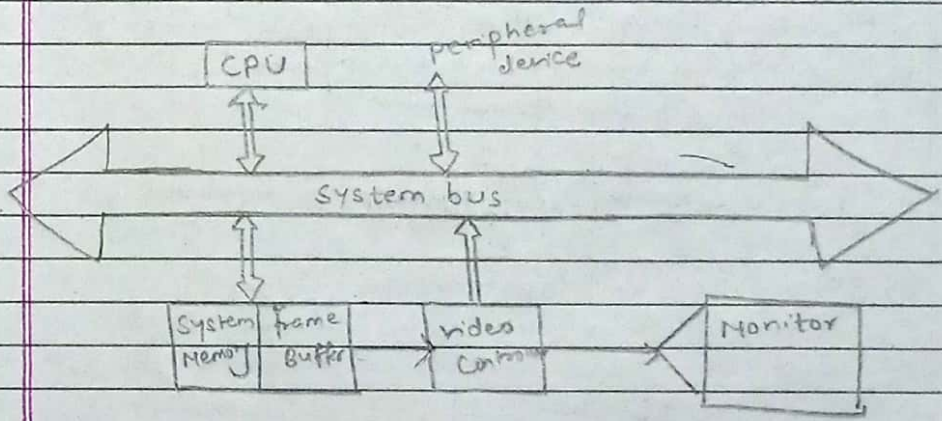


Fig: Architecture of Raster Scan System.

=> Besides CPU, graphics system consists of a special purpose procesor called video controller or display procesor.
=> The display processor controls the operation of the display device
=> A fixed area of system memory is reserved for the frame buffer.
=> The video controller has the direct access to the frame buffer for refreshing the screen.
=> The display processor has its own separate memory called display processor memory (frame buffer)
=> The display processor memory holds data plus the program that perform scan conversion and raster operations.
=> The frame buffer stores displayable image created by scan conversion and raster operations.

Types of Raster-Scan system:

i) Non-interlaced Raster-Scan system
ii) Interlaced " " "

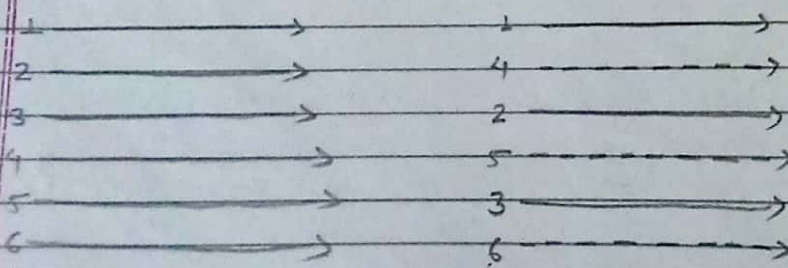


Fig: Non-interlaced

Fig: Interlaced.

Interlaced:

→ In the first pass, the beam sweeps be across every other scan line from top to bottom.
→ Then after the vertical re-trace, the beam sweeps out the remaining scan lines.
→ This is used with slower refreshing rate s.
→ An effective technique for avoiding screen flickering

Non-interlaced:

→ stored in a memory area called refresh buffer or frame buffer.
→ This memory area holds the set of intensity values for all the screen points.
→ Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (scan line) at a time

Random-Scan Displays (Vector displays / stroke writing / Calligraphe displays).

→ In a random scan display unit, electron beam directed towards only to the parts of the screen where a picture is to be drawn.
→ Random scan monitors draw a picture one at a time. So it's also refered as vector displays / - - - .
→ The component line can be drawn or refreshed by a random scan display system in any specific order.

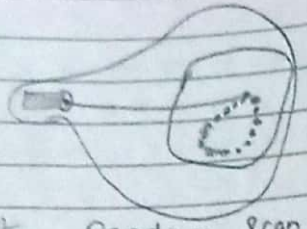


Fig. Random Scan Display.

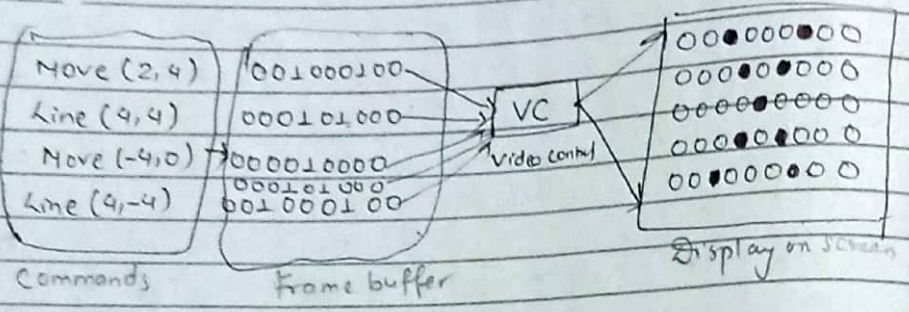


Fig: Raster display method on CRT monitor.

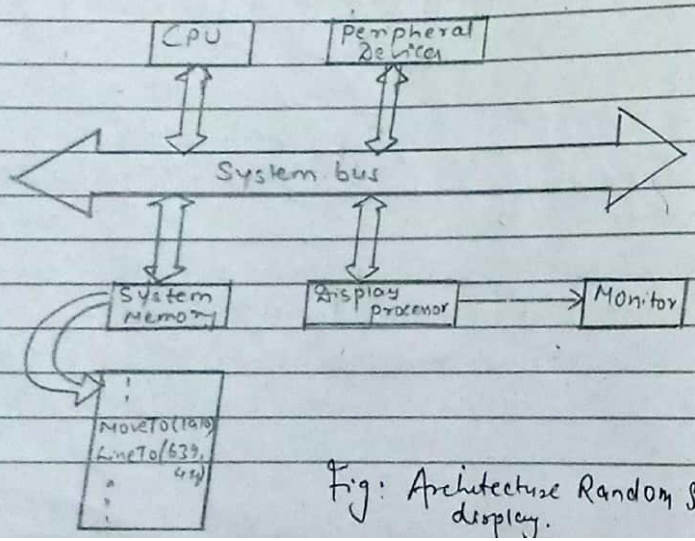


Fig: Architecture Random Scan display.

→ Graphics package creates a display list and stores in systems memory (consists of points and line drawing commands) called display list or display file.

Q Differences between Raster vs Random.

✓ Color CRT:

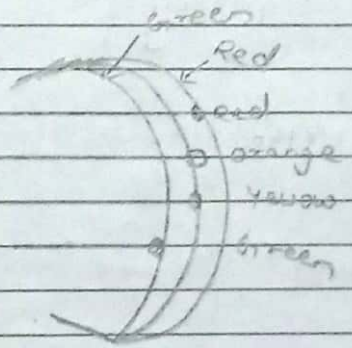
→ It displays color pictures by using a combination of phosphors that emit different-colored light.

→ By combining the emitted light from the different phosphors, a range of colors can be generated.

→ Two basic techniques.

(i) Beam Penetration Method
(ii) Shadow Mask Method.

(1) Beam Penetration Method



→ Cheaper method, used in vector scan displays
→ The inside section of CRT is coated with red (outer) and green (inner) phosphors
→ If electrons are slow, ~~per~~ only penetrate ~~outer~~ inner layers thus green light. If faster, penetrate both layers emitting red light.
→ If electron speed is adjusted, ~~various~~ other colors orange and yellow can be produced.

i) red more than green → orange
ii) green more than red → yellow.

Advantage:
→ Economical way to produce colors

Limitations:
→ Generation of only four colors is possible
→ poor picture quality.

Shadow Mask Method:

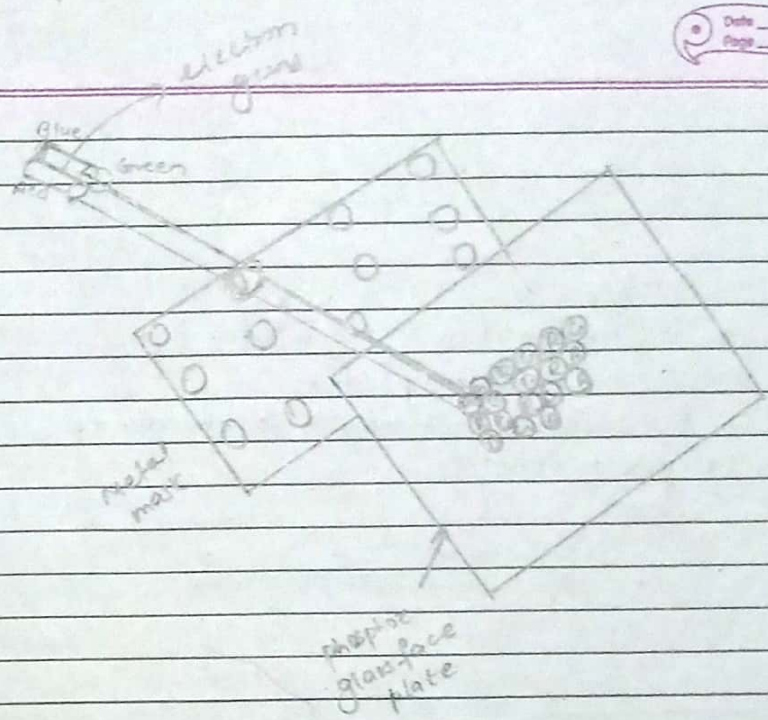
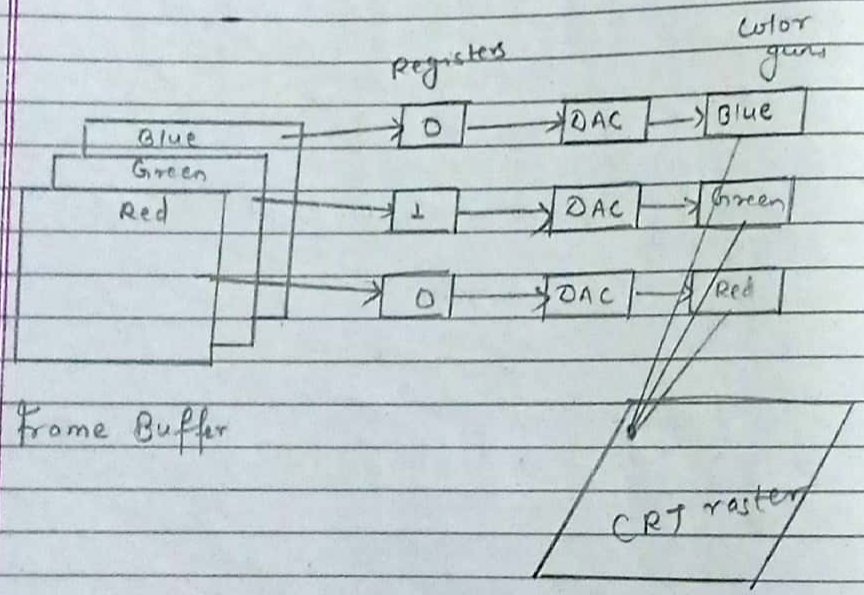


fig: Shadow Mask Method.

Frame Buffer:

→ The information in the buffer typically consists of color values for every pixel.

→ Bit planes or bit depth is the no. of bit corresponding to each pixel.



Frame Buffer

Red	Green	Blue	Color
0	0	0	Black
1	0	0	Red
0	1	0	Green
0	0	1	Blue
1	1	0	Yellow
0	1	1	Cyan
1	0	1	Magneta
1	1	1	White

Color Depth	No. of Displayed colors	Bytes of Storage per pixel	Common Name for Color Depth
4-bit	16	0.5	Standard VGA
8-bit	256	1.0	256-Color mode
16-bit	65,536	2.0	High Color
24-bit	16,777,216	3.0	True Color.

↳

Q.1. If the pixel values are accessed from the frame buffer with an average access time (for one single pixel) of 20 ns and the total resolution of the screen is 1024×800, will there be a flickering effect seen on the screen?

Solution

Here,

Resolution = 1024×800

Access time of single pixel = 20ns = 20×10^{-9} s

To find:

Does screen will have flickering effect or not?

Now,

Access time for one frame = $1024 \times 800 \times 20 \times 10^{-9}$ s

= 0.0163848 s

$\therefore f = \frac{1}{T} = \frac{1}{0.0163848} = 61.035$ Hz

Since the frequency is 61.035 Hz i.e. greater than 60, there will be no flickering effect. //

Q.2. In case of raster system with resolution 1024×1280, how many pixel could be accessed per second in the system by a display controller at a rate of 60 frames per second? What is access time per pixel in the system?

Solution

Here,

Resolution = 1024×1280

frequency = 60

Now,

Access time for one frame, $T = \frac{1}{f} = \frac{1}{60} = 0.016667$ s

Then,

~~Access time for single pixel~~

No. of pixel to be accessed per second = 1024×1280×60

= 78643200 //

Now,

Access time per pixel = ?

78643200 pixels can be accessed in 1 s.

So, 1 pixel can be accessed in $\frac{1}{78643200}$ s

= 1.2716×10^{-8} s //

Q.3. How long would it takes to load a 640x480 frame buffer with 12 bit per pixels if 10^5 bits can be transferred per second.

Solution

Here,

Resolution = 640x480

No. of bits per pixel = 12

No. of bits transferred per second = 10^5

Now,

Total no. of bits in the frame = 640x480x12

= 3686400

10^5 bits can be accessed in 1 s

1 bit can be accessed in $\frac{1}{10^5}$ s.

3686400 bits can be accessed in $\frac{1}{10^5} \times 3686400$ s

= 36.8664 s //

Q.4. If the total number of intensities achievable out of a single pixel on the screen is 1024 and the total resolution of the screen is 1024x800, what will be the required size of frame buffer in this case for the display purpose?

Soln

Here,

Given resolution = 1024x800

No. of bits per pixel is given by,

$2^n = 1024$ where, n = no. of bits

$\therefore n = 10$

Now,

Total bits in screen = 1024x800x10

= 8192000 bits

= $\frac{8192000}{8 \times 1024 \times 1024}$ MB

= 0.09766 MB. //

Q5 Consider three different raster system with resolution 640 by 400, 1280 by 1024 and 2560 by 2048. What size frame buffer (in byte) is needed for each of the system to store 12 bits per pixel? How much storage is required for each system if 24 bits per pixel are to be stored?

Unit: 2

Scan Conversion Algorithm

Digital Differential Analyzer:

→ Scan conversion line algorithm based on calculating either sample the line at unit intervals in one coordinate Δx or Δy

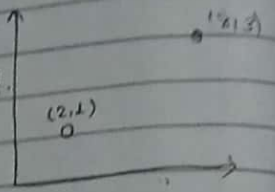
→ Determine the corresponding integer values nearest the line path in another co-ordinate.

Example 1

Digitize the line with end points (2,1) and (8,3) using DDA.

Solution

Let's draw given line from left to right.



Now,

Starting point $(x_1, y_1) = (2,1)$

Ending point $(x_2, y_2) = (8,3)$

$$\text{Slope } (m) = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-1}{8-2} = \frac{2}{6} = \frac{1}{3}$$

Here, slope is less than 1.

$\therefore x_{k+1} = x_k + 1$

$y_{k+1} = y_k + m$

S.N.	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
1.	2	1	(2,1)
2.	2+1=3	1+1/3=1.33≈1	(3,1)
3.	3+1=4	1.33+0.33=1.66≈2	(4,2)
4.	4+1=5	1.66+0.33=1.99≈2	(5,2)
5.	5+1=6	1.99+0.33=2.32≈2	(6,2)
6.	6+1=7	2.32+0.33=2.65≈3	(7,3)
7.	7+1=8	2.65+0.33=2.98≈3	(8,3) Halt.

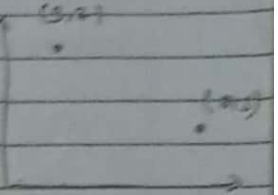
Graph

Example 2

Digitize the line with end points (3,7) and (8,3) with DDA.

Solution:

Let's draw given line from left to right.



Now,

Starting point $(x_1, y_1) = (3,8)$

Ending point $(x_2, y_2) = (8,3)$

$$\text{Slope } (m) = \frac{\Delta y}{\Delta x} = \frac{3-7}{8-3} = \frac{-4}{5} = -\frac{4}{5}$$

Here, slope is less than 1

$\therefore x_{k+1} = x_k + 1$

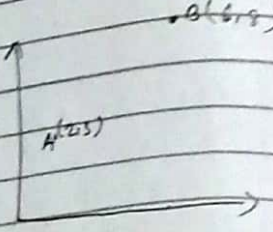
$y_{k+1} = y_k + m$

S.N	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
1.	3	7	(3,7)
2.	3+1=4	7-4/5=6.2≈6	(4,6)
3.	4+1=5	6.2-4/5=5.4≈5	(5,5)
4.	5+1=6	5.4-4/5=4.6≈5	(6,5)
5.	6+1=7	4.6-4/5=3.8≈4	(7,4)
6.	7+1=8	3.8-4/5=3	(8,3) Halt

Graph

Q. Digitize a line with end point A(2,3) and B(6,8), using DDA.

Solution:



Let's draw given line from left to right.

Starting point $(x_1, y_1) = (2,3)$

Ending point $(x_2, y_2) = (6,8)$

Now,

$$\text{Slope, } m = \frac{8-3}{6-2} = \frac{5}{4} = 1.25$$

Here, slope is greater than 1.

So,

$$x_{k+1} = x_k + 1/m$$

$$y_{k+1} = y_k + 1$$

S.N.	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	
1.	2	3	(2,3)	
~~2.~~	~~2+1/1.25=3.8≈4~~			
2.	$2+1/1.25 = 2.8 \approx 3$	$3+1=4$	(3,4)	
3.	$2.8+1/1.25 = 3.6 \approx 4$	$4+1=5$	(4,5)	
4.	$3.6+1/1.25 = 4.4 \approx 4$	$5+1=6$	(4,6)	
5.	$4.4+0.8 = 5.2 \approx 5$	$6+1=7$	(5,7)	
6.	$5.2+0.8 = 6$	$7+1=8$	(6,8)	Halt

Advantages of DDA:

i) Simple to understand.

ii) Requires no special skills for implementation.

iii) Faster than direct method.

Bresenham's line algorithm:

→ It is a more efficient method used to plot pixel position along a straight-line path.

⇒ Advantages of BLA over DDA.

i) In DDA each successive point is computed in floating point, so it requires more time and more space. While

ii) In BLA each successive point is calculated in integer value or whole number. So, it requires less time and less memory space.

ii) In DDA, due to the floating values it needed to be rounded but it doesn't need round off in BLA. So, there is no rounding error.

iii) Due to the rounding error, the line drawn by DDA algorithm is not accurate, while in BLA line is accurate

iv) DDA algorithm cannot be used in other application except line drawing, but BLA can be implemented in other application such as circle, ellipse and other curves.

Algorithm: (BLA for +ve slope & $m<=1$)

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer i.e. plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as

$$P_0 = 2\Delta y - \Delta x$$

4. At each x_k, along the line, starting at $k=0$, perform the following tests:

If $P_k < 0$, then next point to plot is (x_{k+1}, y_k) and

$$P_{k+1} = P_k + 2\Delta y$$

otherwise, the next point to plot is (x_{k+1}, y_{k+1}) and

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 Δx times.

[Note: For $m>1$, just interchange the role of x & y]

Q. Digitize the line with endpoints (20,10) and (30,18) using BLA.

Solution

Here,

Let (20,10) be (x_1, y_1)

(30,18) be (x_2, y_2)

Now,

$$\text{Slope } (m) = \frac{\Delta y}{\Delta x} = \frac{18-10}{30-20} = \frac{8}{10} = \frac{4}{5}$$

Since, slope is +ve and $m \leq 1$, we have:

$$P_0 = 2\Delta y - \Delta x$$
$$= 2 \times 8 - 10$$
$$= 16 - 10 = 6 \quad \text{i.e. } P_0 \geq 0$$

Since, for BLA $|m| \leq 1$, we have,

If $P_k < 0$ [i.e. $d_1 - d_2$ is negative) then,

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k$$

$\therefore P_k = P_k + 2\Delta y$

If $P_k > 0$, then,

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + 1$$

$$P_k = P_k + 2\Delta y - 2\Delta x$$

		20	10	
K	P_{K+1}	x_{K+1}	y_{K+1}	(x_{K+1}, y_{K+1})
0	6	20+1=21	10+1=11	(21, 11)
1	6+2×8-2×10=2	21+1=22	11+1=12	(22, 12)
2	2+2×8-2×10=-2	22+1=23	12	(23, 12)
3	-2+2×8-2×10=14	23+1=24	12+1=13	(24, 13)
4	14+2×8-2×10=10	24+1=25	13+1=14	(25, 14)
5	10+2×8-2×10=6	25+1=26	14+1=15	(26, 15)
6	6+2×8-2×10=2	26+1=27	15+1=16	(27, 16)
7	2+2×8-2×10=-2	27+1=28	16	(28, 16)
8	-2+2×8=14	28+1=29	16+1=17	(29, 17)
9	14+2×8-2×10=10	29+1=30	17+1=18	(30, 18) Halt

Make graph too

Q. Digitize the line with the end points (20, 5) and (25, 10) using BLA

Solution

Here,

Let (20, 5) be (x_1, y_1)

(25, 10) be (x_2, y_2)

Now,

Slope (m) = $\frac{\Delta y}{\Delta x} = \frac{10-5}{25-20} = \frac{5}{5} = 1$

Since, slope is +ve and $m \leq 2$, we have

$P_0 = 2\Delta y - \Delta x$

$= 2 \times 5 - 5 = 10 - 5 = 5$ i.e. $P_0 > 0$

So, solving in the following table

		20	5	
K	P_{K+1}	x_{K+1}	y_{K+1}	(x_{K+1}, y_{K+1})
0	5	20+1=21	5+1=6	(21, 6)
1	5+2×5-2×5=5	21+1=22	6+1=7	(22, 7)
2	5+2×5-2×5=5	22+1=23	7+1=8	(23, 8)
3	5+2×5-2×5=5	23+1=24	8+1=9	(24, 9)
4	5+2×5-2×5=5	24+1=25	9+1=10	(25, 10) Halt

Graph.

Note:

Slope +ve and $|M| > 1$: $(P_0 = 2\Delta x - \Delta y)$

If $P_k < 0$	If $P \geq 0$
$x_{k+1} = x_k$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k + 1$	$y_{k+1} = y_k + 1$
$P_k = P_k + 2\Delta x$	$P_k = P_k + 2\Delta x - 2\Delta y$

Slope -ve

a) $|M| \leq 1$; $(P_0 = 2\Delta y - \Delta x)$

If $P_k < 0$	If $P_k \geq 0$
$x_{k+1} = x_k - 1$	$x_{k+1} = x_k - 1$
$y_{k+1} = y_k$	$y_{k+1} = y_k + 1$
$P_k = P_k + 2\Delta y$	$P_k = P_k + 2\Delta y - 2\Delta x$

b) $|M| > 1$ $(P_0 = 2\Delta x - \Delta y)$

If $P_k < 0$	If $P_k \geq 0$
$x_{k+1} = x_k$	$x_{k+1} = x_k - 1$
$y_{k+1} = y_k + 1$	$y_{k+1} = y_k + 1$
$P_k = P_k + 2\Delta x$	$P_k = P_k + 2\Delta x - 2\Delta y$

Q. Digitize the line with end points (1,0) and (3,3) using BLA.

Solution

Here,

Let (1,0) be (x_1, y_1)

(3,3) be (x_2, y_2)

Now,

$$\text{slope}(m) = \frac{\Delta y}{\Delta x} = \frac{3-0}{3-1} = \frac{3}{2}$$

Since the slope is +ve and $m > 1$. So, we have

$$P_0 = 2\Delta x - \Delta y$$
$$= 2 \times 2 - 3 = 4 - 3 = 1 \text{ i.e. } P \geq 0$$

Now, Solving in the following table:

K	P_{k+1}	x_{k+1} (1 + 1)	y_{k+1} (1 + 3)	(x_{k+1}, y_{k+1})
0	1	1+1 = 2	0+1 = 1	(2, 1)
1	$1 + 2 \times 2 - 2 \times 3 = -1$	2	1+1 = 2	(2, 2)
2.	$-1 + 2 \times 2 = 3$	2+1 = 3	2+1 = 3	(3, 3)

Hall

Q. Digitize the given line with endpoints (5,10) and (10,7) using BLA

Solution

Here,

Let (5,10) be (x_1, y_1)

(10,7) be (x_2, y_2)

Now,

$$\text{Slope } (m) = \frac{7-10}{10-5} = \frac{-3}{5}$$

Since slope is -ve and $|m| \leq 1$. So,

$$P_0 = 2\Delta y - \Delta x$$

$$= 2 \times -3 - 5 = +6 - 5 = 1$$

Since P_0 is +ve, we have the following Cases,

$P_k \leq 0$ $\qquad$ $P_k > 0$

$x_{k+1} = x_k + 1$ $\qquad$ $x_{k+1} = x_k + 1$

$y_{k+1} = y_k$ $\qquad$ $y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2\Delta y$ $\qquad$ $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

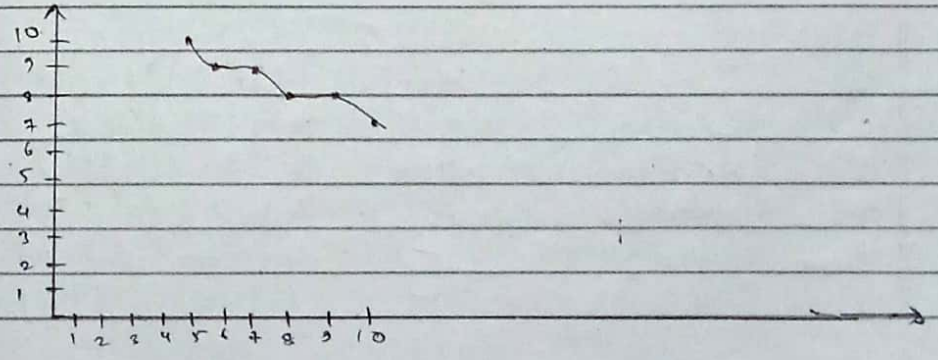
Now,

Calculation Table:

5 10

K	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	1	5+1 = 6	10-1 = 9	(6,9)
1	1+2×3-2×5 = -3	6+1 = 7	9-1 = 8	(7,8)
2	-3+2×3 = 3			

K	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	1	5+1 = 6	10-1 = 9	(6,9)
1	1+2×3-2×5 = -3	6+1 = 7	9	(7,9)
2	-3+2×3 = 3	7+1 = 8	9-1 = 8	(8,8)
3	3+2×3-2×5 = -1	8+1 = 9	8	(9,8)
4	-1+2×3 = 5	9+1 = 10	8-1 = 7	(10,7)
				Halt



Q (5,6) and (6,3)

Q. Digitize the line with end points (3,10) and (6,2) using BLA

Soln

Let (3,10) ⇒ (x_1, y_1)

(6,2) ⇒ (x_2, y_2)

Then,

Slope (m) = $\frac{2-10}{6-3} = \frac{-8}{3}$

Slope is -ve and ~~|m|<1~~ |m| ≥ 1

Now,

$P_0 = 2\Delta x - \Delta y$

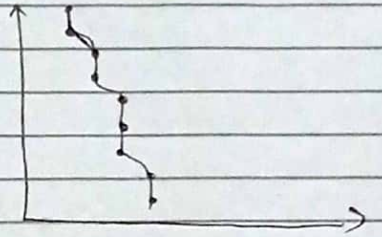
$= 6 - 8 = -2$

For

$P_k < 0$	$P_k \geq 0$
$x_{k+1} = x_k$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$	$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2\Delta x$	$P_{k+1} = P_k + 2\Delta x - 2\Delta y$

k	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	-2	3	10-1=9	(3,9)
1	-2+2×3 = 4	4	9-1=8	(4,8)
2	4+2×3-2×8 = -6	4	8-1=7	(4,7)
3	-6+2×3=0	5	7-1=6	(5,6)
4	-10	5	6-1=5	(5,5)

5	-4	5	4	(5,4)
6	2	6	3	(6,3)
~~7~~	-8	6	2	(6,2) Halt



Q ~~Digi~~ (5,6) and (6,3)

Solution,

Let (5,6) ⇒ (x_1, y_1)

(6,3) ⇒ (x_2, y_2)

Then,

Slope (m) = ~~5-6~~ $\frac{3-6}{6-5} = \frac{-3}{1} = -3$

Slope is -ve and |m| ≥ 1

Now,

$P_0 = 2\Delta x - \Delta y$

$= 2 \times 1 - 3 = 2 - 3 = -1$

For, $P_k < 0$

$x_{k+1} = x_k$

$y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2\Delta x$

$P_k \geq 0$

$x_{k+1} = x_k + 1$

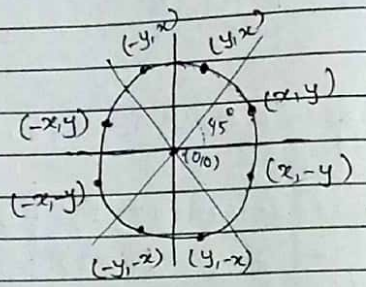
$y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2\Delta x - 2\Delta y$

k	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	-1	5	$6-1=5$	(5, 5)
1	$-1+2\times1=-1+2=1$	$5+1=6$	$5-1=4$	(6, 4)
2	$1+2\times1-2\times3=-3$	6	$4-1=3$	(6, 3) Halt

Circle Algorithm:

→ There is an incremental algorithm for drawing circles - the mid-point circle algorithm.



→ In the mid-point circle algorithm we use eight-way symmetry, so only need to calculate the points for top right eight of the circle, then use symmetry to get rest of the points.

Algorithm:

1. Input radius 'r' and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on origin as:

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$P_0 = \frac{5}{4} - r \approx 1 - r$$

3. At each x_k position, starting at $k=0$, perform the following test:

If $P_k < 0$, the next point of the circle is (x_k+1, y_k) and $P_{k+1} = P_k + 2x_{k+1} + 1$

Otherwise, the next point on circle is (x_k+1, y_k-1) and $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

4. Determine the symmetry point in other seven octants.

5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the co-ordinates value

$$x = x + x_c \qquad y = y + y_c$$

6. Repeat step 3 through 5 until $x \geq y$.

Q. Digitize the circle with radius 10 and centre at origin (0,0)

Soln

Here,

radius $(r) = 10$

centre $(x_c, y_c) = (0, 0)$

Starting point of circle $= (0, r) = (0, 10)$

Now,

Initial decision parameter $(P_0) = 1 - r$
$= 1 - 10$
$= -9$

We have,

$P_k < 0$	$P_k \geq 0$
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$	$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2x_{k+1} + 1$	$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$

S.no.	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1}) at (0,0)
0	-9	$0+1=1$	10	(1,10)
1	$-9+2\times1+1=-6$	$1+1=2$	10	(2,10)
2	$-6+2\times2+1=-1$	$2+1=3$	10	(3,10)
3.	$-1+2\times3+1=6$	$3+1=4$	$10-1=9$	(4,9)
4.	$6+2\times4-2\times9+1=-3$	$4+1=5$	9	(5,9)
5.	$-3+2\times5+1=8$	$5+1=6$	$9-1=8$	(6,8)
6.	$8+2\times6-2\times8+1=5$	$6+1=7$	$8-1=7$	(7,7) Halt

Remaining points of other octants can be determined using symmetry.

Make graph too

Q. Digitize the circle with radius $r=5$ centered (2,3)

Solution

Here,

radius $(r) = 5$

centre $(x_c, y_c) = (2,3)$

Starting point of circle $= (0, r) = (0,5)$

Now,

Initial decision parameter $(P_0) = 1 - r$
$= 1 - 5$
$= -4$

We have

$P_k < 0$	$P_k \geq 0$
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$	$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2x_{k+1} + 1$	$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1}$

S.no.	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1}) at (0,0)	(x_{k+1}, y_{k+1}) at (2,3)
0	-4	$0+1=1$	5	(1,5)	(3,8)
1	$-4+2\times1+1=-1$	$1+1=2$	5	(2,5)	(4,8)
2.	$-1+2\times2+1=4$	$2+1=3$	$5-1=4$	(3,4)	(5,7)
3.	$4+2\times3+1=11$	$3+1=4$	$4-1=3$	(4,3) Halt	(6,6)

Remaining points of other octants can be determined using symmetry.

Q. Digitize a circle with radius 9 and center at (6, 7)

Solution

Here,

radius (r) = 9

centre $(x_c, y_c) = (6, 7)$

Starting point of the circle = $(0, r)$

$= (0, 9)$

Now,

Initial decision parameter, $P_0 = 1 - r$

$= 1 - 9$

$= -8$

we have,

$P_k < 0$	$P_k \geq 0$
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$	$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2x_{k+1} + 1$	$P_k = P_k + 2x_{k+1} - 2y_{k+1} + 1$

Then,

S.NO.	P_{k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1}) at (0,0)	(x_{k+1}, y_{k+1}) at (6,7)
0	-8	$0+1=1$	9	(1, 9)	(7, 16)
1	$-8+2\times1+1=-5$	$1+1=2$	9	(2, 9)	(8, 16)
2.	$-5+2\times2+1=0$	$2+1=3$	$9-1=8$	(3, 8)	(9, 15)
3.	$0+2\times3-2\times8+1=-9$	$3+1=4$	8	(4, 8)	(10, 15)
4.	$-9+2\times4+1=0$	$4+1=5$	$8-1=7$	(5, 7)	(11, 14)
5.	$0+2\times5-2\times7+1=-3$	$5+1=6$	7	(6, 7)	(12, 14)
6.	$-3+2\times6+1=10$	$6+1=7$	$7-1=6$	(7, 6)	(13, 13)

plot another graph

Ellipse:

→ Midpoint Ellipse Algorithm:

The ellipse has 4 way symmetry. The midpoint ellipse method is applied throughout the first quadrant in two region as show in figure. The region-1 just behaves as the circular property and the region 2 is slightly straight curve.

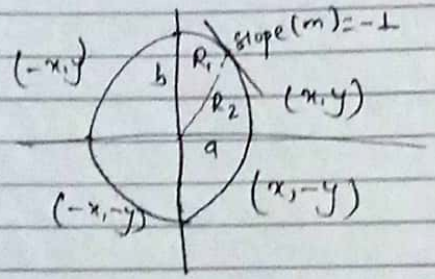


Fig: Ellipse with 4-symmetry & two region in each quadrant.

Algorithm:

1. Input r_x, r_y and center (x_c, y_c) and obtain the first point on ellipse centred at origin as $(x_0, y_0) = (0, r_y)$
2. Calculate the initial value of decision parameter in region 1 as

$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At each step x_k position in region 1, starting at $k=0$, perform the following test:

If $P1_k < 0$, the next point on ellipse centred at (0,0) is (x_{k+1}, y_k) and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$$

Otherwise, the next point along the ellipse is (x_k+1, y_k-1) and,

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with,

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \quad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

And continue until $2r_y^2 x \geq 2r_x^2 y$

4. Calculate the initial value of the decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as $(a = r_x, b = r_y)$

$$P2_0 = b^2 (x_0 + 1/2)^2 + a^2 (y_0 - 1)^2 - a^2 b^2$$

5. At each step y_k in region 2, starting at $k=0$, perform the following test:

↳

If $p2_k > 0$, the next point along the ellipse centered on (0,0) is $(x_k, y_k - 1)$ and,

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$

otherwise,

the next point is $(x_k + 1, y_k - 1)$ and

$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

6. Determine symmetry points in the other three quadrants

7. Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the co-ordinate values:

$$x = x + x_c, \quad y = y + y_c$$

8. Repeat the steps for region 1 until $2r_y^2 x \geq 2r_x^2 y$ and region 2 until $(x_f, 0)$.

Q. Digitize the ellipse with $r_x = 8$, $r_y = 6$ and centre at (0,0)

Solution

Here,

Starting point $(0, r_y) = (0, 6)$

Ending point $(r_x, 0) = (8, 0)$

Centre $(x_c, y_c) = (3, 5)$

Also, $r_x = 8$, $r_y = 6$

For region 1

Initial decision parameter, $(P1_0) = b^2 - a^2 b + 1/4 a^2$

$= 6^2 - 8^2 \times 6 + 1/4 \times 8^2$

$= -332$

we have,

$P_{1k} < 0$	$P_{1k} \geq 0$
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$	$y_{k+1} = y_k - 1$
$P_{1k+1} = P_{1k} + 2b^2 x_{k+1} + b^2$	$P_{1k+1} = P_{1k} + 2b^2 x_{k+1} - 2a^2 y_{k+1} + b^2$

S.NO	P_{1k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	$2b^2 x \geq 2a^2 y$
0	-332	0+1=1	6	(1,6)	False
1	$-332 + 2 \cdot 6^2 \cdot 1 + 6^2 = -224$	1+1=2	6	(2,6)	False
2	$-224 + 2 \cdot 6^2 \cdot 2 + 6^2 = -44$	2+1=3	6	(3,6)	

S.NO	P_{1k+1}	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	$2b^2 x \geq 2a^2 y$
0	-332	0+1=1	6	(1,6)	False
1	$-332 + 2 \cdot 6^2 \cdot 1 + 6^2 = -224$	1+1=2	6	(2,6)	False
2	$-224 + 2 \cdot 6^2 \cdot 2 + 6^2 = -44$	2+1=3	6	(3,6)	False
3	$-44 + 2 \cdot 6^2 \cdot 3 + 6^2 = 208$	3+1=4	6-1=5	(4,5)	False
4	$208 + 2 \cdot 6^2 \cdot 4 + 6^2 - 2 \cdot 8^2 \cdot 5 = -108$	4+1=5	5	(5,5)	False
5	$-108 + 2 \cdot 6^2 \cdot 5 + 6^2 = 288$	5+1=6	5-1=4	(6,4)	False
6	$288 + 2 \cdot 6^2 \cdot 6 - 2 \cdot 8^2 \cdot 4 + 6^2 = 244$	6+1=7	4-1=3	(7,3)	True

For region 2:

Initial point for region 2 $(x_0, y_0) = (7, 3)$

The initial decision parameter is,

$$P2_0 = b^2\left(x_0 + \frac{1}{2}\right)^2 + a^2(y_0 - 1)^2 - a^2b^2$$

$$= 6^2 \times \left(7 + \frac{1}{2}\right)^2 + 8^2(3-1)^2 - 8^2 \times 6^2$$

$$= -23$$

Now,
We know that

If $P2_k > 0$ then,

$P2_{k+1} = P2_k - 2a^2 y_{k+1} + a^2$

$x_{k+1} = x_k$

$y_{k+1} = y_k - 1$

If $P2_k \leq 0$, then,

$P2_{k+1} = P2_k + 2b^2 x_{k+1} - 2a^2 y_{k+1} + a^2$

$x_{k+1} = x_k + 1$

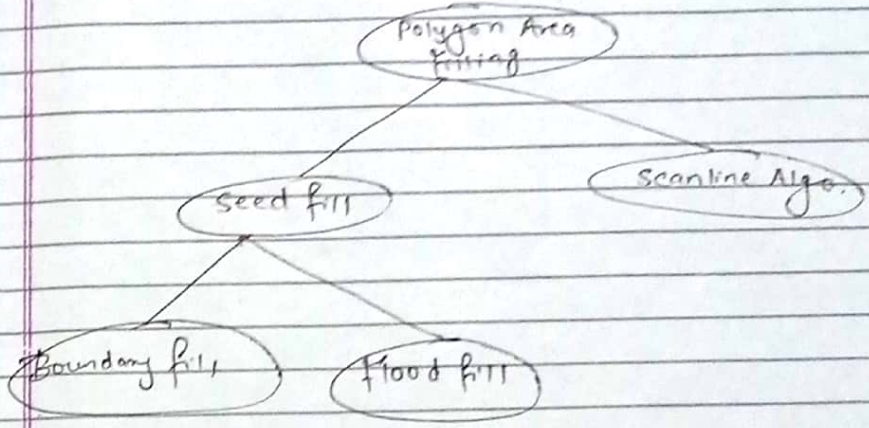
$y_{k+1} = y_k - 1$

Then,

k	$P2_k$	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	-23	7+1=8	3-1=2	(8,2)
1	361	8	1	(8,1)
2	777	8	0	(8,0) Halt

∴ Remaining points can be obtained from the symmet

Polygon Area Filling:



Q. Draw an ellipse where x-radius is 6 and y-radius is at center (3,5).

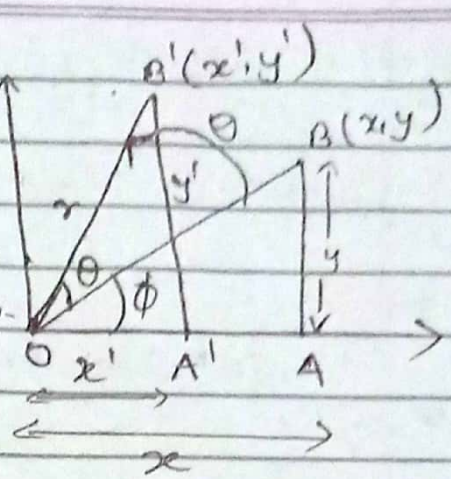
Chapter: 3

2D Geometric Transformation

→ The orientation, size and shape of the output primitives are obtained with geometric transformations that after the co-ordinate description of objects.

→ The basic geometric transformations are translation, rotation and scaling.

Rotation: (Anti-clockwise derivation)



Let (x,y) is the original point, 'r' the constant distance from origin, & 'ϕ' the original angular displacement from x-axis.

Now, the point (x,y) is rotated through angle 'θ' in a counter clockwise direction.

In $\triangle OA'B'$

$$\cos(\theta+\phi) = b/h = x'/r$$

$$\therefore x' = r\cos(\theta+\phi) = r\cos\phi\cdot\cos\theta - r\sin\phi\cdot\sin\theta \quad \text{—(i)}$$

$$\sin(\theta+\phi) = p/h = y'/r$$

$$\therefore y' = r\sin(\theta+\phi) = r\cos\phi\cdot\sin\theta + r\sin\phi\cdot\cos\theta \quad \text{—(ii)}$$

In $\triangle OAB$

In $\triangle OAB$

$$\cos\phi = b/h = x/r$$

$$\therefore x = r\cos\phi$$

$\sin\phi = P/h = y/r$

$\therefore y = r\sin\phi$

Substituting the values in equ^n (i) & (ii), we get

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

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

Representing in matrix form:

$$\therefore \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$[x'\ y'] = [x\ y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

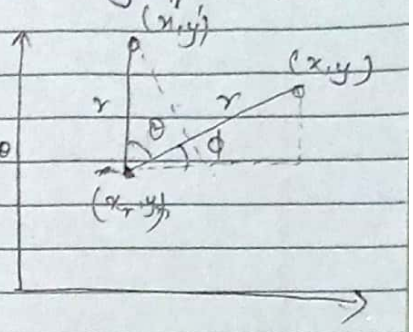
Similarly,

For clockwise:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$[x'\ y'] = [x\ y] \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

⇒ Rotation of a point about an arbitary pivot position can be seen in the figure.



Here

$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$

$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1-\cos\theta & -\sin\theta \\ -\sin\theta & 1-\cos\theta \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

Note: This can also be achieved by translating the arbitary point into the origin and then apply the rotation and finally perform the reverse translation.

Example:

Rotate a line AB whose endpoints are (3,4) and (12,15) about origin through a 45° anticlockwise direction.

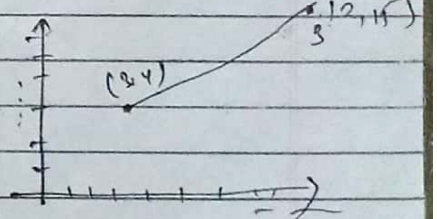
Soln

Here

Let A = (3,4)

B = (12,15)

θ = 45°



Now,

For A:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 3/\sqrt{2} - 4/\sqrt{2} \\ 3/\sqrt{2} + 4/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} -1/\sqrt{2} \\ 7/\sqrt{2} \end{bmatrix}$$

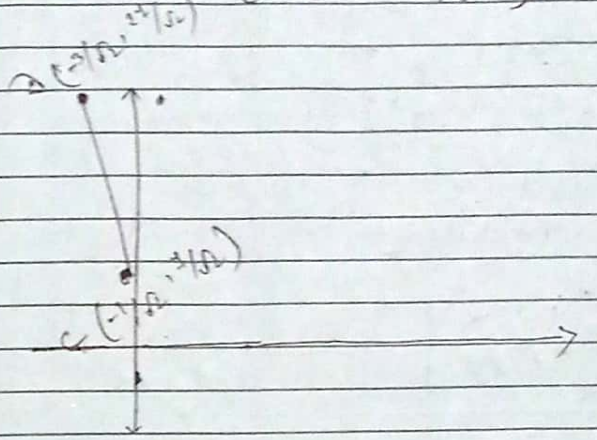
For B:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} \begin{bmatrix} 12 \\ 15 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 12 \\ 15 \end{bmatrix}$$

$$= \begin{bmatrix} 12/\sqrt{2} - 15/\sqrt{2} \\ 12/\sqrt{2} + 15/\sqrt{2} \end{bmatrix} = \begin{bmatrix} -3/\sqrt{2} \\ 27/\sqrt{2} \end{bmatrix}$$



Example:

Rotate line AB whose endpoints are A(2,5) and B(6,12) about origin through a 30° clockwise direction.

Solution

Here

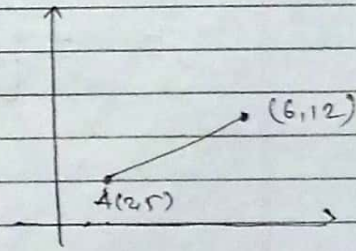
Given:

A = (2,5)

B = (6,12)

So,

For A: (Clockwise)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= \begin{bmatrix} \cos 30^\circ & \sin 30^\circ \\ -\sin 30^\circ & \cos 30^\circ \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 \times 2 + 1/2 \times 5 \\ -1/2 \times 2 + \sqrt{3}/2 \times 5 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3} + 5/2 \\ -1 + 5\sqrt{3}/2 \end{bmatrix}$$

$$= \begin{bmatrix} 4.232 \\ 3.330 \end{bmatrix}$$

For B (Clockwise)

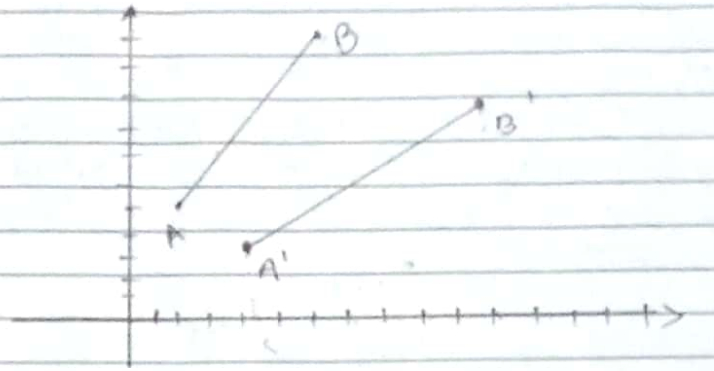
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$= \begin{bmatrix} \cos 30^\circ & \sin 30^\circ \\ -\sin 30^\circ & \cos 30^\circ \end{bmatrix} \begin{bmatrix} 6 \\ 12 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 6 \\ 12 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 \times 6 + 1/2 \times 12 \\ -1/2 \times 6 + \sqrt{3}/2 \times 12 \end{bmatrix}$$

$$= \begin{bmatrix} 3\sqrt{3} + 6 \\ -3 + 6\sqrt{3} \end{bmatrix} = \begin{bmatrix} 11.196 \\ 7.392 \end{bmatrix}$$



Scaling:

→ A scaling transformation alters the size of an object

→ This can be carried out for polygons by multiplying the co-ordinate values (x, y) of each vertex by scaling factors S_x and S_y to produce the transformed co-ordinates (x', y').

- S_x scales object in 'x' direction
- S_y scales object in 'y' direction.

Thus, for equ form,

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Values greater than 1 for S_x S_y produce enlargement
- Values less than 1 for S_x S_y reduce size of objects
- $S_x = S_y = 1$, leaves the size of the object unchanged
- When S_x, S_y are assigned the same value $S_x = S_y$ = etc, then a Uniform Scaling is produced.

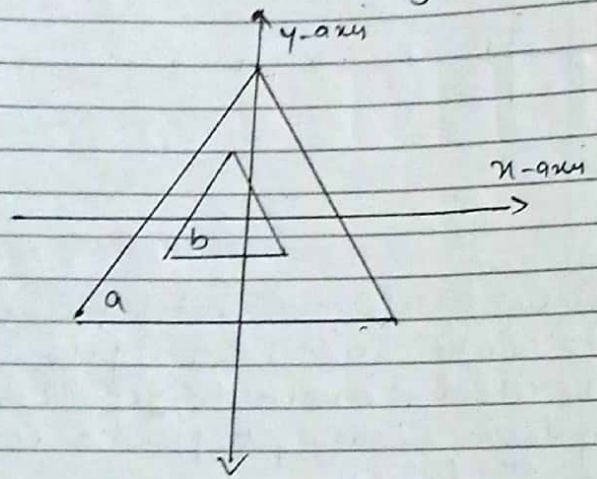


Fig:- Turning a triangle (a) into a triangle (b) with scaling factors $S_x = 1/2$ and $S_y = 1/2$

Other transformations:

→ Basic transformations such as translation, rotation, and scaling are included in most graphics package.

→ Additional transformations are reflection and shearing.

Shearing:

→ It is a non-rigid body transformation that moves objects with deformation.

→ It distorts the shape of object in either 'x' or 'y' or both direction.

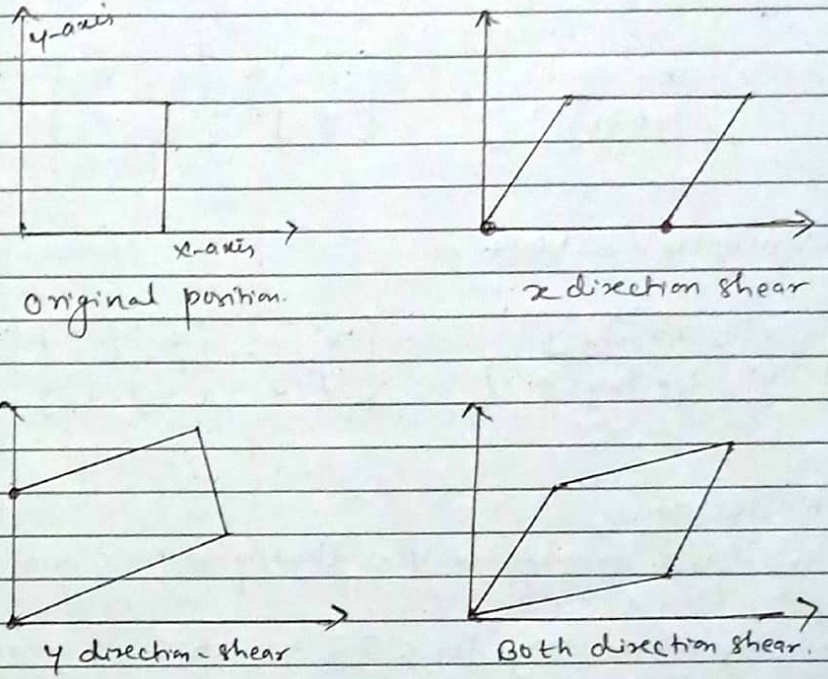


Fig: Two dimensional shearing

In 'x' direction,

$$\left\{ \begin{array}{l} x' = x + s_{hx} \cdot y \\ y' = y \end{array} \right\} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s_{hx} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

In 'y' direction,

$$\left\{ \begin{array}{l} x' = x \\ y' = y + s_{hy} \cdot x \end{array} \right\} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s_{hy} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

In both directions,

$$\left\{ \begin{array}{l} x' = x + s_{hx} \cdot y \\ y' = y + s_{hy} \cdot x \end{array} \right\} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s_{hx} \\ s_{hy} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reflection:

→ It is a transformation that produces a mirror image of an object.

→ The mirror image for a 2D reflection is generated relative to an axis of reflection by rotating the object 180° about the reflection axis.

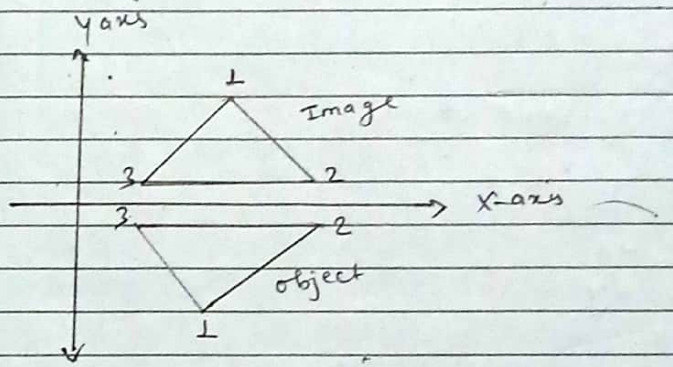
→ We can choose an axis of reflection in the xy-plane or perpendicular to the xy plane.

(i) Reflection about x axis or about line $y=0$

→ Keeps 'x' value same but flips y value of co-ordinate points. So,

$$x' = x$$
$$y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



(ii) Reflection about y-axis or about line $x=0$

→ Keeps 'y' value same but flips x value of co-ordinate points. So,

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(iii) Reflection about origin,

$x' = -x$

$y' = -y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(iv) Reflection about line $y = x$:

Steps:

(i) Clockwise rotation by 45°
(ii) Reflection against x-axis
(iii) Rotate counter-clockwise by 45°.

$$R_f \quad C.W.R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reflection against x-axis, $Rf_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

$$CCW.R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

• Overally / Alternative

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(iv) Reflection about line $y = -x$

Homogeneous Co-ordinates:

→ The matrix representation for translation, scaling and rotation are respectively.

(i) Translation : $P' = T + P$ (Add)

(ii) Scaling : $P' = S \cdot P$ (Multiplication)

(iii) Rotation : $P' = R \cdot P$ (")

for translation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

for rotation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

CCW

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

CW

for Scaling:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

for Shearing:

In x-direction

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In y-direction

$$\begin{bmatrix} 1 & sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In x-y direction

$$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for Reflection:

Reflection about x-axis:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection about y-axis:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection about y=x-axis:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection about y=-x axis:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection about any line y=mx+c:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-(m^2-1)}{(m^2+1)} & \frac{2m}{(m^2+1)} & \frac{-2mc}{(m^2+1)} \\ \frac{2m}{(m^2+1)} & \frac{(m^2-1)}{(m^2+1)} & \frac{2c}{(m^2+1)} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Composite Transformation:

=> With the matrix representation it is possible to setup a matrix for any sequence of transformation as a composite transformation matrix by calculating the matrix product of individual transformation.

=> Also called concatenation, or comporition of mat

=> We take composite transformation by multiplying matrices in order from right to left.

Advantages:

-> It transformations become compact
-> The no. of operations will be reduces.
-> Complexity will be reduced.

✓ Comporite of two translations:

-> Let there are two ~~transformations~~ translations P_1 and P_2 with t_1 t_2 t_3 t_4 as translation vectors.

-> The P_1 and P_2 are represented using Homogeneous matrices and P will be the final transformation matrix obtained after multiplication.

$$P_1 = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 1 & 0 & t_3 \\ 0 & 1 & t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore P = \begin{bmatrix} 1 & 0 & t_1+t_3 \\ 0 & 1 & t_2+t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

Translations are additive and also Rotation is additive.

But, Scaling is multiplicative.

$$S = S_{11} * S_{12} = \begin{bmatrix} S_1 * S_3 & 0 & 0 \\ 0 & S_2 * S_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where

$$S_{11} = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_{12} = \begin{bmatrix} S_3 & 0 & 0 \\ 0 & S_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

✓ Pivot point rotation or rotation about fixed point:

Sequences of steps:

-> Translate pivot point to origin. $[T(-x_f, -y_f)]$
-> Rotate the object about the origin.
-> Translate the object to its original position from origin. It is called as reverse translation $[T(x_f, y_f)]$

Comporite matrix:

$T(x_f, y_f)$ (iii) R_θ (ccw) (ii) $T(-x_f, -y_f)$ (i)

$$= \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

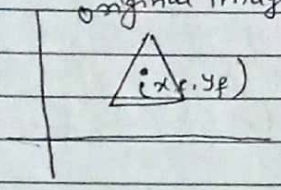
First multiply these

Multiply Here then.

Then,

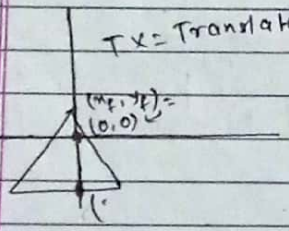
$P' = [\text{Composite matrix}] \times P$

original image



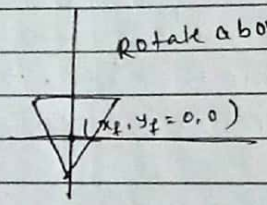
Step I:

T_x = Translate to origin



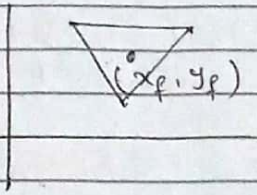
Step II:

Rotate about θ



Step III:

T'_x = Translate to its original place.



→ Scaling relative to fixed point

In place of R_θ, place Scaling matrix

i.e.

Numericals.

1. Rotate the triangle (5,5), (7,3), (3,3) about fixed point (5,4) in counter clockwise (CCW) by 90 degree

Solution:

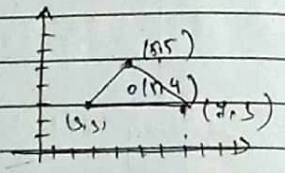
Let,

A(5,5)

B(7,3)

C(3,3) be the points of a triangle

(5,4) be the fixed point



Now,

The composite matrix for the rotation of the points ABC about fixed point (5,4) in CCW by 90 degree is given by

$$\text{Composite matrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0-1+0 & 0+4+0 \\ 1+0+0 & 0+0+0 & -5+0+0 \\ 0+0+0 & 0+0+0 & 0+0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 4 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0+0+0 & -1+0+0 & 4+0+5 \\ 0+1+0 & 0+0+0 & 0-5+4 \\ 0+0+0 & 0+0+0 & 0+0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Now,

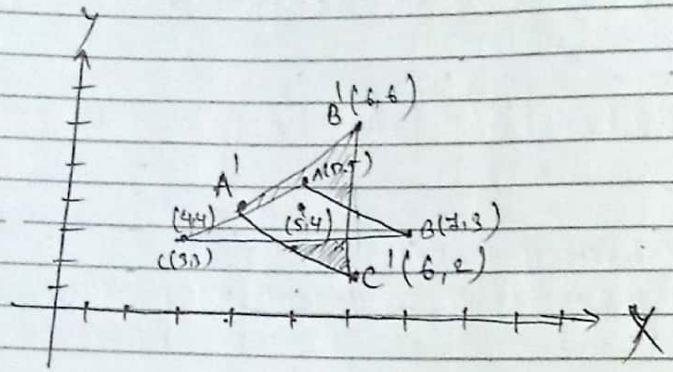
Final image = Composite matrix × object matrix

$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0-5+9 & 0-3+9 & 0-3+9 \\ 5+0-1 & 7+0-1 & 3+0-1 \\ 0+0+1 & 0+0+1 & 0+0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 & 6 \\ 4 & 6 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Hence, the new resulted co-ordinates are (4,4) (6,6) and (6,2)



Q. Rotate a triangle A(5,6), B(6,2) and C(4,1) by 45 degree about an arbitary point (3,3).
in CW

Solution

Composite matrix $= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45^\circ & \sin 45^\circ & 0 \\ -\sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$

$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.707 & 0.701 & -4.242 \\ -0.707 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$= \begin{bmatrix} 0.7071 & 0.7071 & -1.242 \\ -0.707 & 0.7071 & 3 \\ 0 & 0 & 1 \end{bmatrix}$

$\therefore P' = \text{Comp.} \times P$

$= \begin{bmatrix} 0.7071 & 0.7071 & -1.242 \\ -0.707 & 0.7071 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 6 & 4 \\ 6 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$= \begin{bmatrix} 6.5361 & 4.4148 & 2.2935 \\ 3.7076 & 0.1722 & 0.8791 \\ 1 & 1 & 1 \end{bmatrix}$

Therefore the required points are (6.5361, 3.7076), (4.4148, 0.1722) and (2.2935, 0.8791) //

Q.1// Find the co-ordinates of a triangle (5,5), (7,3), (3,3) after translate (1,0) then scaling (3,3) then rotate 45 CW then translate (1,1) and then reflect x=0 line.

Q.2// Reflect an object (2,3), (4,3), (4,5) about line y = x+1.

Q.3// A mirror is placed such that it passes through (0,10) (10,0). Find the mirror image of an object (6,7), (7,6), (6,9)

Q.2 Solution

Here

Let A(2,3)

B(4,3)

C(4,5) be three co-ordinates of an object

To perform:

Reflection about line y = x+1

Composite matrix is given by,

$$T[(0,1)]\, R_f(x=y)\, T[(0,-1)]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

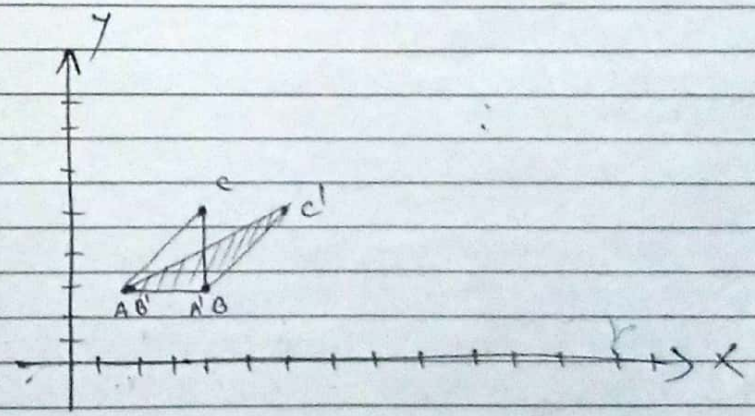
Now, The final image is given by,

P' = Composite matrix x P

$$= \begin{bmatrix} 2 & 4 & 4 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 2 & 6 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

Therefore, the required co-ordinates are (4,3), (2,3), (6,5)



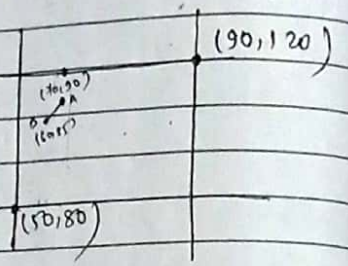
Q. Use Cohen Sutherland line Clipping algorithm to clip a line with end point co-ordinates A(70, 90), B(60, 85) against a clip window with it's lower left corner at (50, 80) and upper right corner at (90, 120).

Solution:

Here,

$(X_{wmin}, Y_{wmin}) = (50, 80)$

$(X_{wmax}, Y_{wmax}) = (90, 120)$



Step 1: Calculating region code of both end-points.

For A (70, 90).

$X - X_{wmin} = 70 - 50 = +ve \rightarrow 0$ (Left)
$X_{wmax} - X = 90 - 70 = +ve \rightarrow 0$ (Right)
$Y - Y_{wmin} = 90 - 80 = +ve \rightarrow 0$ (Bottom)
$Y_{wmax} - Y = 120 - 90 = +ve \rightarrow 0$ (Top)

Hence, the region code = 0000

For B (60, 85)

$X - X_{wmin} = 60 - 50 = +ve \rightarrow 0$ (Left)
$X_{wmax} - X = 90 - 60 = +ve \rightarrow 0$ (Right)
$Y - Y_{wmin} = 85 - 80 = +ve \rightarrow 0$ (Bottom)
$Y_{wmax} - Y = 120 - 85 = +ve \rightarrow 0$ (Top)

Hence, the region code = 0000

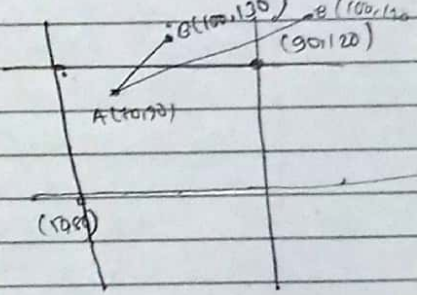
Step 2: ORing the region code obtained:

```
  0 0 0 0
ORing 0 0 0 0
  0 0 0 0
```

Since the output is 0000, the line is completely visible.

Q. Use Cohen Sutherland line Clipping algorithm to clip a line with end point co-ordinates A(70, 90), B(100, 130) against a clip window with its lower left corner at (50, 80) and upper right corner at (90, 120)

Solution,



$(X_{min}, Y_{min}) = (50, 80)$
$(X_{wmax}, Y_{wmax}) = (90, 120)$

So,

Step 1: Calculating region codes.

For A (70, 90): $X - X_{wmin} = 70 - 50 = +ve = 0$ (Left)
$X_{wmax} - X = 90 - 70 = +ve = 0$ (Right)
$Y - Y_{wmin} = 90 - 80 = +ve = 0$ (Bottom)
$Y_{wmax} - Y = 120 - 90 = +ve = 0$ (Top)

Hence, the region code = 0000.

for B(100,130):

$x - x_{wmin} = 100 - 50 = +ve = 0$ (Left)
$x_{wmax} - x = 90 - 100 = -ve = 1$ (Right)
$y - y_{wmin} = 130 - 80 = +ve = 0$ (Bottom)
$y_{wmax} - y = 120 - 130 = -ve = 1$ (Top)

Step II: ORing the region codes:

```
0 0 0 0
0 0 1 0
-------
0 0 1 0
```

```
0 0 0 0
1 0 1 0
-------
1 0 1 0 ≠ 0000
```

Step III: ANDing the region codes:

```
0 0 0 0
1 0 1 0
-------
0 0 0 0
```

Since ANDing results to 0000, this is clipped case.

Step IV: Calculating intersection pnts.

~~Here end result region~~

Here,

In region code of B (i.e. 1010) top bit and right bit are one.

Let us consider top bit:

$y_2 = y_{wmax} = 120$

$x_2 = x_1 + \frac{y_2 - y_1}{m}$

$= 90 + \frac{120 - 90}{4/3}$

$= 80$

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{130 - 90}{100 - 70} = 4/3$

∴ The required intersection point is (80,120).

Let us consider top and right boundary.

$x = x_{wmax} = 90$

So,

$y = y_1 + m(x - x_1)$

$= 90 + 4/3 (90 - 70)$

$= 116.67 = 117$

Hence, the intersection point is (90,117)

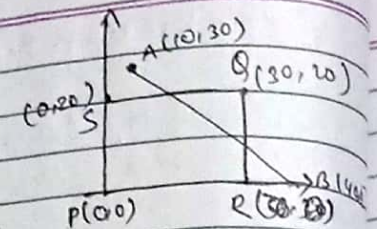
Q. Use CSLC algorithm to clip window P(0,0), Q(30,20) R(30,0) and S(0,20). Determine the visible portion of the line A(10,30) and B(40,0)

Solution

Here

P(0,0) = (x_{wmin}, y_{wmin})

Q(30,20) = (x_{wmax}, y_{wmax})



Now,

Step I: Calculating region codes:

For A(10,30):

$x - x_{wmin}$ = (10-0) = positive → 0 (L)

$x_{wmax} - x$ = 30-10 = +ve → 0 (R)

$y - y_{wmin}$ = (30-0) = +ve → 0 (B)

$y_{wmax} - y$ = (20-30) = -ve → 1 (T)

Hence, the region code = 1000

For B(40,0),

$x - x_{wmin}$ = (40-0) = +ve → 0 (L)

$x_{wmax} - x$ = (30-40) = -ve → 1 (R)

$y - y_{wmin}$ = (0-0) = 0 → 0 (B)

$y_{wmax} - y$ = (20-0) = +ve → 0 (Top)

Hence the region code = 0010

Step II: ORing the region codes:

$$\begin{array}{r} 1000 \\ 0010 \\ \hline 1010 \neq 0000 \end{array}$$

Step III: ANDing the region codes

$$\begin{array}{r} 1000 \\ 0010 \\ \hline 0000 \end{array}$$

Since AND operation results 0000, it is clipped case.

Step IV: To calculate intersection point,

For A(10,30) in the region code (i.e. 1000), top bit is one.

Then

$$\text{Slope}(m) = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0-30}{40-10} = \frac{-30}{30} = -1$$

Now,

$y = y_{wmax} = 20$

$$x = x_1 + \frac{(y - y_1)}{m}$$

$$= 10 + \frac{(20-30)}{-1}$$

$$= 10 + 10$$

$$= 20$$

Hence the intersection point is A'(20, 20)

for B(40,0), in the region code (0010), right bit is one, then

$$x = x_{wmax} = 30$$

$$y = y_1 + m(x - x_1)$$
$$= 0 + (-1)(30-40)$$
$$= 0 + 10 \longrightarrow B' = (30,10)$$

Hence the intersected points are $A'(20,20)$ and $B'(30,10)$. Within these co-ordinates, the line is visible.

Liang-Barsky Algorithm:

→ Line clipping algorithm
→ More efficient than Cohen-Sutherland algorithm and can be extended to 3-dimensional clipping.
→ Faster parametric line-clipping algorithm.

The p and q are defined as:

$p_1 = -(x_2 - x_1)$	$q_1 = x_1 - x_{wmin}$ (left)
$p_2 = (x_2 - x_1)$	$q_2 = x_{wmax} - x_1$ (Right)
$p_3 = -(y_2 - y_1)$	$q_3 = y_1 - y_{wmin}$ (Bottom)
$p_4 = (y_2 - y_1)$	$q_4 = y_{wmax} - y_1$ (Top)

Condition	Position of line
$P_k = 0$	Parallel to the clipping boundaries
$P_k = 0$ && $q_k < 0$	Completely outside the boundary.
$P_k = 0$ and $q_k >= 0$	Inside the parallel clipping boundary
$P_k < 0$	Line proceeds from outside to inside.
$P_k > 0$	Line proceeds from inside to outside.

Parameters t_1 and t_2 can be calculated that define the part of line that lies with the clip rectangle.

When,

(i) $P_k < 0$, t_1 = maximum $(0, q_k/P_k)$ is taken
(ii) $P_k > 0$, t_2 = minimum $(1, q_k/P_k)$ is taken.

If $t_1 > t_2$, the line is completely outside and can be rejected. Otherwise, intersection point (x,y) is:

$$x = x_1 + t_1 * \Delta x$$
$$y = y_1 + t_1 * \Delta y$$

Q. Clip the line A(2,6) and B(12,8) against the clipping window (5,5) and (9,9)

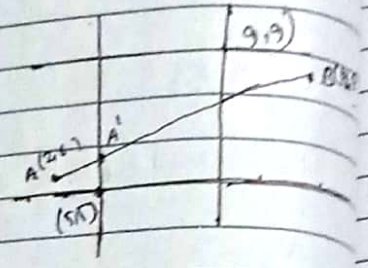
Soln

Let A(2,6) ⇒ (x_1, y_1)

B(12,8) ⇒ (x_2, y_2)

$(x_{wmin}, y_{wmin}) = (5,5)$

$(x_{wmax}, y_{wmax}) = (9,9)$



Then,

$P_1 = -(x_2 - x_1) = -(12-2) = -10$

$P_2 = x_2 - x_1 = 12 - 2 = 10$

$P_3 = -(y_2 - y_1) = -(8-6) = -2$

$P_4 = y_2 - y_1 = (8-6) = 2$

for A:

$q_1 = x_1 - x_{wmin} = 2 - 5 = -3$

$q_2 = x_{wmax} - x_1 = 9 - 2 = 7$

$q_3 = y_1 - y_{wmin} = 6 - 5 = 1$

$q_4 = y_{wmax} - y_1 = 9 - 6 = 3$

Here

$P_k \neq 0$, so line is not parallel to any boundary.

$P_k > 0$ (P_2, P_4)	$P_k < 0$ (P_3, P_1)
$t_2 = \min\left(1, \frac{q_k}{P_k}\right)$	$t_1 = \max\left(0, \frac{q_k}{P_k}\right)$

$t_2 = \min\left(1, \frac{q_2}{P_2}, \frac{q_4}{P_4}\right)$

$= \min\left(1, \frac{7}{10}, \frac{3}{2}\right)$

$\therefore t_2 = 7/10$

$t_1 = \max\left(0, \frac{q_3}{P_3}, \frac{q_4}{P_4}\right)$

$= \max\left(0, \frac{1}{-2}, \frac{3}{2}\right)$

$\therefore t_1 = 3/2$

$t_1 = \max\left(0, \frac{q_1}{P_1}, \frac{q_3}{P_3}\right)$

$= \max\left(0, 3/10, 1/-2\right)$

$= 3/10$

Since $t_2 > t_1$. So, we need to calculate intersection point using,

$x = x_1 + t_1 * \Delta x = 2 + 0.3 \times 10 = 2 + 3 = 5$

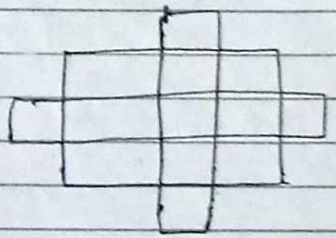
$y = y_1 + t_1 * \Delta y = 6 + 0.3 \times 2 = 6 + 1.4 = 7.4$

$= 6 + 0.6 = 6.6$

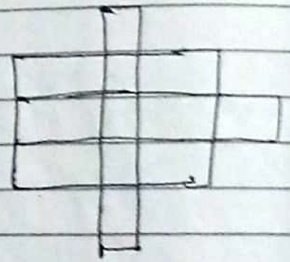
$A'(5, 7.4)$

Polygon Clipping: Sutherland Hodgeman

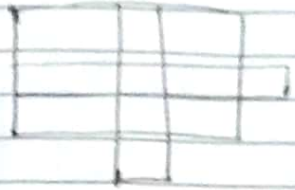
→ The It is used for clipping polygons
→ A single polygon can actually be split into multiple polygons.
→ The algorithm clips a polygon against all edges of the clipping region.
→ The clip region can be any convex polygon in 2D, or any convex polyhedron in 3D.



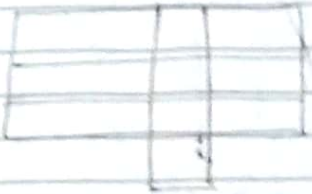
original image.



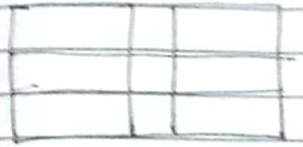
Clipping against left



Clipping against top



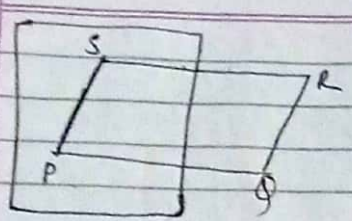
Clipping against right



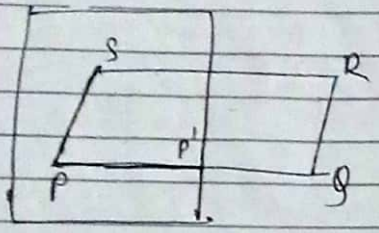
Clipping against bottom.

Rules:

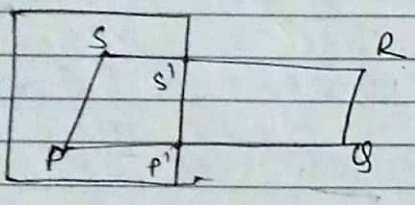
Case	1st vertex	2nd vertex	Output
1	Inside	inside	2nd vertex
2	inside	Outside	Intersection
3	Outside	Outside	None
4	Outside	inside	2nd and Intersection



S and P are inside inside. so, 2nd vertex, P added to the display.



P and Q are inside outside, so intersection point P' added to the display i.e. P'.

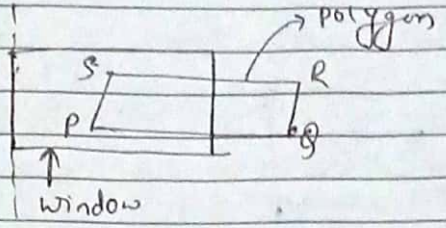


Q and R are outside - outside. so, none are display

and In case of R and S, they are outside - inside so, intersection point and 2nd vertex are displayed i.e. S' and S.

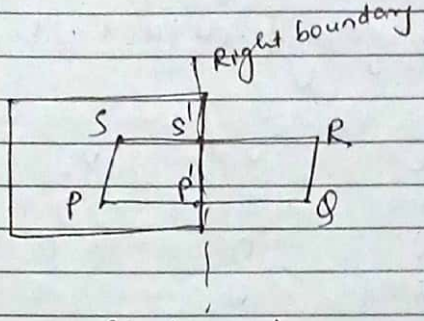
//

example:



(i) Clipping against left boundary

Line	V_1	V_2	Display List
PQ	I	I	Q
QR	I	I	R
RS	I	I	S
SP	I	I	P

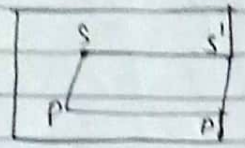


(ii) Clipping against right boundary.

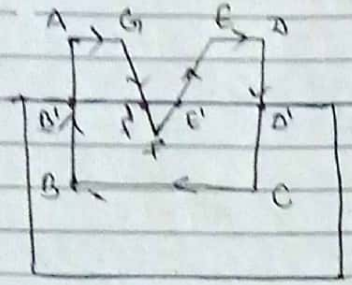
Line	V_1	V_2	Display List
PQ	I	O	P'
QR	O	O	X
RS	O	I	S' S
SP	I	I	P

Similarly the process will be repeated for other vertices

too. The final result will be



Q

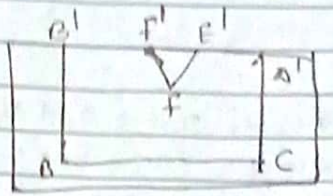


Clipping against the top boundary

Line	V_1	V_2	Display List
BA	I	O	B'
AG	O	O	X
GF	O	I	F'F
FE	I	O	E'
ED	O	O	X
DC	O	I	D'C
CB	I	I	B

Similarly the same process will be repeated for other

boundaries too. The final fig. will be.



→ Disadvantages:

→ Requires a considerable amount of memory.

Unit: 4

3D Graphics System

Why is 3D complex than 2D?

→ More co-ordinates
→ Various combinations of plane and curved surfaces.
→ Light intensities differs from one part to another which is difficult to manage.
→ Visible surface detection is complex
→ Consideration of projection and transparency.

3D geometric transformation:

→ Methods for geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the z-coordinate.

→ Represented by 4x4 matrix

→ Represented as (x, y, z, H)

→ provides Realism.

Translation:

A point is translated from position $P = (x, y, z)$ to position $P' = (x', y', z')$ with the matrix operation as:

$$P' = T \cdot P$$

or

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Parameters t_x, t_y, t_z specify translation distances for the co-ordinate directions x, y and z

equivalent equations

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

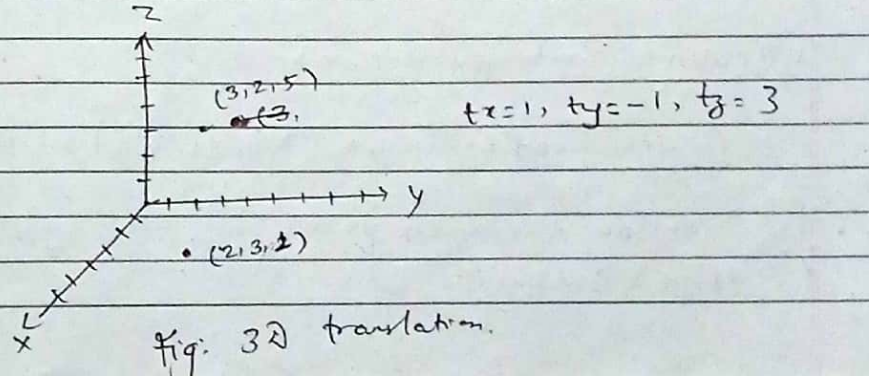


Fig: 3D translation.

Scaling

→ If $s_x = s_y = s_z$; preserves the original shape.
else, the figure get distorted.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

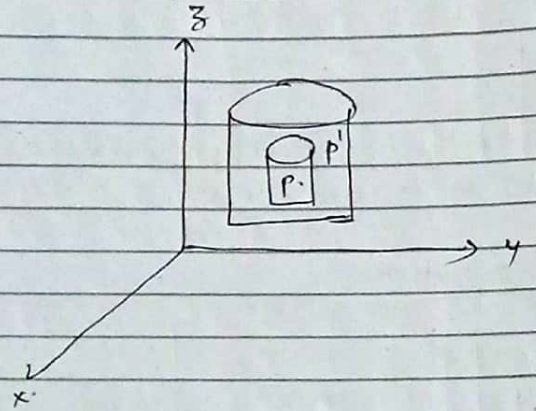


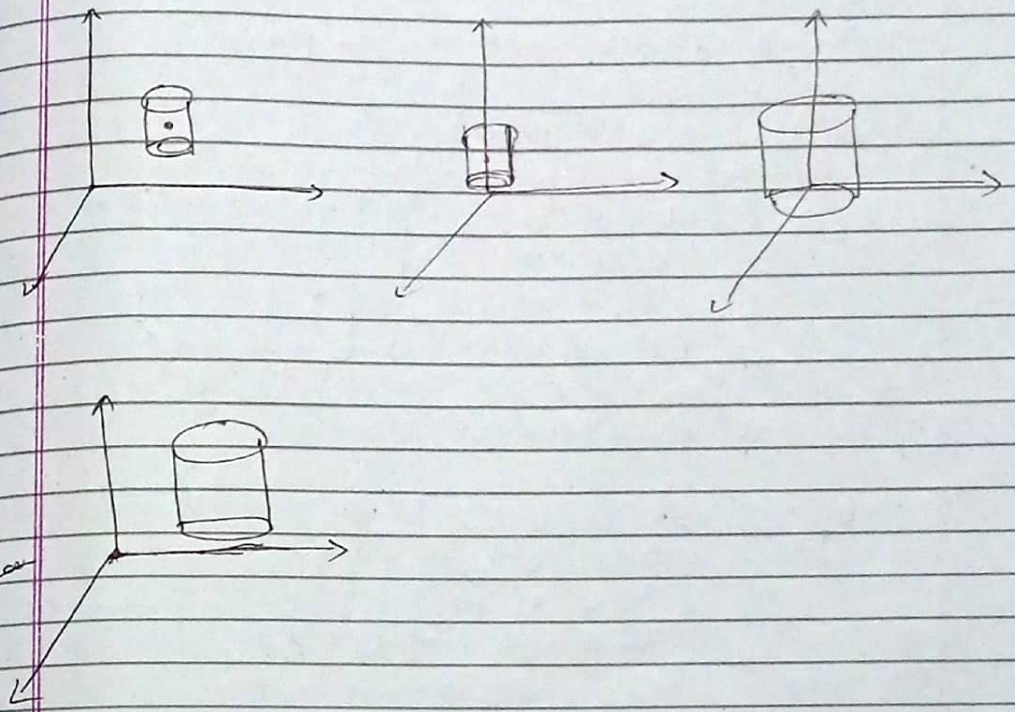
Fig: 3D scaling

#Scaling w.r.t fixed point

Let fixed point be (x_f, y_f, z_f), then

(i) Translate the given points by $(-x_f, -y_f, -z_f)$
(ii) Perform scaling
(iii) Re-translate the points to the fixed point.
ie. Translate by (x_f, y_f, z_f).

Com = $T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f)$



Reflection:

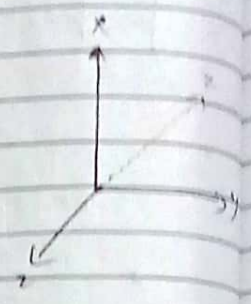
→ It is performed relative to a selected reflection axis or with respect to a selected reflection plane.

→ Set up similarly to those for two dimensions.

→ Can be reflected w.r.t. axis or plane.

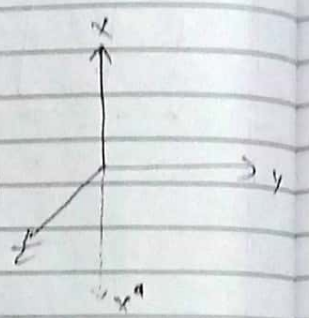
(i) w.r.t. z-axis (xy plane)

$$Rf_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



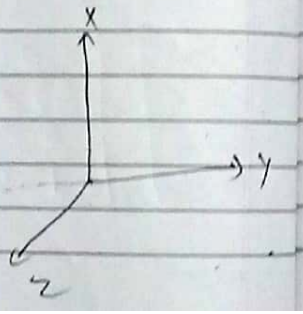
(ii) w.r.t. x-axis (yz plane)

$$Rf_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



(iii) w.r.t. y-axis (xz plane)

$$Rf_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



(iii) w.r.t. a line

(i) Translate to one of the axis, (say x-axis)
(ii) Reflect w.r.t. x-axis
(iii) Re-translate to the original position.

Comp matrix $T(0, y, z)$ Reflect $_{x\text{-axis}}$ $T(0, -y, -z)$

Shearing: (modify object shapes)

$$Sh_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Sh_y = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ b & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Sh_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where a and b are shearing factors.

Rotation:

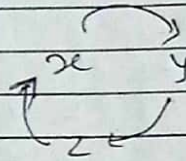
To generate a rotation transformation for an object in 3D space, we require the following.

i) Angle of rotation
ii) Pivot point
iii) Direction of rotation
iv) Axis of rotation.

Axes that are parallel to the co-ordinate axes are easy to handle.

=> Cyclic permutation of the coordinate parameters x, y and z are used to get transformation equations for rotations about the co-ordinates.

$x \to y \to z \to$



(i) 3D x-axis rotation equations are expressed in homogeneous co-ordinate form as:

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$P' = R_x(\theta) \cdot P$

(ii) Y-axis rotation

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(iii) Z-axis rotation

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(iv) Rotation about an axis parallel to one of the coordinate axes (say x)

$$P' = [T^{-1} \cdot R_x(\theta) \cdot T] \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Retranslate to the original point — Rotate — Translate to the origin.

Rotation about any arbitary axis in 3D space

Here

Composite matrix = $T^{-1} R_y^{-1} R_x^{-1} R_z R_y R_x T$

For T:

$$\begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly,

For T^{-1} = $\begin{bmatrix} 1 & 0 & 0 & +t_x \\ 0 & 1 & 0 & +t_y \\ 0 & 0 & 1 & +t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

For R_z = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotation by angle x,

↳

$\sin x = B = B/V$ and

$\cos x = C/V$

Now,

the translation matrix is given by,

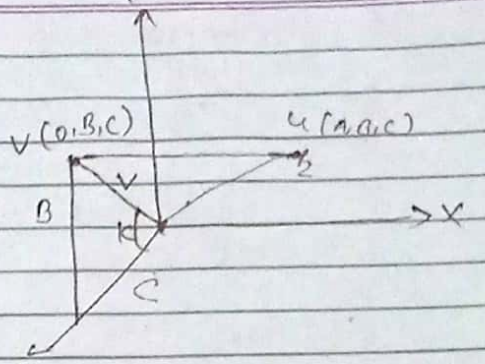


Fig. Identification of rotation angle 'x'

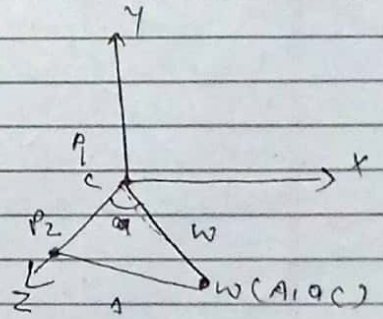
R_x = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & -B/V & 0 \\ 0 & B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Similarly,

R_x' = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & B/V & 0 \\ 0 & -B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

For R_y

Rotation by angle K.



Here

$\sin\alpha = A/W$

$\cos\alpha = C/W$

So,

↳

The translation is given by,

$$R_y = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C/W & 0 & -A/W & 0 \\ 0 & 1 & 0 & 0 \\ A/W & 0 & C/W & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly,

$$R'_y = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C/W & 0 & A/W & 0 \\ 0 & 1 & 0 & 0 \\ -A/W & 0 & C/W & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Composite transformation.

If we're asked to perform three or more different sorts of transformations over a point (P_0) simultaneously which is placed in 3D space, and let us assume that these transformations are T_1, T_2 and T_3 respectively.

(i) $P_0 = (x, y, z)$

$P_1 = T_1 P$

$P_2 = T_2 P_1$

$P_3 = T_3 P_2$

(ii) Composite Transformation

$R = T_3 \cdot T_2 \cdot T_1$

$P_3 = R \cdot P$

Q. Consider we are given with a cuboid "OABCDEFG" over which we want to perform:

Translation distances T_1: $T_x = 2$, $T_y = 3$, $T_z = 2$.
Scaling transformation T_2: $S_x = 2$, $S_y = 1$, $S_z = 3$
Shearing transformation T_3: $S_y = 2$ and $S_z = 1$

Co-ordinates: A(0,0,4), B(0,4,2), C(2,4,0), D(2,2,4), E(2,0,0), F(0,0,2), G(2,0,2)

Solution

Composite matrix = $T_3 \; T_2 \; T_1$

$= Sh_{z\,axis}(2,1) \cdot S(2,1,3) \cdot T(2,3,2)$

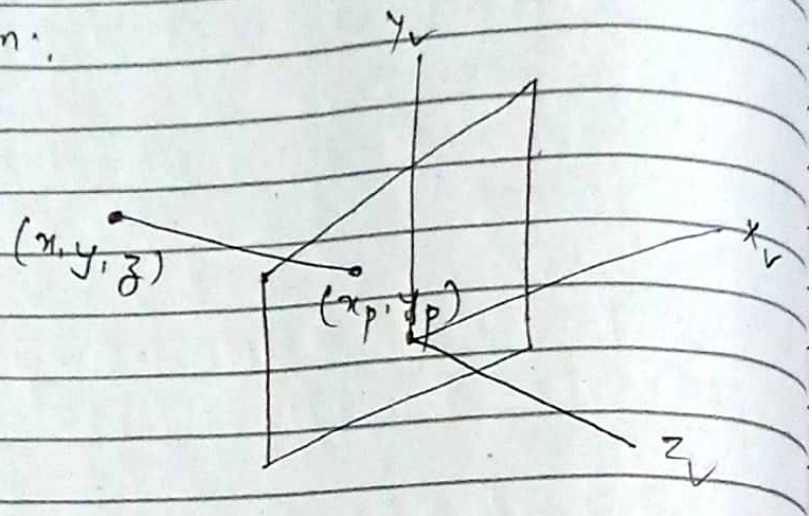
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection:

(i) Oblique projection:

$x_p = x$

$y_p = y$



$x_p = x + z L_1 \cos\theta$

$y_p = y + z L_1 \sin\theta$

(ii) Perspective projection:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \frac{1}{z+d} \begin{bmatrix} d_x \\ d_y \\ 0 \\ z+d \end{bmatrix} = \frac{1}{z+d} \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Unit: 5

3D OBJECT REPRESENTATION

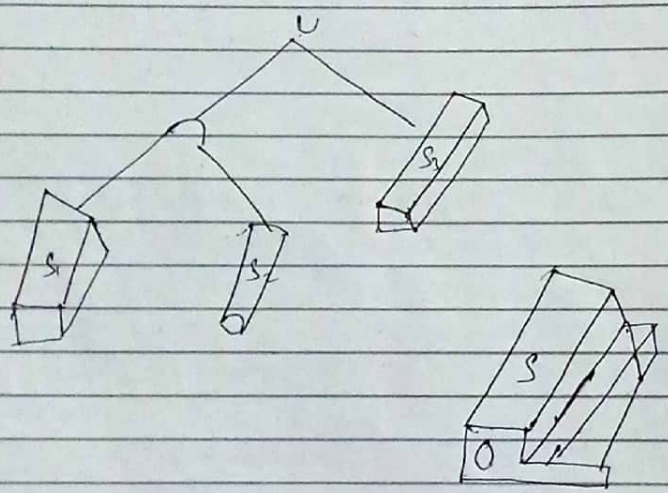
-> objects are represented as a collection of surfaces.
-> 3D object representation is divided into two categories

i> Boundary Representations:

- B-reps:

It describes a 3D object as a set of surface that separates the object interior from the environment

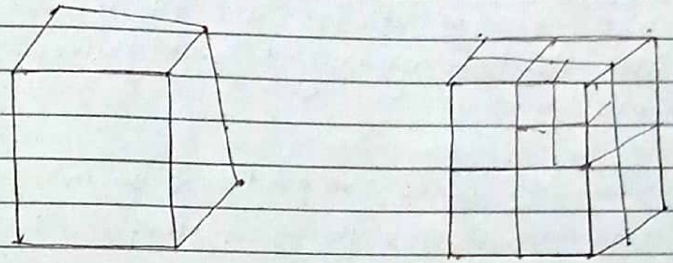
-> Many graphics system use this method.
-> set of polygons are stored for object description.
-> This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.

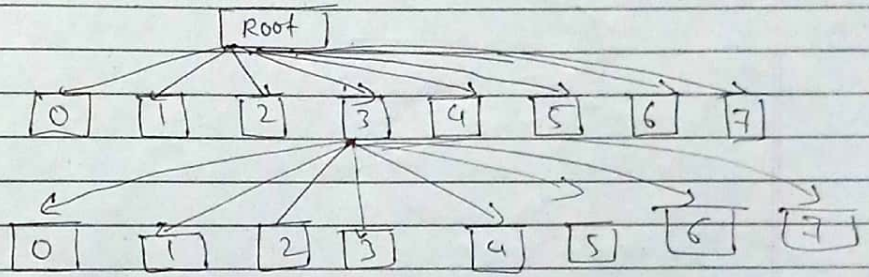


ii> Space-partitioning representations

-> It is used to describe interior properties by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids usually cubes.

-> A common space-partitioning descriptional for a 3D object are: Quad-tree, Octree representations.





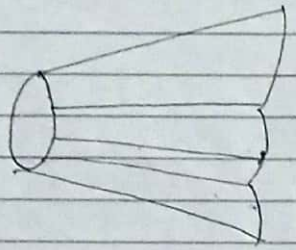
Boundary representation:

→ Polygon surfaces are the most widely used boundary representation.

→ Other methods such as wireframe representation, blobby objects, quadratic surfaces, spline representation, Bezier curve are surfaces are also used.

→ The polygon surfaces are common in design and solid-modeling applications, since their wireframe display can be done quickly to give general indication of surface structure.

→ Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate



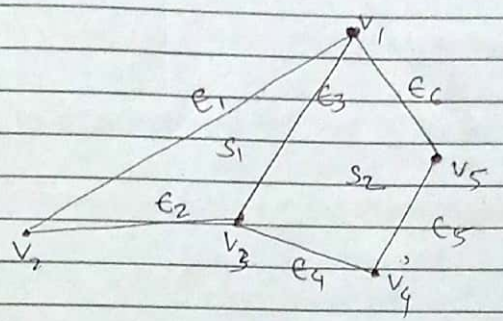
Polygon table:

A convenient organization for storing geometric data is to create 3 lists:

(i) Vertex table: Stores co-ordinates of each vertex in the polygon.

(ii) Edge table: Stores the edge information of each edge of polygon i.e. vertex pair for each edge.

(iii) Surface table: Stores the surface information for each surface i.e. edge list for each surface.



Vertex table:
V_1: x_1, y_1, z_1
V_2: x_2, y_2, z_2
V_3: x_3, y_3, z_3
V_4: x_4, y_4, z_4
V_5: x_5, y_5, z_5

Edge table:
E_1: V_1, V_2
E_2: V_2, V_3
E_3: V_1, V_3
E_4: V_3, V_4
E_5: V_5, V_4
E_6: V_1, V_5

Surface table:
S_1: E_1, E_2, E_3
S_2: E_3, E_6, E_5, E_4

Polygon Meshes:

It is a type of computer graphics technique used for creating 3D models. It is a collection of vertices, edges and faces that define the shape and surface of a 3D object. It is often used in computer games, animation, virtual reality, and computer-aided-design (CAD). This is used to create realistic images.

Advantages:

(i) It can be used to model almost any object.
(ii) Easy to represent as a collection of vertices.
(iii) They are easy to transform
(iv) Easy to draw on computer screen.

Disadvantages:

(i) Curved surfaces can only be approximately described
(ii) Difficult to simulate objects like hair or liquid.

A polygon mesh can be represented in three ways.

(i) Explicit
(ii) Pointer to vertex list and
(iii) Pointer to an edge list.

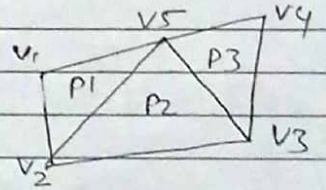
(i) Explicit representation:

→ Each polygon is represented by a list of vertex co-ordinates.

$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n))$

→ The vertices are stored in the order of travelling around the polygon.

→ There are edges between successive vertices in the list and between the last and first vertices



$P_1 = \{v_1, v_2, v_5\}$
$P_2 = \{v_2, v_3, v_5\}$
$P_3 = \{v_3, v_4, v_5\}$

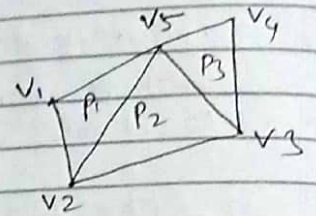
Here, most of the vertices are duplicated, so it is not efficient.

(ii) Pointer to a vertex list:

Each vertex is stored just once, in vertex list

$V = \{ v_1, v_2, v_3, \ldots, v_n \}$
$= \{ (x_1, y_1, z_1), (x_2, y_2 z_2), \ldots, (x_5, y_5, z_5) \}$

$P_1 = \{1, 2, 5\}$
$P_2 = \{2, 3, 5\}$
$P_3 = \{3, 4, 5\}$



(iii) Pointers to an edge list:

→ Each edge in edge list points to the two vertices

Hence, we describe polygon as:

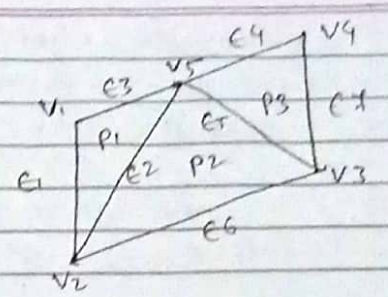
$P = \{ E_1, E_2, E_3, \ldots, E_n \}$

And an edge as:

$E = \{ v_1, v_2, P_1, P_2 \}$

→ If edge belongs to only one polygon, then either P1 or P2 is null.

$V = \{ v_1, v_2, v_3, \ldots, v_n \}$
$= \{ (x_1, y_1, z_1), (x_2, y_2, z_2), \ldots \}$



$E_1 = \{ (v_1, v_2, P_1, N)$
$E_2 = \{ v_2, v_5, P_1, P_2 \}$
$E_3 = \{ v_1, v_5, P_1, N \}$
$E_4 = \{ v_5, v_4, P_3, N \}$
$E_5 = \{ v_5, v_3, P_2, P_3 \}$
$E_6 = \{ v_2, v_3, P_2, N \}$
$E_7 = \{ v_4, v_3, P_3, N \}$

$P_1 = \{ E_1, E_2, E_3 \}$
$P_2 = \{ E_2, E_5, E_6 \}$
$P_3 = \{ E_4, E_7, E_5 \}$

(iv) Plane equations:

→ Equation of the plane in which polygon meshes lies.
→ Used to find the position of spatial points relative to the plane surfaces of an object.
→ Obtained by the vertex co-ordinate and the equation of the plane.

→ The eqn of plane surface is $Ax + By + Cz + D = 0$ where x, y and z are any points on the planee and A, B, C, and D are constants.

→ Obtain the values of A, B, C, and D by solving three plane equations using the co-ordinate values for 3 non-collinear points in the plane.

→ Assume three vertices of the plane (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3):

ie. $\frac{Ax}{D} + \frac{By}{D} + \frac{Cz}{D} + \frac{D}{D} = 0$

$\therefore \frac{A}{D}x + \frac{B}{D}y + \frac{C}{D}z = -1$

So, for three co-ordinates,

$A/D\ x_1 + B/D\ y_1 + C/D\ z_1 = -1$

$A/D\ x_2 + B/D\ y_2 + C/D\ z_2 = -1$

$A/D\ x_3 + B/D\ y_3 + C/D\ z_3 = -1$

Apply Cramer's rule:

$$A = \begin{bmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{bmatrix}, \quad B = \begin{bmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{bmatrix}, \ldots$$

$$C = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}, \quad D = -\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

For any point x, y, z with parameters A, B, C and D, we can say that,

(i) $Ax + By + Cz + D = 0$, means the point is on the plane.

(ii) $Ax + By + Cz + D \neq 0$, means the point is not on the plane

→ if (< 0), the point is inside the surface.

else,

the point is outside the surface.

Wireframe Representation:

→ 3D objects is represented as a list of straight lines, by its two end points (x_1, y_1, z_1) and (x_2, y_2, z_2).

→ Only show the skeletal structure of the objects.

→ Is simple and fast.

→ Not realistic

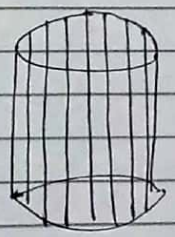


Fig: Wireframe model of a cylinder

Example of wireframe model:

An object is specified by two tables:

i> Vertex Table
ii> Edge table.

Vertex table:

Vertex	X	Y	Z
1	1	1	1
2	1	-1	1
3	-1	-1	1
4	-1	1	1
5	1	1	-1
6	1	-1	-1

Edge table:

Edge	Start vertex	End vertex
1	1	2
2	2	3
3	3	4
4	4	1
5	5	6
6	6	7
7	7	8

Advantages:

i> Simple and easy to create
ii> Requires little computer time for creation
iii> Requires less computer memory; so the cost is reduced.
iv> Clipping is easy.

Disadvantages:

i> Gives only information about the outlook, but doesn't give any information about the complex part.

Blobby Object:

Some objects change their surface characteristics in certain motion and do not maintain a fixed shape, and named by blobby object.

Examples: water droplets, melting objects and muscle shapes in human body.

-> Several models have been developed to handle these kind of objects.

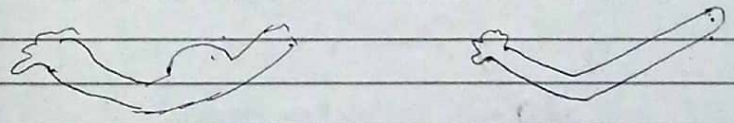


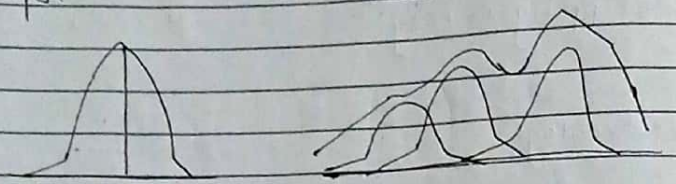
Fig: blobby object.

B.O = Blobby object

B.O

Gaussian Bumps:

One way to represent blobby objects is to model them as a combination of Gaussian density functions, or bumps.



The surface can be defined by:

$$f(x,y,z) = \sum_k b_k * \exp(-a_k * r_k^2) - T = 0$$

where

$r_k^2 = x_k^2 + y_k^2 + z_k^2$, T is some specific threshold

Parameters a and b are used to adjust the blobbiness of the individual objects.

B.O

Meta-balls:

The "meta-ball" (organic-looking n-dimensional objects) model describes complex object as combination of quadratic density functions of the form

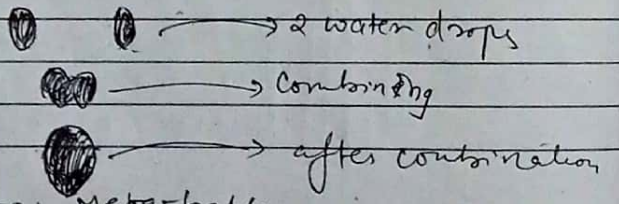


Fig: Meta-balls

→ In solid modeling, polygon meshes are commonly used. In certain instances, however, metaballs are superior.

Quadratic Surfaces:

A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics).

They include:

i) Spheres.
ii) Ellipsoids,
iii) Paraboloids,
iv) Hyperboloids, etc.

Sphere:

In cartesian co-ordinates, a spherical surface with radius r centered on the co-ordinate origin is defined as the set of points (x, y, z) that satisfy the eqn:

$$x^2 + y^2 + z^2 = r^2$$

In parametric form,

$$x = r\cos\phi\cos\theta, \quad -\pi/2 \le \phi \le \pi/2$$
$$y = r\cos\phi\sin\theta, \quad -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$

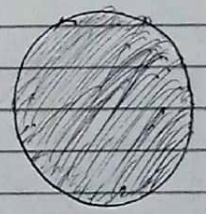


Fig: Sphere

Ellipsoid:

An ellipsoidal surface can be described as an extension of a spherical surface where the radii in three mutually perpendicular directions can have different values.

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

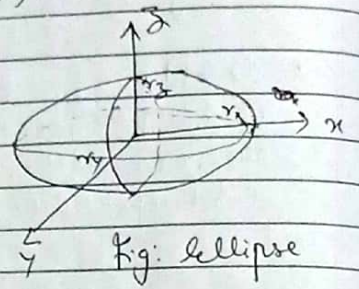


Fig: Ellipse

Superquadrics

→ They are formed by incorporating additional parameters into the quadratic equ's to provide increased flexibility for adjusting object shapes.

→ The no. of additional parameters used is equal to the dimension of the object: one parameter for curves and two parameter for surfaces.

Examples: (i) superellipse
(ii) superellipsoid

Superellipse

We can obtain superellipse by allowing the exponent on the x and y terms to be a variable.

$$\left(\frac{x}{r_x}\right)^{2/s} + \left(\frac{y}{r_y}\right)^{2/s} = 1$$

where parameter s can be assigned any real value. At s=1, we obtain ordinary ellipse.

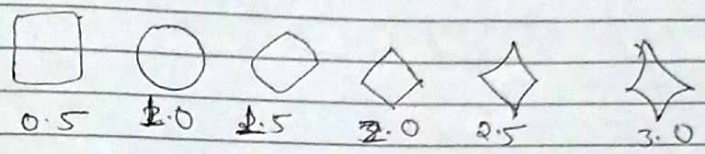


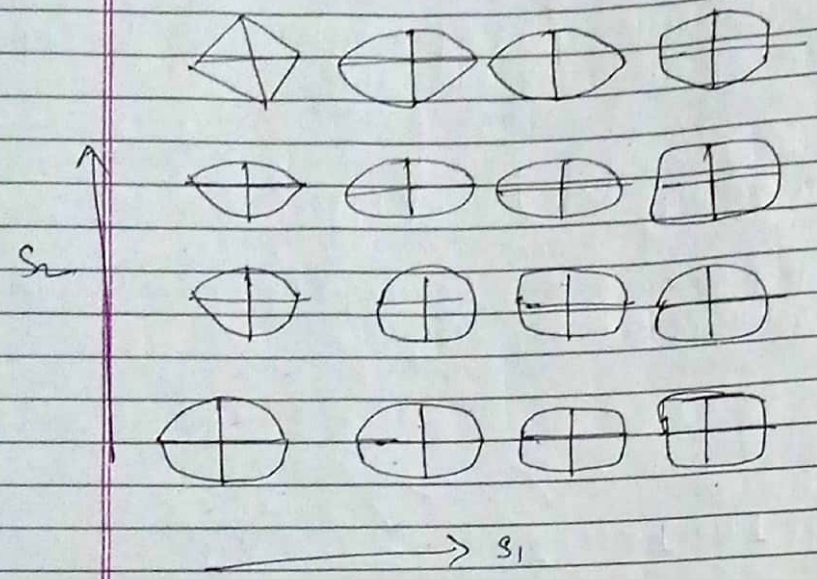
Fig: with different value of s and with $r_x = r_y$.

Superellipsoids:

This can be obtained by adding two exponent parameters as this is a surface.

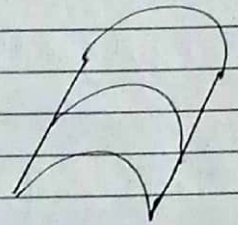
$$\left[\left(\frac{x}{r}\right)^{2/s_2} + \left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{z}{r_z}\right)^{2/s_1} = 1$$

For $s_2 = s_1 = 1$, we have ordinary ellipsoid.



Representing Curves:

→ Curves are the basics for surfaces.



- Explicit form:
  $y = mx + c$
- Implicit form: $ax + by + c = 0$

Implicit: $f(x,y) = c$
Explicit: $y = f(x)$

Drawbacks of Conventional Representations:

i) They represent unbounded geometry.
ii) Difficult to evaluate points along the curve
iii) Depends on co-ordinate system.

Parametric Representation:

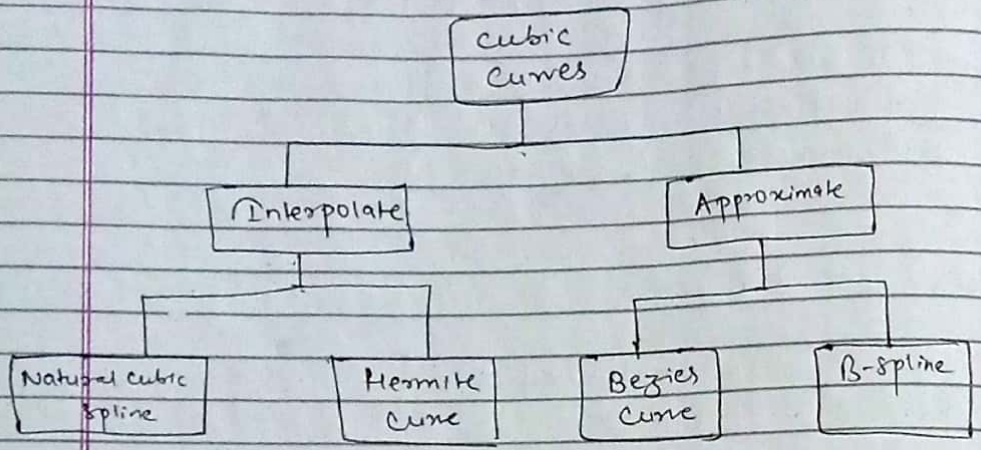
Curves are defined as a function of a single parameter. $Q = Q(t)$

and,
$x = x(t)$
$y = y(t)$
$z = z(t)$

i) Can represent closed curves.
ii) Simple to render; just because t and get new x, y to render.

Why to use cubic polynomials?

→ Lower degree of polynomial give too little flexibility in controlling shapes of the curve while higher order degree poly. give reasonable design and flexibility, but introduce unwanted wriggles and also requires more computation. For this reason, the three degree polynomials are used, known as cubic polynomials.



→ Curves ~~are~~ can be specified by giving a set of co-ordinate positions (control points) which indicate the general shape of the curve.

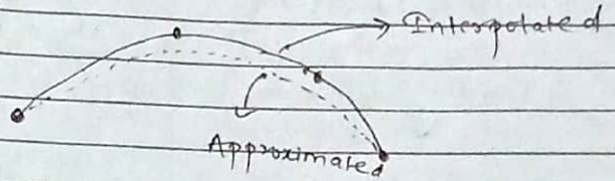
→ These control points are then fitted with parametric functions in one of two ways.

Interpolation and approximation.

→ When the polynomial sections are fitted so that the curve passes through each point the resulting curve is said to interpolate the set of control points.

Control points

→ When the polynomials are fitted to the general control point path without necessarily passing through any control point, the resulting curve is approximate control points

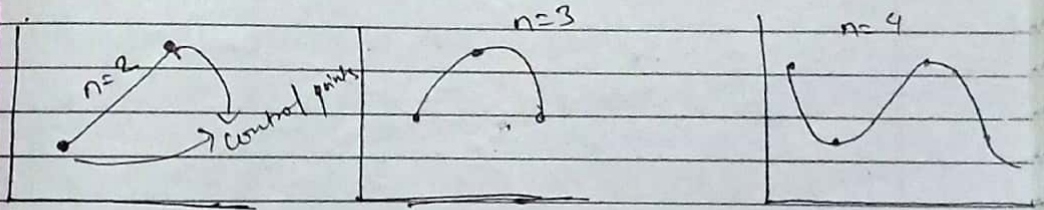


Spline representation:

→ Splines are the smooth curves passing through a set of given points known as control points.

→ These control points determine the general shape of the curve.

→ Mathematically, a spline curve refers to any composite curve formed with polynomial functions satisfying specified boundary conditions at the section endpoints.



First order (linear) Second order (quadratic Third order (cubic)

=> Order of spline is one less than no. of control points.

Spline representation

(i) Boundary Conditions
(ii) Matrix (Characterizing matrix)
(iii) Blending functions or basis functions.

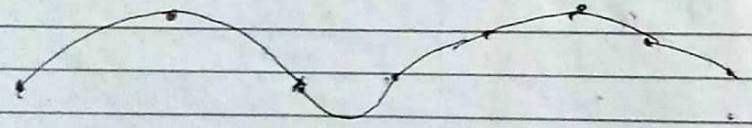
✓ Natural Cubic Splines

-> Given a set of control points, cubic interpolation splines are obtained by fitting the input points with a piecewise cubic polynomial curve that passes through every control points.

-> Suppose we have n+1 control points specified with co-ordinates.

-> $p_k = (x_k, y_k, z_k), k = 0, 1, 2, 3, \ldots, n$

-> A cubic interpolation fit of these points is:



A parametric cubic polynomial that is to be fitted between each pair of control points have following equations

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

Coefficient $a_i b_i c_i d$ can be determined from the above equations

✓ Hermite Spline Curve:

-> This procedure for defining a cubic curve using end points and tangent vector is one form of Hermite interpolation.

-> Hermite spline curve is interpolation spline curve.

-> The Hermite form of the cubic polynomial curve segment is determined by constraints on the end points P_1 and P_4 and the tangent vectors at the end points R_1 & R_4

-> It has local control over the curve.

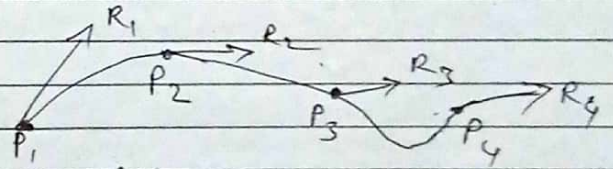


Fig: Hermite spline curve.

We have the general curve equ^n:

$P(t) = at^3 + bt^2 + ct + d$; $0 \le t \le 1$; t is parameter

So, $x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$ .1

$y(t) = a_y t^3 + b_y t^2 + c_y \cdot t + d_y$ .1

$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$ .1

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \underbrace{\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}}_{C}$$

$\therefore Q(t) = T \cdot C$

But as per rule for specifying a spline curve, we need a basis f^n matrix. So,

$Q(t) = T \cdot M \cdot G$, where $[C = M \cdot G]$

basis matrix

Geometry vector

Let for Hermite, we may write it as.

$Q(t) = T \cdot M_H \cdot G_H$

$= [t^3 \; t^2 \; t \; 1] \cdot M_H \cdot G_H$

Tangent of $[t^3 \; t^2 \; t \; 1] = [3t^2 \; 2t \; 1 \; 0]$

$Q_x(t)\big|_{t=0} = P_1(t) = [0 \; 0 \; 0 \; 1] \cdot M_H \cdot G_{Hx}$ — (A)

$Q_x(t)\big|_{t=1} = P_4(t) = [1 \; 1 \; 1 \; 1] \cdot M_H \cdot G_{Hx}$ — (B)

$Q'_{(x)}t\big|_{t=0} = R_1(t) = [0 \; 0 \; 1 \; 0] \cdot M_H \cdot G_{Hx}$ — (C)

$Q'_{(x)}t\big|_{t=1} = R_4(t) = [3 \; 2 \; 1 \; 0] \cdot M_H \cdot G_{Hx}$ — (D)

from A, B, C, D equations

$$\begin{bmatrix} P_1(x) \\ P_4(x) \\ R_1(x) \\ R_4(x) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{Hx} .$$

Or $M_H \cdot G_{Hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{1x}(x) \\ P_4(x) \\ R_1(x) \\ R_4(x) \end{bmatrix}$

So,

$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ and $G_{Hx} = \begin{bmatrix} P_{x_1} \\ P_{x_4} \\ R_{1x} \\ R_{4x} \end{bmatrix}$

$\therefore Q(t) = T \cdot M_H \cdot G_H$

$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$

$= (2t^3 - 3t^2 + 1) P_1 + (-2t^3 + 3t^2) P_4 + (t^3 - 2t^2 + t) R_1 + (t^3 - t^2) R_4$

$= P_1 H_0(t) + P_4 H_1(t) + R_1 H_2(t) + R_4 H_3(t)$

Where, $H_0(t)$, $H_1(t)$, $H_2(t)$ and $H_3(t)$ are Hermite blending functions.

Q. Find the midpoint of a Hermite cubic spline with the two endpoints as (1,1) and (6,5) and corresponding tangent vector as (0,4) and (4,0).

Solution

Given:

$P_1 = (1,1)$, $P_4 = (6,5)$ } endpoints

$R_1 = (0,4)$, $R_4 = (4,0)$ } tangent vectors

Here,

We need to find midpoint, so $t = 0.5$

Now,

In matrix form

$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$

$= \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 6 & 5 \\ 0 & 4 \\ 4 & 0 \end{bmatrix}$

Bezier curve.

→ Developed by Pierre Bezier for designing of Renult automobiles bodies.

→ It is an approximation curve used in various CAD system.

→ Control points are used to design curve.

General Bezier curve for (n+1) control points with co-ordinates $P_k = (x_k, y_k, z_k)$ where $k = 0$ to n.

$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u), \quad 0 \le u \le 1$$

Where

$P(u)$ = Points in Bezier curve.

P_k = Control points.

$BEZ_{k,n}(u)$ = Bezier blending function.

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

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Individual x, y and z co-ordinates are

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{n,k}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{n,k}(u)$$

$$z(u) = \sum_{k=0}^{2} P_k \, BEZ_{n,k}(u)$$

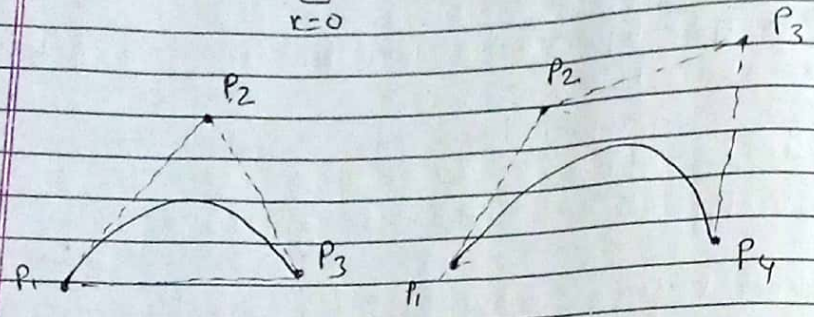


Fig: Bezier curve generated by three and four control points.

Cubic Bezier curve:

- Curve generated by four control points.
- Generated expression is,

$$P(u) = \sum_{k=0}^{3} P_k \cdot BEZ_{k,3}(u) \; ; \; 0 \le u \le 1$$

$$= P_0 BEZ_{0,3}(u) + P_1 BEZ_{1,3}(u) + P_2 BEZ_{2,3}(u) + P_3 \cdot BEZ_{3,3}$$

where

$$BEZ_{(0,3)}(u) = C(3,0) u^0 (1-u)^{3-0}$$

$$BEZ_{(1,3)}(u) = C(3,1) u^1 (1-u)^{3-1}$$

$$BEZ_{(2,3)}(u) = C(3,2) u^2 (1-u)^{3-2}$$

$$BEZ_{(3,3)}(u) = C(3,3) u^3 (1-u)^{3-3}$$

Here

$$C(n,k) = \frac{n!}{(n-k)!\,k!}$$

Solving $BEZ_{n,k}(u)$, we get

$$P(u) = P_0 (1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 \cdot 3u^2(1-u) + P_3 u^3$$

In matrix form,

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_{BEZ} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

where

- M_{BEZ} = Bezier matrix $= \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Q. Construct the Bezier curve of order 3 and four polygon vertices are A(1,1), B(2,3), C(4,3) and D(6,4). Also draw the curve.

Q. Construct Bezier curve for control points (4,2), (8,8) and (16,4).

Solution,

Given control points

$P_0(x_1, y_1) = (4,2)$
$P_1(x_2, y_2) = (8,8)$
$P_2(x_3, y_3) = (16,4)$

Here
n = no. of control points − 1
= 3 − 1 = 2

for $n=2$,
we know that,

$$P(u) = \sum_{k=0}^{n} P_k \cdot BEZ_{k,2}(u)$$

So, when $n=2$

$$P(u) = \sum_{k=0}^{2} P_k \cdot BEZ_{k,2}(u)$$

$$= P_0 \cdot BEZ_{0,2}(u) + P_1 \cdot BEZ_{1,2}(u) + P_2 \cdot BEZ_{2,2}(u)$$

where,

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

$$BEZ_{(0,2)}(u) = C(2,0) \times u^0 (1-u)^2 = (1-u)^2$$

$$BEZ_{(1,2)}(u) = C(2,1) \times u^1 (1-u)^1 = 2u(1-u)$$

$$BEZ_{(2,2)}(u) = C(2,2) \times u^2 (1-u)^0 = u^2$$

On substitution we get,

$$P(u) = P_0(1-u)^2 + P_1 \times 2u(1-u) + P_2 \cdot u^2$$

$$x(u) = x_0(1-u)^2 + 2u x_1 (1-u) + x_2 u^2$$
$$y(u) = y_0(1-u)^2 + y_1 2u(1-u) + y_2 \cdot u^2$$

Now,

u	$x(u)$	$y(u)$	$x(u), y(u)$
0			
0.			

Make curve

B-Spline Curve:

It is a set of piecewise polynomial segments that passes close to a set of control points.

It has two advantages over Bezier curve.

(a) The degree of B-Spline polynomial can be set independently of the no. of control points.

(b) It allows local control over the shape of a spline curve.

General equ^n of B-Spline curve is given by

$$P(u) = \sum_{k=0}^{n} P_k B_{k,d}(u), \quad 0 \le u \le n+d, \quad 2 \le d \le n+1$$

Assignment

Q. Bezier vs B-Spline Curve's properties.

(i) Degree:

The degree of Bezier curve is determined by the number of control points minus one.

B-Spline curve can have a varying degree. A B-Spline curve of degree 'k' is controlled by 'k+1' control points.

(ii) Local Control:

Each control point in a curve directly influences the shape of the curve segments between the end points.

Control points in a B-Spline curve have more localized influence changing one control point primarily affects only a portion of the curve.

(iii) ~~Global vs local influence~~ - Flexibility.

Bezier curves are often easier to work with for simple shapes and low degrees, but they can be limiting for complex shapes.

B-Spline offer greater flexibility for designing complex curves, surfaces and shapes.

(iv) Curve modifications:

Modifying a Bezier curve often requires adjusting control points, which can lead to unintended changes in adjacent curve segments.

B-Spline handle modifications more gracefully allowing adjustments to one part of the curve without significantly affecting others.

(v) Knot vector:

Bez. curve do not use knot vector, solely rely on control points.

B-Splines use a knot vector to define the parameter range and influence of each control point.

Unit: 6

Solid Modeling

Introduction:

→ Solid modeling is a consistent set of principles for mathematical and computer modeling of three dimensional solids.

→ In the solid Modeling, the solid definitions include vertices (nodes), edges, surfaces, weight, and volume.

→ The model is a complete, valid and unambiguous representation of a precisely enclosed and filled volume.

Terms:

(i) Geometry:

Metrics and dimensions of the solid object. Location of the object in a choosen co-ordinate system. e.g.: (x, y, z) co-ordinates of vertices.

(ii) Topology:

Combinational information like connectivity, associativity, and neighborhood information. Invisible relationship information.

Examples: Connectivity matrix.

Methods for solid modeling:

(i) Boundary representation
(ii) Octrees, quadtrees
(iii) Sweep representation
(iv) Constructive solid geometry
...

Boundary representation

(i) Euler's law:

→ This gives a quantitative relationship among faces, edges, vertices, etc.

→ Euler's formula for any polyhedron that doesn't intersect itself, the Number of faces plus the number of vertices (corner points) minus the no. of edges always equals 2.

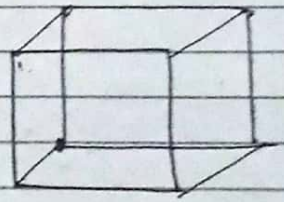
$$\text{i.e. } f + V - E = 2$$

e.g.:

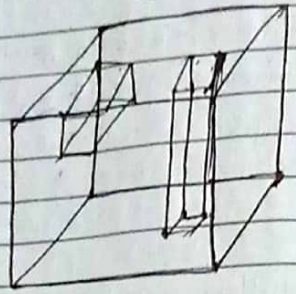
$f = 6, V = 8, E = 12$

So,

$6 + 8 - 12 = 2$



Euler's law fails when the 3D objects has holes.



$$V - E + f - H = 2(C - G)$$

H → No. of holes
C → Complete holes
G → partial holes

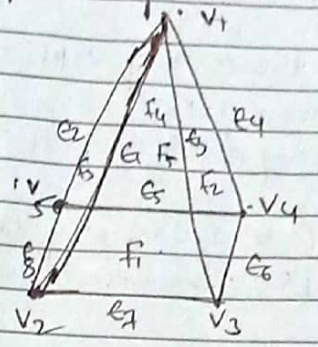
So,

$$24 - 36 + 15 - 3 = 2(1-1) = 0$$

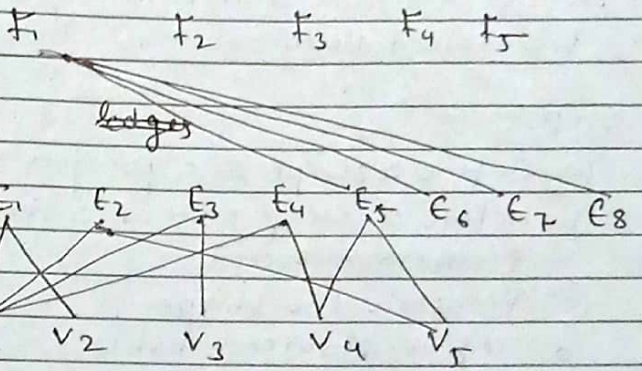
$$\text{or, } 39 - 39 = 0$$

$$\therefore 0 = 0$$

(11) Graphical representation



Solid



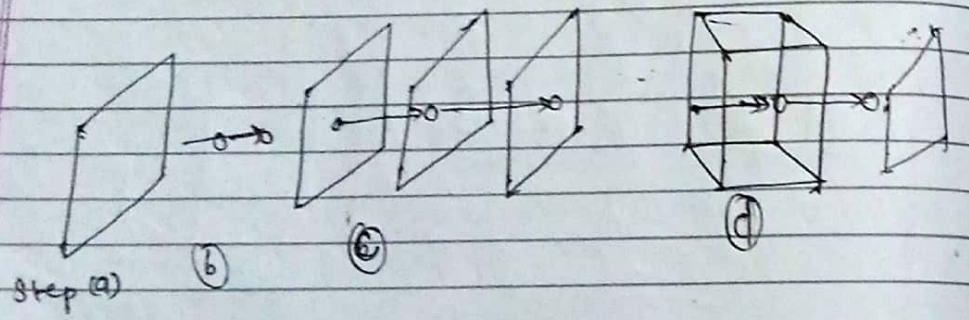
Sweep Representation:

→ It is based on the moving a point, curve, or a surface along a given path.
→ It means sweeping a 2D surface in 3D space to create an object.
→ However, the objects are usually converted into polygon meshes and/or parametric surfaces before storing.

(i) Translational sweep:

→ A 2D shape is translated by a pre-defined translation vector.

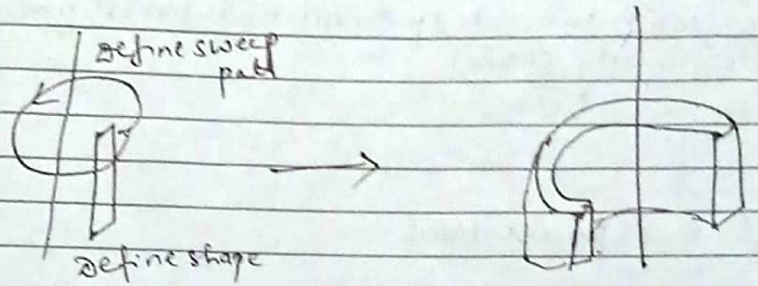
Steps:
(a) Define a shape as a polygon vertex table.
(b) Define a sweep path as a sequence of translation vectors.
(c) Translate the shape.
(d) Define a surface table.



(ii) Rotational sweep:

A 2D shape is rotated around a pre-defined rotational axis.

Steps:
(a) Define a shape as a polygon vertex table.
(b) Define a swee path as a sequence of rotation.
(c) Rotate the shape.
(d) Define a surface table.



Uses:
→ Used for creating solids with uniform thickness in a particular direction and axisymmetric solids by translational and rotational sweeping.

Other variations:

→ We can specify a special path for the sweep as some curve function.
→ Can vary the shape or size of the cross section along the sweep path.

✓ Space partitioning representations:

These are used to describe the interior properties, by partitioning the spatial regions containing an object into a set of small non-overlapping contiguous solids (usually cubes).
Example: Octree

Octree Representation:

(i) Hierarchical tree structures (octree) used to represent solid object in some graphical system.
(ii) Medical imaging and other applications use this method. e.g.: CT scan
(iii) It provides a convenient representation for storing information about object interiors.
(iv) An octree divides region of 3D space into octants and stores 8 data elements in each node of the tree.

(v) Individual elements are called volume element or voxels.

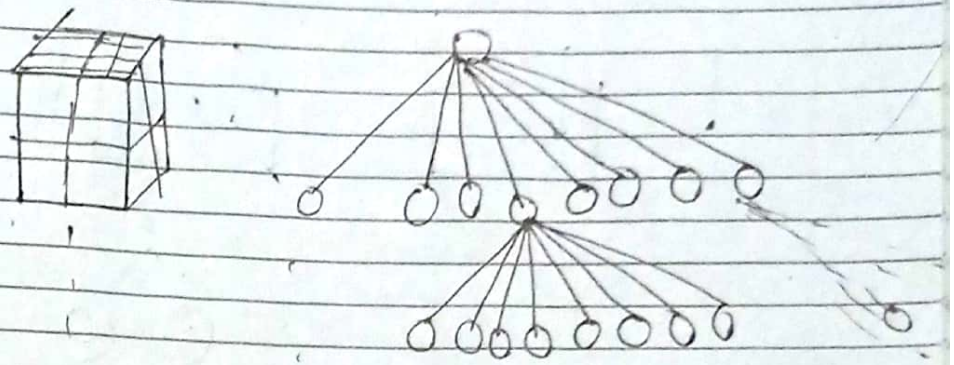


Fig: Octree

✓ Binary Space Partitioning (BSP):

→ It is a 3-D graphics programming technique of dividing a scene into two recursively using hyper-planes.
→ A 3D scene is split in two using a 2-D plane, then that scene is divided in two using a 2-D plane, and so on.
→ The resulting data structure is a binary tree.
→ It is widely used to speed up rendering of 3D scenes, especially in games.

Binary Space partitioning

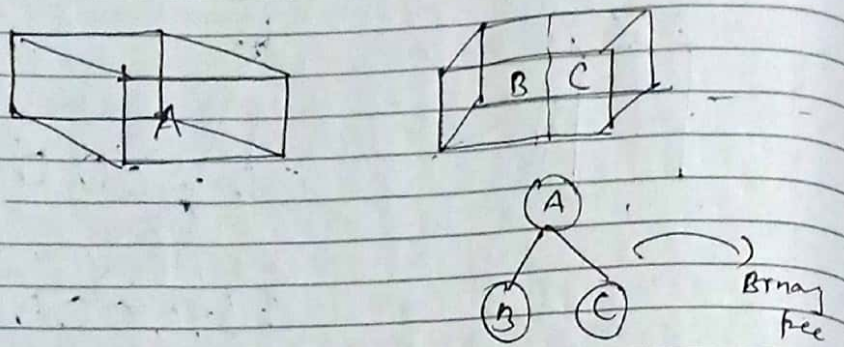


Fig: Binary space partitioning.

Unit: 7

Visible Surface Detection

Hidden Surface Removal (Visible Surface Detection) Method:

→ A major consideration in the generation of realistic graphics displays is identifying those parts of a scene that are visible from a chosen viewing position.
→ There are many algos for efficient identification of visible objects but all are differing according as memory requirement, processing time, special application, etc.

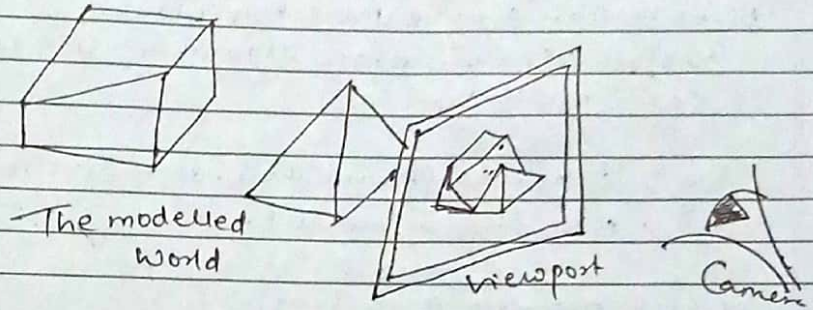


Fig: The basics of 3D graphical processing.

Visible-surface detection algorithms are broadly classified as:

1) Object-space methods (OSM):

→ Deal with object definition.
→ It 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.

2) Image-space methods (ISM):
→ Deal with projected image.
→ 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-division method.

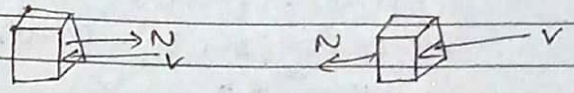
3) List priority Algorithms:
→ This is a hybrid model that combines both object and image precision operations.
→ Here, depth comparison and object splitting are done with object precision and scan conversion is done with image precision.

e.g. Depth-sorting method, BSP-tree method.

Back-face detection Method:

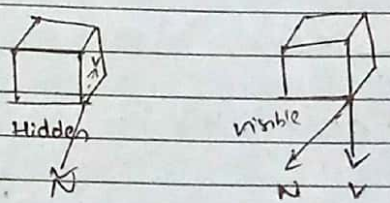
A view vector V is constructed from any point on the surface to the viewpoint, the dot product of this vector and the normal N, indicates visible faces as follows:

Case I: If $V \cdot N < 0$ the face is visible; else face is hidden.



Visible surface Hidden surface

Case II: If $V \cdot N > 0$, the face is visible; else face is hidden.

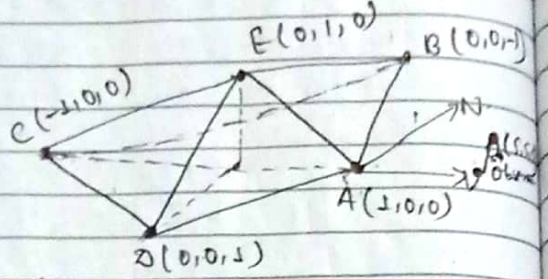


Case III: For other objects, such as the concave polyhedron in fig., more tests need to be carried out to determine whether there are additional faces that are totally or partly obscured by other faces

Numerical

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

Solution



Here

$$\vec{AE} = \vec{OE} - \vec{OA}$$
$$= (0,1,0) - (1,0,0)$$
$$= (0-1)\hat{i} + (1-0)\hat{j} + (0-0)\hat{k}$$
$$= -\hat{i} + 1\hat{j} + 0\hat{k}$$

$$\vec{N} = \vec{AD} \times \vec{AE}$$

$$\vec{AD} = \vec{OD} - \vec{OA}$$
$$= (0,0,1) - (1,0,0)$$
$$= -\hat{i} + \hat{k}$$

Now,

$$\vec{N} = \vec{AD} \times \vec{AE}$$

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

$$\vec{V} = \vec{AP} = \vec{OP} - \vec{OA} = (5,5,5) - (1,0,0)$$
$$= 4\hat{i} + 5\hat{j} + 5\hat{k}$$

So,

$$\vec{V} \cdot \vec{N}$$
$$= (4\hat{i} + 5\hat{j} + 5\hat{k}) \cdot (-\hat{i} - \hat{j} - \hat{k})$$
$$= -4\hat{i} - 5\hat{j} - 5\hat{k}$$
$$= -4 - 5 - 5$$
$$= -14 < 0 \text{ ; hidden.}$$

Depth-Buffer (Z-Buffer Method):

→ It is commonly used image-space approach which compares surface depths at each pixel position (i.e. one point at a time across the surface) on the projection plane.

→ It is also refered to as the z-buffer method, since object depth is usually measured from the view plane along the z-axis of a viewing system.

→ z value ranges from 0 at the back clipping plane to z_{max} at the front clipping plane.

→ This is usually for polygon surface, because depth values can be computed very quickly and the method is easy to implement. It can also be applied to non-planar surfaces.

→ It requires two buffer areas.

→ Depth buffer: Stores depth values (z-values) for each (x,y) position as surfaces are processed.

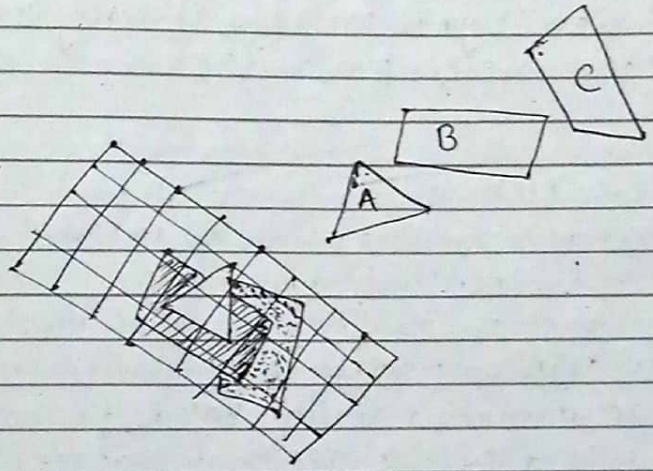
→ Refesh buffes: It stores the intensity values for each position.

→ Initially, all positions in the depth buffer are set to 0 (minimum depth), and the refesh buffes is initialized to the background intensity.

→ Each surface listed in the polygon tables is then processed, one scan line at a time, calculating the depth (z-value) at each (x,y) pixel position.

→ The calculated depth is compared to the value previously stored in the depth buffer at that position.

→ If the calculated depth is greater than the value stored in the depth buffer, the new depth value is stored, and the surface, and the surface intensity at that position is determined and placed in the same xy location in the refesh buffes.



Algorithm:

(i) Initialize the depth buffer and refresh buffer so that for all buffer position (x,y),
depth (x,y) = 0 or (zmin) and refresh (x,y) = Ibackground

(ii) For each position on each polygon surface compare depth values to previously stored values in the depth buffer to determine visibility.

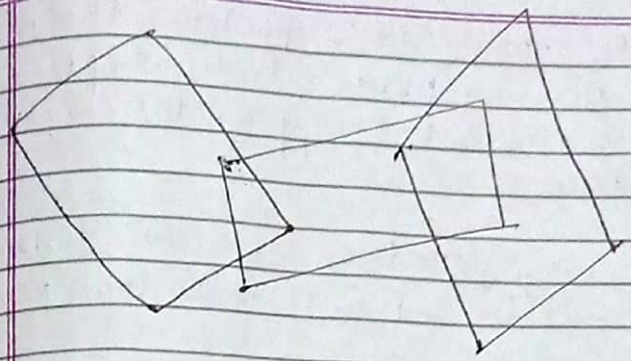
a. Calculate the depth 'z' for each (x,y) position on the polygon.
b. If z > depth (x,y) then set,
depth = z and refresh (x,y) = Isurface (x,y)

Where,
Ibackground (x,y) is background intensity,
Isurface (x,y) is projected intensity value for surface at pixel position (x,y).

A-Buffer Method:

→ Accumulation buffer is an extension of the ideas in the depth-buffer method.

→ A drawback of the depth-buffer method is that it deals only with opaque surfaces and cannot accumulate intensity values for more than one surface or transparent surfaces.

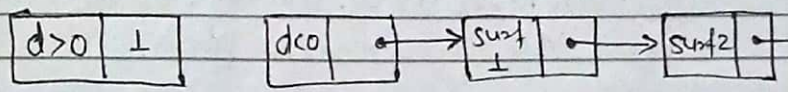


→ Viewing an opaque surface through a transparent surface requires multiple surface-intensity contributions for pixel positions.

→ The A-buffer method calculates the surface intensity for multiple surfaces at each pixel position.

→ Each position in the A buffer has two fields:
depth field and intensity field.

Algo:



Depth field | Intensity field | Depth field | Intensity field

Case-I | Case-II

Case I: If the depth field is positive, the no. stored at that position is the depth of a single surface overlapping the corresponding pixel area.

- The intensity field then stores the RGB components of the surface color at that point.

Case II: If the depth field is negative, this indicates multiple-surface contributions to the pixel intensity.

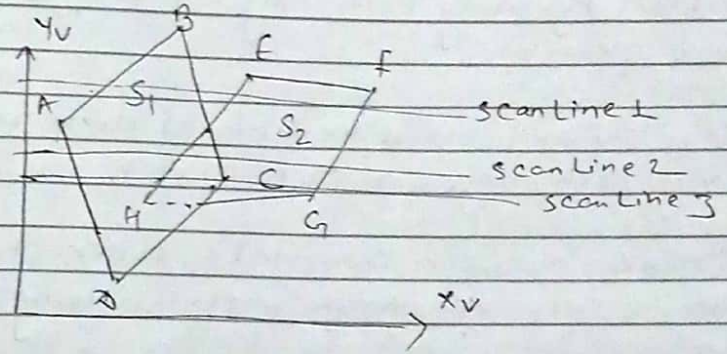
- The intensity field then stores a pointer to a linked list of surface data, as in second figure.
- Data for each surface in the linked list includes: RGB intensity components, opacity parameter, depth and pointer to next surface
- Using the opacity factors and percent of surface overlaps, we can calculate the intensity of each pixel as an average of the contributions from the overlapping surfaces.

Scan-Line method:

→ This image-space method for removing hidden surfaces is an extension of the scan-line algorithm for filling polygon interiors, here we deal with multiple surfaces rather than one.

→ Each scan line is procesed 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.



→ 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,

Depth-Sorting method / painter's algorithm:

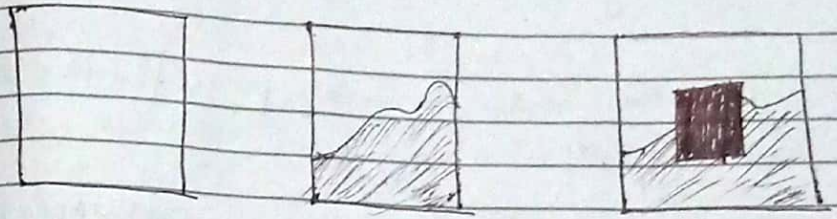
→ It is method for solving the hidden-surface problem by using both image-space and object-space operations.

→ Surfaces are sorted in order of decreasing depth (using both image and object space).

→ Surfaces are scan converted in order, starting with the surface of greatest depth (using image space).

→ This is known as the painter's algorithm because an artist first paints the background colors, then the most distant objects are added, then the nearer objects, and so forth.

→ Thus, each layer of paint covers up the previous layers.



→ We first sort surfaces according to their distance from the view plane.

→ The intensity values for the farthest surface are then entered into the refresh buffer.

→ Taking each succeeding surface in turn (in decreasing depth order), we "paint" the surface intensities onto the frame buffer over the intensities of the previously processed surfaces.

→ As long as no overlaps occur, each surface is processed in depth order until all have been scan converted.

→ If a depth overlap is detected at any point in the list, the surfaces should be re-ordered.

→ Here are some tests for each surface that overlaps with surface s.

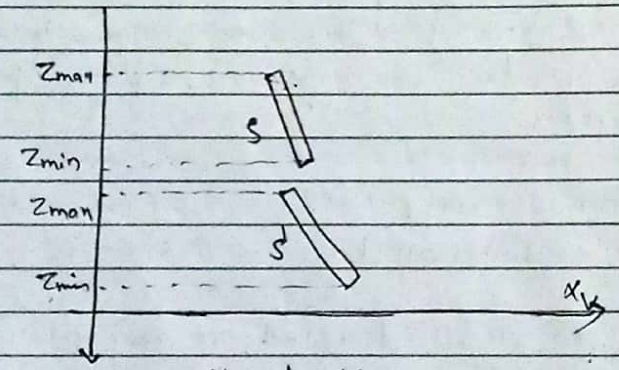
→ If any one of these tests is true, no reordering is necessary for that surface.

(a) Bounding rectangle in xy plane for two surfaces don't overlap

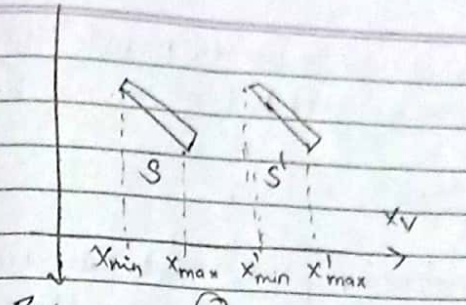
(b) Surface S is completely behind overlapping surface relative to viewing position

(c) Overlapping surface 's' is completely in-front of S relative to viewing position.

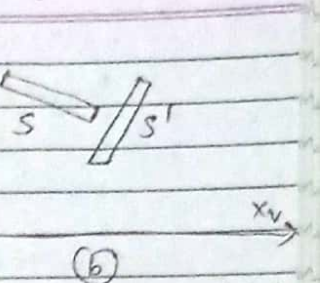
(d) Projection of 2 surfaces onto view plane doesn't overlap.



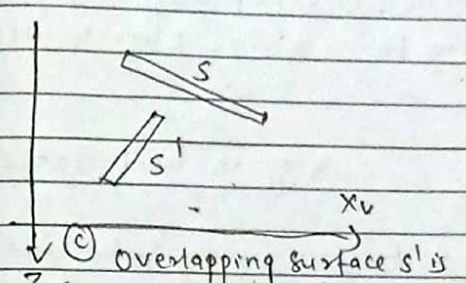
No depth overlap



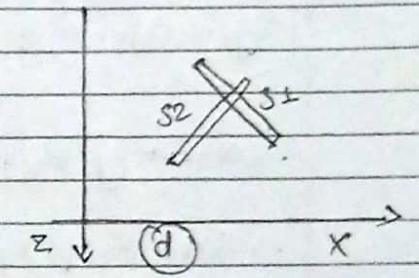
Depth overlap but no overlap in x direction



S is completely behind overlapping surface S'



Overlapping surface S' is completely in front of S.



Binary Space Partition Tree Method (BSP Tree Method)

→ It is an efficient method for determining object visibility by painting surfaces onto the screen from back to front as in the painter's algorithm.

→ A BSP tree is a recursive sub-division of space.

→ Useful when the view reference point changes, but object in a scene are at fixed position.

Construct a BSP Tree:

→ Select the partitioning plane and partition the space into two sets of objects; surfaces that are inside / front and outside / back wrt the partitioning plane.

→ In case of intersection, i.e., if object is intersected by partitioning plane then divide object into two objects.

→ Identify surfaces in each of the half spaces.

→ Each half space is then recursively subdivided using one of the polygon in the half space as partitioning plane. The process is continued till there is only one polygon in each half space.

→ Then the subdivided space is represented by a binary tree with the original polygon as the root.

→ Objects are represented as terminal nodes with front objects as left branches and back objects as right branches.

Display the BSP tree.

→ If the view point is in front of the root polygon

Unit : 8

Illumination Models And Surface Rendering Methods.

→ Once visible surface has been identified by hidden surface algorithm, an illumination model or lighting model and or shading model is used to calculate the intensity of light that we should see at a given point on the surface of a object for realistic displaying of 3D scene.

↳ Components of illumination model:

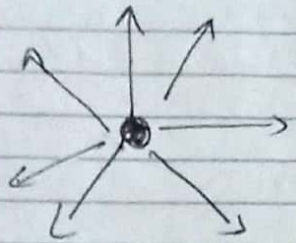
- Light sources:
  type, color, and direction of the light source.
- Surface properties:
  reflectance, Opaque/transparent, shiny/dull.

↳ Light Source:
Object that radiates energy are called light sources, such as sun, lamp, bulb, fluorescent tube, etc.

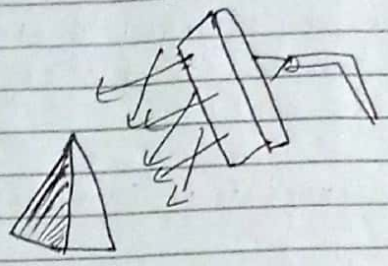
↳ Point Source:
→ It is the simplest light emitter e.g. light bulb
→ When light source model is a reasonable approximation for sources whose dimensions are small compared to the size of objects in a scene.

Distributed light source:

→ Area of source is not small compared to the surfaces in the scene eg. fluorescent lamp.



Point Light Source



Distributed light source

Types of reflection:

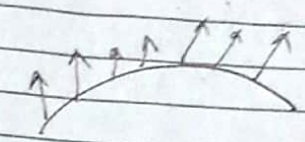
(i) Diffuse reflection:

Surface that are rough or grainy, tend to scatter the reflected light in all direction which is called diffuse reflection.

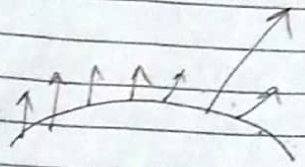
(ii) Specular reflection:

When light sources create high lights, or bright spots called specular reflection.

Eg: Shiny surface.



Diffuse reflection



Specular reflection.

Illumination models:

→ Illumination models are used to calculate light intensities that we should see at a given point on the surface of an object.

Some illumination models are

1. Ambient light
2. Diffuse reflection
3. Specular reflection and phong model

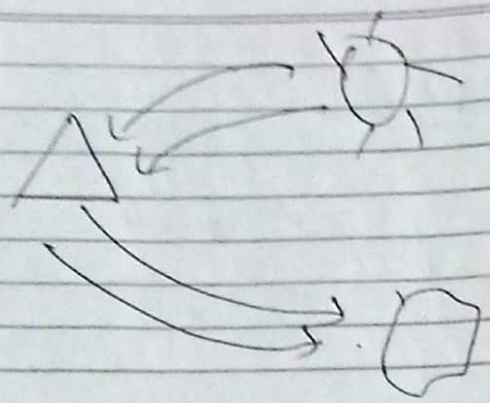
Ambient light:

→ This is a simplest illumination model.

→ A surface that is not exposed directly to light source still will be visible if nearby objects are illuminated.

→ The combination of light reflections from various surfaces produce a uniform illumination called ambient light or background light.

→ It usually refers to natural light, coming through windows, etc.



-> Ambient light has no spatial or directional characteristics, and amount on each object is a constant for all surfaces and all directions.
-> Multiple reflections of nearby objects yields a uniform illumination.

-> In this model, illumination can be expressed by an illumination equation:

$$I = I_a K_a$$

where I_a is intensity of ambient light.
K_a is object's intrinsic intensity

The amount of light reflected from an object's surface is determined by K_a, the ambient-reflection coefficient. K_a ranges from 0 to 1.

2. Diffuse reflection:

-> If surfaces are rough, incident light is scattered with equal intensity in all directions.
-> Surfaces appear equally bright from all direction.
-> Such surfaces are called ideal diffuse reflectors (also referred to as Lambertian reflectors).

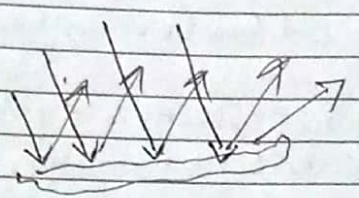


Fig: Diffuse reflection.

-> Illuminating object by a point light source, where rays enumerate uniformly in all directions from a single point.

If a surface is exposed only to ambient light, we can express the intensity of the diffuse reflection at any point on the surface as.

$I_{amb,diff} = K_d \cdot I_a$, where I_a = intensity of ambient light

If surface is exposed to a point source then

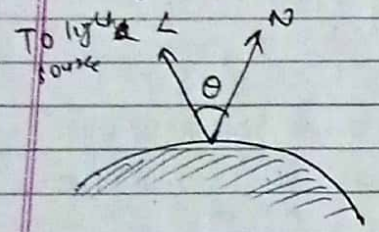
intensity of diffuse reflection can be calculated by using Lambert's Cosine law.

Lambertian Cosine law:

→ If N is the unit normal vector to a surface and L is the unit direction vector to the point light source from a position on the surface then

$\cos\theta = N \cdot L$ (Lambertian Cosine law)

and the diffuse reflection equation for a single point source illumination is:



$I_{diff} = k_d I_l \cos\theta = k_d I_l (N \cdot L)$

I_l : the intensity of the light source

k_d : diffuse reflection coefficient

N: the surface normal (unit vector)

L: the direction of light source (unit vector)

When $\cos\theta$ is
+ve: surface is illuminated
-ve: light source is behind the surface.

We can combine the ambient and point source intensity calculations to obtain an expression for the total diffuse reflection. Thus, we can write the total diffuse reflection equation as:

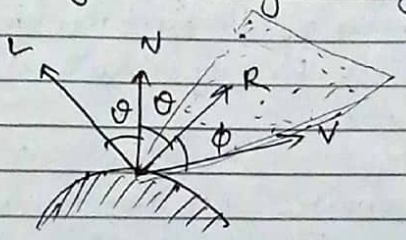
$I_{diff} = k_d \cdot I_a + k_d I_L (N \cdot L)$

ambient — diffuse for single point

Specular reflection and phong model:

→ When we look at an illuminated shiny surface, such as polished metal, we see a highlight or bright spot, at certain viewing direction. Such phenomena is called specular reflection.

→ It is the result of total or near total reflection of incident light in a concentrated region around "specular reflection angle = angle of incidence."



n_s -> specular reflection parameter.

-> For ideal reflector (perfect mirror), incident light is reflected only in the specular reflection direction.

V & R concide ($\phi = 0$).

-> Shiny surface have narrow ϕ and dull surface wider ϕ.

-> An empirical model for calculating specular-reflection range developed by Phong Bui Tuong-called Phong specular reflection model (or simply Phong model), sets the intensity of specular reflection proportional to

$\cos^{n_s}\phi$ $\quad$ $\cos^{n_s}\phi$ -> 0 to 90°.

$0 \le \phi \le \frac{1}{2}$

-> Specular reflection parameter n_s is determined by type of surface.

-> Intensity of specular reflection depends upon material properties of the surface and θ. Other factors such as the polarization and color of the incident light.

-> For monochromatic specular intensity variations can approximated by SR coefficient $W(\theta)$.

$I_{spec} = W(\theta)\, I_l \cos^{n_s}\phi$

for a glass, we can replace $W(\theta)$ with constant K_s.

$I_{spec} = K_s I_l \cos^{n_s}\phi$

$= K_s I_l (V \cdot R)^{n_s}$ since $\cos\phi = V \cdot R$

✓ Intensity Attenuation, Color Consideration, Transparency, Shadow.

✓ Surface Rendering Method (Surface Shading Method):

-> It is the process of applying illumination model to obtain the pixel intensities for all the projected surface portions in a scene.

-> Objects such as curved surfaces and Polyhedra (not curved surfaces) usually use Polygon-mesh approximation

-> Illumination model is applied to fill the interior of polygons.

Three widely used approaches

(i) Constant Intensity Shading Method (flat shading)
(ii) Gouraud Shading method (Intensity Interpolation)
(iii) Phong Shading Method (Normal vector Interpolation).

Constant Intensity Shading Method:

→ The fast and simplest model for shading rendering a polygon is constant intensity shading also called flat shading

→ In this approach, the illumination model is applied only once for each polygon to determine a single intensity value.

→ The entire polygon is then displayed with the single intensity value.

→ It doesn't provide realistic displaying.

Provides an accurate rendering for an object if all of the following assumptions are valid:

→ Polygon surface must be one face of a polyhedron and is not a section of a curved surface.

→ The light source is sufficiently far so that $\vec{N} \cdot \vec{L}$ is constant across the polygon face.

→ The viewing position is sufficiently far from the surface so that $\vec{V} \cdot \vec{R}$ is constant over the surface.

Algorithm:

(1) Divide polygon surface into polygon meshes.

(2) Determine surface unit normal vectors for each polygon

(3) Calculate intensity value for a point for each surface (usually at the center).

(4) Apply this intensity value to all the points of that surface.

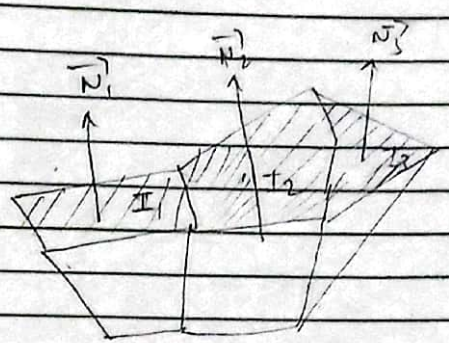


Fig: Flat shading

Drawback:

intensity discontinuity at the edges of polygons.