Date
Page

Scanned by CamScanner

Computes Graphics

→ It is a field of computes science that deals with generation of visual representation of data by ~~using~~ computer using specialized hardware & software.

→ It deals with creation, storage, and manipulation of graphical objects.

→ Graphical objects include static pictures into complex animation

→ It can also be defined as;

CG = Graphical Algorithms + Data Structure + Programming language

Graphical Algorithms:-

Procedures to generate graphical objects.

Eg:- Line Drawing Algorithm, Circle generating algo, Visible surface detection algorithm, Clipping Algorithms, Filling algorithms

Data Structures :-

→ Special data structures such as quad-tree and octrees.

Programming Language :-

Higher level programming language to create and manipulate graphical objects.

Eg :- Maya, C, C++, OPENGL

Scanned by CamScanner

Graphical System :-

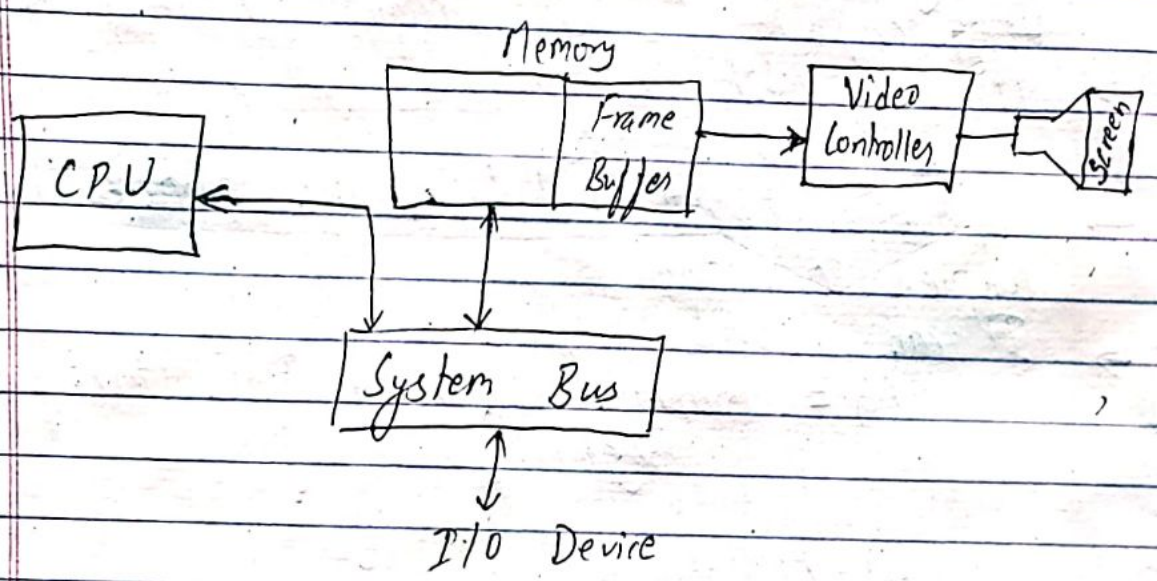


Fig :- Graphical System

CPU :-

It performs digitization process (determining appropriate color for each pixel), and stores in the memory area called Frame Buffer. And finally, video controller access the content of frame buffer and produces the graphics.

Scanned by CamScanner

Basic Concept

) Pixel :-> An unit of displaying area is called pixel.
→ Smallest ~~point~~ of picture is called pixel.
→ Discrete point of screen is called pixel.

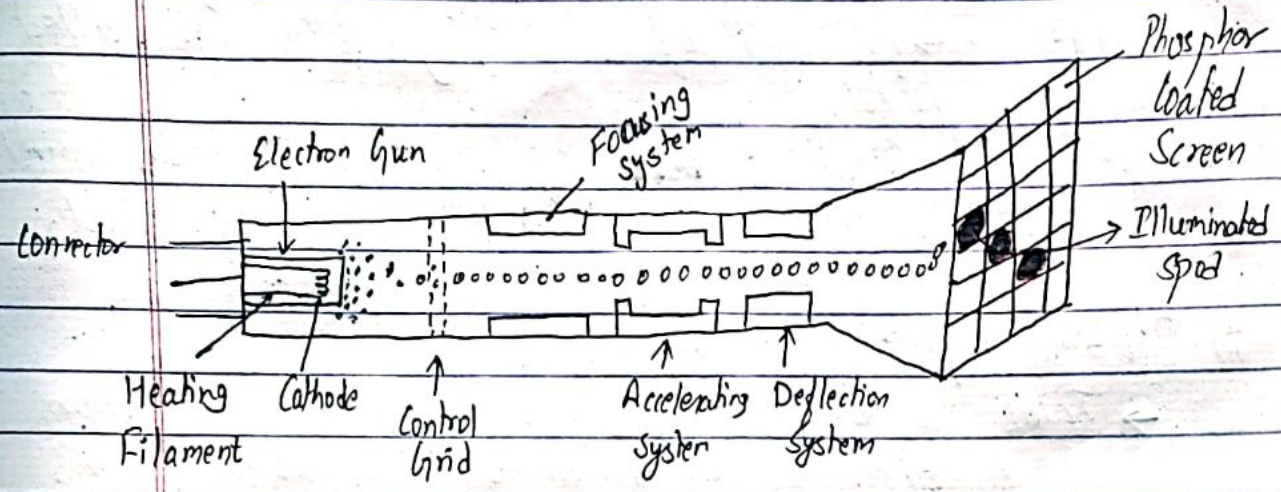
) Bit - depth / Color - depth
→ Total no. of bits required to store information of a pixel is called bit-depth or color depth.
→ ~ is measured in bit / pixel.
→ If 1 bit / pixel = $2^1 = 2$
8 bit / pixel = $2^8 = 256$

) Raster
→ A row of pixel is refered to as a scan line.
→ A collection of scan line is called ~.

Applications :-
→ Computer - Aided Design (CAD)
→ Entertainment
→ Medical
→ Education & Trainings
→ Visualization
→ Cartography
→ Graphical User Interface (GUI)
→ Simulation & Modeling

Scanned by CamScanner

Cathode Ray Tube (CRT)



Major components of CRT

i) Electron Gun
It is a source of an electron that contains cathode and heating filament.

ii) Control Grid
It controls the emission of electron i.e. it controls the voltage level of electron gun.

iii) Focusing System
It guaranties the convergence of electron beam on a screen point (pixel).

iv) Accelerating System
It accelerates electron beam towards the phosphor coated screen

Scanned by CamScanner

Deflection System
It controls the scanning (displaying) of electron beam both horizontally and vertically.

Working Mechanism of CRT

Electron gun emmits electron and when beam of electron strikes the phosphor coated screen it generates a pattern of illuminated spots which forms images on the screen.

To maintain display of the graphical object electron beam is redirected back over the same points hence also called Refresh CRT.

Properties of CRT

1) Pessistence :-
→ Time it takes to decay $\frac{1}{10}$ of original intensity of electron beam is called ~,
→ A phosphor with lower pessistance rate requires higher refresh rate
→ A phosphor with higher pessistance rate requires lower refresh rate.

2) Refresh Rate
→ The total no. of times a screen is refreshed in 1 second is called ~,
→ It is measured in cycles/sec or hertz.

Scanned by CamScanner

iii) Resolution

→ The total no of pixels that can be drawn on an output device (screen) without any overlapping is called ~.

→ It is represented by width x height
where width → tot no of pixel columns
height → tot. no. of pixel row or scanline

→ A display device with ~ 1024 x 1024 means there are 1024 width and height respectively.

iv) Aspect Ratio

→ The ratio between width and height is ~

$$AR = \frac{width}{height}$$

A system with 1280 x 1024 pixels and width 12 bits per pixel. Calculate the size of frame buffer in MB ?

Sol/

Here; Resolution = 1280 x 1024
Bit - depth = 12 bit/pixel

Size of frame buffer = 1280 x 1024 x 12 bits

$$= \frac{\quad}{8 \times 1024 \times 1024} \text{ MB}$$

Scanned by CamScanner

Consider a system having 12 inch by 10 inch screen with resolution of 100 pixels / inch. in each direction. Calculate the size of frame buffer if 6 bit / pixel is used ?

Sol^n

The dimension of screen is 12 x 10 inch with resolution of 100 pixel / inch, Then,

total no. of pixel is 12x100 x10x100 pixels

The bit depth per pixel is 6 bit / pixel. Then

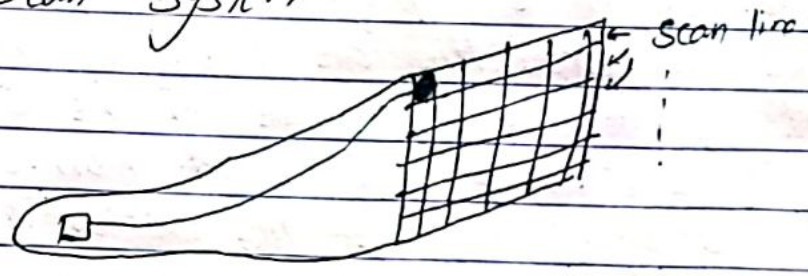
size of frame buffer

$$= \frac{12 \times 100 \times 10 \times 100 \times 6}{8 \times 1024 \times 1024} \text{ MB}$$

$$= 0.86 \text{ MB}$$

Types of Refresh CRT's

i) Raster-scan System



→ In this type of refresh CRT, picture definition is stored in frame buffer and content of frame buffer is accessed by video controller.

→ Displaying of graphical objects always starts from top-left corner of the screen.

→ Electron gun illuminates one pixel at a time from top to bottom.

→ Each scan-line is processed from top to bottom consequent

→ If 1 bit/pixel is used, frame buffer is called bit map and if multiple bits per pixel is used that frame buffer is called pix-map.

→ There are two-types of raster scan system :-

* Non-interlaced raster scan system
* Interlaced raster scansystem

Non-Interlaced Raster Scan System

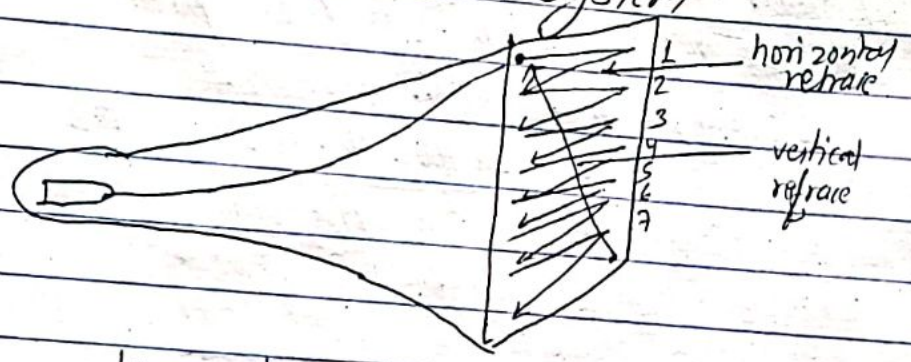


fig:- Non-interlaced RSS

Graphical object is displayed in one pass ie both odd and even scan lines are displayed. The process of returning to the left side of the screen after displaying a scan-line is called horizontal retrace. After displaying a frame, the electron beam returns to the top-left corner of the screen to begin the next frame which is called vertical retrace. NB)

Scanned by CamScanner

Interlaced Raster Scan System

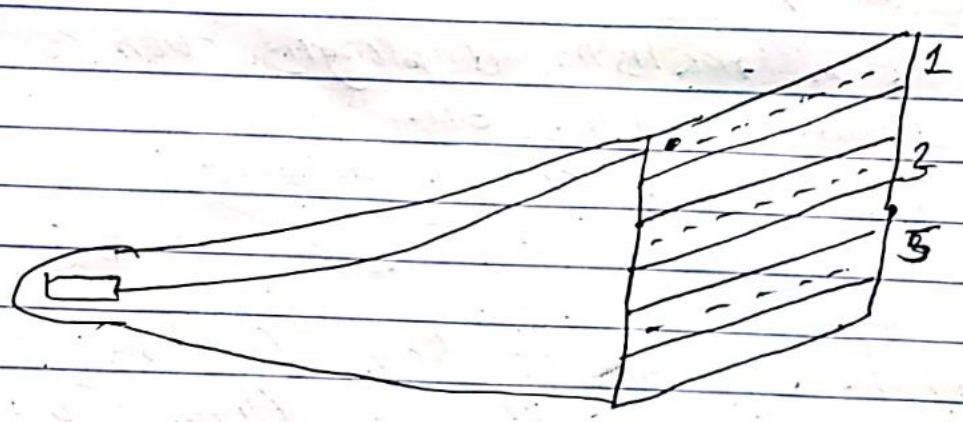


fig:- Interlaced RSS

→ It follows interlaced refresh procedure i.e. at first displays odd scan lines from top to bottom and after vertical retrace, even scan lines are displayed.
→ Graphical object is displayed in two passes for odd and even.

) Consider a RSS with resolution 1024 × 1024. If the video controller refreshes the screen at the rate of 50 frames/second. How many pixels should be accessed per second and what is the access-time per pixel of the system.

Sol:-

Here :- Resolution = 1024 × 1024 pixels

∴ No. of pixels in 1 frame = resolution = 1024×1024 pixel

Refresh rate = 50 frames/sec

No. of pixels accessed in 1 second = 1024 × 1024 × 50 pixel

= 52428800 pixel

∴ Access time per pixel = $\frac{1}{52428800}$ = 1.90×10^{-8} sec per pixel

Scanned by CamScanner

Application of Raster Scan System

→ It is used in realistic displaying such as home television system, printers etc.

→ It can not produce smooth lines.

Refresh-rate of Raster Scan System

→ The refresh rate of RSS lies between 30 to 60 frames/sec

Q) How many time spend scanning across each row of pixels during screen refreshed or a RSS with resolution 1280 x 1024 and refresh rate 60 frame/sec?

Sol/

Here,

Resolution = 1280 x 1024 pixels

Refresh rate = 60 frames/sec

Total no. of rows = 1024

Then,

Time spend scanning across each row of pixels;

$$= \frac{1}{60 \times \cancel{1280}\ 1024}$$

$$= \frac{1}{61440} \text{ sec}$$

Scanned by CamScanner

Q) How long would it take to load 640 x 480 frame buffer with 12 bit/pixel if 10^5 bit can be transferred in one sec.

Sol/

Here,

No. of pixels = 640 x 480

bit-depth = 12 bit/pixel

Transfer-rat = 10^5 bit/sec

Total no. of bits required to store = 640 x 480 x R

10^5 bits can be transferred in 1 sec

1 bit can be transferred in $\frac{1}{10^5}$ sec

640 x 480 x 12 bits can be transferred in

$\frac{1}{10^5} \times 640 \times 480 \times 12$ sec

= 36.86 sec

Scanned by CamScanner

Random Scan System

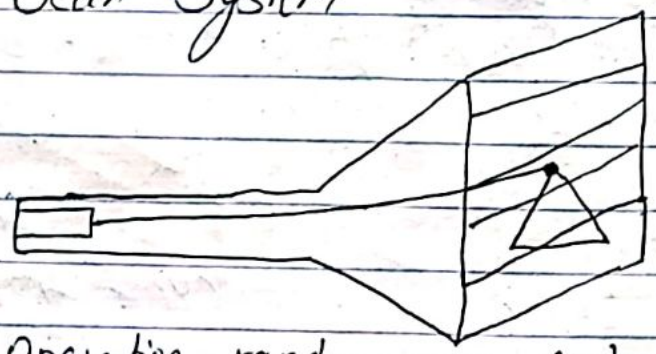


fig:- Operation random-scan system

→ In this type refresh CRT, electron beam is directed only towards that part of the screen where graphical object is to be drawn

→ Sometimes also referred to vector-scan system because it produces an entire line segment at a time

→ ~ is resolution independent, hence produces smooth lines.

→ Frame buffer contains set of line drawings commands and refer to as a display list or file.

→ The refresh rate of ~ is 30 – 50 fps.

Color CRT's

→ There are two common methods to displays with CRT :-

i) Beam - penetration method

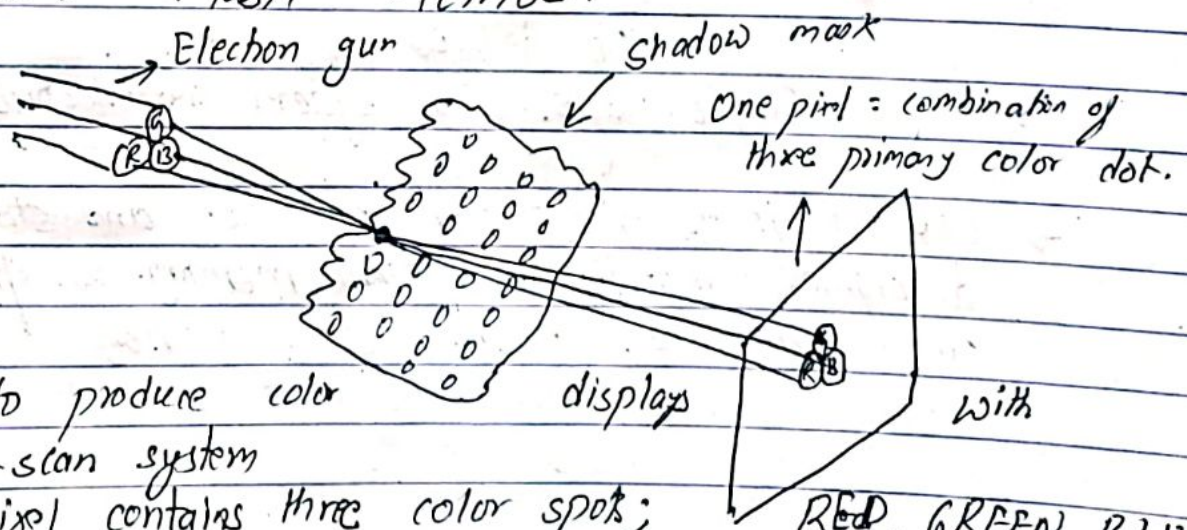
ii) Shadow mask method

Scanned by CamScanner

Beam Penetration Method

- It is used to produce color with random-scan system
- The color produced on the screen depend upon how fas electron beam penetrates the phosphor coated screen.
- Two types of phosphor ; red & green are coated onto the screen.
- Red colored phosphor forms the outes layes and green colored phorphor forms the innes layes
- Slow electron beam excites only outes red layes, hence produces red color
- Fast electron beam penetrates outes red layes and excites innes green layes, hence only green color is produced.
- At intermediate beam speed two additional color are produce ; orange & yellow

Shadow Mask Method.



- Used to produce color displays with raster-scan system
- One pixel contains three color spots; RED, GREEN, BLUE
- Three electron guns are used ; one for each color spot.
- Color dots are arranged in precise geometric pattern.

Scanned by CamScanner

→ * Delta pattern :-

* Stripe pattern :-

→ A thin metal plates with fine holes called shadow mask is insested between screen & electron guns

→ Shadow mask insures electron guns ~~only~~ excites only associated color.

→ It produces much wides range of colors than beam penetration color.

R, G, B : 0 - 255

Apprchitecture of Raster-scan System

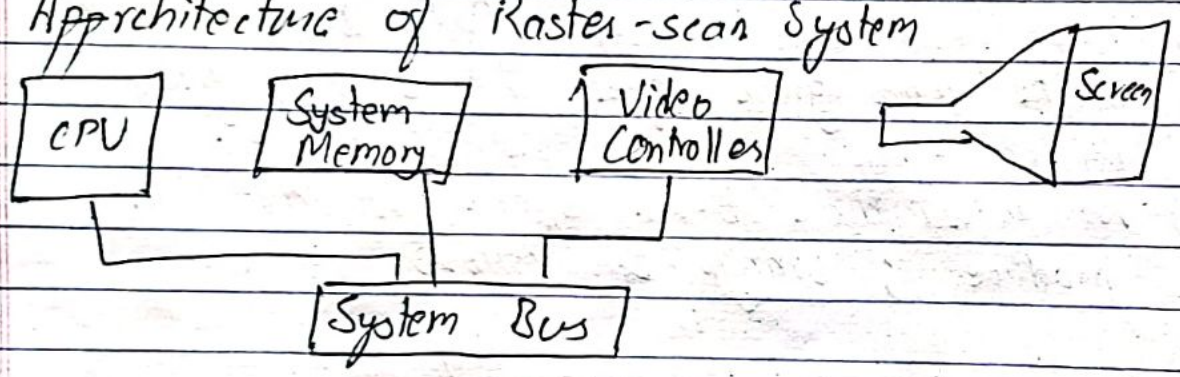


fig:- Simple Raster-scan system Architecture

→ CPU performs digitization operation and stores objects description randomly in system memory. i.e. frame buffer can be anywhere in the system memory.

→ Video controller retrive the contains of frame buffer via system bus and sets the intensity of each pixel on the screen.

→ It requires higher access time.

Scanned by CamScanner

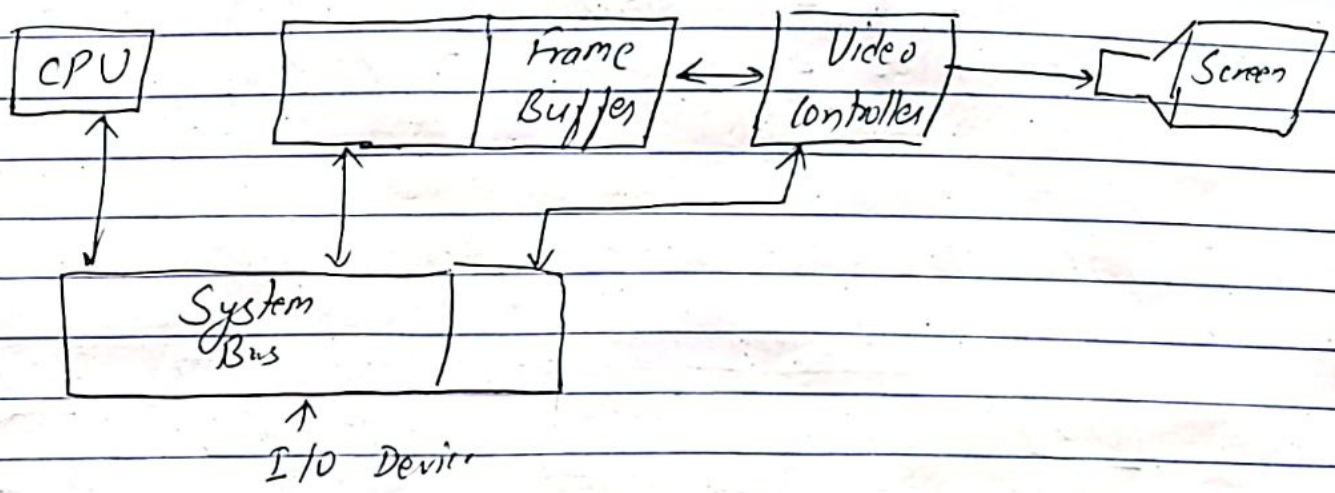
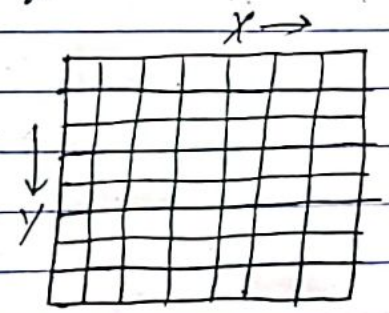


fig :- General Raster-screen system Architecture in which frame buffer is dual ported for video controller.

Working Mechanism (refresh-operation) of video controller



→ In raster-scan system display area is treated as 2D co-ordinate system.

→ Two registers : X-register and Y-register are used

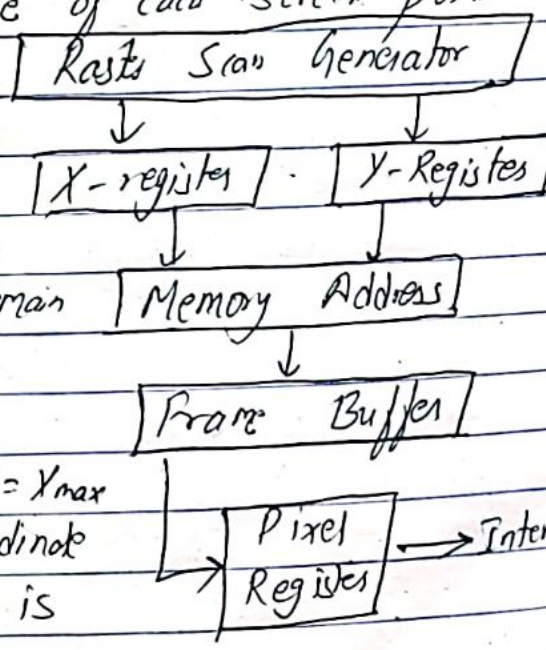
→ X - register stores x-coordinate system and Y- register stores Y-coordinate value of each screen point

→ Initially X-register is set to '0' and Y - register is set to '0'

→ The intensity associated with pixel '0,0' is retrieved from frame buffer.

→ For the next pixel Y-coordinates remain same & x-coordinate is simply incremented by 1.

→ This process is repeated until $X = X_{max}$ For the next scan line x-coordinate is reset to '0' & Y co-ordinate is



Scanned by CamScanner

incremented to 1 is (0,1)

→ This process is repeated until $y = y_{max}$

Architecture of Random Scan System

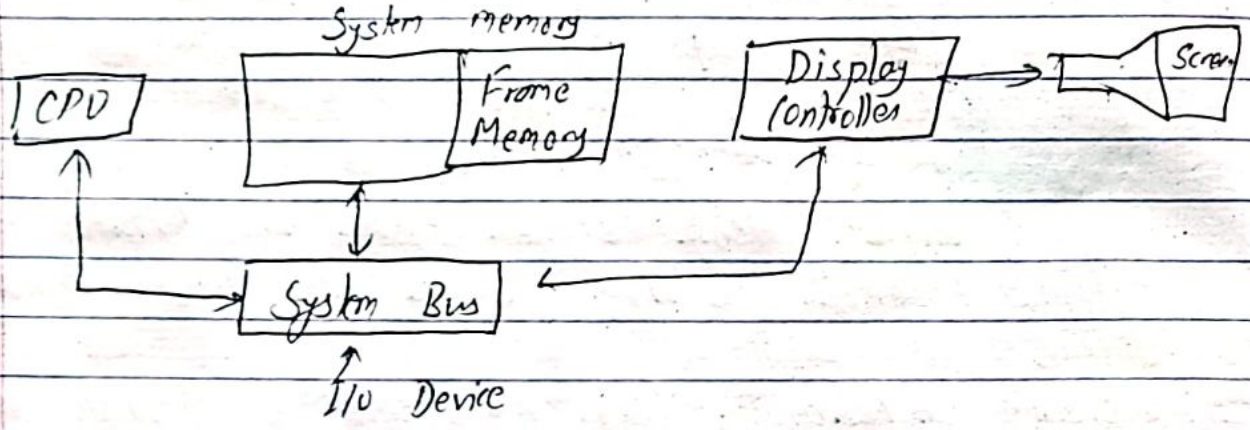


fig:- Architecture of Random Scan System

→ Additional Processor called display controller is used to read commands from frame buffer.

→ Display controller cycles through each command of frame buffer and produces the graphical

Scan Conversion Algorithm

→ Scan conversion is the process representing graphical object by the combination of discrete pixel.

→ Various scan-conversion algorithms exist, most widely used are :-

* Digital Differential Analyzer Algorithm (DDA)
* Bresenham's Line Drawing Algorithm
* Mid-point Circle Drawing Algorithm
* Mid-point Ellipse Drawing Algorithm

DDA

→ This algorithm is based on calculation of x-increment and y-increment to produce a line segment

→ One co-ordinate is incremented by 1, and corresponding other co-ordinate is determined for each intermediate pixel.

Algorithm

* Input endpoints (x_1, y_1) (x_2, y_2)
* Determine dx, dy as follows :

$$dx = x_2 - x_1$$
$$dy = y_2 - y_1$$

* If (abs (dx) > abs (dy))
  set step-length = abs (dx)
  else
  set step-length = abs (dy)
* Calculate x-increment & y-increment as follows.

$$x\text{-increment} = \frac{dx}{\text{step-length}}$$

$$y\text{-increment} = \frac{dy}{\text{step-length}}$$

Scanned by CamScanner

* $x = x_1$, $y = y_1$
  Plot pixel (x, y) ⇔ putpixel (x, y, color);
* for $k = 1$ to step-length
  $x = x + x\text{-increment}$
  $y = y + y\text{-increment}$
  plot pixel (round(x), round(y))
* STOP

Q) Trace DDA Algorithm for (4,2) and (10,4)

Sol^n/

Here, $X_1 = 4$ $\quad Y_1 = 2$

$X_2 = 10$ $\quad Y_2 = 4$

$dx = X_2 - X_1 = 10 - 4 = 6$

$dy = Y_2 - Y_1 = 4 - 2 = 2$

$\therefore dx = 6$, $dy = 2$

$\because abs(dx) > abs(dy)$

$\therefore$ step-length $= abs(dx) = 6$

Now,

$$X\text{-inc} = \frac{dx}{\text{step-length}} = \frac{6}{6} = 1$$

$$Y\text{-inc} = \frac{dy}{\text{step-length}} = \frac{2}{6} = \frac{1}{3} = 0.333$$

Now, successive intermediate pixels are calculated as follows

K	(X_{K+1}, Y_{K+1})	Plot (X_{K+1}, Y_{K+1})
1	(5 , 2.33)	(5,2)
2	(6 , 2.66)	(6,3)
3	(7 , 2.99)	(7,3)
4	(8 , 3.32)	(8 , 3)
5	(9 , 3.65)	(9 , 4)
6	(10 , 3.98)	(10, 4)

Hence, (4,2), (5,2), (6,3), (7,3), (8,3), (9,4), (10,4) are the req^d points

Scanned by CamScanner

Q) Trace DDA Algorithm for (-3, 8) (2, 4)

Sol/

Here, $X_1 = -3$ $Y_1 = 8$

$X_2 = 2$ $Y_2 = 4$

$dX = X_2 - X_1 = 2 - (-3) = 5$

$dY = Y_2 - Y_1 = 4 - 8 = -4$

$\therefore dX = 5, dY = -4$

$\because$ abs(dX) > abs(dY)

$\therefore$ step-length = abs(dX) = 5

Now

X-inc = dX / steplength = 5/5 = 1

Y-inc = dY / step length = -4/5 = -0.8

Now, successive intermediate pixels are calculated as follows

k	(X_{k+1}, Y_{k+1})	Plot (X_{k+1}, Y_{k+1})
1	(-2, 7.2)	(-2, 7)
2	(-1, 6.4)	(-1, 6)
3	(0, 5.6)	(0, 6)
4	(1, 4.8)	(1, 5)
5	(2, 4)	(2, 4)

Hence, (-3, 8) (-2, 7) (-1, 6) (0, 6) (1, 5) (2, 4) are reqd points

Scanned by CamScanner

Q) Digitize endpoint (-1, -4) & (-8, 2) using DDA algorithm

Sol/

Here, $x_1 = -1$ $\quad y_1 = -4$

$x_2 = -8$ $\quad y_2 = 2$

$dx = x_2 - x_1 = -8 - (-1) = -7$

$dy = y_2 - y_1 = 2 - (-4) = 6$

$\therefore dx = -7, \; dy = 6$

$\because abs(dx) > abs(dy)$

$\therefore$ steplength $= abs(dx) = 7$

Now

$$X\text{-inc} = \frac{dx}{\text{steplength}} = \frac{-7}{7} = -1$$

$$Y\text{-inc} = \frac{dy}{\text{steplength}} = \frac{6}{7} = 0.857$$

Now, successive intermediate pixels are calculated as follows

k	(X_{k+1}, Y_{k+1})	Plot (X_{k+1}, Y_{k+1})
1	(-2, -3.143)	(-2, -3)
2	(-3, -2.286)	(-3, -2)
3	(-4, -1.429)	(-4, 1)
4	(-5, -0.572)	(-5, 0)
5	(-6, 0.29)	(-6, 0)
6	(-7, 1.14)	(-7, 1)
7	(-8, 2)	(-8, 2)

Hence, (-1, -4), (-2, -3), (-3, -2), (-4, 1), (-5, 0), (-6, 0), (-7, 1) & (-8, 2) are reqd points

Scanned by CamScanner

Q) Digitize endpoints (4, -6) & (-2, -8) using DDA Algorith.

Sol/

Here,

$X_1 = 4$ $\quad Y_1 = -6$

$X_2 = -2$ $\quad Y_2 = -8$

$dX = X_2 - X_1 = -2 - 4 = -6$

$dY = Y_2 - Y_1 = -8 - (-6) = -2$

$\because$ abs (dX) > abs (dY)

$\therefore$ steplength = abs (dX) = 6

Now

$$X\text{-inc} = \frac{dX}{\text{steplength}} = \frac{-6}{6} = -1$$

$$Y\text{-inc} = \frac{dY}{\text{steplength}} = \frac{-2}{6} = -0.333$$

Now, successive intermediate pixels are calculated as follows

k	(X_{k+1}, Y_{k+1})	Plot (X_{k+1}, Y_{k+1})
1	(3, -6.33)	(3, -6)
2	(2, -6.67)	(2, -7)
3	(1, -7)	(1, -7)
4	(0, -7.33)	(0, -7)
5	(-1, -7.67)	(-1, -8)
6	(-2, -8)	(-2, -8)

Hence, (4, -6) (3, -6) (2, -7) (1, -7) (0, -7) (-1, -8) (-2, -8) are req^d points.

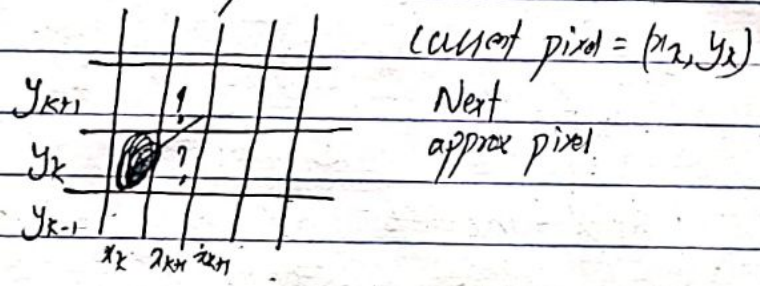
Scanned by CamScanner

Bresenham's Line Drawing Algorithm

→ An efficient & accurate scan conversion algorithm to produce line segment.

→ It only involves integer calculations hence produces smooth line compared to DDA i.e. it doesnot require round off operation

→ To determine successive intermediate pixels, it introduces a parameter called decision parameter.



1) * Input endpoint (x_0, y_0), $R(x_n, y_n)$

* Calculate Δx, & Δy as follows

$$\Delta x = |x_n - x_0|$$
$$\Delta y = |y_n - y_0|$$

* If $(x_n - x_0 < 0$ & $y_n - y_0 > 0)$ or $(x_n - x_0 > 0$ and $y_n - y_0 < 0)$

$a = -1$

else

$a = 1$

* If $(\Delta x \geq \Delta y)$

↳ if $(x_n < x_0)$ swap $(x_0, y_0) \longleftrightarrow (x_n, y_n)$

↳ plot first pixel (x_0, y_0)

↳ Determine initial decision parameter P_0 as

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

↳ Repeat Δx times starting from $k = 0$

if $(P_k < 0)$ { plotpixel $(x_k + 1, y_k)$

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

else

plotpixel (x_k+1, y_k+a)

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

else [means $\Delta y > \Delta x$ $\because |m| = \frac{\Delta y}{\Delta x}$]

↳ if $(y_n < y_0)$

- swap $(x_0, y_0) \longleftrightarrow (x_n, y_n)$

↳ Plot pixel (x_0, y_0)

↳ Determine initial decision parameter as;

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

↳ Repeat Δy times, starting from $k = 0$

if $(P_k < 0)$

- plotpixel (x_k, y_k+1)

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

else

plotpixel (x_k+a, y_k+1)

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

Scanned by CamScanner

Trace BSA or BLA algorithm for (10, 2) and (16, 5)

Sol^n/

Here, $x_0 = 10$, $y_0 = 2$

$x_n = 16$ $y_n = 5$

$\Delta x = |x_n - x_0| = |16 - 10| = 6$

$\Delta y = |y_n - y_0| = |5 - 2| = 3$

Now, $x_n - x_0 > 0$ and $y_n - y_0 > 0$

$\therefore a = 1$

$\because \Delta x \geq \Delta y$

$\therefore$ x-co-ordinate is simply incremented by 1

$\therefore$ ~~first~~ plot pixel $(x_0, y_0) = (10, 2)$

Now

initial decision parameter

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

$= 2 \times 3 - 6 = 0$

$\therefore P_0 = 0$

Successive decision parameter & intermediate pixel calculation are shown into table below

k	P_k	(x_{k+1}, y_{k+1})	
0	0	(11, 3)	$P_{0+1} = P_0 + 2\Delta y - 2\Delta x \Rightarrow P_1 = 0 + 2\times3 - 2\times6 = -6$
1	-6	(12, 3)	$P_{1+1} = P_2 = P_1 + 2\Delta y = -6 + 2\times3 = 0$
2	0	(13, 4)	$P_3 = P_2 + 2\Delta y - 2\Delta y = -6$
3	-6	(14, 4)	$P_4 = \quad = 0$
4	0	(15, 5)	$P_5 = \quad = -6$
5	-6	(16, 5)	~~P_6~~

k
0
1
2
3
4
5
6
7

Scanned by CamScanner

Date
Page

Trace BSA algorithm for (10,4) & (2,8)

Sol^n

Here, $x_0 = 10$, $y_0 = 4$

$x_n = 2$ $y_n = 8$

$\Delta x = |x_n - x_0| = |2-10| = 8$

$\Delta y = |y_n - y_0| = |8-4| = 4$

Now, $x_n - x_0 < 0$ and $y_n - y_0 > 0$

$\therefore a = -1$

$\because \Delta x \geq \Delta y$

$\therefore$ x - coordinate is simply increment by 1

~~$\therefore$ plot pixel~~ $\because x_n < x_0$

swap $(x_0, y_0) \leftrightarrow (x_n, y_n)$

ie $(x_0, y_0) = (2, 8)$ $(x_n, y_n) = 10, 4$

plot pixel $(x_0, y_0) = (2, 8)$

Now Initial parameter $(P_0) = 2\Delta y - \Delta x$

$= 2 \times 4 - 8 = 0$

Successive decision parameter & intermediate pixel calculation are shown into table below

P_k	(x_{k+1}, y_{k+1})	
0	(3, 7)	$P_1 = 0 + 2\times 4 - 2\times 8 = -8$
-8	(4, 7)	$P_2 = -8 + 2\times 4 = 0$
0	(5, 6)	$P_3 = -8$
-8	(6, 6)	$P_4 = 0$
0	(7, 5)	$P_5 = -8$
-8	(8, 5)	$P_6 = 0$
0	(9, 4)	$P_7 = -8$
-8	(10, 4)	$P_8 =$

Scanned by CamScanner

Trace BSA for (20,10) & (10,4)

Sol/

Here, $x_0 = 20$ $y_0 = 10$

$x_n = 10$ $y_n = 4$

Now, $\Delta x = |x_n - x_0| = |10-20| = 10$

$\Delta y = |y_n - y_0| = |4-10| = 6$

$\because x_n - x_0 < 0$ & $y_n - y_0 < 0$

$\therefore a = 1$

$\because \Delta x > \Delta y$ so x-coordinate is incremented by 1

$\because x_n < x_0$

swap $(x_0, y_0) \Longleftrightarrow (x_n, y_n)$

$(x_0, y_0) = (10, 4)$, $(x_n, y_n) = (20, 10)$

plot first pixel $(x_0, y_0) = (10, 4)$

Now

Initial parameter $(P_0) = 2\Delta y - \Delta x = 2\times6 - 10 = 2$

Intermediate pixel calculation and decision parameter calculation are shown in table below.

k	P_k	(x_{k+1}, y_{k+1})	
0	2	(11, 5)	$P_1 = 2 + 2\times6 - 2\times10 = -6$
1	-6	(12, 5)	$P_2 = -6 + 2\times6 = 6$
2	6	(13, 6)	$P_3 = 6 + 12 - 20 = -2$
3	-2	(14, 6)	$P_4 = -2 + 12 = 10$
4	10	(15, 7)	$P_5 = 10 + 12 - 20 = 2$
5	2	(16, 8)	$P_6 = 2 + 12 - 20 = -6$
6	-6	(17, 8)	$P_7 = -6 + 12 = 6$
7	6	(18, 9)	$P_8 = 6 + 12 - 20 = -2$
8	-2	(19, 9)	$P_9 = -2 + 12 = 10$
9	10	(20, 10)	

Scanned by CamScanner

Digitize a line with endpoints (4, 12) & (16, 6)

Sol/

Here, $x_0 = 4$ $y_0 = 12$

$x_n = 16$ $y_n = 6$

Now, $\Delta x = |x_n - x_0| = |16 - 4| = 12$

$\Delta y = |y_n - y_0| = |6 - 12| = 6$

$\because x_n - x_0 > 0$ & $y_n - y_0 < 0$

$a = -1$

$\because \Delta x > \Delta y$ so x - coordinate is incremented by 1

plot first pixel $(x_0, y_0) = (4, 12)$

Now Initial decision parameter $(P_0) = 2\Delta y - \Delta x$

$= 2 \times 6 - 12 = 0$

Intermediate pixel calculation & decision parameter calc^n are shown in the table below

k	P_k	(x_{k+1}, y_{k+1})	
0	0	(5, 11)	$P_1 = 0 + 2\times6 - 2\times12 = -12$
1	-12	(6, 11)	$P_2 = -12 + 2\times6 = 0$
2	0	(7, 10)	$P_3 = -12$
3	-12	(8, 10)	$P_4 = 0$
4	0	(9, 9)	$P_5 = -12$
5	-12	(10, 9)	$P_6 = 0$
6	0	(11, 8)	$P_7 = -12$
7	-12	(12, 8)	$P_8 = 0$
8	0	(13, 7)	$P_9 = -12$
9	-12	(14, 7)	$P_{10} = 0$
10	0	(15, 6)	$P_{11} = -12$
11	-12	(16, 6)	

Scanned by CamScanner

Digitize a line with end points (10, 18) and (15, 8).

Sol/

Here $x_0 = 10$ $y_0 = 18$

$x_n = 15$ $y_n = 8$

$\Delta x = |x_n - x_0| = |15 - 10| = 5$

$\Delta y = |8 - 18| = |(y_n - y_0)| = 10$

$\because x_n - x_0 > 0$ & $y_n - y_0 < 0$

$\therefore a = -1$

$\because \Delta x < \Delta y$

$\therefore$ y-coordinate is simply incremented by 1

$\because y_n > y_0$

swap $(x_0, y_0) \longleftrightarrow (x_n, y_n)$

$\therefore (x_0, y_0) = (15, 8)$, $(x_n, y_n) = (10, 18)$

plot pixel $(x_0, y_0) = (15, 8)$

Now

Initial decision parameter $P_0 = 2\Delta x - \Delta y$

$= 2 \times 5 - 10 = 0$

Successive decision parameter & intermediate pixel calculation are shown in table below

K	P_K	(x_{K+1}, y_{K+1})	
0	0	(14, 9)	$P_1 = 0 + 2\times5 - 2\times10 = -10$
1	-10	(14, 10)	$P_2 = -10 + 2\times5 = 0$
2	0	(13, 11)	$P_3 = 0 + 2\times5 - 2\times10 = -10$
3	-10	(13, 12)	$P_4 = \quad = 0$
4	0	(12, 13)	$P_5 = \quad = -10$
5	-10	(12, 14)	$P_6 = \quad = 0$
6	0	(11, 15)	$P_7 = \quad = -10$
7	-10	(11, 16)	$P_8 = \quad = 0$
8	0	(10, 17)	$P_9 = \quad = -10$

Scanned by CamScanner

-10 $(10, 18)$

Trace B.S.A for $(-2, 4)$ and $(4, 14)$

Sol

Here, $x_0 = -2$ $y_0 = 4$

$x_n = 4$ $y_n = 14$

$\Delta x = |x_n - x_0| = |4 - (-2)| = 6$

$\Delta y = |y_n - y_0| = |14 - 4| = 10$

$\because$ $y_n - y_0 > 0$ & $x_n - x_0 > 0$

$a = 1$

$\because$ $\Delta y > \Delta x$

$\therefore$ y-coordinate is simply incremented by 1

Now plot pixel $(x_0, y_0) = (-2, 4)$

Initial decision parameter $P_0 = 2\Delta x - \Delta y$

$= 2 \times 6 - 10$

$= 2$

Successive decision parameters & intermediate pixel calculation are shown in table below.

K	P_K	(x_{K+1}, y_{K+1})	
0	2	$(-1, 5)$	$P_1 = 2 + 2\times6 - 2\times10 = -6$
1	-6	$(-1, 6)$	$P_2 = -6 + 2\times6 = 6$
2	6	$(0, 7)$	$P_3 = 6 + 2\times6 - 2\times10 = -2$
3	-2	$(0, 8)$	$P_4 = -2 + 2\times6 = 10$
4	10	$(1, 9)$	$P_5 = 10 + 2\times6 - 2\times10 = 2$
5	2	$(2, 10)$	$P_6 = -6$
6	-6	$(2, 11)$	$P_7 = 6$
7	6	$(3, 12)$	$P_8 = -2$
8	-2	$(3, 13)$	$P_9 = 10$
9	10	$(4, 14)$	

Scanned by CamScanner

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

Trace & DDA and BSA for endpoint (4, 16) & (12, 10)

Soln

Here $(x_0, y_0) = (4, 16)$, $(x_n, y_n) = (12, 10)$

Using DDA

$dx = x_n - x_0 = 12 - 4 = 8$

$dy = y_n - y_0 = 10 - 16 = -6$

$\because$ abs (dx) > abs (dy)

steplength = abs (dx) = 8

Now

$x\text{-inc} = \frac{dx}{\text{steplength}} = \frac{8}{8} = 1$

$y\text{-inc} = \frac{dy}{\text{steplength}} = \frac{-6}{8} = -0.75$

Using BSA

$\Delta x = |x_n - x_0| = |12 - 4| = 8$

$\Delta y = |y_n - y_0| = |10 - 16| = 6$

$\because x_n - x_0 > 0$ & $y_n - y_0 < 0$

$\therefore a = -1$

$\because \Delta x > \Delta y$

x is incremented by 1

plot pixel $(x_0, y_0) = (4, 16)$

Now

Initial parameter $(P_0) = 2\Delta y - \Delta x$

$= 2 \times 6 - 8$

$= 4$

Successive intermediate pixel are calculated as

k	(x_{k+1}, y_{k+1})	Plot (x_{k+1}, y_{k+1})
1	(5, 15.25)	(5, 15)
2	(6, 14.5)	(6, 14)
3	(7, 13.75)	(7, 14)
4	(8, 13)	(8, 13)
5	(9, 12.25)	(9, 12)
6	(10, 11.5)	(10, 11)
7	(11, 10.75)	(11, 11)
8	(12, 10)	(12, 10)

$2\Delta y = 12$ $2\Delta x = 16$

k	P_k	(x_{k+1}, y_{k+1})	
0	4	(5, 15)	$P_1 = 4 + 12 - 16 = -0$
1	0	(6, 14)	$P_2 = -4 + 12 - 16 = 8$
2	8	(7, 14)	$P_3 = 4 + 12 - 16 = 4$
3	4	(8, 13)	$P_4 = 4 + 12 - 16 = 0$
4	0	(9, 12)	$P_5 = 0 - 4 + 12 = 8$
5	8	(10, 11)	$P_6 = 8 - 6 = 2$
6	2	(11, 11)	$P_7 = 2 + 12 - 16 = 4$
7	4	(12, 10)	

Scanned by CamScanner

Date
Page

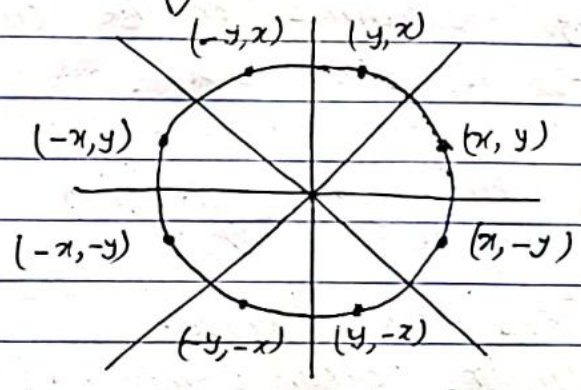
Circle

→ It is an object that is defined by a set of points that lie all at a equidistance r from center (x_c, y_c)

→ General circle equation can be used to represent a circle:

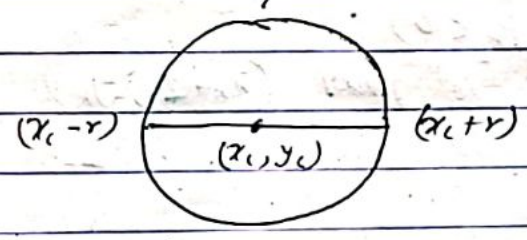
$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad \text{--- (1)}$$

Methods to generate circle



* Direct Method

In this method one-coordinate is incremented by 1 and it's corresponding other co-ordinate is determined using general circle equation.



$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

$$y - y_c = \pm\sqrt{r^2 - (x - x_c)^2}$$

$$\therefore y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

Here, x-co-ordinate is incremented by 1 starting from $x_c - r$ to $x_c + r$

Scanned by CamScanner

* Trignometric method

In this method x & y-coordinate can be calculated as follows

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

To determine x, y coordinate on the circumference θ is incremented from 0° to 2π

* Mid-Point Circle Drawing Algorithm

→ Input radius r, and center (x_c, y_c)

→ Obtain first point on the circumference by assuming circle is centered at (0,0)

i.e. $(x_0, y_0) = (0, r)$

→ Determine initial decision parameter as

$$P_0 = \begin{cases} 1 - r & \rightarrow [\text{ if } r \text{ is integer}] \\ 5/4 - r & \rightarrow [\text{ if } r \text{ is float value}] \end{cases}$$

→ Repeat following ; until $(x \geq y)$; starting from $k=0$

If $(P_k < 0)$

- plot pixel (x_k+1, y_k)
- set $P_{k+1} = P_k + 2x_{k+1} + 1$

else

- plot pixel (x_k+1, y_k-1)
- set $P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$

→ Determine symmetry point in other seven octants

i.e. for each

(x, y)	(y, x)
$(-x, y)$	$(y, -x)$
$(-x, -y)$	$(-y, -x)$
$(x, -y)$	$(-y, x)$

Scanned by CamScanner

→ For each (x, y) calculate actual point as

$$x = x + x_c$$
$$y = y + y_c$$

Digitize a circle described by equation $(x-5)^2 + (y-4)^2 = 16$

Sol^n

$$x_c = 5$$
$$y_c = 4$$
$$r = 4$$

let us assume that circle is centered at $(0,0)$

$\therefore (x_0, y_0) = (0, r) = (0, 4)$

Now, initial decision parameter

$$P_0 = 1 - r = 1 - 4 = -3$$

Successive decision parameter and point on the circumference are calculated in table below :-

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	P_{k+1}
0	-3	$(1, 4)$	2	8	$P_1 = P_0 + 2x + 1 = 0$
1	0	$(2, 3)$	4	6	$P_2 = 0 + 4 - 6 + 1 = -1$
2	-1	$(3, 3)$	6	6	$P_3 = -1 + 6 + 1 = 6$
~~3~~	~~6~~	~~$(4, 2)$~~	~~8~~	~~4~~	~~$P_4 = 6 + 8 - 4 + 1 = 1$~~

Now

(x, y)	symmetry points	Actual points i.e. $(x + x_c, y + y_c)$
	$(0, 4)$	$(5, 8)$
$0, 4$	$(0, -4)$	$(5, 0)$
	$(4, 0)$	$(9, 4)$
	$(-4, 0)$	$(1, 4)$

Scanned by CamScanner

(1,4)

(1, 4)
(-1, 4) , (4, 1)
(-1, -4) , (4, -1)
(1, -4) , (-4, -1)
(-4, 1)

(6, 8) , (9, 5)
(4, 8) , (9, 3)
(4, 0) , (1, 3)
(6, 0) , (1, 5)

(2,3)

(2, 3)
(-2, 3) , (3, 2)
(-2, -3) , (3, -2)
(2, -3) , (-3, -2)
(-3, 2)

(7, 7) , (8, 6)
(3, 7) , (8, 2)
(3, 1) , (2, 2)
(7, 1) , (2, 6)

Scanned by CamScanner

Date
Page

Digitize a circle with 4 unit and centre at (-2, 6)

Sol/

$x_c = -2$

$y_c = 6$

$r = 4$

Let us assumed that circle is centered at (0, 0)

$\therefore (x_0, y_0) = (0, r) = (0, 4)$

Now, initial decision parameter

$P_0 = 1 - r = 1 - 4 = -3$

Successive decision parameter and point on the circumference are calculated in table below

K	P_K	(x_{K+1}, y_{K+1})	$2x_{K+1}$	$2y_{K+1}$	P_{K+1}
0	-3	(1, 4)	2	8	$P_1 = -3 + 2 + 1 = 0$
1	0	(2, 3)	4	6	$P_2 = 0 + 4 - 6 + 1 = -1$
2	-1	(3, 3)	6	6	$P_3 = -1 + 6 + 1 = 6$

Now

x, y	Symmetry points	Actual points $(x + x_c, y + y_c)$
(0,4)	(0, 4)	~~(5, 8)~~ (-2, 10)
	(0, -4)	(-2, 2)
	(4, 0)	(2, 6)
	(-4, 0)	(-6, 6)
(1,4)	(1, 4), (4, 1)	(-1, 10), (2, 7)
	(-1, 4), (4, -1)	(-3, 10) (2, 5)
	(-1, -4), (-4, -1)	(-3, 2) (-6, 5)
	(1, -4) (-4, 1)	(-1, 2) (-6, 5)
(2,3)	(2, 3), (3, 2)	(0, 9), (1, 8)
	(-2, +3) (-3, 2)	(-4, 9), (1, 4)
	(-2, -3) (-3, -2)	(-4, 3), (-5, 4)
	(2, -3) (-3, 2)	(0, 3), (-5

Scanned by CamScanner

#) Digitize a circle described by equation
$(x-3)^2 + (y-5)^2 = 100$

Sol)

Comparing given eq° with $(x-x_c)^2 + (y-y_c)^2 = r^2$

$x_c = 3$

$y_c = 5$

$r = 10$

let us assume that circle is centred at $(0,0)$

∴ first pixel on the circle circumference $(x_0, y_0) = (0, r) = (0, 10)$

Now initial parameter $(P_0) = 1 - r = 1 - 10 = -9$

Successive decision parameters and pixel calculations are shown in the table below :-

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	
0	-9	(1, 10)	2	20	$P_1 = -9 + 2 + 1 = -6$
1	-6	(2, 10)	4	20	$P_2 = -6 + 4 + 1 = -1$
2	-1	(3, 10)	6	20	$P_3 = -1 + 6 + 1 = 6$
3	6	(4, 9)	8	18	$P_4 = 6 + 8 - 18 + 1 = -3$
4	-3	(5, 9)	10	18	$P_5 = -3 + 10 + 1 = 8$
5	8	(6, 8)	12	16	$P_6 = 8 + 12 - 16 + 1 = 5$
6	5	(7, 7)			

Now

(x, y)	Symmetry points	Actual points $(x+x_c, y+y_c)$
(1, 10)	(1, 10), (10, 1)	(3, 15), (13, 5)
	(-1, 10), (10, -1)	(2, 15), (13, 4)
	(-1, -10), (-10, -1)	(2, -5), (-7, 4)
	(1, -10), (-10, 1)	(4, -5), (-7, 5)
(2, 10)	(2, 10), (10, 2)	(5, 15), (13, 7)
	(-2, 10), (10, -2)	(1, 15), (13, 3)
	(-2, -10), (-10, -2)	(1, -5), [illegible]
	(2, -10), (-10, 2)	[illegible]

[illegible] we can obt[illegible]

Scanned by CamScanner

Date
Page

Q. Given a circle centered at (10, 10) and radius 5 units. Find all points in its first octant.

Sol^n

$(x_c, y_c) = (10, 10)$, $r = 5$

let us assume that circle is centered at (0, 0)

∴ first pixel on the circle circumference $(x_0, y_0) = (0, 5)$

Now, initial parameter $(P_0) = 1 - r = 1 - 5 = -4$

Successive decision parameter and pixel calculation are shown in table below

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	
0	-4	(1, 5)	2	10	$P_1 = -4 + 2 + 1 = -1$
1	-1	(2, 5)	4	10	$P_2 = -1 + 4 + 1 = 4$
2	4	(3, 4)	6	8	$P_3 = 4 + 6 - 8 + 1 = 3$
3	3	(4, 3)	8	6	$P_4 = 3 + 8 - 6 + 1$

Now

(x, y)	Symmetry points	Actual points $(x + x_c, y + y_c)$
(1, 5)	(1, 5)	(1 + 10, 5 + 10) = (11, 15)
(2, 5)	(-1, 5)	(2 + 10, 5 + 10) = (12, 15)
(3, 4)	(-1, -5)	(3 + 10, 4 + 10) = (13, 14)
(0, 5)		(0 + 10, 5 + 10) = (10, 15)

Find all points on its circumference of the circle centered at (3, 4) & radius 4 units.

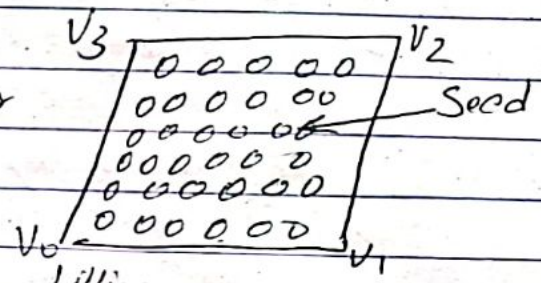
Digitize a circle described by a equation $(x + 4)^2 + (y - 2)^2 = 2$

Scanned by CamScanner

Date
Page

Scanned by CamScanner

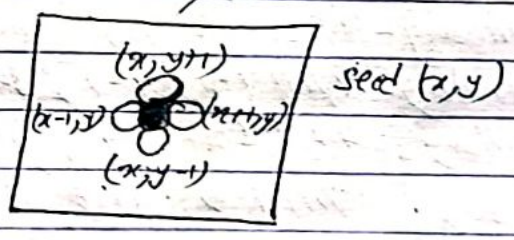
Polygon filling



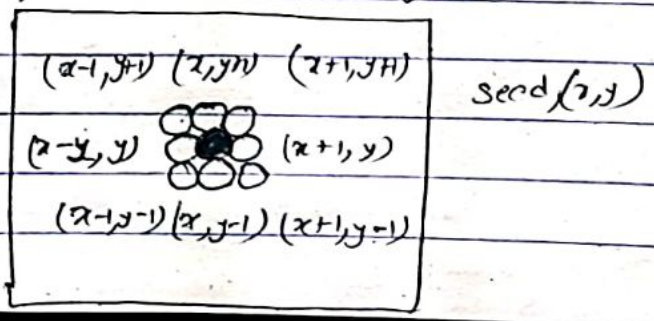
It is the process of assigning a color to all the pixels that lies inside the polygon other than background color. It is also reffered to as a area filling.
There are two approach to fill a polygon.

Seed Fill Approach

In this method, any arbitary pixel that is known to be inside of the polygon is selected as a seed.
A color that is to be assigned to all the pixels inside a polygon is called fill color
First seed pixel is assigned a fill colo, and its neighbouring pixels are checked
The neighbouring pixels are checked on the basis of either four connected or eight connected.
In four connected approach, neighbouring 4 pixels are checked i.e. it can produced 4 new seeds at a time



In 8 connected approach, 8 neighbousing pixels are checked



Scanned by CamScanner

→ There are two variation of seed fill approach

- Boundary Fill Algorithm

→ In this method; seed pixel is selected and its neighbouring pixels are checked

→ It first checks the color of seed pixel and if it is other than boundary color, or fill color then it is assigned a fill color

→ It requires monocolor polygon boundary which is other than fill color

→ If a polygon has multicolored boundary than all boundary must be assigned a single color which must be other than fill color

C-function for boundary fill algorithm

```
void boundary-fill (int x, int y, int fill-color, int b-color)
{
    int value = getpixel (x,y);
    if (value != fill-color && value != b-color)
    {   putpixel (x,y, fill-color);
        boundary-fill (x-1,y, fill-color, b-color);
        boundary-fill (x, y-1, fill-color, b-color);
        boundary-fill (x+1, y, fill-color, b-color);
        boundary-fill (x, y+1, fill-color, b-color);
    }
}
```

Scanned by CamScanner

- Flood Fill Algorithm

→ Arbitrary interior pixel is selected as a seed, and its neighbouring pixels are checked.

→ In this method, it is not mandatory to have monocolor boundary i.e boundary can have different color.

→ But it requires all interior pixels to have single color other than boundary color and fill color.

→ It starts filling from seed pixel and checks neighbouring pixels until all interior pixels have fill color

C - function for flood fill algorithm

```
void flood-fill4 (int x, int y, int fill-color, int old-color)
{
    int value = getpixel (x, y);
    if (value == old-color)
    {
        putpixel (x, y, fill-color);
        flood-fill4 (x-1, y, fill-color, old-color);
        flood-fill4 (x, y-1, fill-color, old-color);
        flood-fill4 (x+1, y, fill-color, old-color);
        flood-fill4 (x, y+1, fill-color, old-color);
    }
}
```

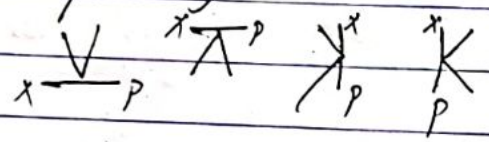
Scanned by CamScanner

- ~~Scan line~~ Inside - Outside Test for the polygon

→ A reference point X that is known to be outside of the polygon is taken and a point in question $P(x,y)$ is connected by straight line segment.

→ If a line segment makes even no. of intersection with polygon then $P(x,y)$ is outside of the polygon. otherwise $P(x,y)$ is inside of the polygon.

→ If following cases occurs then intersection count is 2.



→ In following cases intersection count is 1

Scan line Method

→ In this method all intersection points of scan-line with polygon are determined, and those pair of intersection are ~~called~~ sorted according to increasing order of x-coordinate.

→ Next, calculate all pixel points that lie on the path of pair of intersection i.e calculate intermediate points by using line drawing algorithm.

Scanned by CamScanner

Scan Line Method [Continued]

For each intermediate points that lie on the path of pair of intersection perform inside outside test
If point lies inside the polygon then that point is filled with fill color, otherwise that point is ignored
This process is repeated starting from $y = y_{min}$ i.e minimum value of (y-coordinate) polygon vertices to $y = y_{max}$

$$x_{k+1} = x_k - \frac{1}{m} \qquad y_{k+1} = y_k + 1$$

Scanned by CamScanner

Date
Page

Scanned by CamScanner

2D Transformations

→ 2D transformation are geometrical operations that are applied to object description (i.e. object vertices) to change position, size, orientation, shape

Rigid - Body Transformation

→ Transformations that only change position, orientation are called rigid body transformation.

→ Eg :- Translation, Rotation, Reflection

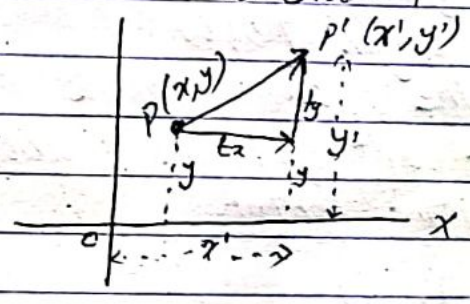
→ ~~Non - rigid body~~

Non Rigid Body Transformation

→ Can change shape, size or both are called non-rigid body transformation

Eg :- Sheasing, Scaling

Translation



→ It is a transformation that moves object in a straight line path

→ The amount displacement is given by a pair (t_x, t_y) called translation vector or shift vector

t_x = displacement along x-direction

t_y = " " y - "

→ Let $P(x,y)$ be the point that is to be translated

Scanned by CamScanner

by with translated vector $T(t_x, t_y)$ and $P'(x', y')$ be the resultant point

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

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

$$\therefore P' = T(t_x, t_y) + P$$

Rotation

→ A transformation that moves object in a circular path is called rotation

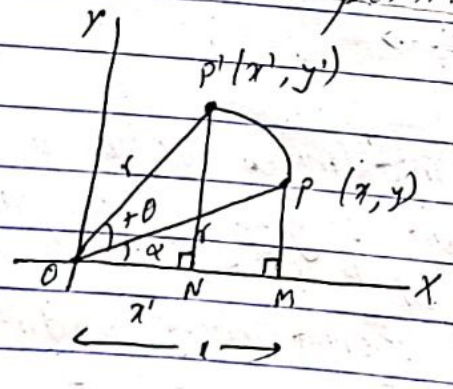
→ A point about to which object is rotate is called pivot point.

→ An angle that defines orientation change with respect to pivot point is called rotation angle.

→ There are two types of rotations

* Rotation about origin $(0,0)$

Let $P(x,y)$ be the point that is to be rotated by an angle $+\theta$ (i.e. anti-clockwise) about origin, and $P'(x', y')$ be resultant point.



Let α be the angle made by OP with OX

$\therefore \quad OP = OP' = r$

ΔOPM

$$\sin\alpha = \frac{PM}{OP} = \frac{y}{r} \qquad \cos\alpha = \frac{OM}{OP} = \frac{x}{r}$$

$$\therefore y = r\sin\alpha \quad \text{— (i)} \qquad \therefore x = r\cos\alpha \quad \text{(ii)}$$

$\Delta OP'N$

$$\sin(\theta+\alpha) = \frac{NP'}{OP'} = \frac{y'}{r}$$

$$\therefore y' = r\sin(\theta+\alpha) \quad \text{— (iii)}$$

$$\cos(\theta+\alpha) = \frac{ON}{OP'} = \frac{x'}{r}$$

$$\therefore x' = r\cos(\theta+\alpha) \quad \text{— (iv)}$$

$$y' = r\sin(\theta+\alpha)$$
$$= r[\sin\theta\cdot\cos\alpha + \cos\theta\cdot\sin\alpha]$$
$$= (r\cos\alpha)\sin\theta + (r\sin\alpha)\cos\theta$$
$$\therefore y' = x\sin\theta + y\cos\theta$$

$$x' = r\cos(\theta+\alpha)$$
$$= r[\cos\theta\cdot\cos\alpha - \sin\theta\sin\alpha]$$
$$= \cos\theta(r\cos\alpha) - (r\sin\alpha)\sin\theta$$
$$= x\cos\theta - y\sin\theta$$

In Matrix form

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

$\therefore P' = R(\theta).P$ where $R(\theta)$ is rotation matrix with angle θ about origin.

Scanned by CamScanner

When rotation is in clockwise direction i.e. $-\theta$.

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

In Matrix form

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

$$\Rightarrow P' = R(-\theta) \cdot P$$

Scanned by CamScanner

Date
Page

Scanned by CamScanner

Scaling

It is a transformation that alters the size of an object.

→ To perform scaling; we require scaling factor i.e. scale factor i.e. scale factor along x-axis and scale factor along y-axis.

→ let S_x & S_y be the scale factor along x- and y-direction respectively

i) If $S_x = S_y$ then it is called uniform scaling

ii) If $S_x \neq S_y$ then it is called differential scaling

Scaling w.r. to origin

→ let $P(x, y)$ be the point that is to be scaled w.r to origin by scale factor S_x & S_y & let $P'(x', y')$ be the resultant point then;

$$x' = x \cdot S_x$$

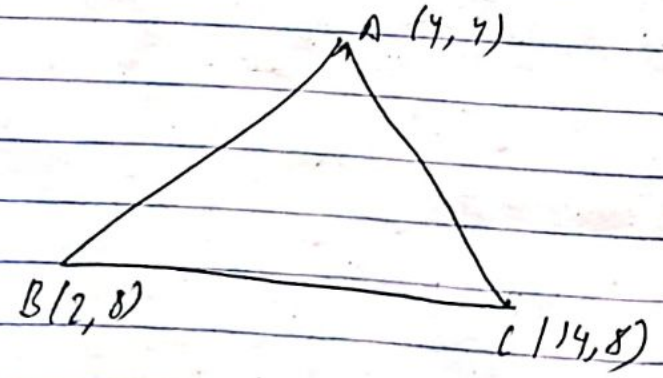
$$y' = y\, S_y$$

In Matrix form

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

$$\therefore P' = S(S_x, S_y) \cdot P$$

where $S(S_x, S_y)$ is Scaling matrix w.r.t origin



Scanned by CamScanner

Date
Page

⇒ Given a triangle A (4,4), B(2,8) & C(14,8). Scale this triangle by scaling factor S(2,3). Determine the final co-ordinate.

A (4,4) ⟶ A' (4 × 2, 4 × 3) = (8, 12)
B (2,8) ⟶ B' (2 × 2, 8 × 3) = (4, 24)
C (14,8) ⟶ C' (14 × 2, 8 × 3) = (28, 24)

Ans

Scanned by CamScanner

Date
Page

Scanned by CamScanner

Homogeneous Coordinates

It is a way of representing n-dimensional ~~cart~~ cartesian coordinate into n+1 projective coordinates i.e. n+1 tupels are used to represent a n-dimensional cartesian points.

When $n = 2$, then a point (x, y) is represented as (x_h, y_h, h) in homogeneous coordinates

$$\rightarrow \text{And} \quad \left.\begin{aligned} x &= \frac{x_h}{h} \\ y &= \frac{y_h}{h} \end{aligned}\right\} h = 1 \text{ (For 2D \& 3D)}$$

Eg: $(2, -4) \longrightarrow (2, -4, 1)$

$$\begin{bmatrix} 2 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}$$

Why homogeneous coordinates?

It provides a uniform way to represent a sequence of transformation by matrix multiplication and allows us to compute a single net (total) transformation of the sequence

Thus it ~~returns~~ reduces memory space requirement by eliminating intermediate steps.

Scanned by CamScanner

Homogeneous Coordinate Representation of Basic Transformation

1) Translation

→ let $P(x,y)$ be the point that is to translated by translation vector $T(t_x,t_y)$ and $P'(x',y')$ be the resultant point.

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$

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

In Matrix form

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

$$\therefore P' = T_H(t_x,t_y) \cdot P$$

where $T_H(t_x,t_y)$ is translation matrix in homogeneous coordinates

Translate a point $P(7,8)$ by translation vector $T(-2,4)$ so

Here, $P(7,8) = P(x,y)$

$P'(x',y') = ?$ & $T(t_x,t_y) = T(-2,4)$

$x' = x + t_x = (7 + -2, \cancel{8+4}) = 5$

$y' = y' + t_x = (8 + 4) = 12$

$P'(x',y') = (5,12)$

Again, Using Hom. Coor

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$
$$T_H(t_x,t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

Scanned by CamScanner

$\therefore P' = T_H(t_x, t_y) \cdot P$

$$= \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 \\ 12 \\ 1 \end{bmatrix}$$

2.) Rotation

→ Let $P(x,y)$ be the point that is to rotated about origin with rotation angle θ and $P'(x',y')$ be the resultant point

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P(x',y') \xrightarrow{H} P'(x',y',1)$$

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

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

In matrix form

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

$\therefore P' = R_H(\theta) \cdot P$

For inverse rotation

$P' = R_H(-\theta) \cdot P$

Scanned by CamScanner

3.) Scaling

Let $P(x,y)$ be the point that is to scaled w.r.t origin with scaling factor $S(S_x, S_y)$ and $P'(x',y')$ be the resultant point

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$

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

In Matrix form

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

$$\therefore P' = S_H(S_x, S_y) \cdot P$$

For inverse scaling

$$S_H\left(\frac{1}{S_x}, 1/S_y\right)$$

Translate a triangle $A(2,4)$, $B(2,10)$ and $C(10,8)$ with translation vector $(-2,4)$ and then rotate with angle $45°$ about origin. Finally, scale the object w.r.to origin with scaling factor $(2,4)$. Find the final coordinate of an objects

Sol^n

Here, $A(2,4)$, $B(2,10)$ & $C(10,8)$

$T(-2,4)$, $R(45°)$, $S(2,4)$

Now

Net-transformation $= S_H(2,4) \cdot R_H(45°) \cdot T(-2,4)$

$$= S_H(2,4) \cdot \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & & \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 4 \end{bmatrix}$$

Scanned by CamScanner

Date
Page

$$= S_{11}(2,4) \cdot \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & -2/\sqrt{2} + -4/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & -2/\sqrt{2} + 4/\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & -6/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & 2/\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2/\sqrt{2} & -2/\sqrt{2} & -12/\sqrt{2} \\ 4/\sqrt{2} & 4/\sqrt{2} & 8/\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -6\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{2} & 4\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore N_T = \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -6\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{2} & 4\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore A' = N_T \cdot A$$

$$= \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -6\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{2} & 4\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2\sqrt{2} + -4\sqrt{2} + -6\sqrt{2} \\ 4\sqrt{2} + 8\sqrt{2} + 4\sqrt{2} \\ 0 + 0 + 1 \end{bmatrix}$$

$$= \begin{bmatrix} -8\sqrt{2} \\ 16\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} -11.31 \\ 22.63 \\ 1 \end{bmatrix}$$

$$B' = N_T \cdot B$$

$$= \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -6\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{2} & 4\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 10 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2\sqrt{2} + -10\sqrt{2} + -6\sqrt{2} \\ 4\sqrt{2} + 10\sqrt{2} + 4\sqrt{2} \\ 0 + 0 + 1 \end{bmatrix}$$

$$= \begin{bmatrix} -14\sqrt{2} \\ 18\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} -19.79 \\ 25.45 \\ 1 \end{bmatrix}$$

$$C' = N_T \cdot C = \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -6\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{2} & 4\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} 10\sqrt{2} - 8\sqrt{2} - 6\sqrt{2} \\ 20\sqrt{2} + 16\sqrt{2} + 4\sqrt{2} \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} -4\sqrt{2} \\ 40\sqrt{2} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} -5.65 \\ 56.56 \\ 1 \end{bmatrix} \Leftarrow$$

Scanned by CamScanner

Given a triangle A(0,0) B(1,7) & C(9,2). Rotate the object by 60° about origin, then translate by T(-4,6) and scale w.r.t origin with S(4,2). What will be the new vertex value

Sol

Here, A(0,0), B(1,7) & C(9,2)

R(60°), T(-4,6) & S(4,2)

Now

Net-transformation $= S_M(4,2) \cdot T_M(-4,6) \cdot R_M(60°)$

$$= S_M(4,2) \cdot \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 60° & -\sin 60 & 0 \\ \sin 60 & \cos 60 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= S_M(4,2) \cdot \begin{bmatrix} \cos 60 & -\sin 60 & -4 \\ \sin 60 & \cos 60 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & -\sqrt{3}/2 & -4 \\ \sqrt{3}/2 & 1/2 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore N_T = \begin{bmatrix} 2 & -2\sqrt{3} & -8 \\ \sqrt{3} & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix}$$

$\therefore A' = N_T A$

$$= \begin{bmatrix} 2 & -2\sqrt{3} & -8 \\ \sqrt{3} & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -8 \\ 12 \\ 1 \end{bmatrix}$$

$$B' = \begin{bmatrix} 2 & -2\sqrt{3} & -8 \\ \sqrt{3} & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 7 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 - 14\sqrt{3} - 8 \\ \sqrt{3} + 7 + 12 \\ 1 \end{bmatrix} = \begin{bmatrix} -30.24 \\ 20.73 \\ 1 \end{bmatrix}$$

Scanned by CamScanner

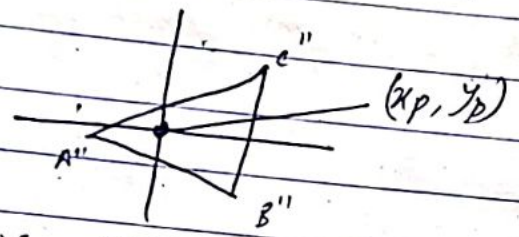
Date
Page

$$C' = \begin{bmatrix} 2 & -2\sqrt{3} & -8 \\ \sqrt{3} & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 9 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 18 - 4\sqrt{3} - 8 \\ 9\sqrt{3} + 2 + 12 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 3.071 \\ 29.58 \\ 1 \end{bmatrix}$$

Scanned by CamScanner

General Pivot Point Rotation

→ Let object is to be rotated about general pivot (x_p, y_p) and θ be the rotation angle



→ It follows sequence of transformations:

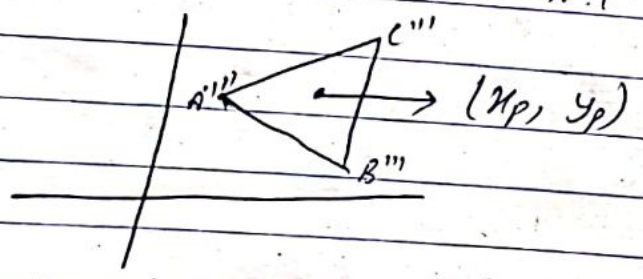
i) Translate such that pivot point coincides with coordinate origin

i.e apply $T(-x_p, -y_p)$

ii) Perform rotation about origin with an angle θ.

ie. apply $R(\theta)$

$$M_T = T_M(x_p, y_p) \cdot R_M(\theta) \cdot T_M(-x_p, -y_p)$$



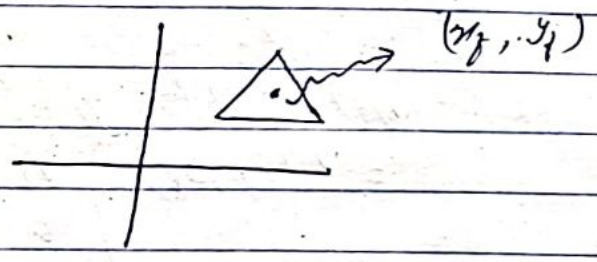
iii) Apply $R(\theta)$ (x_p, y_p)

$(-x_p, -y_p)$

Scanned by CamScanner

Scaling w.r.to arbitrary fixed point

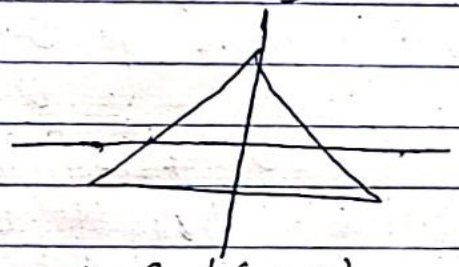
→ If an object is to be ~~selected~~ scaled with scale factor $S(S_x, S_y)$ w.r. to arbitrary fixed point (x_f, y_f), then it involves following sequence of transformations.



*) Original position

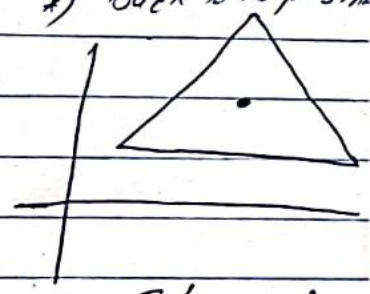
$\Rightarrow T_H(-x_f, -y_f)$

*) Scaling



$\Rightarrow S_H(S_x, S_y)$

*) Back to its position



$\Rightarrow T_H(x_f, y_f)$

$$N_T = T(x_f, y_f) \cdot S_H(S_x, S_y) \cdot T_H(-x_f, -y_f)$$

Scanned by CamScanner

Perform scaling w.r.to $(4,-2)$ with scale factor $(1/2, 1/4)$. Given points are $A(4,8)$, $B(16,8)$ & $C(8,14)$ #1

Soln

Here, $S_f = (1/2, 1/4)$ #2

Fixed point $(x_f, y_f) = (1/2, 1/4)$

Now

Net transformation $(M_t) = T(x_f, y_f)\, S_n(S_x, S_y) \cdot T(-x_f, -y_f)$ #3

$$= T(x_f, y_f) \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= T(x_f, y_f) \begin{bmatrix} 1/2 & 0 & -1/4 \\ 0 & 1/4 & -1/8 \\ 0 & 0 & 1 \end{bmatrix}$$ #1

$$= \begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & -1/4 \\ 0 & 1/4 & -1/8 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/2 & 0 & -1/4 + 1/2 \\ 0 & 1/4 & -1/8 + 1/4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1/4 & 1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

Date
Page

Prove that two successive translations are additive
i.e. $T_H(t_{x_2}, t_{y_2}) \cdot T_H(t_{x_1}, t_{y_1}) = T_H(t_{x_1}+t_{x_2}, t_{y_1}+t_{y_2})$

Prove that two successive rotations are additive
i.e. $R_H(\theta_2) \cdot R_H(\theta_1) = R_H(\theta_1+\theta_2)$

Prove that two successive scalings are multiplicative
i.e. $S_H(S_{x_2}, S_{y_2}) \cdot S_H(S_{x_1}, S_{y_1}) = S_H(S_{x_1} \cdot S_{x_2}, S_{y_1} \cdot S_{y_2})$

Soln

let $P(x,y)$ be the point that is to be translated with translation vectors $T_H(t_{x_1}, t_{y_1})$ and $T_H(t_{x_2}, t_{y_2})$ and let $P'(x', y')$ be the resultant point.

$\therefore P' \cancel{(x', y')} = \{T_H(t_{x_2}, t_{y_2}) \cdot T_H(t_{x_1}, t_{y_1})\} P$

Now

$T_H(t_{x_2}, t_{y_2}) \cdot T_H(t_{x_1}, t_{y_1})$

$$= \begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & t_{x_1}+t_{x_2} \\ 0 & 1 & t_{y_1}+t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

$= T_H(t_{x_1}+t_{x_2}, t_{y_1}+t_{y_2})$

★ Translate an object $A(4,-2)$, $B(6,6)$ & $C(-2,0)$ with translation vectors $(4,-3)$ and $(6,-8)$ Determine final coordinates of triangle.

Scanned by CamScanner

#2 Sol/

Let $P(x,y)$ be the point that is to be rotated with rotation vectors $R_H(\theta_1)$ and $R_H(\theta_2)$ and let $P'(x',y')$ be the resultant point.

$$\therefore \; P' = \{R_H(\theta_2) \cdot R_H(\theta_1)\} P$$

Now

$$R_H(\theta_2) \cdot R_H(\theta_1)$$

$$= \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_2 \cdot \cos\theta_1 - \sin\theta_2 \cdot \sin\theta_1 & -\cos\theta_2 \cdot \sin\theta_1 - \sin\theta_2 \cos\theta_1 & 0 \\ \sin\theta_2 \cdot \cos\theta_1 + \cos\theta_2 \cdot \sin\theta_1 & -\sin\theta_2 \cdot \sin\theta_1 + \cos\theta_2 \cdot \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= R_H(\theta_1+\theta_2)$$

Scanned by CamScanner

Scanned by CamScanner

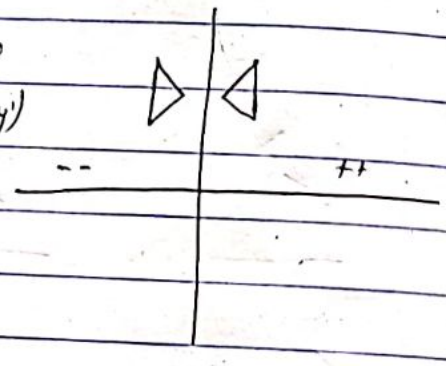
Reflection

In 2D following types of reflection are performed

i.) Reflection about a line where $x=0$ i.e. y-axis

→ y-coordinate remains unchanged and sign of x-coordinate is altered.

→ let $P(x,y)$ be point that is to reflected about y-axis, and $P'(x',y')$ be the resultant point



In Matrix form

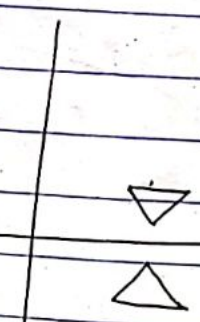
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\therefore P' = R_y \cdot P$

ii) Reflection about a line $y=0$ ie x-axis

→ x-coordinate remains unchanged and sign of y-coordinate is altered

→ let $P(x,y)$ be point that is to reflected about x-axis, and $P'(x',y')$ be the resultant point



In Matrix form

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

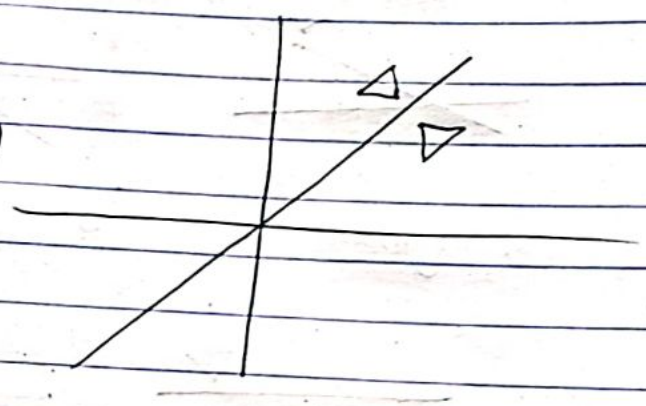
$\therefore P' = R_x P$

Scanned by CamScanner

) Reflection about a line $y = x$

→ value of x and y-coordinate are swapped

→ Let $P(X, Y)$ be point that is to reflected about $y = x$ and $P'(X', Y')$ be the resultant point.



In Matrix form

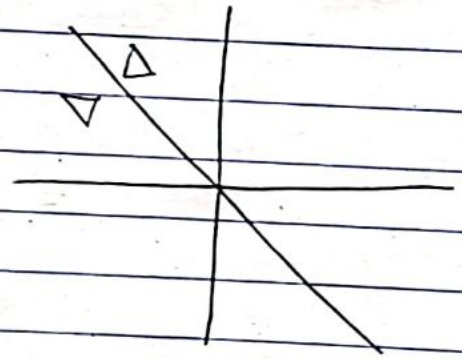
$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$P' = R_{y=x} \cdot P$$

) Reflection about a line $y = -x$

→ value of x and y-coordinate are swapped as well as sign of x, & y are changed.

→ let $P(X, Y)$ be the point that is to reflected about $y = -x$ & $P(X', Y')$ be the resultant point



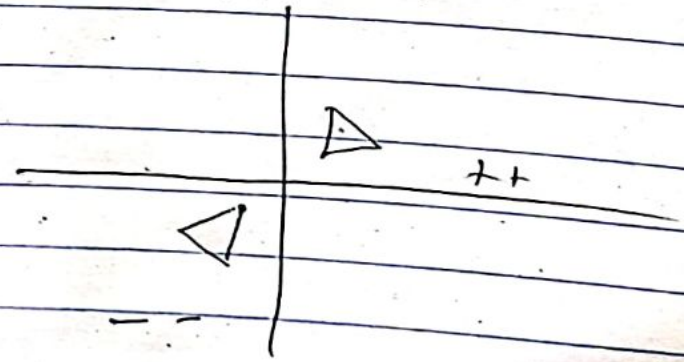
In Matrix form

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

) Reflection about a line $\perp^r$ to origin

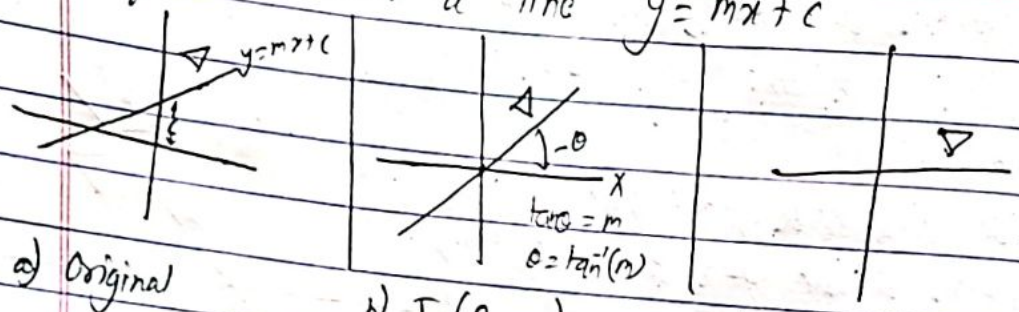
In Matrix form

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

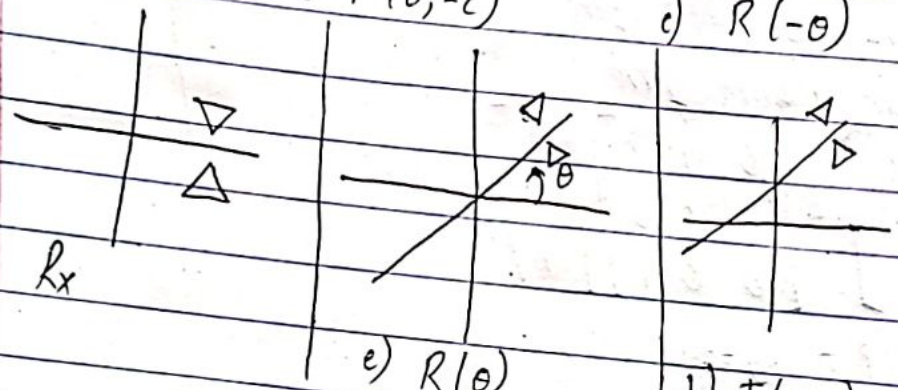


Scanned by CamScanner

vi) Reflection about a line $y = mx + c$



a) Original b) $T(0,-c)$ c) $R(-\theta)$



d) R_x e) $R(\theta)$ f) $T(0,c)$

$\therefore$ Net Transformation $(N_T) = T(0,c).R(\theta)R_x\,R(-\theta)T(0,-c)$

$= T(0,c).R(\theta).R_x \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$

$= T(0,c)\,R(\theta) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Scanned by CamScanner

Reflect an object $A(4,2)$, $B(3,-1)$, $C(2,8)$ about a line $y = 2x - 4$

Sol

Here, Given line is $y = 2x - 4$

$\therefore m = 2$, $c = -4$

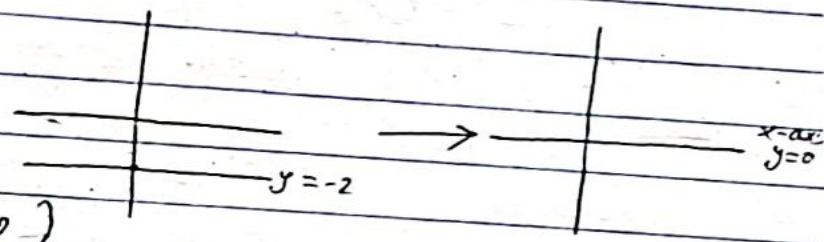
$\therefore \theta = \tan^{-1}(m) = \tan^{-1}(2) =$

$N_T = T(0,-4)\, R(\theta)\, R_x\, R(\theta)\, T(0,4)$

Reflect a triangle $M(4,4)$, $N(6,-8)$ & $O(-3,7)$ about $y = -2$

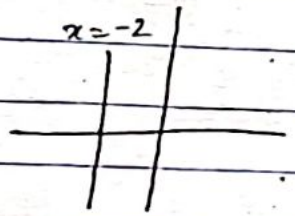
Sol

Here, $y = -2$



$N_T = T(0,-2)\, R_x\, T(0,2)$

Determine net transformation of reflection about $x = -2$



$N_T = T(-2,0)\, R_y\, T(2,0)$

Scanned by CamScanner

Date
Page

Scanned by CamScanner

Shearing :-

→ It is the transformation that distords the shape of an object
→ It is also called skewing
→ There are three types of shearing in 2D

i) X - shear

→ It preserves y-coordinate, and only x-coordinate is altered.
→ X-coordinate is shifted by an amount proportional to y-coordinate
→ let $P(x, y)$ be the point that is to be sheared along x-direction and $P'(x', y')$ be the resultant point.

$$X' = X + Sh_x \cdot Y$$
$$Y' = Y$$

where Sh_x is called shear parameter along x-direction

In Matrix form

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$P' = S(Sh_x) \cdot P$$

ii) Y- Shear

→ It preserves x-coordinate and only y-coordinate is altered
→ y-coordinate is shifted by an amount proportional to x-coordinate

$$X' = X$$
$$Y' = Y + Sh_y \cdot X$$

In Matrix form

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Scanned by CamScanner

iii) XY - Shear

→ Both X and Y - coordinate are altered

→ $x' = x + Sh_{xy} y$

$y' = y + Sh_{xy} x$

In matrix form

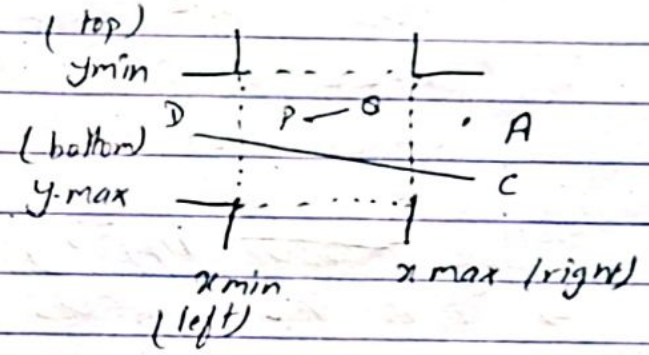
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_{xy} & 0 \\ Sh_{xy} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

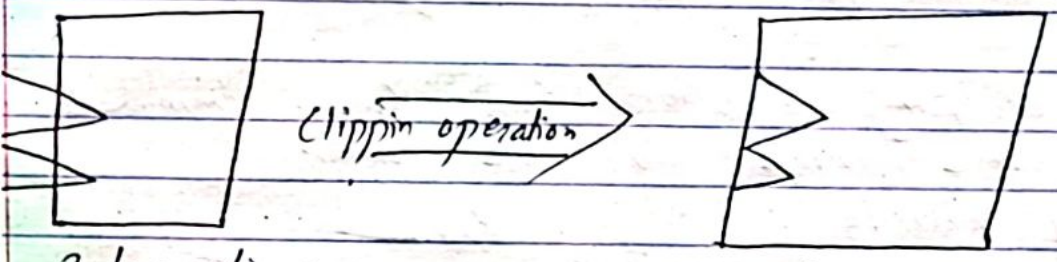
Scanned by CamScanner

Clipping

It is the process of discarding (cutting off) those parts of an object that lie outside of the region against which clipping operation is performed is a clipping window

A window can be specified using left boundary i.e. x-min, top boundary ie. y-min, right boundary i.e. x-max & bottom-boundary ie y-max.





Before clipping

After clipping

Application of Clipping in Computer Graphics

In CG clipping is necessary to discard unwanted graphics from the consideration to reduce memory space requirement and no. of calculation.

Animation

Video editing

Visible surface detection

Scanned by CamScanner

Types of Clipping

i) Point clipping

→ Given a clipping window X_{min}, X_{max}, Y_{min} & Y_{max} and coordinate point $P(x,y)$

→ $P(x,y)$ is accepted for display iff

$$X_{min} \le X \le X_{max}$$
$$Y_{min} \le Y \le Y_{max}$$

→ Otherwise point is clipped i.e. $P(x,y)$ is discarded

ii) Line ~~Reader~~ Clipping

→ It is the process of discarding those parts of a line that lie outside of a specified region (i.e. clipping window).

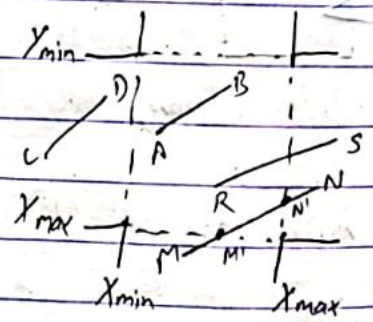
→ Given a clipping window (X_{min}, X_{max}, Y_{min} & Y_{max}), then there are three possible cases.

•) Completely visible

→ Line lies completely inside the window

→ Accept and display the line

→ E.g.: AB in the figure



•) Completely invisible

→ Line lies completely outside of the clipping window

→ Discard the line segment

→ Eg :- CD in figure

•) Partial Visibility

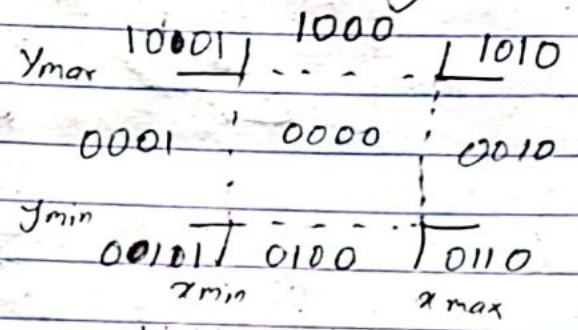
→ Part of the line segment lies inside of the clipping window

→ Clipping candidate → Eg : MN & RS

Scanned by CamScanner

TBRL

Cohen - Sutherland Line Clipping Algorithm



Entire coordinate system is divided into 9 reasons and each reasons is assigned a four bit pattern which is called region code or uprcode (out code).

T B R L

3rd bit 2nd bit 1st bit 0th bit

~~0th bit~~

→ For each vertex $P(x,y)$ region code is determined as follow

OP =
if $(x < x_{min})$ then
- left bit is 1

otherwise - left bit is 0

if $(x > x_{max})$
- right bit is 1

otherwise - right bit is 0

if $(y < y_{min})$
- ~~top bit~~ bottom bit is 1

otherwise - bottom bit is 0

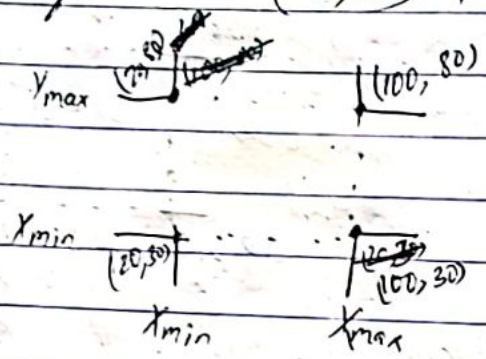
if $(y > y_{max})$
- top bit is 1

otherwise - top bit is 0

Scanned by CamScanner

TBRL

Given a clipping window upper ~~left~~ right corner (100, 80) & lower left corner (20, 30). Determine the region code.



i) A (30, 120), B (80, 40)
ii) C (25, 70) D (95, 40)

Region code for A = 1000
Region code for B = 0000
C = 0000
D = 0000

Scanned by CamScanner

Algorithm

1) Input window boundaries (x_{min}, x_{max}, y_{min}, y_{max}) and line endpoints P_1 (x_1, y_1) & P_2 (x_2, y_2)
2) Determine region code for P_1 & P_2 and store in OP_1, & OP_2 respectively
3) If $OP_1 \vee OP_2 = 0000$
   - line lies completely inside window
   - line is accepted for display
4) If $OP_1 \wedge OP_2 \neq 0000$
   - line lies completely outside window
   - Discard the line segment
5) Otherwise, find the intersection point (s) and determine region code for each intersection point and repeat until step 3 or 4 above is satisfied

Determination

i) Intersection with horizontal boundary

→ Let $P(x, y)$ be the intersection point with horizontal boundary {case, (a, b)}

$$y = y_{min} \text{ or } y_{max}$$

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

$$\therefore x = x_1 + \frac{1}{m}(y - y_1)$$

Where $m = \dfrac{y_2 - y_1}{x_2 - x_1}$

Scanned by CamScanner

ii) Intersection with vertical boundary

let $P(x, y)$ be the intersection point with vertical boundary. (case ©, ⓓ)

$$X = X_{min} \text{ or } X_{max}$$

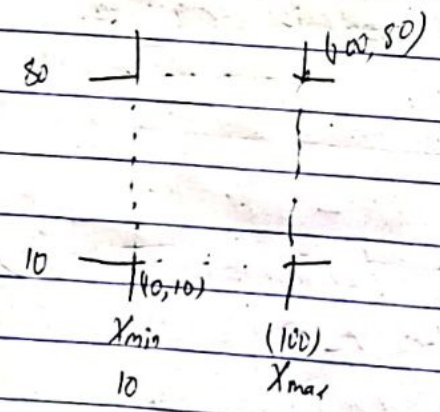
And

$$Y - Y_1 = m(X - X_1)$$

$$Y = Y_1 + m(X - X_1)$$

Given a clipping with lower left corner (10,10) & upper right corner (100, 80).

P_1 (6,20) & P_2 (80,90)



let OP_1 be the region code for P_1 and OP_2 be the region code for P_2

$OP_1 =$ $6 < 10 \rightarrow 1$ (L)

$6 > 100 \rightarrow 0$ (R)

$20 < 10 \rightarrow 0$ (B)

$20 > 80 \rightarrow 0$ (T)

$\therefore OP_1 = 0001$

Scanned by CamScanner

$OP_2 =$

$80 < 10 \rightarrow 0$ (L)

$80 > 100 \rightarrow 0$ (R)

$90 < 10 \rightarrow 0$ (B)

$90 > 80 \rightarrow 1$ (T)

$\therefore OP_2 = 1000$

Now,

$$OP_1 \vee OP_2 = \begin{array}{r} 0001 \\ 1000 \\ \hline 1001 \end{array} \neq 0000$$

$\therefore$ Line doesnot lie completely inside window

Again,

$$OP_1 \wedge OP_2 = \begin{array}{r} 0001 \\ 1000 \\ \hline 0000 \end{array}$$

$\therefore$ line doesnot lie completely outside window

Hence, we need to compute intersection points

Let $P(x,y)$ be the intersection point with vertical boundary

$\therefore x = x_{min} = 10$

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

$$y - 20 = \frac{y_2 - y_1}{x_2 - x_1}(10 - 6)$$

$$y = \frac{90 - 20}{80 - 6}(4) + 20 = \frac{70}{74} \times 4 + 20 = 23.78$$

let $M(x_n, y_n)$ be the intersection point with horizontal boundary

$\therefore y_n = y_{max} = 80$

And $y_n - y_1 = m(x_n - x_1)$

$$80 - 20 = \frac{90 - 20}{80 - 6}(x - 6)$$

$$\Rightarrow 60 \times \frac{74}{70} + 6 = x \Rightarrow x = 69.42$$

Scanned by CamScanner

Now region code for $P(x,y)$

$OP =$

$10 < 10 \rightarrow 0$ (L)

$10 > 100 \rightarrow 0$ (R)

$23.78 < 10 \rightarrow 0$ (B)

$23.78 > 80 \rightarrow 0$ (T)

And region code for $M(x_m, y_m)$

$OM =$

$69.42 < 10 \rightarrow 0$ (L)

$69.42 > 100 \rightarrow 0$ (R)

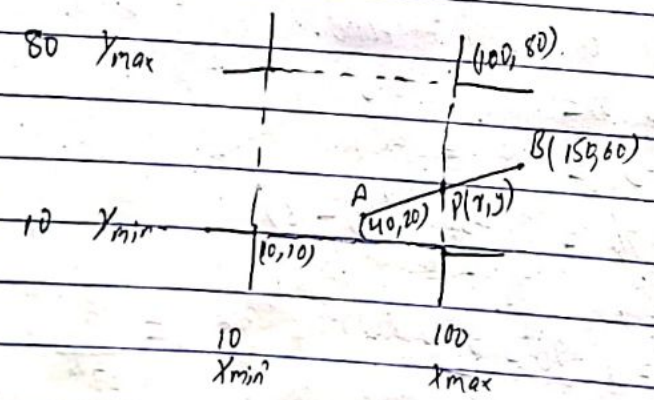
$80 < 10 \rightarrow 0$ (B)

$80 > 80 \rightarrow 0$ (T)

Now,

$$OP \vee OM = \begin{array}{r} 0000 \\ 0000 \\ \hline 0000 \end{array}$$

$\therefore$ line is ~~completely~~ clipped at $P(10, 23.78)$ & $M(69.42, 80)$



Region code for $A(40, 20)$

$OA =$

$40 < 10 \rightarrow 0$ (L)

$40 > 100 \rightarrow 0$ (R)

$20 < 10 \rightarrow 0$ (B)

$20 > 80 \rightarrow 0$ (T)

$\therefore OA = 0000$

$\therefore$ Point A lies completely inside

Scanned by CamScanner

Region code for B (150, 60)

OB =

$150 < 10 \rightarrow 0$ (L)

$150 > 100 \rightarrow 1$ (R)

$60 < 10 \rightarrow 0$ (B)

$60 > 80 \rightarrow 0$ (T)

$\therefore$ OB = ~~0010~~ 0010

NOW

OA $\vee$ OB = 0000
0010
0010 $\neq$ 0000

$\therefore$ line doesnot lie completely inside

OA $\wedge$ OB = 0000
0010
0000

$\therefore$ line doesnot lie completely outside

Hence, we need to determine intersection Point

Let P (x, y) be the intersection point as shown in fig then

$x = x_{max} = 100$

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

$y - 20 = \frac{60 - 20}{150 - 40}(100 - 40)$

$y = \frac{40}{110} \times 60 + 20 = 41.8181$

Region code for P (x, y) = (100, 41.8181)

OP =

$100 < 10 \rightarrow 0$ (L)

$100 > 100 \rightarrow 0$ (R)

$41.8181 < 10 \rightarrow 0$ (B)

$41.8181 > 80 \rightarrow 0$ (T)

$\therefore$ OP = 0000

Scanned by CamScanner

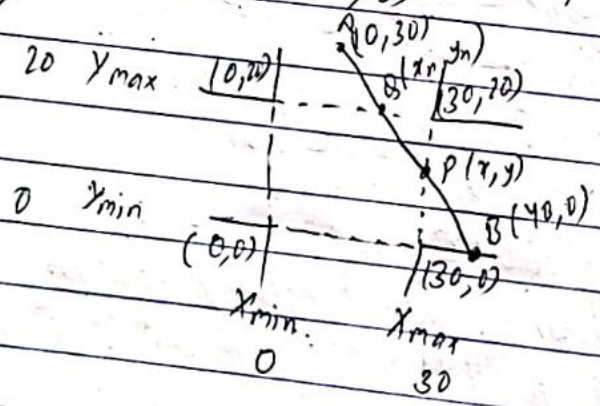
Now,

$$OA \vee OP = \begin{array}{r} 0000 \\ 0000 \\ \hline 0000 \end{array}$$

Hence, line AB is clipped at A(40,20) & P(100, 41.8)

Given a clipping window P(0,0), Q(30,0) R(30,20) & S(0,20). Use Cohen's - Sutherland line clipping Algorithm to determine the visible portion of the line. A(10,30) & B(40,0)

Sol



Region code for A (10, 30)

OA =

$10 < 0 \rightarrow 0$ (L)

$10 > 30 \rightarrow 0$ (R)

$30 < 0 \rightarrow 0$ (B)

$30 > 20 \rightarrow 1$ (T)

$\therefore$ OA = 1000

Region code for B (40, 0)

OB =

$40 < 0 \rightarrow 0$ (L)

$40 > 30 \rightarrow 1$ (R)

$0 < 0 \rightarrow 0$ (B)

$0 > 20 \rightarrow 0$ (T)

$\therefore$ OB = 0010

Scanned by CamScanner

Now, $OA \vee OB = \begin{array}{r} 1000 \\ 0010 \\ \hline 1010 \end{array} \neq 0000$

$\therefore$ Line doesnot lie completely inside

And

$OA \wedge OB = \begin{array}{r} 1000 \\ 0010 \\ \hline 0000 \end{array}$

$\therefore$ Line doesnot lie completely outside

Hence, we need to determine intersection points

Let $P(x, y)$ be the intersection point with vertical boundary

$\therefore x = x_{max} = 30$

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

$y - 30 = \frac{0 - 30}{40 - 10}(30 - 10)$

$y = \frac{-30}{30} \times 20 + 30 = 10$.

$\therefore P(x, y) = (30, 10)$

Let $Q(x_n, y_n)$ be the intersection point with horizontal boundary

$\therefore y_n = y_{max} = 20$

And $y_n - y_1 = m(x_n - x_1)$

$20 - 30 = \frac{0 - 30}{40 - 10}(x_n - 10)$

$x_n = \frac{-10}{-1} + 10$

$= 20$

$\therefore P_n(x_n, y_n) = (20, 20)$

Scanned by CamScanner

Region code for $P(x,y) = (30,10)$

$OP =$

$30 < 0 \rightarrow 0$ (L)

$30 > 30 \rightarrow 0$ (R)

$10 < 0 \rightarrow 0$ (B)

$10 > 20 \rightarrow 0$ (T)

$\therefore OP = 0000$

Region code for $Q(x_n, y_n) = (20,20)$

$OQ =$

$20 < 0 \rightarrow 0$ (L)

$20 > 30 \rightarrow 0$ (R)

$20 < 0 \rightarrow 0$ (B)

$20 > 20 \rightarrow 0$ (T)

$\therefore OQ = 0000$.

Again

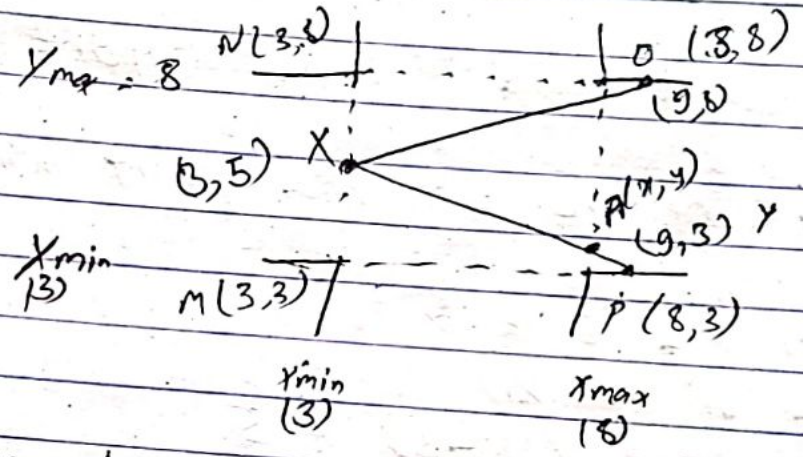
$$OP \vee OQ = \begin{array}{r} 0000 \\ 0000 \\ \hline 0000 \end{array}$$

$\therefore$ Line ~~PQ~~ AB is clipped at $P(30,10)$ & $Q(20,20)$

Scanned by CamScanner

Clip the line XY with endpoints X(3,5), Y(9,3) against clipping boundary MNOP M(3,3) N(3,8), O(8,8) and P(8,3) using cohen-Shutherland line clipping Algorithm.

Sol/



X = (3,5), Y (9,3)

Region code for X (3,5)

OX = 3 < 3 → 0 (L)
3 > 8 → 0 (R)
5 < 3 → 0 (B)
5 > 8 → 0 (T)

∴ OX = 0000

Region code for Y (9,3)

OY = 9 < 3 → 0 (L)
9 > 8 → 1 (R)
3 < 3 → 0 (B)
3 > 8 → 0 (T)

∴ OY = 0010

Now

OX ∨ OY = 0000
0010
0010 ≠ 0000

∴ line doesnot lie completely inside

$$OX \wedge OY = \begin{array}{r} 0000 \\ 0010 \\ \hline 0000 \end{array}$$

Line doesn't lie completely outside

Hence we need to find intersection point

let $A(x, y)$ be the intersection point with vertical boundary

ie $\therefore x = 8 = x_{max}$

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

$$y = \frac{3-5}{9-3}(8-3) + 5$$

$$= \frac{-2}{6}(5) + 5$$

$$= 5 - 10/6$$

$$= 3.3333$$

$\therefore A(x, y) = (8, 3.333)$

Region code for $A(8, 3.3333)$

$$OA = \begin{array}{lll} 8 < 3 \rightarrow 0 & (L) \\ 8 > 8 \rightarrow 0 & (R) \\ 3.3333 < 3 \rightarrow 0 & (B) \\ 3.3333 > 8 \rightarrow 0 & (T) \end{array}$$

$OA = 0000$

Now

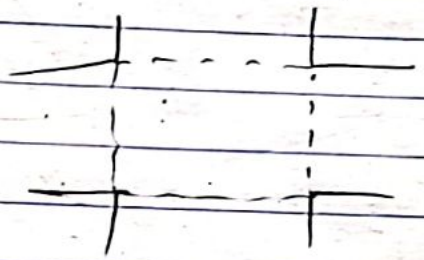
$$OX \vee OA = \begin{array}{r} 0000 \\ 0000 \\ \hline 0000 \end{array}$$

Hence, line XY is clipped at $X(5, 5)$ and $A(8, 3.333)$

Date
Page

Use Cohen - Sutherland Algorithm to clip the line with endpoint (50, 45) and (25, 15). Window lower left: 10,20 & upper right (30, 40)

Sol



Scanned by CamScanner

Scanned by CamScanner

Liang - Barsky Line Clipping Algorithm

→ An efficient line clipping algorithm which is based on parametric form of a line.

→ It reduces no. of intersections to be calculated hence performs much better than Cohen - Sutherland algorithm

→ let (x_1, y_1) and (x_2, y_2) be the endpoints then equation of a line passing from these two points can be written in parametric form as

$$x = x_1 + t\Delta x \quad \text{where } 0 \le t \le 1$$
$$y = y_1 + t\Delta y \qquad \Delta x = x_2 - x_1, \ \Delta y = y_2 - y_1$$

Then for any point (x, y) to be inside of the clipping window we must have

$$x_{min} \le x_1 + t\Delta x \le x_{max}$$
$$y_{min} \le y_1 + t\Delta y \le y_{max}$$

Eqn (2) involves four inequalities

$$x_{min} \le x_1 + t\Delta x \Rightarrow -t\Delta x \le x_1 - x_{min} \quad (L)$$
$$x_1 + t\Delta x \le x_{max} \Rightarrow t\Delta x \le x_{max} - x_1 \quad (R)$$
$$y_{min} \le y_1 + t\Delta y \Rightarrow -t\Delta y \le y_1 - y_{min} \quad (B)$$
$$y_1 + t\Delta y \le y_{max} \Rightarrow t\Delta y \le y_{max} - y_1 \quad (T)$$

Above inequalities can be written as

$$tp_i \le q_i, \quad \text{for } i = 1, 2, 3, 4$$

where

$$p_1 = -\Delta x, \quad q_1 = x_1 - x_{min}$$
$$p_2 = \Delta x_1, \quad q_2 = x_{max} - x_1$$
$$p_3 = -\Delta y_1, \quad q_3 = y_1 - y_{min}$$
$$p_4 = \Delta y_1, \quad q_4 = y_{max} - y_1$$

Scanned by CamScanner

→ If $P_1 = 0 \Rightarrow -\Delta x = 0 \Rightarrow \Delta x = 0 \Rightarrow x_2 - x_1 = 0 \Rightarrow x_2 = x_1$

→ If $P_2 = 0 \Rightarrow \Delta x = 0 \Rightarrow x_2 - x_1 = 0 \Rightarrow x_2 = x_1$

→ If $P_3 = 0 \Rightarrow -\Delta y = 0 \Rightarrow \Delta y = 0 \Rightarrow y_2 - y_1 = 0 \Rightarrow y_2 = y_1$

→ If $P_4 = 0 \Rightarrow \Delta y = 0 \Rightarrow y_2 - y_1 = 0 \Rightarrow y_2 = y_1$

If $(P_i = 0)$ then
- line segment is parallel to window boundary

If $(P_1 = 0)$
* If $(q_1 < 0)$
$\Rightarrow x_1 - x_{min} < 0 \Rightarrow x_1 < x_{min}$
- line lies outside of x_{min} boundary 1.)

If $(P_2 = 0)$ 2.)
* If $(q_2 < 0)$ 3.)
$\Rightarrow x_{max} - x_1 < 0 \Rightarrow x_{max} < x_1$
- line lies outside x_{max} boundary

If $(P_3 = 0)$ 4.)
* If $(q_3 < 0)$
$\Rightarrow y_1 - y_{min} < 0 \Rightarrow y_1 < y_{min}$
- line lies outside of y_{min} boundary

If $(P_4 = 0)$ then 5.)
* If $(q_4 < 0)$
$\Rightarrow y_{max} - y_1 < 0 \Rightarrow y_{max} < 0$
- line lies outside of y_{max} boundary 6.)

∴ If $(P_i = 0)$
* If $(q_i < 0)$
- line lies completely outside of window

If $P_i = 0$ and
$q_i \geq 0$ then line lies inside of ith boundary

If $P_i \neq 0$ then we need to compute intersection point

To calculate - intersection point, this algorithm determines two value of t based on those values line is clipped

Algorithm

) Input $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$
) Input clipping window boundaries $(x_{min}, x_{max}, y_{min}, y_{max})$
) Determine P_i and q_i as follows:-

$P_1 = -\Delta x$, $q_1 = x_1 - x_{min}$ (Left Boundary)
$P_2 = \Delta x$, $q_2 = x_{max} - x_1$ (Right ")
$P_3 = -\Delta y$, $q_3 = y_1 - y_{min}$ (Bottom ")
$P_4 = \Delta y$, $q_4 = y_{max} - y_1$ (Top ")

) If $(P_i = 0)$ and $q_i < 0$ for any i
→ Line is parallel to window boundary and is outside of window so discard line segment.

) ~~Stet~~ Set two parameters t_1 & t_2 as
$t_1 = 0$
$t_2 = 1$

) Determine r_p as follows:

$$r_i = \frac{q_i}{P_i} \qquad i = 1, 2, 3, 4$$

Scanned by CamScanner

7) Find all r_i for which $p_i < 0$
set $t_1 = \max(0, r_i)$

8) Find all r_i for which $p_i > 0$
set $t_2 = \min(1, r_i)$

9.) If $(t_1 > t_2)$
→ line is completely outside window so discard line segment

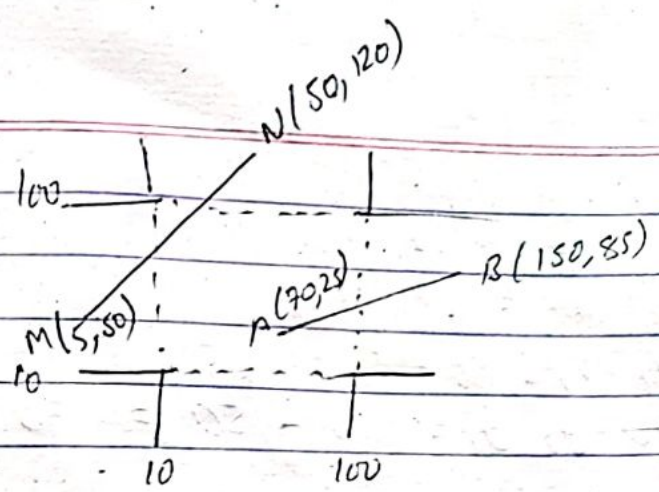
else
{
If $(t_1 = 0 \ \& \ t_2 = 1)$
— line is completely inside window, so display the line segment

else
{ clip the line segment and determine (x_1', y_1') and (x_2', y_2') by

$x_1' = x_1 + t_1 \Delta x$
$y_1' = y_1 + t_1 \Delta y$
$x_2' = x_1 + t_2 \Delta x$
$y_2' = y_1 + t_2 \Delta y$
}
}

10 End

Scanned by CamScanner



For line AB

Here, $A(70, 25)$ $B(150, 85)$

$x_1 = 70,\ y_1 = 25$ $x_2 = 150,\ y_2 = 85$

$\Delta x = x_2 - x_1 = 150 - 70 = 80$

$\Delta y = y_2 - y_1 = 85 - 25 = 60$

Now

$p_1 = -\Delta x = -80,\quad q_1 = x_1 - x_{min} = 70 - 10 = 60,\quad r_1 = \frac{q_1}{p_1} = \frac{60}{-80} = \frac{-3}{4}$

$p_2 = \Delta x = 80,\quad q_2 = x_{max} - x_1 = 100 - 70 = 30,\quad r_2 = \frac{q_2}{p_2} = \frac{3\cancel{0}}{8\cancel{0}} = \frac{3}{8}$

$p_3 = -\Delta y = -60,\quad q_3 = y_1 - y_{min} = 25 - 10 = 15\quad r_3 = \frac{q_3}{p_3} = \frac{15}{-85} = \frac{-1}{4}$

$p_4 = \Delta y = 60,\quad q_4 = y_{max} - y_1 = 100 - 25 = 75\quad r_4 = \frac{q_4}{p_4} = \frac{75}{60} = \frac{5}{4}$

Now $t_1 = 0,\ t_2 = 1$

Again,

$t_1 = \max(0, -3/4, -1/4) = 0$

$t_2 = \min(1, 3/8, 5/4) = 3/8$

Here, $t_1 < t_2$ so we need to find intersection point. let, (XX_1, YY_1) & (XX_2, YY_2) be intersection point

$XX_1 = X_1 + t_1 \Delta x = 70 + 0 \times 80 = 70$

$YY_1 = Y_1 + t_1 \Delta y = 25 + 0 \times 60 = 25$

$XX_2 = X_1 + t_2 \Delta x = 70 + 3/8 \times 80 = \cancel{180}\ 100$

$YY_2 = Y_1 + t_2 \Delta y = 25 + 3/8 \times 60 = 47.5$

Now AB is clipped at (XX_1, YY_1) & (XX_2, YY_2) ie $(70, 25)$ & $(100$

For line MN

$M(5, 50) = (x_1, y_1)$

$N(50, 120) = (x_2, y_2)$

$\Delta x = x_2 - x_1 = 50 - 5 = 45$

$\Delta y = \cancel{120} - y_2 - y_1 = 120 - 50 = 70$

Now

$p_1 = -\Delta x = -45$ $\quad q_1 = x_1 - x_{min} = 5 - 10 = -5$ $\quad r_1 = \frac{q_1}{p_1} = \frac{-5}{-45} = \frac{1}{9}$

$p_2 = \Delta x = 45$ $\quad q_2 = x_{max} - x_1 = 100 - 5 = 95$ $\quad r_2 = \frac{q_2}{p_2} = \frac{95}{45} = \frac{19}{9}$

$p_3 = -\Delta y = -70$ $\quad q_3 = y_1 - y_{min} = 50 - 10 = 40$ $\quad r_3 = \frac{q_3}{p_3} = \frac{40}{-70} = \frac{-4}{7}$

$p_4 = \Delta y = 70$ $\quad q_4 = y_{max} - y_1 = 100 - 50 = 50$ $\quad r_4 = \frac{q_4}{p_4} = \frac{50}{70} = \frac{5}{7}$

Now

$t_1 = 0$

$t_2 = 1$

Again

$t_1 = \max(0, 1/9, -4/7) = 1/9$

$t_2 = \min(1, 19/9, 5/7) = 5/7$

Here; $t_1 < t_2$, So we need to find intersection points.

let (XX_1, YY_1) & (XX_2, YY_2) be intersection point then

$XX_1 = X_1 + t_1 \Delta x = 5 + 1/9 \times 45 = 10$

$YY_1 = Y_1 + t_1 \Delta y = 50 + 1/9 \times 70 = 58$

$XX_2 = X_1 + t_2 \Delta x = 5 + 5/7 \times 45 = 37$

$YY_2 = Y_1 + t_2 \Delta y = 50 + 5/7 \times 70 = 100$

Scanned by CamScanner

Use Liang - Barsky to clip the given lines

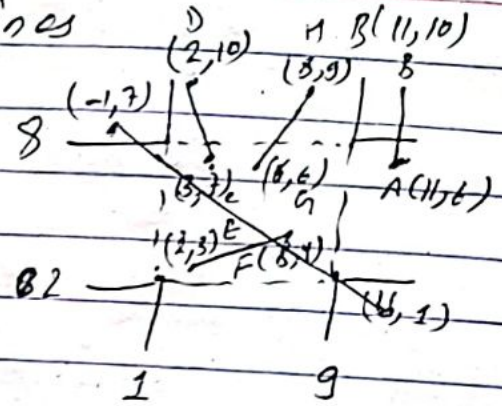
i) A (11, 6) & B (11, 10)

ii) E (2, 3) & F (8, 4)

iii) C (3, 7), D (2, 10) iv) G (6, 6) & H (8, 9)

$x_{min} = 1$, $x_{max} = 9$

$y_{min} = 2$, $y_{max} = 8$



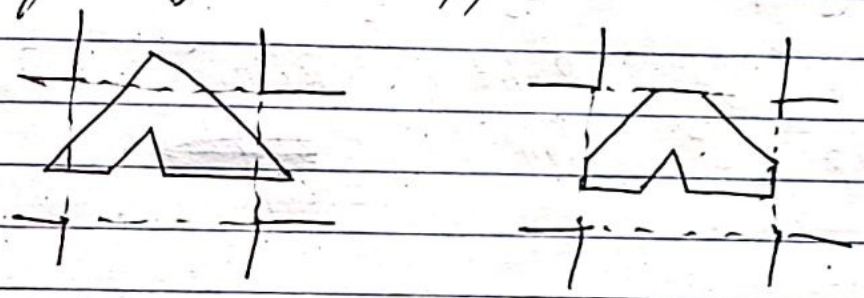
Date
Page

Scanned by CamScanner

Polygon Clipping

→ Polygon is a geometrial object consisting no. of points called vertices and an equal no. of line called edges

→ Polygon clipping is the process of discarding those parts of a polygon that lie outside of a specified region i.e. clipping window



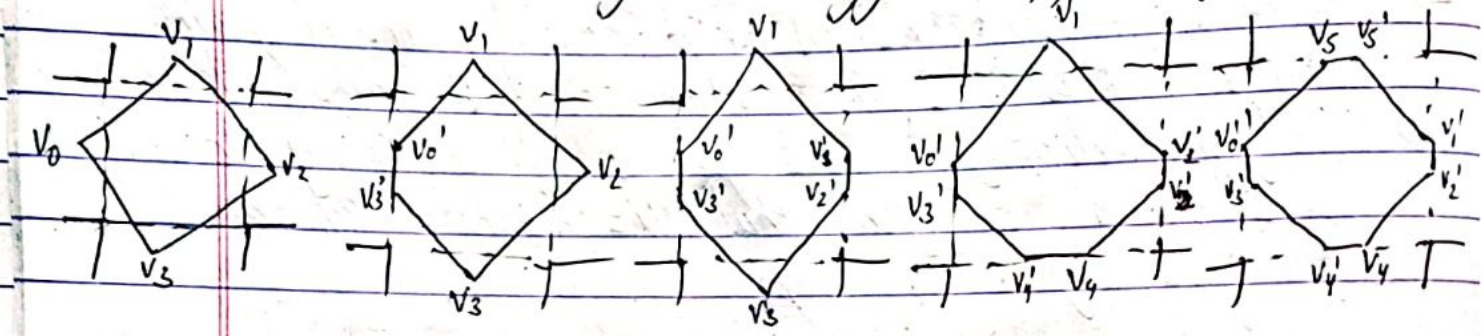
original polygon After clipping

→ It includes
- Retaining existing edges
- Discarding existing edges
- Adding new edges
- Dividing existing edges

→ Polygon clipping often results multiple polygons.

Scanned by CamScanner

Sutherland - Hodgeman Polygon Clipping Algorithm



Original polygon | Left Clip | Right Clip | Bottom ~~boundary~~ clip | Top clip

→ This is a recursive algorithm that follows divide and conquer to clip the given polygon.

→ Original polygon is clipped against left boundary, right, bottom & top boundary respectively.

→ After performing left clip, the set of output vertices are taken as input for the right clip. This process is repeated and top clip produces finally clipped polygon

→ This algorithm uses inside-outside test to determine which are inside of the clipping window boundary.

→ There are four cases for a particular window boundary

i) ~~Vi is~~ Out-In Pair : V_i is outside of boundary and V_{i+1} is inside of window boundary. In this case, intersection point V_0' and V_1 is added to the output vertex list

ii) In-In Pair :- Both V_i & V_{i+1} are inside of the window boundary.

In this case only V_{i+1} is added to output vertex list.

iii) In-Out Pair :- V_i is inside and V_{i+1} is outside of window boundary.

In this case, intersection point V_{i+1}' is added to output vertex list.

iv) Out-Out Pair :- Both V_i & V_{i+1} are outside of the window boundary.

Hence, add nothing to the output vertex list.

→ After clipping a polygon often it results in a open polygon hence new edges are introduced to form a closed polygon which is called POST PROCESS

Algorithm

1.) For each clip boundary, perform following, starting from left boundary (x_{min}) :

i) Create input vertex list

i.e. input vertex list = $\{V_0, V_1, V_2, \ldots V_n\}$ when $V_n = V_0$

ii) Create initially empty output vertex list

i.e. Output vertex list = { }

iii) For each pair of adjacent vertices V_i & V_{i+1} perform insid-outside test

a) If Out-In Pair : V_i is outside & V_{i+1} is inside

→ Add intersection point V_i' & V_{i+1} to OVL.

b) If In-In Pair : Both V_i & V_{i+1} are inside

→ Add V_{i+1} to output vertex list

Scanned by CamScanner

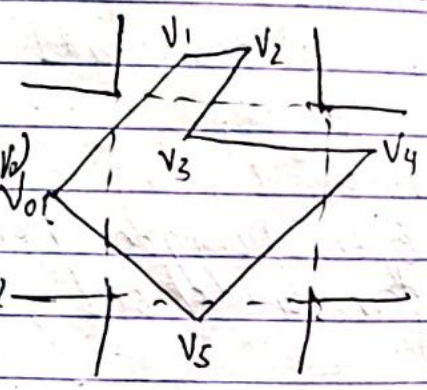
c) If In - Out Pair : V_i inside & V_{i+1} outside
→ Add intersection point V_{i+1} to OVL

d) If Out-Out Pair : Both V_i & V_{i+1} outside
→ Add nothing to OVL

iv) Apply Post process if necessary

2.) Repeat same process for right boundary, bottom boundary and top boundary.

For left boundary (X_{min})



Input vertex list = $\{V_0, V_1, V_2, V_3, V_4, V_5, V_0\}$

Output vertex list = $\{\ \}$

For pair $[V_0, V_1] \xrightarrow{Out-In}$ Add V_0' & V_1 to OVL

For pair $[V_1, V_2] \xrightarrow{In-In}$ Add V_2 to OVL

For pair $[V_2, V_3] \xrightarrow{In-In}$ Add V_3 to OVL

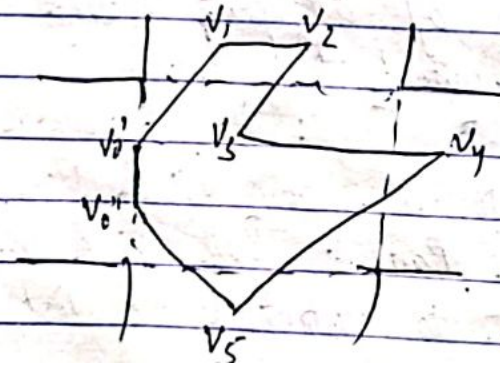
For pair $[V_3, V_4] \xrightarrow{In-In}$ Add V_4 to OVL

For pair $[V_4, V_5] \xrightarrow{In-In}$ Add V_5 to OVL

For pair $[V_5, V_0] \xrightarrow{In-Out}$ Add V_0'' to OVL

$\therefore$ OVL = $\{V_0', V_1, V_2, V_3, V_4, V_5, V_0''\}$

$\therefore$ Post process = connect V_0'' & V_0'



Scanned by CamScanner

For right boundary (X_{max})

Input vertex list = { V_0', V_1, V_2, V_3, V_4, V_5, V_0'', V_0' }

Output vertex list = { }

For pair $[V_0', V_1] \xrightarrow{In-In}$ Add V_1 to OVL

$[V_1, V_2] \xrightarrow{In-In}$ Add V_2 to OVL

$[V_2, V_3] \xrightarrow{In-In}$ Add V_3 to OVL

$[V_3, V_4] \xrightarrow{In-Out}$ Add V_4' to OVL

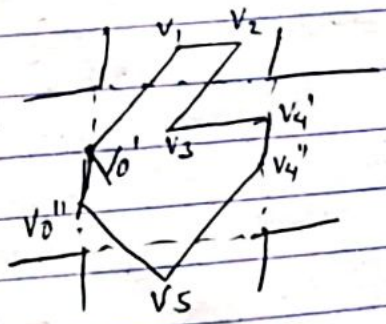
$[V_4, V_5] \xrightarrow{Out-In}$ Add V_4'', ~~to OVL~~ V_5 to OVL

$[V_5, V_0''] \xrightarrow{In-In}$ Add V_0'' to OVL

$[V_0'', V_0'] \xrightarrow{In-In}$ Add V_0' to OVL

$\therefore$ OVL = { V_1, V_2, V_3, V_4', V_4'', V_5, V_0'', V_0' }

$\therefore$ Post process = connect V_4' & V_4''



For bottom boundary (Y_{min})

IVL = { V_1, V_2, V_3, V_4', V_4'', V_5, V_0'', V_0' }

OVL = { }

For pair :-

$[V_1, V_2] \xrightarrow{In-In}$ Add V_2 to OVL

$[V_2, V_3] \xrightarrow{In-In}$ Add V_3 to OVL

$[V_3, V_4'] \xrightarrow{In-In}$ Add V_4' to OVL

$[V_4', V_4''] \xrightarrow{In-In}$ Add V_4'' to OVL

$[V_4'', V_5] \xrightarrow{In-Out}$ Add ~~V_5', to~~ V_5' to OVL

$[V_5, V_0''] \xrightarrow{Out-In}$ Add V_5'' & V_0'' to OVL

$[V_0'', V_0'] \xrightarrow{In-In}$ Add V_0' to OVL

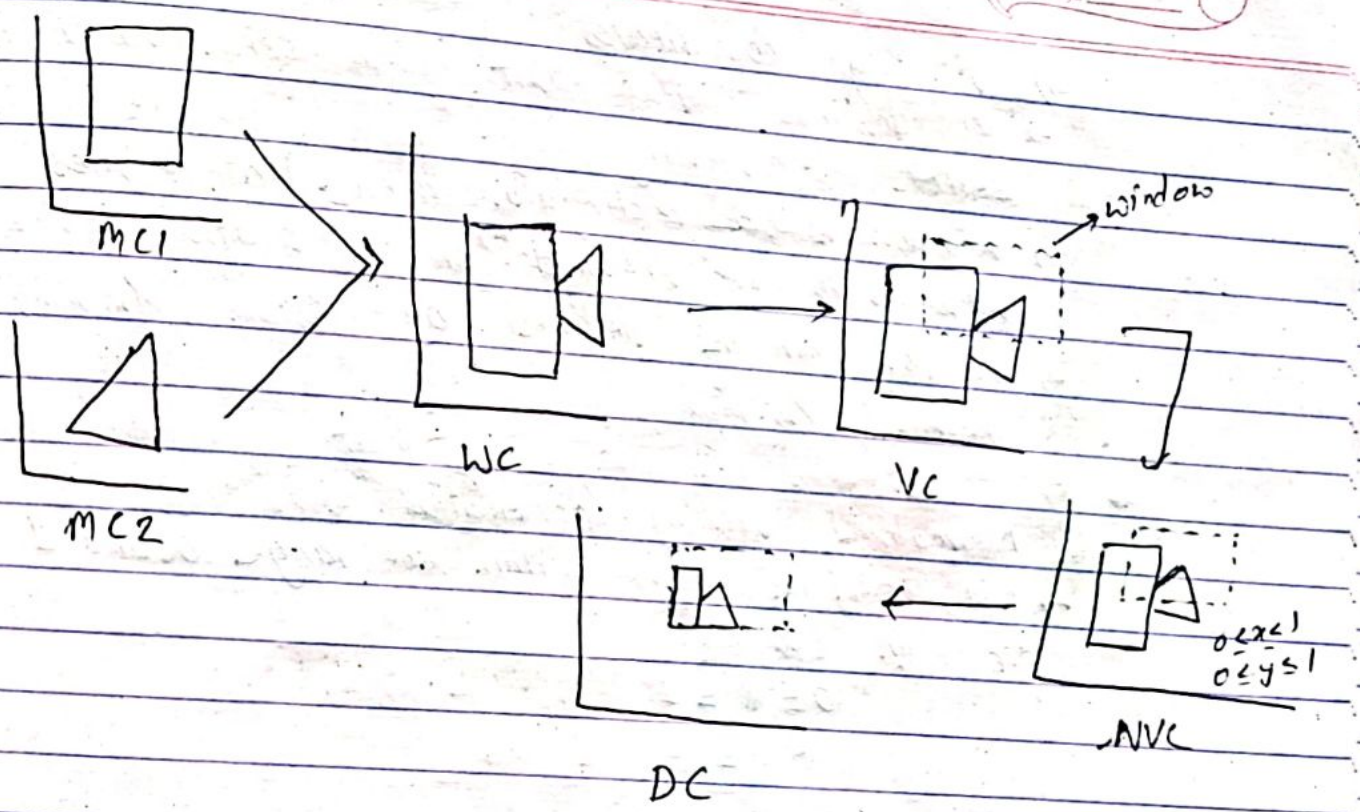
$[V_0', V_1] \xrightarrow{In-In}$ Add V_1 to OVL

Scanned by CamScanner

Date
Page

Scanned by CamScanner

2D Viewing



→ It is the process of producing different views of a 2D scene i.e. translated view, rotated view, etc.

→ Graphical package allows us to define part of the scene to be displayed and part of screen where defined view is displayed

→ In 2D multiple coordinate systems are used to create 2D scene

i) Modeling Coordinates / Master coordinates

→ It defines shape and size of ~~eb~~ individual objects to create a graphical ~~screen~~ scene

→ Arbitrary unit are used to specify MC's.

ii) World Coordinate System

→ It defines relative placement of objects to create a total picture.

Scanned by CamScanner

iii) Viewing Coordinates

→ It defines that part of a scene which is selected for viewing

→ Window is a rectangular area that defines what is to be viewed on an output device

→ Window can be defined by $X_{w\text{-}min}$, $X_{w\text{-}min}$, $Y_{w\text{-}min}$, $Y_{w\text{-}max}$

iv) Normalized Viewing Coordinates

→ It specifies VC's within the range 0 and 1 i.e for each (x, y),

$$0 \le x \le 1$$
$$0 \le y \le 1$$

v) Device Coordinates

→ It defines resolution of an output device.

→ Viewport is a sub-area where selected part of 2D scene is displayed

→ That is, it defines an area where window is mapped

→ Viewport shows where on an output device window is displayed.

→ It can be defined by X_{dmin}, X_{dmax}, Y_{dmin} & Y_{dmax}

→ Clipping is necessary to discard that part of a 2D scene that lie outside of a window

→ Clipping can be performed in world coordinate or device coordinate

Scanned by CamScanner

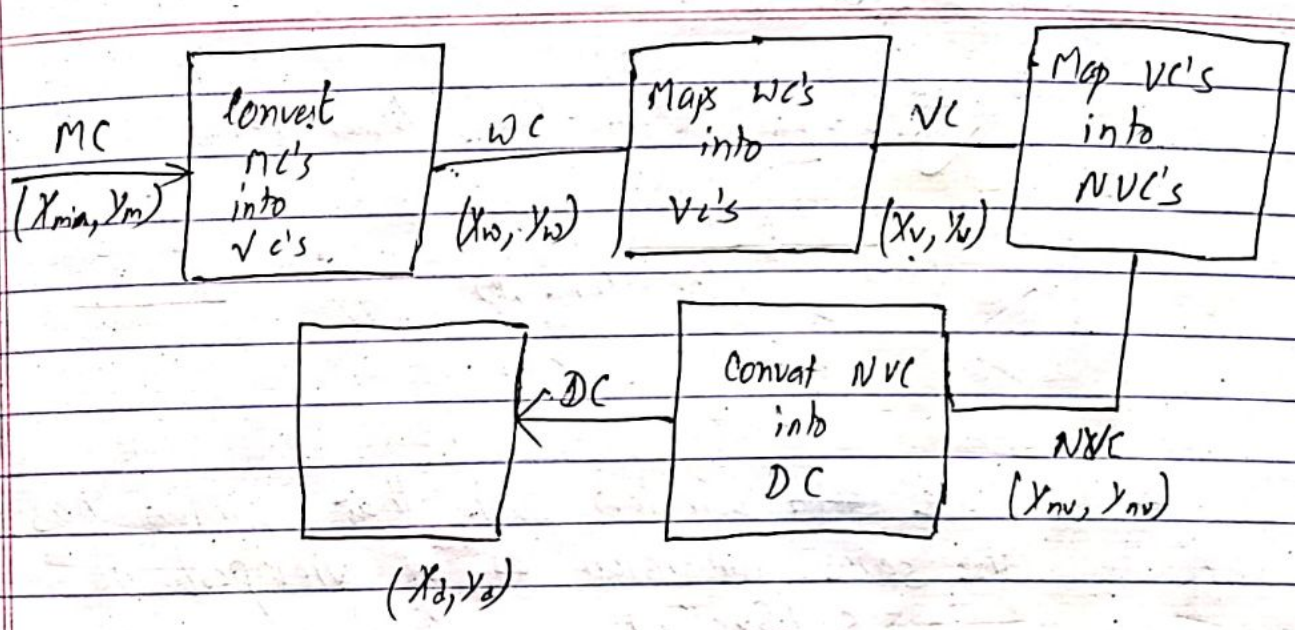
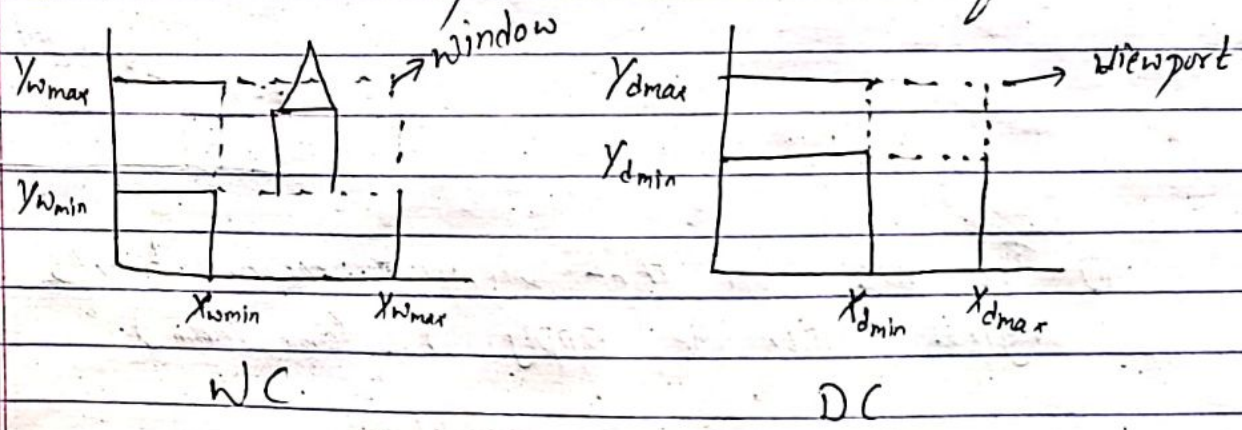


fig:- 2D viewing pipeline

Window to Viewport coordinate transformation.

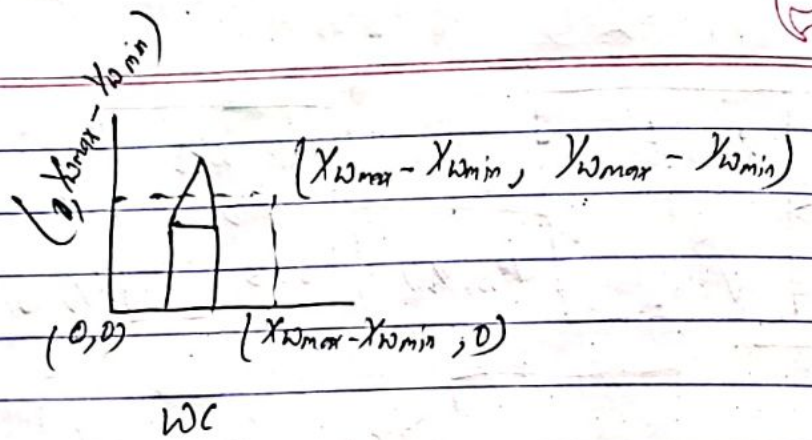


→ Let window is defined by world coordinates X_{wmin}, X_{wmax}, Y_{wmin}, Y_{wmax}. & Viewport is defined by device coordinates X_{dmin}, X_{dmax}, Y_{dmin}, & Y_{dmax}.

→ Window to viewport transformation can be obtained as follows

i) Translate window such that lower left corner coincides with coordinate origin

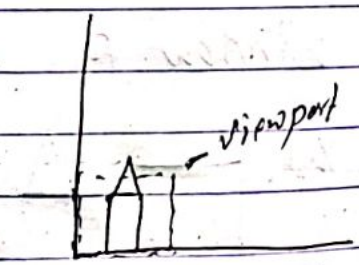
ie apply $T(-X_{wmin}, -Y_{wmin})$



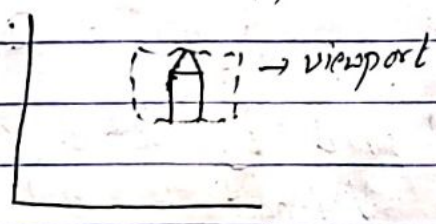
ii) Scale ~~such~~ window such that window has the same dimension as that of viewport ie
ie. $S(S_x, S_y)$

where, $S_x = \dfrac{X_{dmax} - X_{dmin}}{X_{wmax} - X_{wmin}}$

$S_y = \dfrac{Y_{dmax} - Y_{dmin}}{Y_{wmax} - Y_{wmin}}$



iii). Translate such that viewport reaches to its original position ie apply $T(X_{dmin}, Y_{dmin})$



$\therefore$ Net - transformation $(N_T) = T(X_{dmin}, Y_{dmin}) . S(S_x, S_y) . T(-X_{wmin}, -X_{wmin})$

let $P(x,y)$ be the point inside the window & $P'(x',y')$ be the equivalent point inside viewport than

$[P' = N_T \cdot P]$

Given a window lower left at (1, 2) and upper right at (16, 12). Given the triangle with vertices (3, 2), (10, 7.5 & (5, 5). Calculate the appropriate position of this triangle in the viewport with lower left (100, 100) & upper right at (400, 200).

8 min

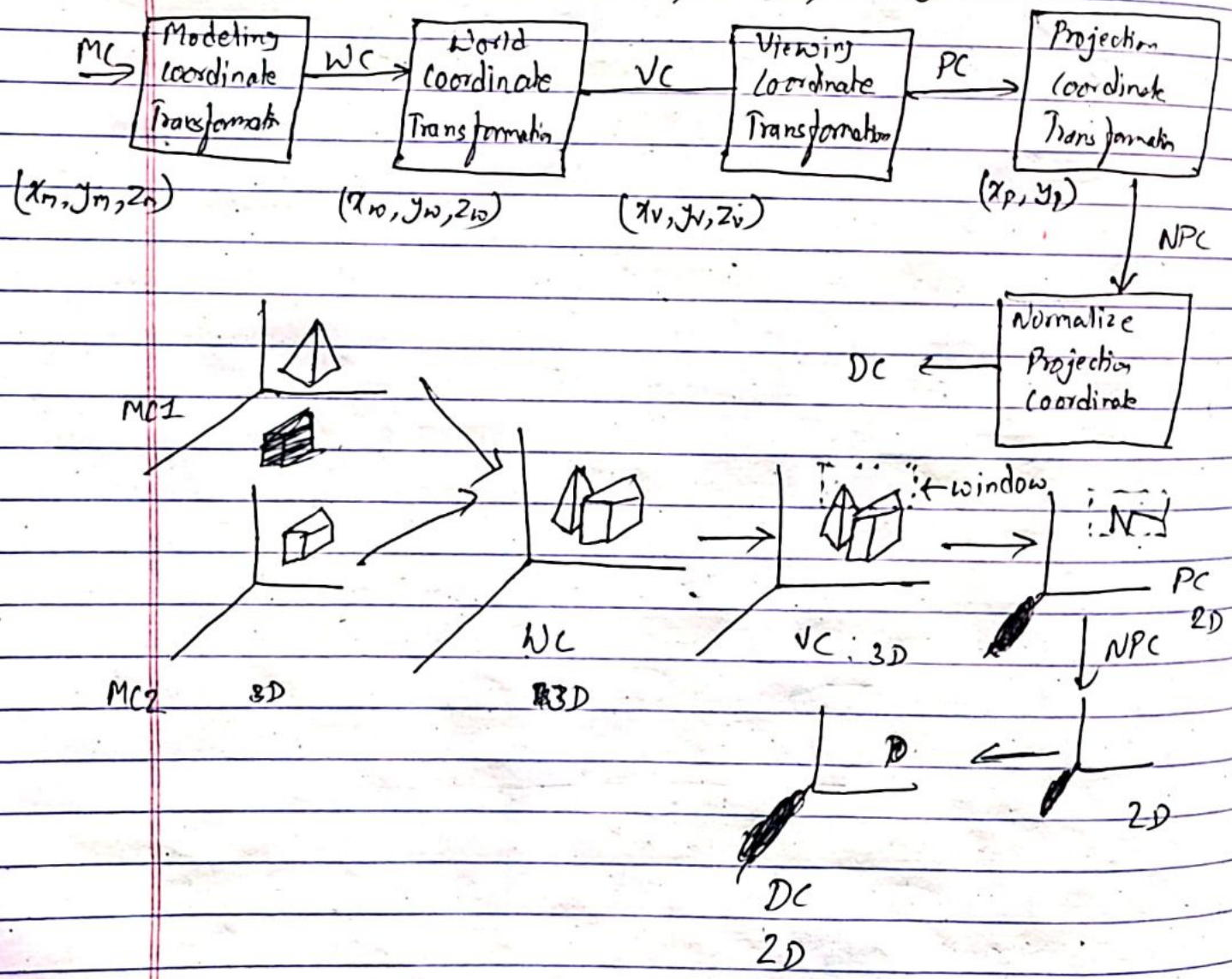
Scanned by CamScanner

3D Viewing

→ It is the process of mapping selected part of 3D world coordinate into 2D device coordinate.

→ In other words, it is the process of producing different views of a 3D scene into 2D device coordinates.

→ It is similar to taking photograph by a camera



Scanned by CamScanner

Projection

→ Mathematically, projection can be defined as conversion of m-dimensional coordinate system into n-dimensional coordinate system, where $m > n$.

→ In Computer Graphics, the conversion of 3D objects descriptions into 2D object descriptions is called projection

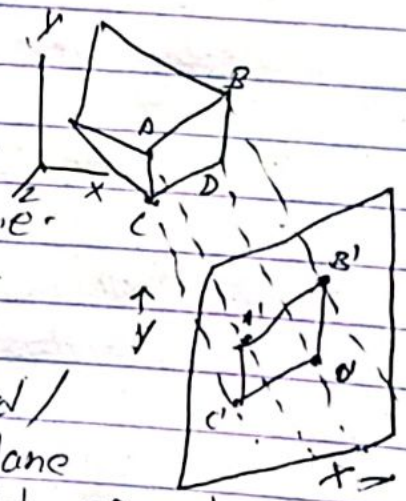
→ It produces a shadow image of a 3D object.

→ There are two types of projection in CG:-

i) Parallel projection

ii) Perspective projection

Projectors



→ Light rays / straight line segments casted from 3D object vertices onto 2D plane.

→ Also referred to as a projection vector

→ View Plane / Projection Plane

- A 2D plane where 3D object is mapped / displayed / projected is called view plane
- Generally, a 3D object is mapped into 2D object by determining intersection points of projectors with projection plane and intersection point are connected by straight line segments in the view plane.

→ Center of Projection (COP)

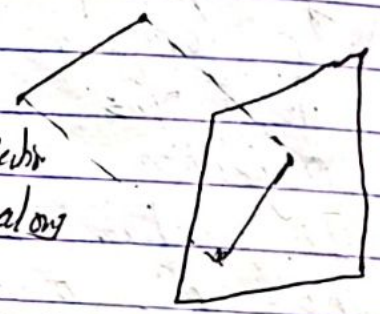
- Projectors converseence points is called center of projection.
- If projectors are parallel then COP lies at infinity

Scanned by CamScanner

- In this case projection is denoted by direction of projection (DOP)
- Normally, COP denotes human eye or camera position

Parallel Projection

→ Projectors are parallel ~~to~~, hence COP lies at infinity



→ Projectors remain parallel after projection

→ 3D object descriptions are projected along parallel lines.

→ It is often ~~catesa~~ characterized by DOP

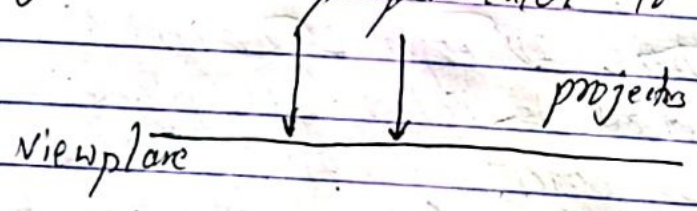
→ It preserves relative proportion of 3D object hence used in mathematical drawings

→ But it cannot provide realistic displaying

Types of Parallel Projection

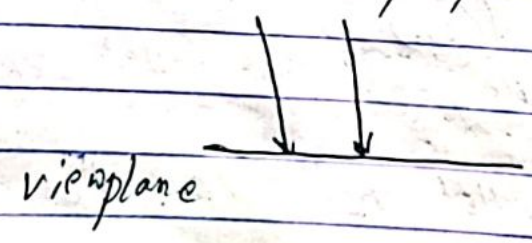
a) Orthorgraphic Parallel Projection

→ Projectors are perpendicular to viewplane



b) Oblique Parallel Projection

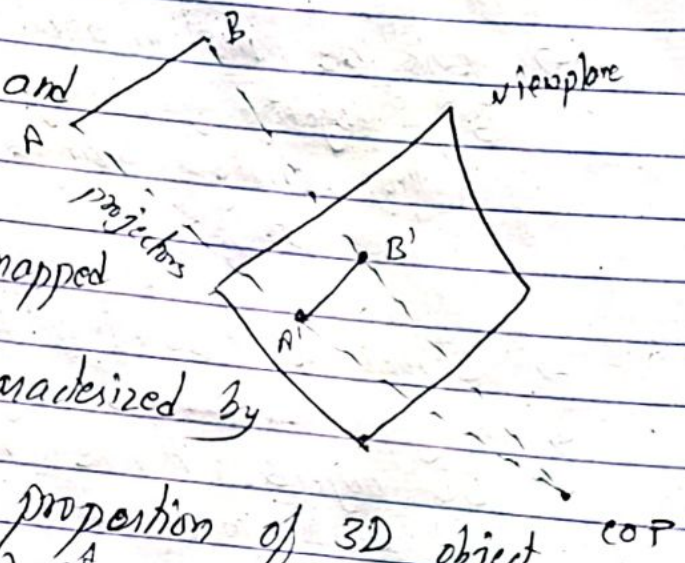
→ Projectors are not perpendicular to viewplane



Scanned by CamScanner

Perspective Projection

- Projectors are not parallel and hence COP is at finite distance from a viewplane
- 3D object descriptions are mapped along converging lines
- This type of projection is characterized by center of projection COP.
- It cannot preserve relative proportion of 3D object but provides realistic view of a 3D object



Scanned by CamScanner

3D Object Representation

→ It is the mathematical model to represent complex 3D Object by using simple / basic 2D primitives (line, circle, triangle, rectangle, spline, arc, polygon faces etc)

→ Sometimes also reffered to as a solid modeling

→ A graphical scene may contain different types of 3D objects, hence we require different set of methods to model those objects.

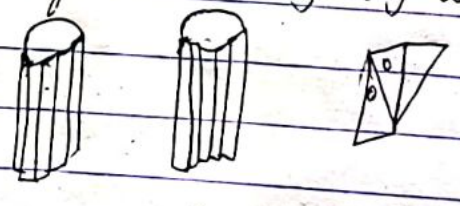
→ There are two broad categories of 3D object representation

•) Boundary Representation (B - Reps)

→ It only represents Object boundary by means of faces in that particular object.

→ It only associates object shape, hence cannot represent interior properties of object. (number of descrete points, distance between descrete points etc)

Eg:- Polygon Surfaces, spline patches, quadric surfaces, Blobby objects, Bezier curve etc.



•) Space Partitioning Method.

→ This method represent object interior properties i.e space content / within object boundary.

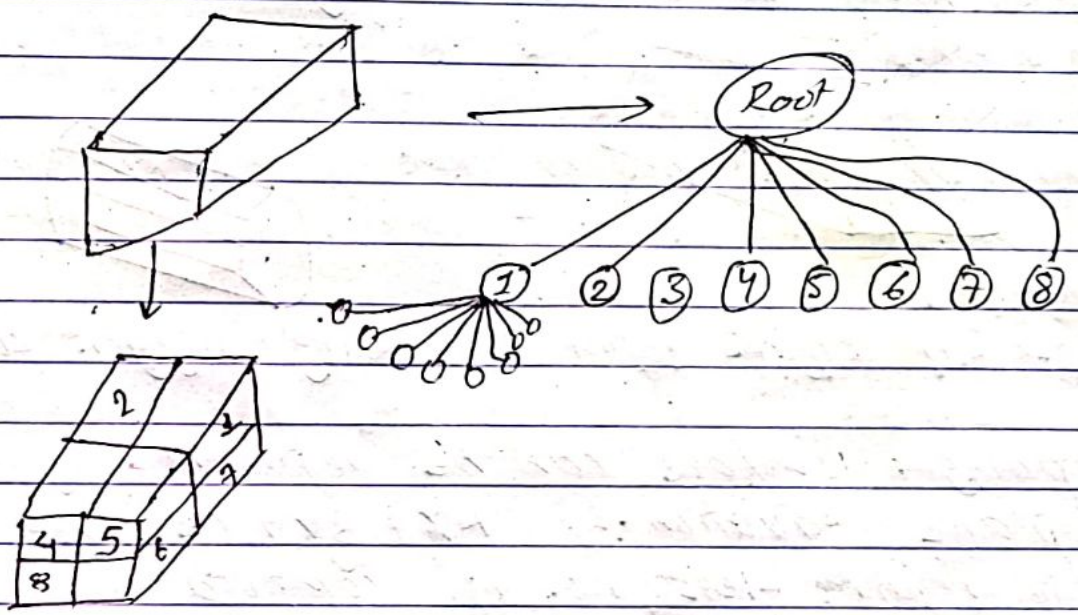
→ Entire 3D object is recursively sub-divided into non-overlapping region. (usually cubes)

→ Recursive sub-division is repeated until each

region represents a single pixel.

→ It is represented as a tree structure.

Eg :- Quad tree method, Octree Method



Scanned by CamScanner

Polygon Surfaces :-

→ It is the most widely used boundary representation.

→ In this method a 3D object is represented by collection of polygon faces as shown in figure

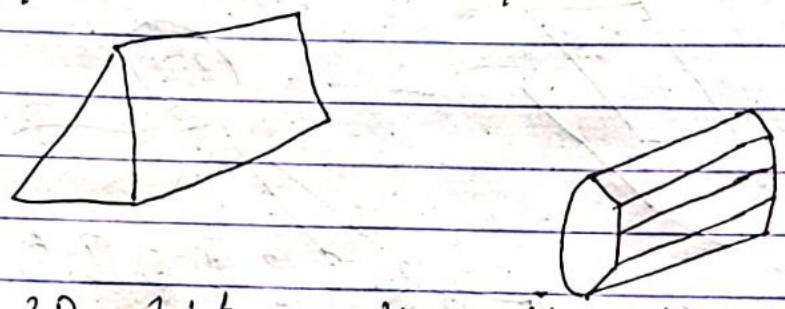


fig :- 3D objects represented using polygon surfaces

→ Each polygon surface can be represented by a ~~polygon~~ linear plane equation :- $Ax + By + Cz + D = 0$
Hence, it requires less no. of computation.

→ Also, Polygon surface has a linear boundaries, so clipping and surface rendering (determination of appropriate color for a particular surface) become easier.

→ Hence, polygon surfaces are considered as a standard graphical object.

→ Polygon surfaces can be represented by using :

•) Polygon Table

→ It has two parts :-

i) Geometric table :- → Contains vertex table, Edge, surface table.

→ Vertex table :- contains coordinate information of each vertex

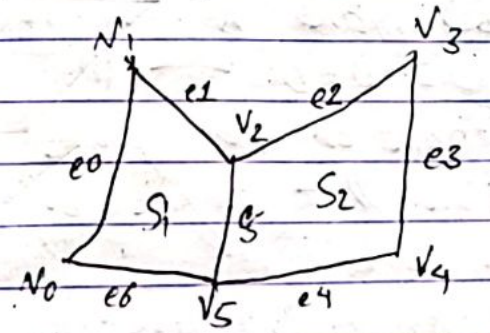
→ Edge table :- Contains edge information. Stores pointers back to vertex table

→ Surface table :- Contains information about each face. Stores pointers back to edge table.

Scanned by CamScanner

ii) Attribute table :-

→ It stores surface characteristics such as surface, reflectivity, surface transparancy, shiny or rough surface etc.



Vertex Table
V_0 : x_0, y_0, z_0
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_0 : V_0, V_1
e_1 : V_1, V_2
e_2 : V_2, V_3
e_3 : V_3, V_4
e_4 : V_4, V_5
e_5 : V_2, V_5
e_6 : V_0, V_5

Surface Table
S_1 : e_0, e_1, e_5, e_6
S_2 : e_5, e_2, e_3, e_4

Scanned by CamScanner

Plane Equation

→ Another widely used method to represent polygon surfaces

→ In this method a 3D plane is expressed as $Ax + By + Cz + D = 0$ — (i)

where (x, y, z) is the point on that plane and A, B, C & D are spatial parameters that define orientation & inclination of a plane.

→ Let (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3) are three non-collinear points on the plane described by eq (i) #

Now

$$\left.\begin{aligned} Ax_1 + By_1 + Cz_1 + D &= 0 \\ Ax_1 + By_2 + Cz_2 + D &= 0 \\ Ax_3 + By_3 + Cz_3 + D &= 0 \end{aligned}\right] \text{— (ii)}$$

$$\left.\begin{aligned} \left(\frac{A}{D}\right)(-x_1) + \frac{B}{D}(-y_1) + \left(\frac{C}{D}\right)(-z_1) &= 1 \\ \left(\frac{A}{D}\right)(-x_2) + \frac{B}{D}(-y_2) + \left(\frac{C}{D}\right)(-z_2) &= 1 \\ \left(\frac{A}{D}\right)(-x_3) + \frac{B}{D}(-y_3) + \left(\frac{C}{D}\right)(-z_3) &= 1 \end{aligned}\right\} \text{— (iii)}$$

Now; solving eqn (iii) using Cramers rule

$$D = \begin{vmatrix} -x_1 & -y_1 & -z_1 \\ -x_2 & -y_2 & -z_2 \\ -x_3 & -y_3 & -z_3 \end{vmatrix} = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

$$A = \begin{vmatrix} 1 & -y_1 & -z_1 \\ 1 & -y_2 & -z_2 \\ 1 & -y_3 & -z_3 \end{vmatrix} = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}$$

Scanned by CamScanner

$$B = \begin{vmatrix} -x_1 & 1 & -z_1 \\ -x_2 & 1 & -z_2 \\ -x_3 & 1 & -z_3 \end{vmatrix} = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} -x_1 & -y_1 & 1 \\ -x_2 & -x_2 & 1 \\ -x_3 & -y_3 & 1 \end{vmatrix} = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Determine equation of the plane that contains three non collinear points $(-4, 8, 4)$, $(2, -4, 6)$ $(9, 7, 5)$

Scanned by CamScanner

Polygon Mesh :-

→ It is a collection of vertices, edges, and polygon faces such that each edge is shared by atmost two polygon faces, and each vertex is shared by at least two edges.

→ In other words, it is a collection of homogeneous polygon faces.

→ Generally, polygon face consist of triangles quadrilaterals or convex polygon

→ If polygon mesh is formed by triangles, it is called triangle strip and if a mesh is formed by using quadrilaterals only then it is called quadrilateral mesh

→ If total no. of vertices are n, in a triangle mesh than it can represent n-2 connected triangles.

→ If a polygon mesh is formed by an array n * m vertices then it contains (n-1) * (m-1) connected quadrilaterals

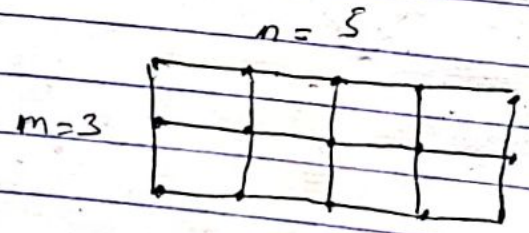
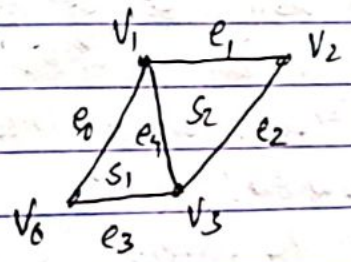


fig:- A quadrilateral mesh with (5 x 3) array of vertices and (5-1) x (3-1) connected quadrilaterals.

→ Often a polygon mesh is represented by a pointers to an edge list method

→ In this method, a vertex list an edge list, and a face list is maintained.

→ An edge list contains vertex information and face information.



$V = \{V_0, V_1, V_2, V_3, V_0\}$

$E = \{e_0, e_1, e_2, e_3, e_4\}$

→ An edge is represented by two vertices and faces that share that particular edge.

$e_0 = (V_0, V_1, S_1, \lambda)$

$e_1 = (V_1, V_2, S_2, \lambda)$

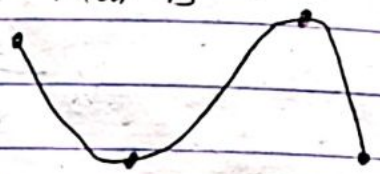
$e_4 = (V_1, V_3, S_1, S_2)$

Scanned by CamScanner

Spline Curve

→ Another widely used boundary representation in which curve is generated by using set of points called control points or knots.

→ Spline curve is a composite curve that is generated by using parametric polynomial functions of the form:-



$$X = x(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3 + \dots + a_n u^n$$
$$Y = y(u) = b_0 + b_1 u + b_2 u^2 + b_3 u^3 + \dots + b_n u^n$$
$$Z = z(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3 + \dots + c_n u^n$$

where $0 \le u \le 1$

→ There are two categories of spline curve

- Interpolated Spline

→ If a spline curve passes through each control point then it is called interpolated spline. Eg:-

→ This types of spline curve are used to design animation paths.

fig:- Interpolated spline using four control points

- Approximated spline curve

→ ~~n~~ are used to design automobiles body, aircraft wings etc.

→ Control point only define the shape of the curve

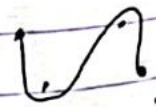


fig: approximated spline curve using ~~through~~ four control points.

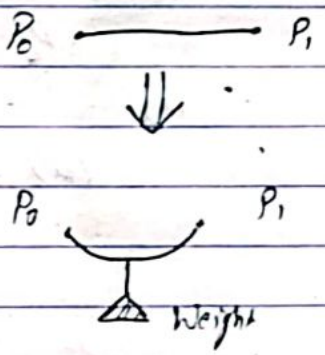
Scanned by CamScanner

→ Generally cubic splines are the most widely used splines.

→ That is, each point on the spline curve is expressed using degree 3 polynomial

$$X = x(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$$
$$Y = y(u) = b_0 + b_1 u + b_2 u^2 + b_3 u^3$$
$$Z = z(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$$

→ Generally, polynomial functions are used to define sharpness of the curve which are called blending function or basis function



Scanned by CamScanner

Bezier Curve

→ Most widely used approximated spline curve which uses blending functions to generate a curve

→ It is a curve defined by weighted sum of control points and blending functions. #

→ Mathematically,

Given $n+1$ control points $P_0(x_0, y_0, z_0)$, $P_1(x_1, y_1, z_1)$ ... $P_n(x_n, y_n, z_n)$ then a Bezier curve is given by

$$C(u) = \sum_{i=0}^{n} P_i . BEZ_{i,n}(u) \quad \text{—— (i)}$$

where, P_i is i^{th} control point,

$BEZ_{i,n}$ is i^{th} Bezier blending function and is defined by Bernstein polynomial as;

$$BEZ_{i,n}(u) = C(n,i) . u^i . (1-u)^{n-i} \quad \text{—— (ii)}$$

where

$C(n,i)$ is the Binomial coefficient and is given

$$C(n,i) = \frac{n!}{i!\,(n-i)!} \quad \text{—— (iii)}$$

where $0 \le u \le 1$

→ Rewriting eqn (i) in terms of x, y & z-coordinate we obtain

$$X(u) = \sum_{i=0}^{n} X_i . BEZ_{i,n}(u) \quad \text{—— (iv)}$$

$$Y(u) = \sum_{i=0}^{n} Y_i . BEZ_{i,n}(u) \quad \text{—— (v)}$$

Scanned by CamScanner

$$Z(u) = \sum_{i=0}^{n} Z_i \cdot BEZ_{i,n}(u) \quad \text{—— (vi)}$$

Design a Bezier curve using control points

$P_0 (-7, 4, 1)$ $P_1 (9, 8, 4)$

$P_2 (2, 2, -2)$ $P_3 (4, 6, -4)$

where $u = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$

Soln/

let $c(u)$ be the Bezier curve defined by P_0, P_1, P_2 & P_3.

Now

$$c(u) = \sum_{i=0}^{3} P_i \cdot BEZ_{i,3}(u)$$

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

Now, solving basis functions :-

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

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

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

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

Now, substituting these value in eqn (i)

$$\therefore c(u) = P_0 \cdot (1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 \cdot 3u^2(1-u) + P_3 \cdot u^3 \quad \text{—— (ii)}$$

Now, resolving eqn (ii) in x, y, z - coordinate

Scanned by CamScanner

At $u = 0$

$$c(0) = \cancel{(1)^3}\,(-7, 4, 1)(1)^3 + (9, 8, 4) \times 3 \times 0^2 (1-0)^2$$
$$+ (2, 2, -2) \times 3 \times 0^2 (1-0) + (4, 6, -4) \times 0^3$$
$$= (-7, 4, 1)$$

Scanned by CamScanner

Properties of Bezier Curve

1.) Bezier curve always passes through first and last control points :-

ie First point on the Bezier curve when $u=0 \Rightarrow P_0$
Last point on the Bezier curve when $u=1 \Rightarrow P_n$

2.) The degree of Bezier curve is ~~become~~ one less than the number of control point i.e. if n control point are used to design a Bezier curve then its degree is $n-1$.

3.) Transformation such as Translation, Scaling, Rotation etc can be applied to Bezier curve.

4.) The produced Bezier curve is always contained within convex hull (polygon formed by connecting adjacent control points) of control points.

5) The sum of Bezier ~~curve is always contained within~~ basis function is equal to 1

ie. $$\sum_{i=0}^{n} BEZ_{i,n}(u) = 1$$

Scanned by CamScanner

Design a Bezier curve through vertices (1,1), (2,3), (4,3) and (6,4) where $u = \left\{\frac{1}{4}, \frac{2}{5}, \frac{3}{4}\right\}$

Sol^n/

let $C(u)$ be the Bezier curve then

$$C(u) = \sum_{i=0}^{3} P_i \, BEZ_{i,3}(u)$$

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

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

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

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

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

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

$$= (1,1)(1-u)^3 + (2,3) \, 3u(1-u)^2 + (4,3) \, 3u^2(1-u)^2 + (6,4) \, u^3$$

$$= (1-u)^3 + 6u(1-u)^2 + 12(1-u)u^2 + 6u^2, \; (1-u)^3 + 9u(1-u)^2 + 9u^2(1-u) + 4u^3)$$

At $u = 1/4$

$$\therefore C\left(\frac{1}{4}\right) = \left\{\left(1-\frac{1}{4}\right)^3 \times 6 \times \frac{1}{4}\left(1-\frac{1}{4}\right)^2 + 12\left(1-\frac{1}{4}\right)\left(\frac{1}{4}\right)^2 + 6 \times \left(\frac{1}{4}\right)^3, \left(1-\frac{1}{4}\right)^3 + 9 \times \frac{1}{4}\left(1-\frac{1}{4}\right)^2 + 9\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right) + 4 \times \left(\frac{1}{4}\right)^3\right.$$

$$= (0.75)^2 + 1.5 \times (0.75)^2 + 12 \times (0.75)(0.25)^2 + 6 \times (0.25)^3, (0.75)^3 + 2.25 \times (0.75)^2 + 9(0.25)^2 \times 0.75 + 4 \times (0.25)^3$$

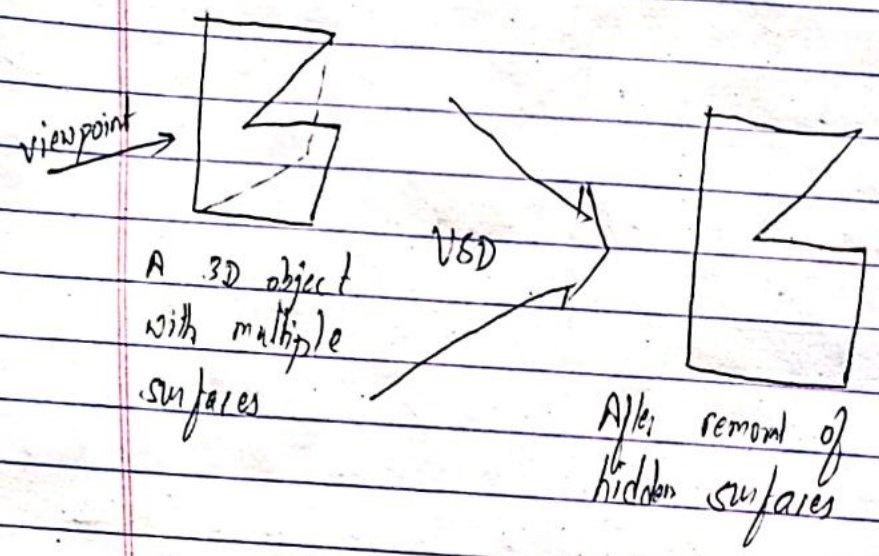
$=$

Scanned by CamScanner

Date
Page

Scanned by CamScanner

Visible Surface Detection (VSD)



→ VSD / HSR is the process of determining all visible parts of a 3D scene w.r.t particular viewpoint and discarding all hidden surfaces from the scene

→ To implement ~~visible~~ VSD often face coherence is used

→ VSD is an important aspect of CG to produce realistic displays and reduce memory requirement.

→ Multiple VSD algorithm exists but can be categorized into two classes.

i) Object Space Method

→ Visible Surfaces are identified in world co-ordinate

→ ~~To deter~~ Object Surfaces are compared to each other to determine visible surfaces w.r.t viewpoint

→ It requires n^2 comparisons to determine visible surfaces among n- surfaces

→ Eg:- Back-Face Detection, Depth-Sorting Algorithm

Scanned by CamScanner

ii) Image Space Method

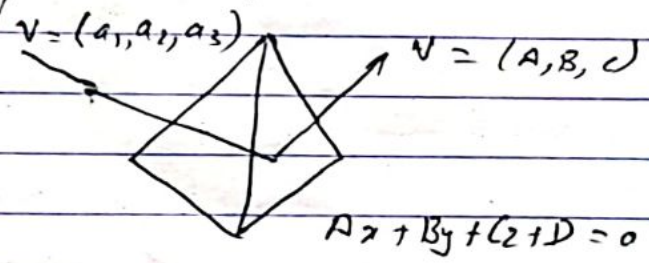
→ Implemented in device co-ordinate i.e after projection of 3D object onto 2D device co-ordinate.

→ For each pixel x, y nearest object surface is determined and color of that surface is entered into frame buffer

→ If there are m x n pixels ~~and~~ and o number of objects then it requires m x n + o no. of comparison.

→ Eg :- Z- buffer Method, A- Buffer Method, Scan-line Method. etc.

Back-Face Detection Algorithm / Plane Equation Method :-



Simplest object space method in which all the back faces are determined for a particular viewpoint. and

And determined back faces are eliminated, hence this method is also called back face elimination method.

If $Ax + By + Cz + D = 0$ is the candidate surface and $\vec{V}_{view}$ the viewing vector specifying the viewpoint, then that face is back face if

$$\vec{V}_{view} \cdot \vec{N} > 0$$

where $\vec{N}$ is the surface normal vector on $Ax + By + Cz + D = 0$

$$\vec{V}_{view} \cdot \vec{N} = |\vec{V}_{view}|\,|\vec{N}| \cos\theta$$

If $\vec{V}_{view} = \langle -9, 8, 4 \rangle$ is the viewpoint and $2x + 3y + 6z + 8 = 0$ is the plane surface then determine this surface is back-face or not.

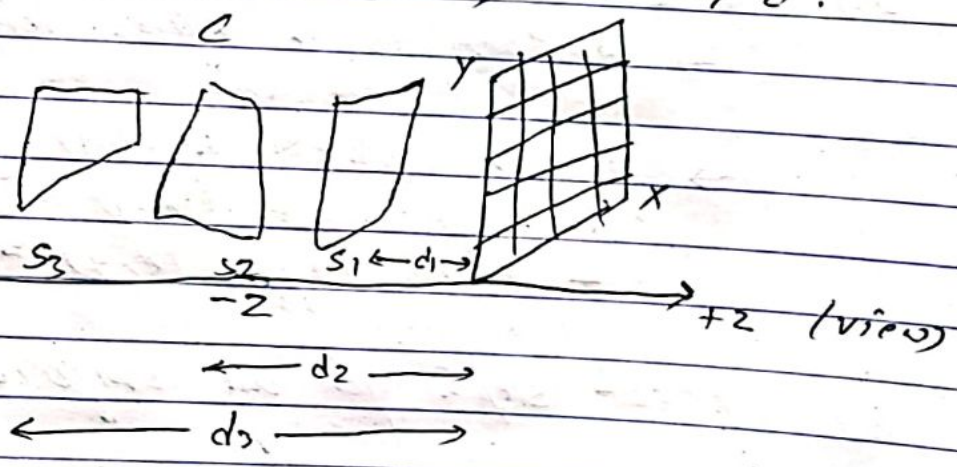
Sol

Z- Buffer Method / Depth - Buffer Method :-

- Most widely used image space method in which visible surfaces are determined based on depth calculation of object surfaces.
- It is also called Z- Buffer Method because ~~Z is~~ usually depth is calculated or measured along Z- coordinate.
- For each x, y, z on the object surface its projection point on the projection plane is determined.
- The depth of particular object is basically a distance between a viewplane and that object surface.

Depth for a particular object surface $Ax + By + Cz + D = 0$ is calculated as;

$$Z = \frac{-(Ax + By + D)}{C}, \text{ where } C \neq 0.$$



Visible surfaces are determined based on depth of an object i.e for a particular pixel a surface with smallest depth is considered as a visible surface. Smallest depth refers maximum Z- coordinate. (viewpoint is along +ve Z- direction)

It requires two types of buffer :-

i) Frame Buffer :-

For each pixel (x, y) frame buffer stores intensity value i.e. frame-buffer $(x, y) = I$

ii) Depth Buffer :- It stores depth values of each surface for pixel (x, y)

i.e. depth-buffer $(x, y) = d$

Algorithm

1.) For each pixel (x, y), initialize buffer values as ;

frame-buffer $(x, y) = I_{background}$

depth-buffer $(x, y) = -\infty$

2.) Process each surface S randomly, for each pixel (x, y)

→ Calculate depth value for that surface for pixel (x, y) i.e. d

→ If $d >$ depth-buffer (x, y)

- frame-buffer $(x, y) = I_S$
- depth-buffer $(x, y) = d$

3.) Repeat step 2 for all surfaces

Scanned by CamScanner

Pros & Cons of Z-Buffer Method

It doesnot require pre-sorting of surfaces

It doesnot require acceleration hardware

It can only deal with opaque surfaces ie. only one surface is visible for a particular path.

A - Buffer Method

Extension of Z-Buffer method that deals with all kinds of surfaces.

It ~~stands for~~ is called accumulation buffer method because it can accumulate color information of all background surfaces that are visible through foreground transparent surface.

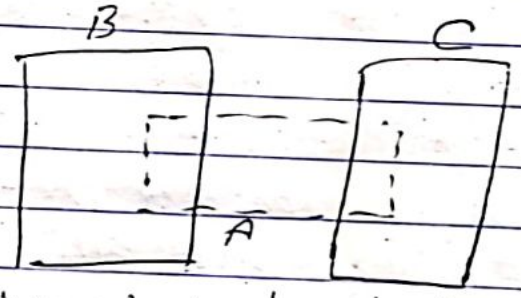
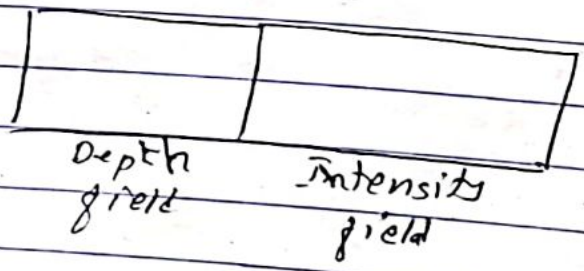


fig:- viewing two background surfaces B & C through a transparent surface A.

It extends frame buffer of foreground surface hence it can point all the background surfaces that are visible.

A buffer has two fields



(1) Depth field :-

→ It stores +ve or -ve depth value for a particular surface.

→ If it is +ve then that surface is opaque surface, otherwise it is transparent / translucent surface.

Scanned by CamScanner

ii) Intensity Field :- It stores surface intensity value or pointer to background surfaces.

ⓐ → If depth is +ve then intensity field contains color value of that surface.

ⓑ If depth is -ve then intensity field contains pointer to all background visible surfaces

→ Intensity field can contain

- RGB information
- Surface Opacity
- Spatial Orientation
- Pointer value to other surfaces

d>0	I_A

ⓐ If surface is opaque

d<0	→	B	→	C	→

ⓑ If surface is transparent.

Pros & Cons of A - Buffer

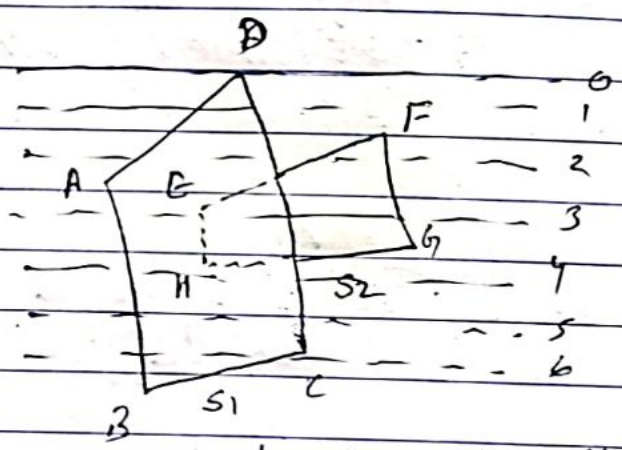
→ It deals with all types of surfaces.

→ Requires more computation than Z-buffer method.

→ Depth computation is always performed even if single surface is visible surface.

Scanned by CamScanner

Scan line Method



An efficient image space method that determine visible surface based on intersection of scan lines with polygon surfaces.

In this method depth calculation is performed only when there is a overlapping of multiple surfaces for a single pixels.

In all other cases depth calculation is not require and that surface which intersects with current pixel is visible for that pixel.

This algorithm starts processing from 0th scan line to Y-max.

It requires following types of data structures.

- Polygon Table :-
  It stores surface information
- Edge Table :-
  It stores edge information of each surface.
- Active Edge List (AEL) :-
  It stores all the edges that are intersected by current scanline.
- Flag :- A flag is maintained for each surface that can be set either 'ON' or 'OFF'.
  - For a particular polygon surface, flag is set 'ON' at left intersection point and flag is set 'OFF' at right intersection point.

Scanned by CamScanner

→ If flag is on for multiple surfaces then depth calculation is performed, otherwise depth calculation is not required.

Algorithm

1.) Initialize data structure :-

ⓐ Polygon Table :- Store A, B, C, D and RGB of each surface.

ⓑ Edge Table :- Store all edges for the polygon surface

ⓒ AEL :- { }

ⓓ Flag :- OFF for each surface

2.) For each scan line starting from 0th scanline

- For each pixel (x, y)

ⓐ Update AEL

ⓑ Update Flag for each surface

ⓒ If flag is 'ON' for single surface then intensity of that is stored in the frame buffer.

ⓓ If multiple flags are 'ON', then calculate depth for each surface and enter the intensity of that surface which has the smallest depth.

Pros & Cons of Scanline Method

→ Depth calculation is reduced.

→ Can deal with all types of surfaces

→ It doesn't require pre-sorting of Polygon Surfaces

→ Requires additional memory than A & Z-Buffer Method.

Scanned by CamScanner

Depth Sorting Algorithm / Painter's Algorithm

Another widely used object space method ~~in~~ ~~which~~ also called Painter's algorithm because simulates working behaviour of Oil painter. It first calculates depth values for each surface and all the surfaces in the scene are sorted decreasing order of depth values.

First it enters intensity value of the surface with is the farthest surface. and next it enters intensity value of next farthest surface and so on.

Finally, a surface which has the smallest depth is entered into the frame buffer. Frame buffer contains intensity values of visible surfaces only.

It requires additional process than other image space method.

Only applicable for a scene that contains few number of surfaces.

Scanned by CamScanner

Binary Space Partitioning Tree Method / BSP Tree method

→ It is an enhancement of depth sorting Algorithm, that divides entire 3D scene into two parts by selecting any polygon surface as a dividing / partitioning plane.

→ This method is best suited when a 3D scene is static but viewpoint changes frequently.

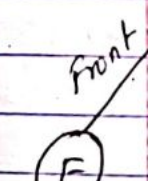
→ This image space method first displays polygon surfaces that are far from the viewpoint and enters their intensity into frame buffer.

→ It can be divided into two steps :-

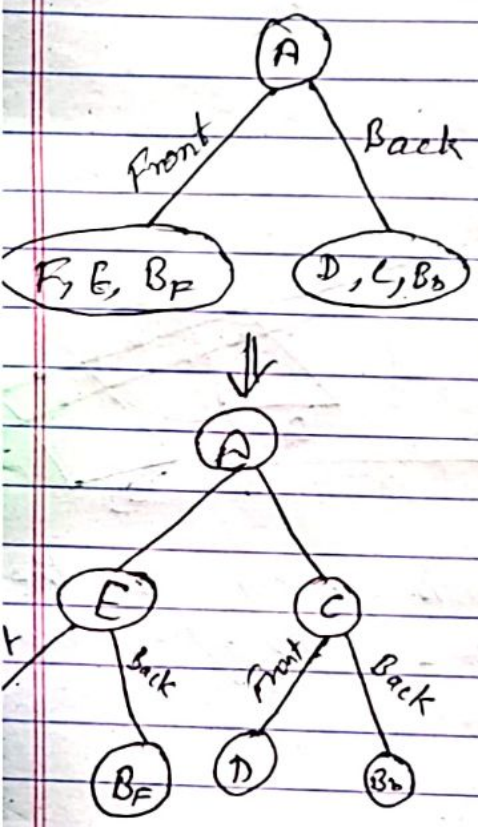
- Construction of BSP Tree
  - A ~~3D Scene~~ space that contains 3D scene is recursively divided into two parts i.e. FRONT & BACK with respect to viewpoint, by selecting any ~~a~~ surface as a dividing plane.
  - The recursive sub division process is repeated until each half space contains exactly one object or zero.
  - If particular object intersects partitioning plane than that object is divided into two parts ie Front part & Back Part

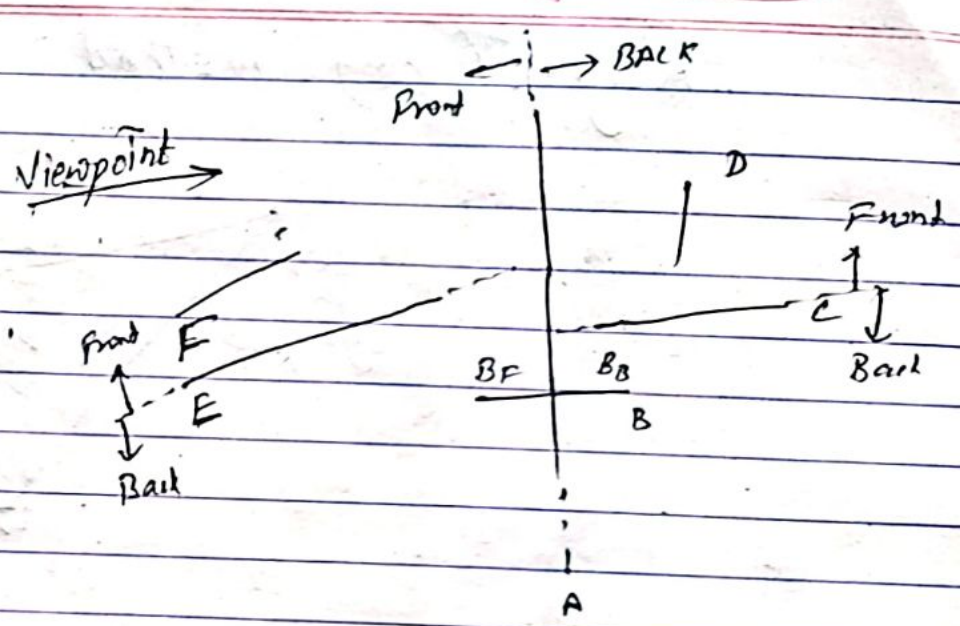
→ This recursive sub-division process is represented as a binary tree with initial partitioning plane as a root node.

→ FRONT objects is ~~are~~ represented as a left node of a tree and BACK object is represented as a right node of tree



Scanned by CamScanner

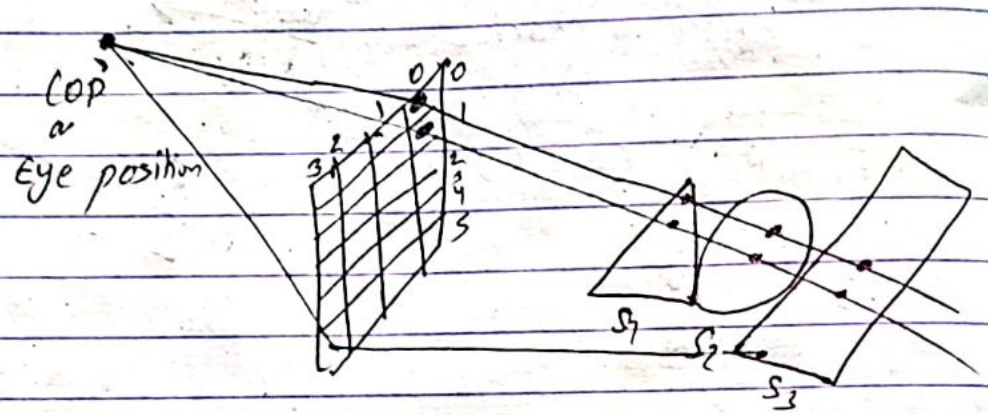




- Display of BSP tree
  - If viepoint is in front of the root node, then BSP tree is displayed in reverse inorder traversal i.e. right, root, and left.
    B_B, C, D, A, B_F, E, F
  - If viewpoint is in back of the root node, then BSP tree is displayed in ~~normal~~ inorder traversal. i.e. left, root and right
    F, E, B_F, A, D, C, B_B

Scanned by CamScanner

Ray Tracing method



→ light rays are casted from each pixel to the scene and visible surfaces are identified based on nearest intersection point.

For pixel
For pixel
For pixel

→ The number of casted light rays depends on the no. of pixels in the viewplane i.e. resolution

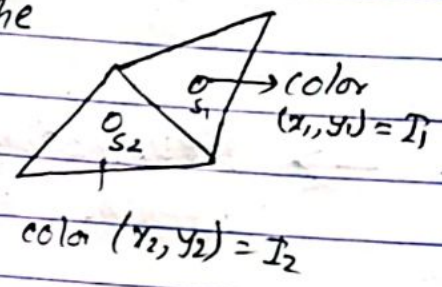
Scanned by CamScanner

Illumination Models and Surface Rendering

→ Illumination models are mathematical model to determine color calculation / intensity calculation of a single pixel within a particular surface.

→ Sometimes also referred to as a shading model.

→ The color seen on particular point of the surface depends on various optical parameters



i) Light Source Type

There are two lighting models in CG.

a) Point light Source

→ light rays radially diverge from a lightsource and there is a point of a ~~given~~ origin of light rays.

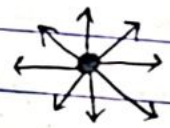


fig:- point light source

→ light intensity appears equal in all directions.

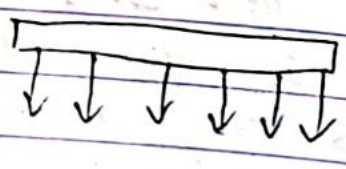
Eg:- Bulb, Sun

~~ii) Dis~~ b) Distributed light Source

→ There is no point origin and light ~~orig~~ rays are parallel to each other

→ light rays are focused to particular direction

Eg:- Laser light, Torch, light in the light house



Scanned by CamScanner

ii) Surface Characteristics

* It determines type of reflection from a particular surface.
* Surface can be rough, shiny, transparent and can produce specular reflection or diffuse reflection

→ Illumination model are necessary in CG for producing reliastic displays.

Types of Illumination Model

1.) Ambient light

$$\boxed{I = K_a I_a}$$

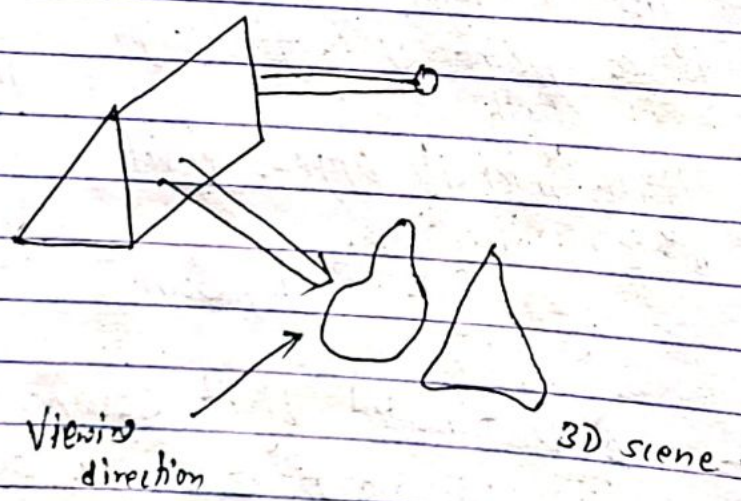


fig:- 3D scene visible due to ambient light

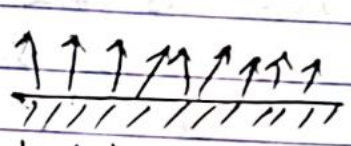
→ Objects that are not in direct ~~con~~ contact with light source, can still be visible if nearby objects are illuminated this is called ambient light,

→ It is constant in all direction irrespective of viewing direction.

Scanned by CamScanner

2.) Diffuse reflection

→ It is reflection due to rough regular surfaces.
→ Reflection of light is equal in all direction



→ It is often calculated using Lambert's cosine law
→ It is proportional to cosine of angle angle between surface normal vector and light direction vector.
ie $I \propto \cos\theta$

3.) Specular Reflection

→ It is reflection due to shiny surfaces.

fig : specular reflection

→ Maximum intensity appears in particular direction.
→ It is calculated using Phong model.

4.) Intensity Attenuation

→ The rate of decrease in intensity w.r.t distance between light source and objects is called intensity attenuation.
→ If point light source is used the intensity attenuation is given by $\frac{1}{d^2}$

Scanned by CamScanner

→ If distributed light source is used then intensity attenuation factor is given by a function

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

where a_0, a_1 & a_2 are surface parameters
d is distance betn object and distributed light source.

Surface Rendering / Surface Shading :

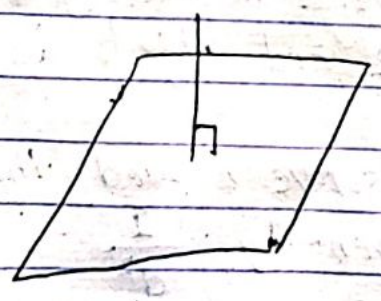
→ It is the process of calculating intensity for a particular surface
→ It calculates intensities of all surface points.
→ There are 3 approaches for surface rendering

i) Flat Shading / Constant Intensity Shading

→ In this method illumination model is applied by selecting arbitary pixel inside the surface and calculated intensity is applied to all other pixels inside the surface
→ It requires less computations but can't produce reliastic images.

Algorithm

→



Scanned by CamScanner

→ Calculate surface normal vector for each surface.
→ Apply illumination model to particular interior pixel to determine intensity value.
→ Assign calculated intensity value to all other pixels on the surface.

ii) Gouraud Shading / Intensity Interpolation Method

→ In this method, 1st average surface normal vector at each vertex is determined as
→ ~~In this method, av~~

$$\vec{N_V} = \frac{\vec{N_1} + \vec{N_2} + \dots + \vec{N_n}}{|\vec{N_1} + \vec{N_2} + \dots + \vec{N_n}|}$$

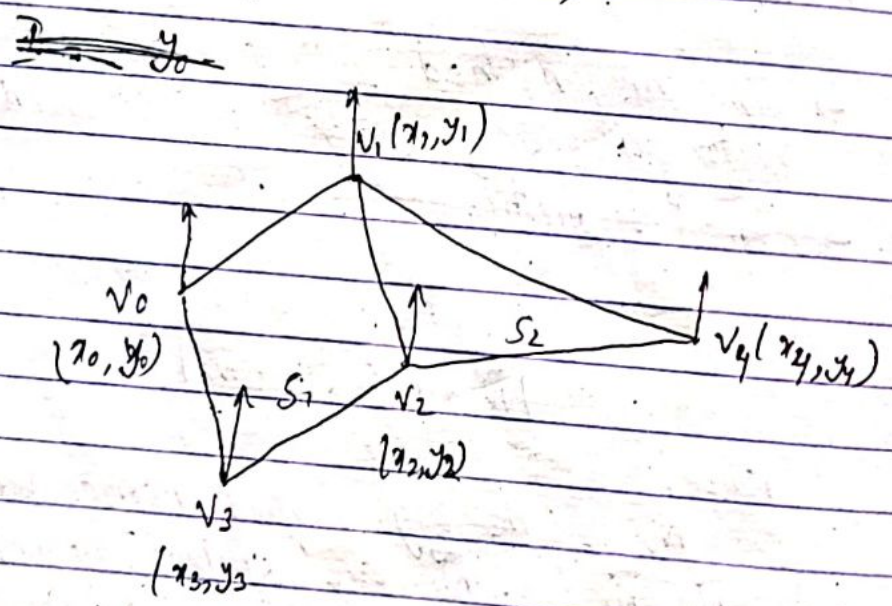
where,
$\vec{N_V}$ is the avg. surface normal vector at vertex V, and $\vec{N_1}, \vec{N_2} \dots \vec{N_3}$ are surface normal vectors on surfaces $S_1, S_2 \dots S_n$. Here vertex V is shared by all n surface.

→ Then illumination model are applied to each vertex to determine intensity value at that vertex.
→ These calculated intensities are interpolated to determine intensity value of all other pixels.
→ It provides more realistic graphics than flat shading but suffered by Mach-Band Effect. [Appereance of dark & bright spots at the corner of the objects surface.

Scanned by CamScanner

Algorithm

1.) Calculate average surface normal vector at each vertex.
2.) Apply illumination model at vertices to calculate intensity value.
3.) Apply interpolation to vertex intensities by using vertex coordinates and intensity at that vertex to determine intensities for all other pixel.



Phong Shading / Normal Vector Interpolation

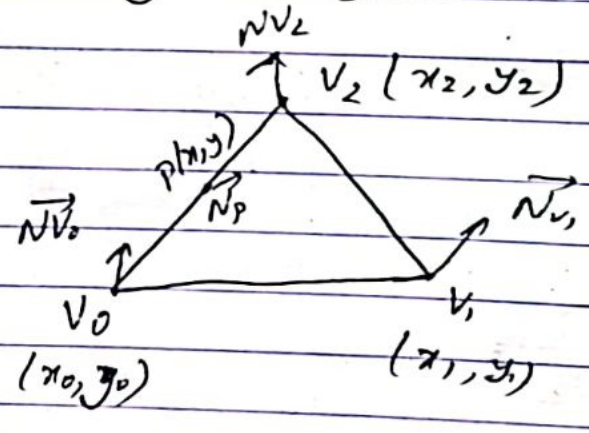
An improvement of Gouraud Shading suggested by Phong Bui Toung that interpolates average surface normal vector ~~that interpo~~ to calculate color of the surface.
It is the most efficient shading method but requires large number of computation.

Algorithm

Calculate avg. surface normal vector at each vertex.
Calculate / Apply interpolation to determine average normal vector for all other points by using average normal vector at vertices

$$\vec{N_P} = \frac{(y_2 - y)}{(y_2 - y_0)} \vec{N_{V_0}} + \frac{(y - y_0)}{(y_2 - y_0)} \vec{N_{V_2}}$$

$$\vec{N_{P_m}} = \frac{(x_m - x_0)}{(x_1 - x_0)} \vec{N_{V_1}} + \frac{x_1 - x_m}{(x_1 - x_0)} \vec{N_{V_0}}$$



Apply illumination model at each pixels to determine intensity value

Scanned by CamScanner