Introduction to computer Graphics

Computer Graphics is a subfield of computer science which studies methods for digitally synthesizing and manipulating visual content.

Today, There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer Graphics so widespread. We find computer graphics used routinely in such diverse area in science, engineering, medicine, business, industry, government, art, entertainment, advertising, education and training

Application of Computer Graphics:

a) Computer-Aided Design:

A major use of computer graphics is in design process, particularly for engineering and architectural systems, but almost all products are now computer designed. Generally, refered to as CAD, computer aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, texttiles and many more other products

b) Presentation Graphics:

Another major use of computer Graphics, used to produce illustration for reports or

to generate 35 mm slides or transparancies for use with projectors. Presentation graphics is comonly used to summarize financial, statistical, mathematical, scientific and economic data for research reports, managerial reports, customer information bulletins, and other types of reports. Typical examples of presentation graphics are barchart, line graphs, surface graph, pie-charts and other displays showing relationship between multiple parameter.

c) Computer Art:

Computer graphics methods are widely used in both fine art and commercial art application. Artist use a variety of computer methods, including special purpose hardware, artist's paintbrush programs such as Lumena, other paint package such as PixelPaint and SuperPaint, specially developed software like photoshop, CorelDraw. symbolic mathematics package like Mathematica that provides facilities for designing object shapes and specifying object motion.

d) Entertainment

Computer graphics methods are now commonly used in making motion pictures, music videos and television shows. Sometimes graphics scenes are displayed by themselves and sometimes graphics scenes are combined with actors and live scenes. Music videos, Animation, movies use graphics in several ways. & games.

e) Education and Training:

Computer-generated models of physical, financial and economic systems are often used as educational aids. Models of physical systems, physiological systems, population trends or equipment, such as color-coded diagram can help trainess to understand the operation of the system.

For some training, special systems like simulator for practice sessions or training of ship captains, aircraft pilots, heavy equipment operators and air traffic control personnel.

f) Visualization

Scientist, engineers need to analyze large amounts of data or study the behaviour of certain po processes. satellite cameras are amassing large data files than they can be interpreted. scanning these files and converting into visual form, the trends and patterns are often immediately apparent. producing graphical representation for scientific, engineering data sets is generally referred to as scientific visualization.

A collection of data can contain scalar values, vectors, higher order tensors. or Mathematician, scientist use visual techniques to analyze mathematical function ie Graphs.

g) Image Processing :

Image Processing is a subdomain of computer Graphics. Although methods used in computer graphics and image processing overlap, computer graphics is used to create pictures. Image processing applies technique to modify or interpet existing pictures, suchas photographs and TV scans. image processing events such as: rearrange picture parts, enhance color seperation, improve quality of shading, adding effects etc.

Medical image processing
↳ CT scan, MRI, X-Ray etc.

h) Graphical User Interfaces:

It is common now for software packages to provide a graphical ~~user~~ interface. A major component of a graphical interface is a window manager that allows a user to display multiple-window areas. Each window can contain a different process that can contain graphical or non-graphical displays. To make a particular window active, we simply click in that window using an interactive pointing device.

Interface can contain menu and dialog ~~device~~. box. An icon is a graphical symbol that is designed to look like the processing option it represents.

* Generally, Computer Graphics is composed of two different words ie computer + Graphics where
computer is a electronic device which takes input, process it and gives human readable output
Graphics is that in which output is in visual form.
Computer Graphics can be defined as a electronic device that takes input, process it and gives visual form output. Computer graphics is uptrend, so it is widely accepted technology. Human mind is oriented move towards image rather than text. The computer graphics cover almost all field except audio field like radio and music (persistence of image). video is also image ie still image of certain frame per sec (Generally 24 FPS).

pixel is minute element of picture. pixel is also called picture ~~imo~~ element.

13 MP = 13M pixel
= 13×10^6 pixel

Mega pixel is a factor of quality but it is not ultimate.

Computer Graphics is a creation of image → CGI (Computer generated imagery)

24 bit for 1 pixel → true colors.

Augmented Reality

Augmented reality is the blending of virtual reality and real life, as developers can create images within application that blend in with content in real coord. with AR, users are able to interact with virtual contents in real world and are able to distinguish between the two

examplge: flying shartiman, Life of pie.

Virtual Reality

virtual reality is all about the creation of a virtual world that users can interact with. This virtual world should be designed in such a way that users would find it difficult to tell the difference from what is real and what is virtual. furthermore, VR is usually achieved by the wearing of a VR helmet or ~~google~~ goggles similar to the oculus Rift.

History of a computer Graphics :

The history of the computer graphics can be study as a ~~chronological~~ chronical development of hardware and software. The evolution of computer Graphics under various terms are expect on following points

- The whirlwind computer developed in 1950 at Massachusetts institute of Technology (MIT) had computer drive CRT displays for output, both for operator use and campra producing hard copy

- The SAGE air defence 1950's was the first to use command and control CRT displays console on which operators identified target with lightpen

- The begining of Modern interactive graphics, however were found in IVAN SUTHERLAND's seminal doctoral work on the sketchpad drawing system (used for drawing CRT symbol)

- By the mid Sixties, a number of research project and commercial product had appeared as the potentially CAD activities in computer, automobile and aerospace grew enormously for automating drafting-insensitive activities. The general motor system for automobile design and intek Digiter System for lens design were pioneers in showing the efforts utilizing graphics interaction in interactive cycles common in engineering.

Chapter 2. Graphics Hardware

Cathode Ray Tubes:

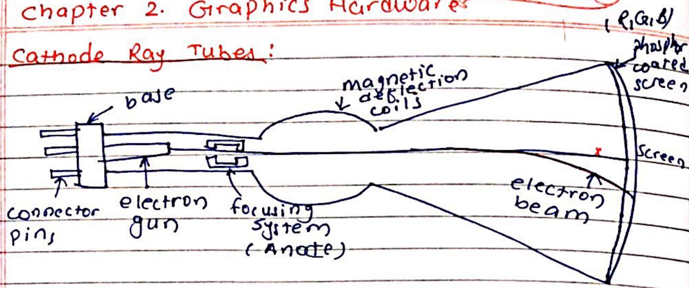
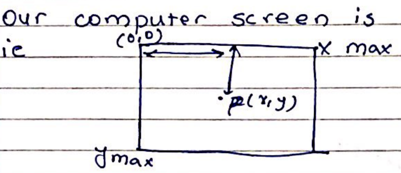


Fig: Basic Design of magnetic do

Our computer screen is Graph paper.
ie

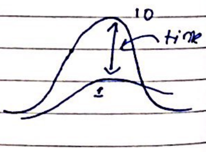


magnetic deflection coil is ~~to~~ used deflect the electron beam in which cordinate of screen should · repel → upward, attract ← downward

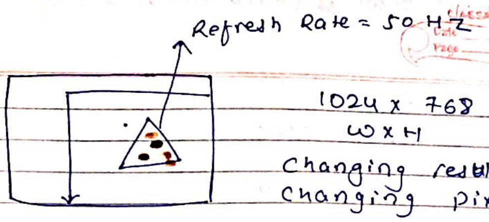
Frame buffer → To store frame content.
↳ which color.

Black and white → 1 electron ~~beam~~ Gun
Color → 3 electron (R,G,B)

Persistence : (अद्भुत)



its unit is in nanosecond
Intial intensity decrease to $\frac{1}{10}$. It gradually fade out. persistence time



1024 x 768
W x H

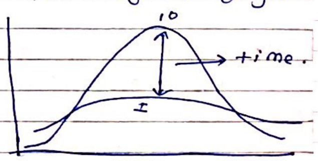
Changing resolution is changing pixel

Eg: जस्तै persistence time 1/10 छ भने त्यो भन्दा पहिला ~~क~~ कोटी strike गर्यो भने त्यो image देखिरहन्छ

Refresh Rate = 50 HZ = 50 FPS
in 1 sec, 50 images are strike on same point.

Persistence

Persistence is defined as the time it takes the emitted light from the screen to decay to 1/10th of its original Intensity.



Types of Persistence:
- lower Persistence
- Higher Persistence

we prefer lower persistence → animation
for still images → Higher Persistences.

Resolutions : The maximum number of points that can be displayed without overlap on a CRT is Referred to as the resolution. A more precise de resolution is no. of points per centimeter that can be plotted horizantally and

vertically.

Eg: 1024 x 768
W x H

1024

768

Aspect Ratio :

This number gives the ratio of vertical point to ~~the~~ horizontal point necessary to produce equal length line in both directions on the screen.

Eg: 16:9, 3:4

3:4 → 1cm 1cm

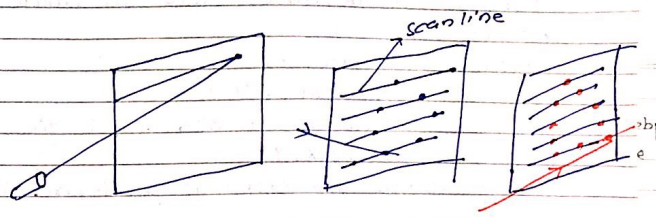
↳ it needs 3 pixel in vertical
4 pixel in horizontal

N.B) Calculate the size of frame buffer for resolution of 1024 x 768 and pixel depth is 8 bit

Number of pixels = 1024 x 768
pixel depth = 8 bit

Now
size of frame buffer = 1024 x 768 x 8 bit
= 768 KB
= 1024 x 768 x 8 / 8 x 1024 x 1024 MB

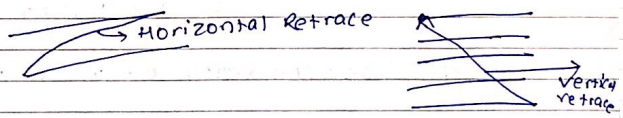
Raster scan display :



left → right 50 Hz → 1 sec, 50 scan

Fig: Raster (Image) scan system displaying triangle.

Scan line is a imaginary straight line in a screen.



Scan Conversion :

This line has infinite no. of dot. so conversion of infinite no. of dot into finite no. of pixel

Analog to discrete pixel is scan conversion.

→ Scan conversion is a process of converting basic, low level objects into their corresponding pixel map representations. This is often an approximation to the object, since the frame buffer is a discrete grid.

Color data is stored in a frame Buffer. This is sometimes called an image map or bitmap.

primitive operations:

- setpixel (x, y, color)

→ sets the pixel at position (x, y) to given color

- getPixel (x, y)

→ Gets the ~~pix~~ color at ~~pixit~~ pixel at pos(x, y)

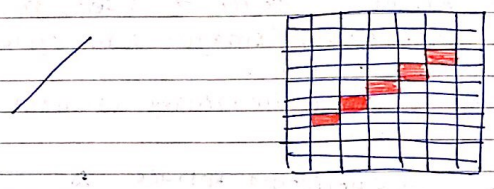


fig: Scan conversion of a circle

Raster and

Frame Buffer

Picture definition is stored in a memory area called refresh Buffer or frame Buffer. This memory area holds the setup intensity values for all the screen point.

1024 × 768 obj-e

1	0	1	0	0
0	0	1	0	0

On = 1 , black
Off = 0 , white

fig: Frame Buffer for monochromatic display.

Voltage control ~~A~~ control ~~JT~~ digital ~~9~~ ← frequency

for 2 color → 1 bit

for 8 color = 2^3 = ~~8~~ 3 bit required.

Stored intensity values are then retrieved from the refresh buffer and painted on the screen, one row at a time.

Pixel :

Each screen point is reffered to as pixel or pel which is the ~~cert~~ shortend form of picture element. The capability of a raster scan system to store intensity information for each screen point makes it well suited for realistic display of complex picture.

Illustrator → vector scan.

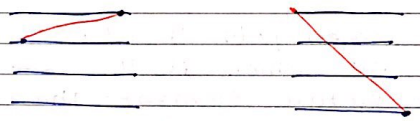
Photoshop → raster scan.

Bitmap and pixmap :

On the Black and white system, with 1 bit per pixel, frame buffer is commonly called a Bitmap. For system with multiple bit per pixel (pixel depth), frame buffer is often refferred to as a pix map

Horizontal retarace and vertical retrace

At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying next scan line. The return to the left of the screen, after refreshing each screen line is called the Horizontal retrace of the electron beam. And at the end of each frame, the electron beam returns to the top left corner of the screen to begin next frame is called vertical retrace



(HR) (VR)

If Frequency = 50Hz

VR = 4g

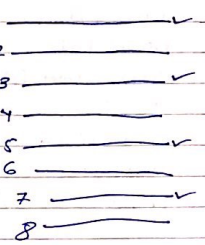
∴ VR = F-1

Resolution = 1024 x 768
= W x H

Frequency = 50Hz

HR = 767 x 50
= (H-1) x f

Interlaced refresh procedure : (flicker free)



Scan line = 8

Frequency = 30

first passe: odd = 1,3,5,7

second passe: even = 2,4,6,8

Time = $\frac{1}{F}$

T = ½ t ∴ f = 2f

adjacent scan line must contain same information. while reducing time to half, frequency is doubled.

→ On some raster scan systems, each frame is displayed in two passes using an interlaced refresh procedure. In the first pass, the beam sweep accross every scan line from top to bottom. Then after the vertical retrace, the beam sweep across remaining scan line.

Interlacing of the scan lines, in this way, allows us to see entire screen display in one half the time, it would have taken sweep across all the line at once from top to bottom. Interlacing is primarily used with CRT with slower refresh rate. On an older 30 frame per second (30 FPS), non interlaced display, the flicker is noticable but with interlacing we can increase the frequency nearly to the 60 Hz. This is an effective technique for avoiding flicker provided that adjacent lines contain similar display information

Numerical Problems: DPI = Dot per inch

a) Consider three different raster system with resolution
640 × 480, 1280, 1024, 2560 × 2048
what is the size of frame buffer is if pixel depth is 12 bit and true color (24 bit).

→ for 12 bit

a) size of frame = $640 \times 480 \times 12$
$= \frac{225}{512}$ MB

b) size of frame = $1280 \times 1024 \times 12$
$= \frac{15}{8}$ MB

c) size of frame = $2560 \times 2048 \times 12$
$= \frac{15}{2}$ MB

for 24 bit

a). size of frame = $640 \times 480 \times 24$
$= \frac{225}{256}$ MB

b) size of frame = $\frac{15}{4}$ MB

c) size of frame = 15 MB

∴ The size of frame = Resolution × pixel depth

2016 fall

b) Consider two raster system with resolution of 640 x 480, 1280 x 1024. How many pixel could be accessed per second in each of the system by a display controller that refreshes the screen at a rate of 60 FPS or 60 Hz. What is the access time for individual pixel in each system.

For first system

Resolution = 640 x 480

Total pixel = 307200 pixels

pixel accessed in one second (P)

= Total pixel x refresh rate frequency

= 307200 x 60

= 18432000 pixels per sec

Access time for individual pixel in each system = $\frac{1}{P}$

= $\frac{1}{18432000}$

= 5.4×10^{-8} sec

= 0.54×10^{-9} sec

= 0.54 ns

For second system

Resolution = 1280 x 1024

Total pixel = 1310720 pixel

pixel accessed in one second =

= total pixel x refresh rate

= 78643200 pixels

Access time for individual pixel

= $\frac{1}{P}$

= $\frac{1}{78643200}$ sec

c) Find out the aspect ratio of raster system of 8 x 10 inches screen and 100 pixels per inch

→ Screen resolution = 8 x 10"

Aspect Ratio = $\frac{\text{width}}{\text{height}}$

= $\frac{8}{10}$

= $\frac{4}{5}$

= 4:5

d) How much time is spent scanning across each row a pixels during screen refresh, on a raster system with a resolution 1280 x 1024 and refresh rate of 60 Hz.

→ Resolution = 1280 x 1024

scan line = 1024

60Hz = in 1 sec, 60 frame so

time = $\frac{1}{60}$ sec

Time Spent = $\frac{1}{1024 \times 60}$

= 1 scan line x Time for 1 scan line

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

= $\frac{1}{61440}$

e) Consider a RGB raster system is to be designed using 8 inch by 10 inch screen with a resolution of 100 pixel per inch in each direction. If we want to store 8 bit per pixel in the frame Buffer, How much storage (in bytes) do we need for the frame buffer ?

→ Screen size = (8 x 10) inch

Screen resolution = (8x100) x (10x100)

= 800 x 1000

Total pixel (p) = 800000

Bit per pixel (8) = 8 bit

Total Bit to store pixel in Frame Buffer

= Total pixel x Bit per pixel

= 800000 x 8 bits

Total Bytes = 800000 bytes

∴ Therefore Total storage required in frame Buffer = 800000 bytes

Video Display Devices

Typically, primary output devices in graphics system is video monitor whose operation is based mostly on ~~started~~ standard cathode Ray Tube (CRT) design

Cathode Ray Tubes (CRT)

CRTs are most common display devices on computer. A CRT is an evacuated glass tube, with a heating element on one end and a ~~philos~~ phosphor-coated screen on the other end.

- A beam of electrons (cathode rays) emitted by an electron gun, passes through focusing and deflection systems that direct beam toward specified positions on the phosphor coated ~~system~~ screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture. " One way to keep the phosphor glowing by quickly directing the electron beam back over the same points. This type of display is called a refresh CRT

- The primary components of an electron gun in a CRT are the heated metal cathode and control grid. Heat is supplied to cathode by directing a current through a coil of wire called Filament, inside the cathode.

This causes the electrons to be 'boiled off', the hot cathode surface. Negatively charged are then accelerated towards the ~~Philosphor~~ phosphor coating by a high positive voltage.

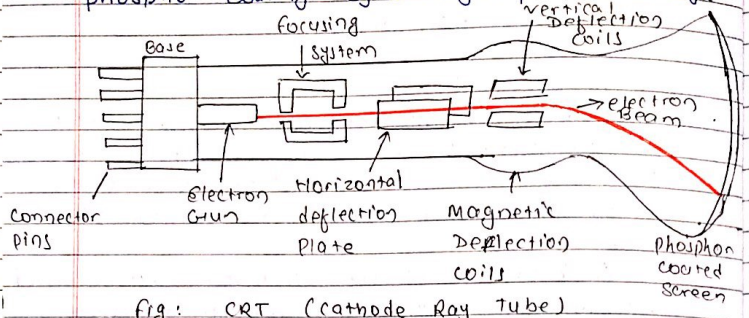


fig: CRT (Cathode Ray Tube)

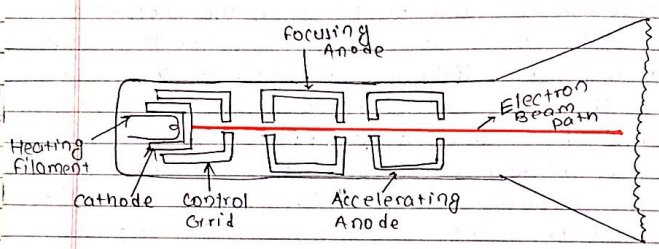


fig: Operation of an electron Gun

Control grid is an electrode used in amplifying thermonic waves, is used to control the flow of electrons from cathode to anode.

The intensity of the electron beam is controlled by setting voltage levels on the control grid. A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stoping them from passing through the small hole at the end of control grid structure. A smaller negative voltage on the control grid simply decreases the number of electrons passing through. Since the amount of light emitted by the phosphor coating depends on the electrons striking the ~~bea~~ screen, we control the brightness of a display by varying the on the control grid. We specify the intensity level for individual screen positions with graphics software commands

Display Technologies:

A. Raster Scan Display:

→ The most common type of graphics monitor employing a CRT is the raster scan display. based on television Technology.

→ In raster-scan the electron beam is swept accross the screen, one row at a time from top to bottom. Number of scan line per second is called horizontal scan rate.

→ As electron beam moves accross each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

→ Picture definition is stored a memory called frame buffer or refresh buffer. Frame Buffer holds all the intensity value for screen points.
→ Stored intensity values are then retrieved from the frame buffer and 'painted' on the screen one row (scan line) at a time.
→ Each screen point is refferred to as a pixel or pel (picture element).
→ Availability of frame buffer makes raster scan display as well suited for the realistic display.
Example: Monitors, Home Television, Printers.

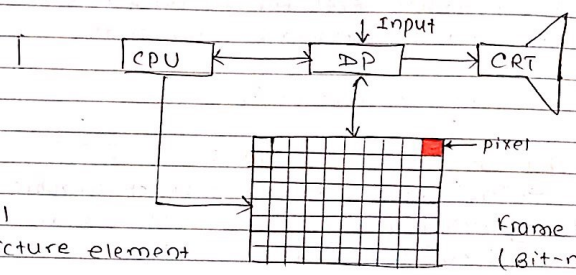
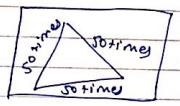


Figure: Raster scan display system

→ The frame buffer in Black/white system stores a pixel with one bit per pixel so it is termed as bitmap.
The frame buffer in multi bit per pixel storage is called pixmap

→ Refreshing on Raster-Scan display is carried out at the rate of 60 or higher frames per second. Sometimes refresh rates are described in units of 'cycles per second' or 'hertz' Hz where cycle corresponds to one ~~form~~ frame.

Random scan Display (vector/stroke writting or calligraphic)
In Raster scan display, refresh buffer
In Random scan Display, Display file
50 Hz mean each line drawn 50 times
if it exceeds 60 Hz, phosphor begins to burnt out.



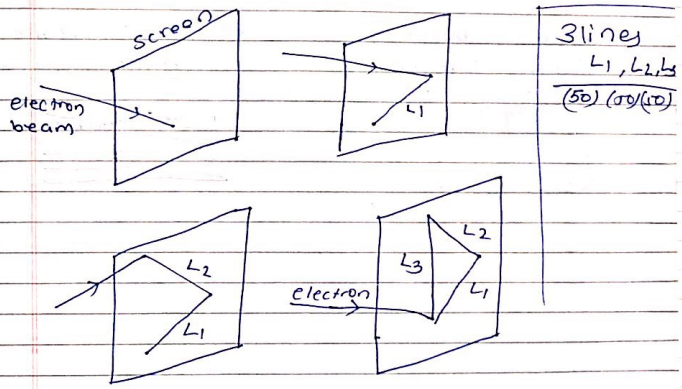


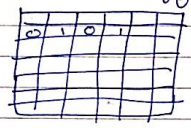
Fig: Random scan display.
(composition of lines

When operated as a random scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random scan Monitors draw a picture one line at time and for this reason, are also referred to as vector display or stroke writing or calligraphic display. The component lines picture can be drawn and refreshed by random scan system in any specified order

3 lines

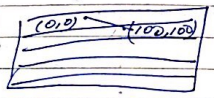
Raster

Refresh buffer / frame buffer.



Random

refres display file
display list / display program



Architecture of raster scan system and random scan system

How many individual component?
communication between component?
block digram.

Raster scan

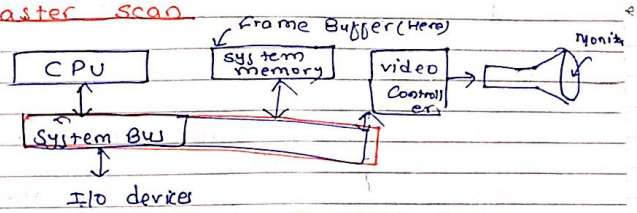


Fig: Architecture of raster system

Video controller → (Digital to Analog) voltage

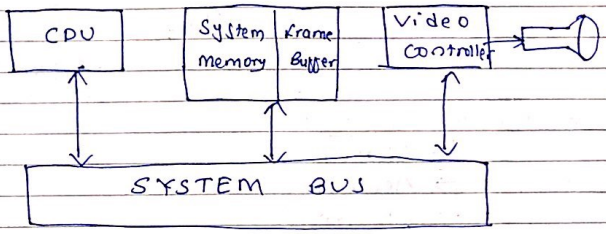


Fig: Architecture of raster system with fixed memory for frame buffer

Architecture of Raster scan:

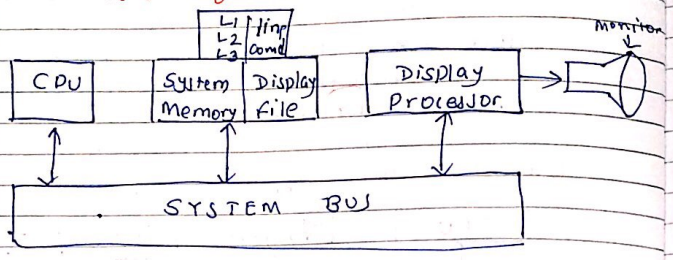
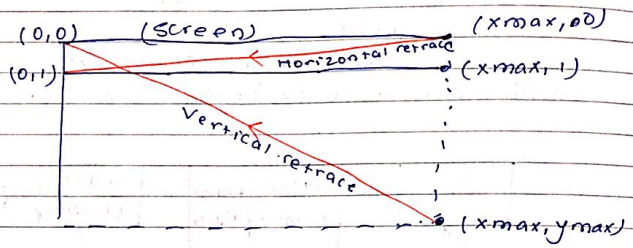


Fig: Random scan system.



* Touch panel:

- → Optical
- → Acoustic
- → Electrical
- → Capacitive

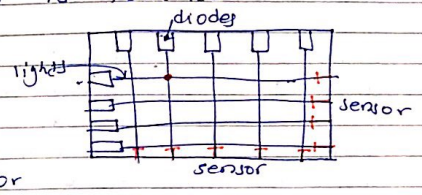
Capacitive:

(+) based on capacitor, it store charge coloumn ~~मा~~ charge store ~~हुन्छ~~।

Uniform capacitance through all panel when we touch portion, voltage drop and backend ~~मा~~ (x,y) cordinate.

Optical touch Panel:

Optical touch panel employ line of infrared ray, light emiting diode (LED) along one vertical edge and along one horizontal edge of the frame. They opposite horizontal and vertical edges contain light detector. These detectors are used to records which beam are interrupted when panel is touched.



• → touches which x sensor and which y sensor are blocked (x,y)

Fig: optical touch panel

The two crossing beams that are interrupted identifying horizontal and vertical cordinate of the screen select position selected. Position can be selected with an accuracy of an about one fourth (¼) inch.

Acoustic touch Panel:

In Acoustical touch panel, High frequencies sound waves are generated in the horizontal and vertical direction accross a glass plate

330 mls

Touching the screen causes part of each wave to be reflected from the finger to the emitter

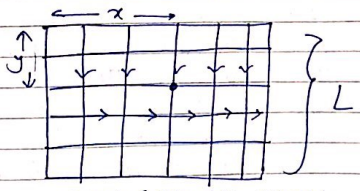
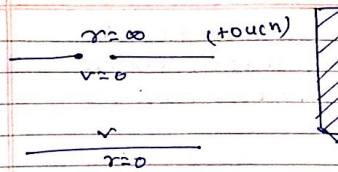


Fig: Acoustic panel

The screen position at the point of contact is calculated from a measurement of the time interval between the transmission of each wave and its reflection to the emitter

3) Electrical touch Panel :

An electrical touch panel is constructed with two transparent plates separate by the small distance. one of the plate is coated with conducting material and other plate is coated with resistive material. When the outer plate is touched, it is forced in to contact with inner plate. This contact creates the voltage drop across the resistive plate that is converted to the cordinate values of the selected screen position.



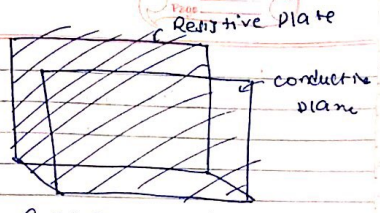
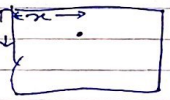


Fig: Electrical touch panel

**) light Pen :

It generates pulse of signal photo electric (sensor) generates high spark of light. sensor in tip when greater intensity. Sense is been detected in screen. It active high intensity light. when it touch then cordinate is get.



light pen work only on CRT

Color generation Techniques:
> Beam Penetration method.
> Shadow mask method.

1) Beam Penetration Method:

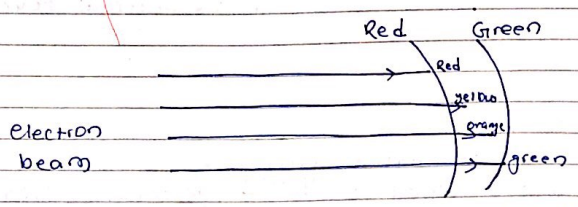
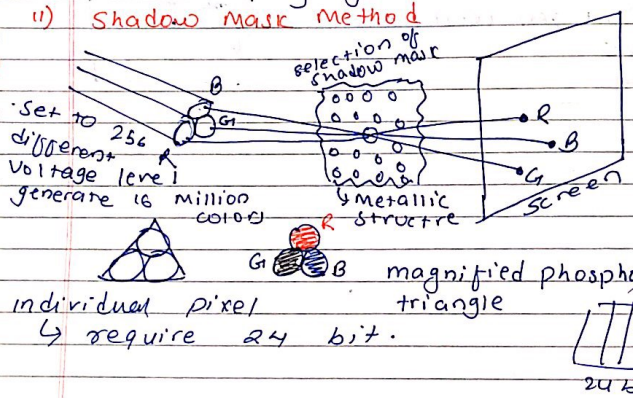


Fig: Beam penetration Method

Used on Random Scan.
Cost minimum

→ It can only generate few colors

11) Shadow Mask Method



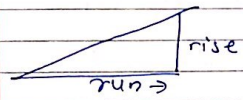
individual pixel

↳ require 24 bit.

Red	Green	Blue	Output
ON	OFF	OFF	RED
OFF	OFF	ON	Blue
OFF	ON	OFF	Green
ON	ON	ON	White
OFF	OFF	OFF	Black

Chapter - 3 2D Algorithm

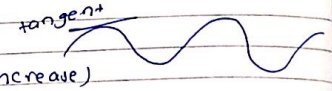
1) line drawing



$y = mx + c$

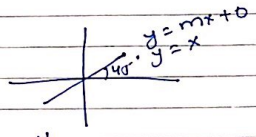
$\text{slope} = \frac{\text{rise}}{\text{run}}$

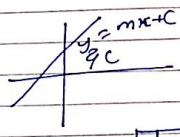
$\text{Slop} = \tan\theta = m = \frac{\text{rise}}{\text{run}} = \frac{\Delta y}{\Delta x} =$

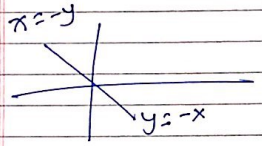


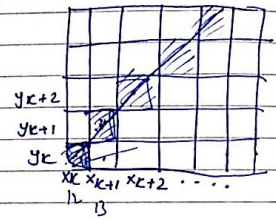
+ve slope
(x increase → y increase)
x decrease → y decrease)

-ve slop
(x increase → y decrease)
(y increase → x decrease)









i) Digital Differential Analyzer (DDA)

a) **Case 1 :**

$|m| < 1 \quad |m| < \frac{\Delta y}{\Delta x} \quad \Delta x > \Delta y$

we sample the line along x-direction

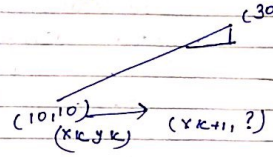
$m = \frac{\Delta y}{\Delta x} \quad \text{---} \quad (1)$

$x_{k+1} = x_k + 1 \quad \text{---} \quad (II)$

$y_{k+1} = y_k + \Delta y \quad \text{---} \quad (III)$

$m = \frac{\Delta y}{\Delta x}$

$\therefore m = \Delta y \quad [\Delta x = 1 \therefore x_{k+1} - x_k]$



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

Repeat Δx times [ie 30-10]

b) **Case 2**

$|m| > 1 \quad |m| > \frac{\Delta y}{\Delta x} \quad \Delta y > \Delta x$

Here we sample doing y-direction

$y_{k+1} = y_{k+1} \quad \text{---} \quad (I)$

$m = \frac{\Delta y}{\Delta x}$

$\Delta x = \frac{1}{m} \quad \text{---} \quad (II) \quad [\Delta y = 1]$

$x_{k+1} = x_k + \Delta x$

$x_{k+1} = x_k + \frac{1}{m}$

Repeat Δy times

Case-III $|m| = 1$

we can sample on either direction

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

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

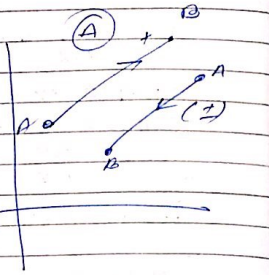
repeat Δx or Δy times

In General

Case I : $|m| < 1$

$x_{k+1} = x_k \pm 1$ (Δx)

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



Case-II

$y_{k+1} = y_k \pm 1$ (Δy)

$x_{k+1} = x_k \pm \frac{1}{m}$

Case-III

$x_{k+1} = x_k \pm 1$ (Δx or Δy)

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

(N.P) Digitalize line with endpoints (10,15) to (15,30) ($m > 1$)

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{30-15}{15-10} = 3$

$|m| > 1$

The process is repeat Δy times

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

$x_{k+1} = x_{k+1} \frac{1}{m}$

(10,15)

S.N	x_{k+1}	y_{k+1}
1	$10.33 \approx 10$	16
2	$10.66 \approx 11$	17
3	$10.99 \approx 11$	18
4	$11.32 \approx 11$	19
5	$11.65 \approx 12$	20

Algorithm:

		y_{k+1}
6	$10.98 \approx 12$	21
7	$12.31 \approx 12$	22
8	$12.64 \approx 13$	23
9	$12.99 \approx 13$	24
10	$12.32 \approx 13$	25
11	$13.64 \approx 14$	26
12	$13.97 \approx 14$	27
13	$13.30 \approx 14$	28
14	$14.63 \approx 15$	29
15	$14.95 \approx 15$	30

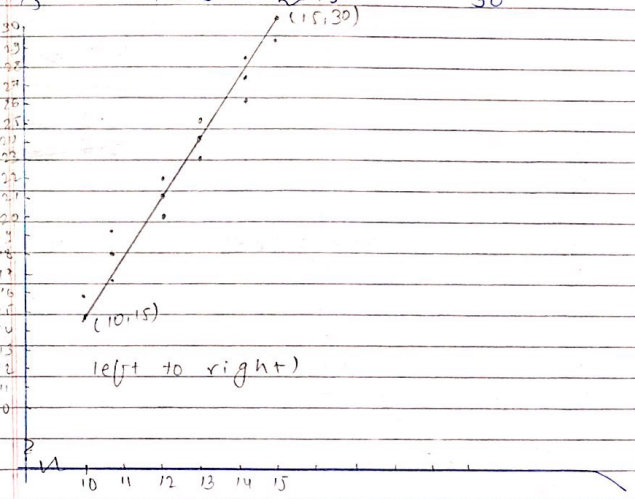


fig: Digitized line with end points (10,15) and (15,30)

Example for $|m| < 1$

Digitalize line with end point (15,15) to (10,18)

let $(x_1, y_1) = (10, 18)$

$(x_2, y_2) = (15, 15)$

$\therefore |m| = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-3}{5}$

$\therefore |m| = \frac{3}{5} = 0.6$

which is $|m| < 1$

The process is repeat Δx time (x increases)

15 | 15

S·N	$x_{k+1} = x_k + 1$	$y_{k+1} = y_k - m$
1	15	14.4
2		13.8
3		13.2
4		12.6
5		

10 | 18

S·N	$x_{k+1} = x_k + 1$	$y_{k+1} = y_k - m$
1	11	17.4 ≈ 17
2	12	16.8 ≈ 17
3	13	16.2 ≈ 16
4	14	15.6 ≈ 16
5	15	15 ≈ 15

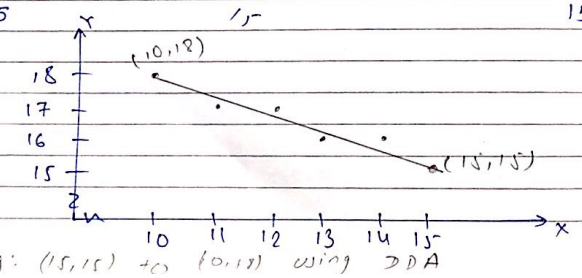


Fig: (15,15) to (10,18) using DDA

(Stair casing effect)

Digitalize line with endpoint (20,10) and (30,18)

let $(x_1, y_1) = (20, 10)$

$(x_2, y_2) = (30, 18)$

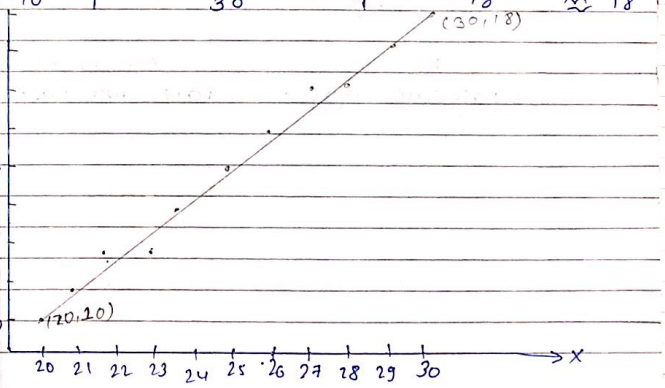
so $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8}{10} = 0.8$

$\therefore |m| < 1$

so, we repeat Δx times (10)

20 | 10

S·N	$x_{k+1} = x_k + 1$	$y_{k+1} = y_k + m$
1	21	10.8 ≈ 11
2	22	11.6 ≈ 12
3	23	12.4 ≈ 12
4	24	13.2 ≈ 13
5	25	14 ≈ 14
6	26	14.8 ≈ 15
7	27	15.6 ≈ 16
8	28	16.4 ≈ 16
9	29	17.2 ≈ 17
10	30	18 ≈ 18



Algorithm

1. Declare variable x_1, y_1 and x_2, y_2, delX, delY as real and k as integer
2. Perform
   $dx = x_2 - x_1$
   $dy = y_2 - y_1$
3. Test if $|dy| < dx$ then
   steps = |dx|
   else
   steps = |dy|
4. set delX = dx/steps
   delY = dy/steps
   $x = x_1$
   $Y = Y_1$
5. Plot (x, y)
6. for k = 1 to steps
   $x = x + delX$
   $Y = Y + delY$
   plot (x, y)

int (a+0.5)
12.3 = 12
12.6 = 13
12.6+0.5 = 13.1 = 13

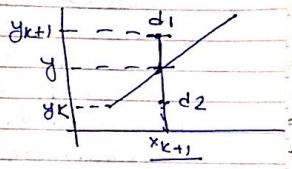
Drawback
Staircase effect

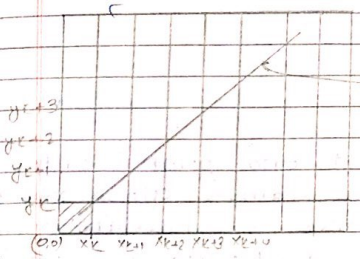
Advantage
fast and easy to compute

(Scan conversion Analog → Digital)

Bresenhams line Drawing algorithm (BLA)

$d_1 < d_2$ (x_{k+1}, y_{k+1})
$d_2 < d_1$ (x_{k+1}, y_k)





$y = mx + b$

$d_2 = y_{k+1} - y$
$d_1 = y - y_k$

Fig: section of a line represented by turning on discrete pixel position

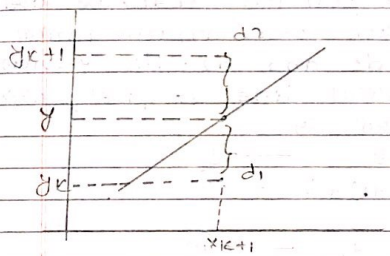


Fig: Distance between pixel position and the line y cordinate at sampling position x_{k+1}

Case-I

$|m| < 1$

candidate pixels :

(x_{k+1}, y_k) and

(x_{k+1}, y_{k+1})

if $d_1 < d_2$

(x_{k+1}, y_k)

else

(x_{k+1}, y_{k+1})

Starting from left end point (x_0, y_0) of a given line, we sample along x-direction and find respective y-cordinate which is closest to line path. As we sampling along x-direction. our choises are (x_{k+1}, y_k) and (x_{k+1}, y_{k+1})

At sampling position x_{k+1}, we label vertical separation from mathematical line path as d_1 and d_2. The y-cordinate on the mathematical line at pixel position x_{k+1} is calculated as

$$y = mx + b \quad \text{---(i)}$$

$$y = m(x_{k+1}) + b \quad \text{---(ii)}$$

then

$$d_1 = y - y_k \quad \text{---(iii)}$$

$$d_2 = y_{k+1} - y \quad \text{---(iv)}$$

From (ii) and (iv)

$$d_1 = m(x_k + 1) + b - y_k \quad \text{---(3)}$$

From (iv) and (ii)

(y_{k+1}, y_{k+1}) are variable

$$d_2 = (y_{k+1}) - y$$

$$= y_k + 1 - m(x_k + 1) - b \quad \text{---(4)}$$

The difference between those parameters

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1 \quad \text{---(5)}$$

A decision parameter P_k for the kth step in the line algorithm can be obtained by rearranging eqn (5). so that it involves only integer calculations we accomplish this task by substituting $m = \frac{\Delta y}{\Delta x}$ where Δy and Δx are horizontal and vertical separations in $m < 1$, Δx is greater than Δy so, multiply by Δx

$$P_k = \Delta x(d_1 - d_2) \quad \text{---(6)}$$

$$= 2\Delta y x_k - 2\Delta x y_k + c \quad \text{---(7)}$$

where $c = 2\Delta y + \Delta x(2b - 1)$

The sign of P_k is same as the sign of $(d_1 - d_2)$ since. $\Delta x > 0$ for our example.

if the pixel of y_k is closer to the line path at y_k is closer to the line path than the pixel at y_{k+1} (ie $d_1 < d_2$) then decision parameter P_k is negative, hence we plot lower pixel otherwise upper pixel.

$y = mx + b$

$b = y - mx$

$P_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2b-1)$

Therefore we can obtain successive decison parameter using incremental integer calculation

$P_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$ — (8)

Subtracting eqn (7) from (8)

$P_{k+1} - P_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$

$P_{k+1} = P_k + 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$

we calculate intial decision parameter P_0 for (x_0, y_0) and with m evaluated as $\frac{\Delta y}{\Delta x}$

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

P_0 (x_0, y_0) (x_{k+1})

$P_k = \Delta x(d_1 - d_2)$

$P_k = 2\Delta y x_k - 2$

we know,

$P_k = 2\Delta y x_k - 2\Delta x y_k + c$

$= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2b-1)$

$= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2(y - mx) - 1)$

$= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + 2\Delta x y - 2\Delta x m x - \Delta x$

$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y - 2\Delta x m x - \Delta x$

[∵ Replacing $x_k = x_0$ and $y_k = y_0$]

$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta x m x_0 - \Delta x$

since $m = \Delta y / \Delta x$

$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta x$

$= 2\Delta y - \Delta x$

BLA Algorithm for $|m| < 1$

1) Input two line end points and store left end point in (x_0, y_0)

2) load (x_0, y_0) into the frame buffer ie Plot first point

3) calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain starting value of decision parameter

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

4) At each x_k along the line path starting at $k = 1$ perform the following test

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

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

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

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

5) Repeat step 4 Δx times.

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

If $P_k < 0$ (x_{k+1}, y_k)

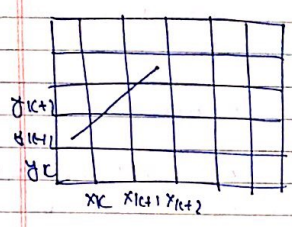
$P_{k+1} = P_k + 2\Delta y$

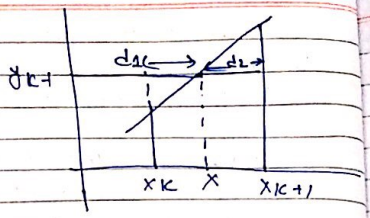
else, (x_{k+1}, y_{k+1})

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

BLA

Case - II $|m| > 1$





let (x_k, y_k) be the pixel position determined then the next pixel to the plotten either (x_{k+1}, y_{k+1}) or (x_k, y_{k+1}).

let d_1 and d_2 be the separation of the pixel positions (x_k, y_{k+1}) and (x_{k+1}, y_{k+1}) for the actual line path.

$y = mx + b$

The actual value of x is: $x = (y-b)/m$

Sampling position at y_{k+1}

from figure:

$d_1 = x - x_k$

$d_2 = x_{k+1} - x$

let us define a decision parameter P_k and P_k is defined by

$P_k = \Delta y (d_1 - d_2)$

$\Delta y > 0$

$P_k < 0$ if $d_1 < d_2$

$P_k \geq 0$ if $d_1 \geq d_2$

$P_k = \Delta y (d_1 - d_2)$

$= \Delta y (2x - x_{k+1} + x)$

$= \Delta y (2x - 2x_k + 1)$ [since $x_{k+1} = x_k + 1$]

$= \Delta y \{ 2\left(\frac{y_{k+1} - b}{m}\right) - 2x_k + 1 \}$

$= \frac{\Delta y}{m} \{ 2y_{k+1} - b) - 2x_k m + m \}$

$= \Delta x \{ 2(y_{k+1} - b) - 2x_k (\Delta y / \Delta x) - \Delta y / \Delta x \}$

$= 2\Delta x (y_{k+1} - b) - 2 \Delta y \cdot x_k - \Delta y$

$= 2\Delta x y_k - 2\Delta y x_k + 2(1-b)\cdot \Delta x - \Delta y$

$P_k = 2\Delta x y_k - 2\Delta y x_k + c$ ——— (ii)

let $c = 2(1-b)\Delta x - \Delta y$

for next step

$P_{k+1} = 2\Delta x y_{k+1} - 2\Delta y x_{k+1} + c$ ——— (iii)

from (ii) and (iii)

$P_{k+1} - P_k = 2\Delta x (y_{k+1} - y_k) - 2\Delta y (x_{k+1} - x_k)$

$P_{k+1} = P_k + 2\Delta x (y_{k+1} - y_k) - 2\Delta y (x_{k+1} - x_k)$

where $x_{k+1} - x_k = 0$ or 1

If $P_k \geq 0$

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

we plot $x_{k+1} = x_{k+1}, y_{k+1} = y_k + 1$

$P_k < 0$

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

$x_{k+1} = x_k$

for intial parameter

$P_0 = 2\Delta x y_0 - 2\Delta y x_0 + c$

$= 2\Delta x y_0 - 2\Delta y x_0 + 2(1-b)\Delta x - \Delta y$

$= 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - 2b\Delta x - \Delta y$

$= 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - 2(y_0 - mx_0)\Delta x - \Delta y$

$= 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - 2\Delta x y_0 + 2(\Delta y/\Delta x)\cdot x_0 \cdot \Delta x - \Delta y$

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

(20,10) to (30,18)

Soln

$|m| = 0.8 < 1$

$\therefore |m| < 1$

$\Delta y = 8$

$\Delta x = 10$

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

$P_0 = 2\Delta y - \Delta x = 16 - 10 = 6$.

Wrong

K	P_K	x_{K+1} (20)	y_{K+1} (10)	P_1
0	6	21	11	$P_1 = 6+16-20 = 2$

K	P_K	x_{K+1} (20)	y_{K+1} (10)	P
0	6	21	11	$P_1 = 6+16-20 = 2$
1	2	22	12	$P_2 = 2+16-20 = -2$
2	-2	23	12	$P_3 = -2+16 = 14$
3	14	24	13	$P_4 = 14-4 = 10$
4	10	25	14	$P_5 = 10-4 = 6$
5	6	26	15	$P_6 = 6-4 = 2$
6	2	27	16	$P_7 = 2-4 = -2$
7	-2	28	16	$P_8 = -2+16 = 14$
8	14	29	17	$P_9 = 14-4 = 10$
9	10	30	18	

$|m| < 1$

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

If $P_K < 0$ (x_{K+1}, y_K)

✓ $P_{K+1} = P_K + 2\Delta y$

else (x_{K+1}, y_{K+1})

✓ $P_{K+1} = P_K + 2\Delta y - 2\Delta x$

Repeat Δx

(15,18) to (10,15)

let

~~$(x_0, y_0) = (10,15)$~~ $\Delta x = 10-15 = -5$

~~$(x_1, y_1) = (15,18)$~~ $\therefore \Delta x = 5$

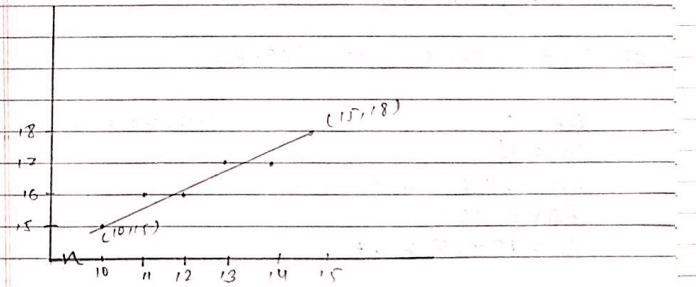
$\Delta y = 15-18 = -3$

~~$\therefore |m| =$~~ $\therefore \Delta y = 3$

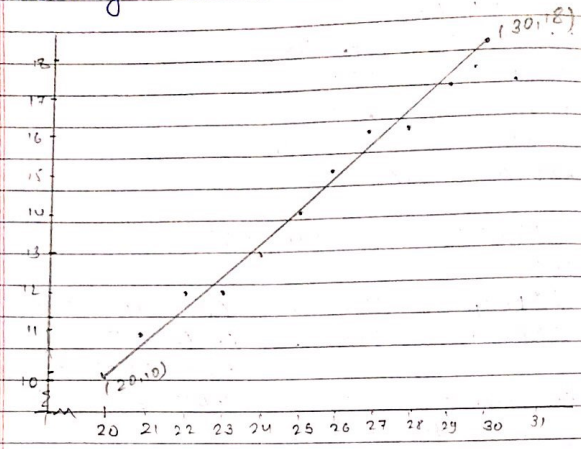
$m = 3/5 < 1$

$P_0 = 1$ repeat Δx +1 time

K	P_K	x_{K+1} (+0)	y_{K+1} (15)	P
0	1	11	16	$P_1 = -3 < 0$
1	-3	12	16	$P_1 = 3$
2	3	13	17	$P_2 = -1 < 0$
3	-1	14	17	$P_3 = 5$
4	5	15	18	



Plotting (20,10) to (30,18)



For |m| > 1

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

If $P_k \geq 0$ (x_{k+1}, y_{k+1})

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

If $P_k < 0$ (x_k, y_{k+1})

↳ $P_{k+1} = P_k + 2\Delta x$

Repeat Δy times

Plot (10,15) to (15,30)

$\Delta x = 5$

$\Delta y = 15$

$|m| > 1 : (3)$

$2\Delta x - 2\Delta y = -20$

$2\Delta x = 10$

$P_0 = 10 - 15 = -5$

$2\Delta x - 2\Delta y = -20$

$2\Delta x = 10$

K	P_k	x_{k+1}	y_{k+1}	P
		10	15	
0	-5	10	16	$P_1 = -5+10 = 5 > 0$
1	5	11	17	$P_2 = -15 < 0$
2	-15	11	18	$P_3 = -5 < 0$
3	-5	11	19	$P_4 = 5$
4	5	12	20	$P_5 = -15$
5	-15	12	21	$P_6 = -5$
6	-5	12	22	$P_7 = 5$
7	5	13	23	$P_8 = -15$
8	-15	13	24	$P_9 = -5$
9	-5	13	25	$P_{10} = 5$
10	5	14	26	$P_{11} = -15$
11	-15	14	27	$P_{12} = -5$
12	-5	14	28	$P_{13} = 5$
13	5	15	29	$P_{14} = -15$
14	-15	15	30	

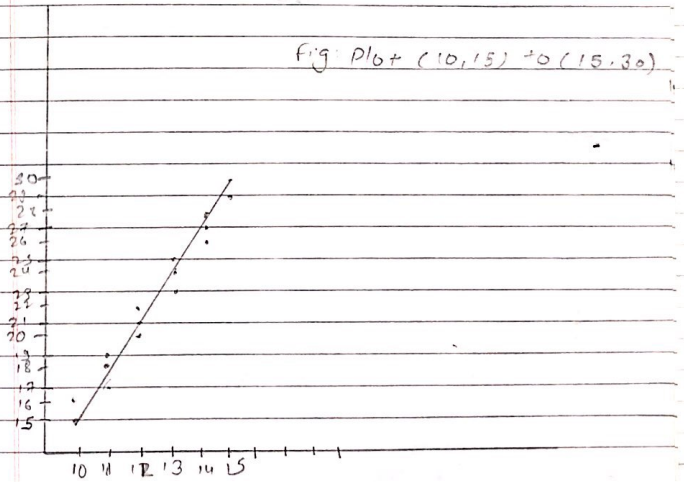


Fig Plot (10,15) to (15,30)

(10,24) (18,18)

$|m| = \frac{18-24}{18-10} = 0.75 < 1$

$\Delta x = 8 \qquad 2\Delta y = 12$

$\Delta y = 6 \qquad 2\Delta y - 2\Delta x = -4$

$P_0 = 4$

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

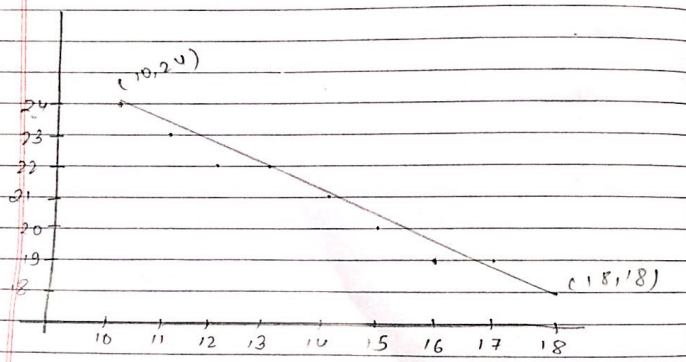
If $P_k < 0$ x_{k+1}, y_k

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

else x_{k+1}, y_{k-1}

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

		10	24	
K	P_k	x_{k+1}	y_{k+1}	P
0	4	11	23	0
1	0	12	22	-4
2	-4	13	22	8
3	8	14	21	4
4	4	15	20	0
5	0	16	19	-4
6	-4	17	19	8
7	8	18	18	



Circle generation algorithm (Breshanms mid point algorithm)

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

$x^2 = r^2 - y^2$

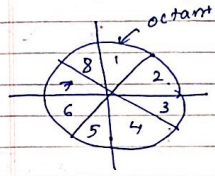
$x = \pm\sqrt{r^2 - y^2}$

x
y

y is independent and x is dependent

Mid point circle drawing algorithm

8 way symmetry



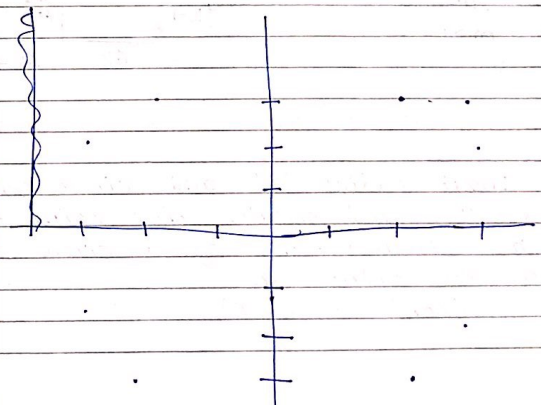
(x,y) (x,-y) (-x,y) (-x,-y)

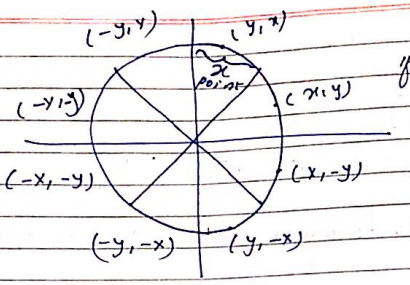
(y,x) (-y,x) (y,-x) (-y,-x)

let x = 2 and y = 3

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

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





$x \times 7 + x = 8x$

if $5 \times 7 + 5 = 40$

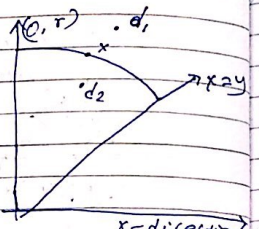
let $x^2 + y^2 = r^2$

$f_{circle} = x^2 + y^2 - r^2$

$f_{circle} = 0$ (x)

$f_{circle} < 0$ (d_2)

$f_{circle} > 0$ (d_1)

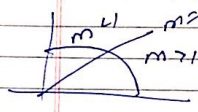


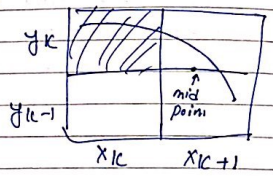
Candidate pixels

(x_{k+1}, y_{k-1}) $P_k > 0$

or

(x_{k+1}, y_k) $P_k < 0$





Midpoint Circle drawing Algorithm:

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

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

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

2) Calculate the intial value of decison parameter as

$P_0 = 5/4 - r \sim 1 - r$

3) At each x_k starting at $k=0$, perform the following test. If $P_k < 0$ the next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and

$P_{k+1} = P_k + 2x_{k+1} + 1$

otherwise, the next point along the circle is (x_{k+1}, y_{k-1}) and

$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_{k-2}$

4) Determine symmetry points in seven other octant

5) Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c)

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

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

Note: circle

$P_0 = 1 - r$

$P_k < 0 \quad (x_{k+1}, y_k)$

$P_{k+1} = P_k + 2x_{k+1} + 1$

else x_{k+1}, y_{k-1}

$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

Repeat until $x \geq y$

N.P) Digitize a circle $x^2 + y^2 = 10^2$

Soln:

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

$= (0, 10)$

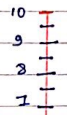
parameter $P_0 = 1 - r = 1 - 10 = -9$

K	P_k	x_{k+1} (0)	y_{k+1} (10)	$2x_{k+1}$	$2y_{k+1}$	P_{k+1} (mid-point)
0	-9	1	10	2	20	$-9+2+1 = -6$
1	-6	2	10	4	20	$-6+4+1 = -1$
2	-1	3	10	6	20	$-1+6+1 = 6$
3	6	4	9	8	18	$6+8+1-18 = -3$
4	-3	5	9	10	18	$-3+10+1 = 8$
5	8	6	8	12	16	$8+12+1-16 = 5$
6	5	7	7	14	14	

$x \geq y$

Applying 8 way symmetry

(x, y)	0, 10	1, 10	2, 10	3, 10	4, 9	5, 9	6, 8	7, 7
(x, -y)	0, -10	1, -10	2, -10	3, -10	4, -9	5, -9	6, -8	7, -7
(-x, y)	0, 10	-1, 10	-2, 10	-3, 10	-4, 9	-5, 9	-6, 8	-7, 7
(-x, -y)	0, -10	-1, -10	-2, -10	-3, -10	-4, -9	-5, -9	-6, -8	-7, -7
(y, x)	10, 0	10, 1	10, 2	10, 3	9, 4	9, 5	8, 6	7, 7
(y, -x)	10, 0	10, -1	10, -2	10, -3	9, -4	9, -5	8, -6	7, -7
(-y, x)	-10, 0	-10, 1	-10, 2	-10, 3	-9, 4	-9, 5	-8, 6	-7, 7
(-y, -x)	-10, 0	-10, -1	-10, -2	-10, -3	-9, -4	-9, -5	-8, -6	-7, -7



6) Determine the symetry points in other three ~~quadrants~~ quadrants

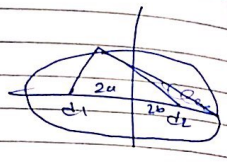
7) Move each calculated pixel position (x, y) into the elliptical path centered on (x_c, y_c) and plot the cordinates values

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

8) Repeat the steps for region 1 until ~~$2r_y^2 x$~~ $2r_y^2 x \geq 2$

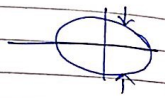
$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$



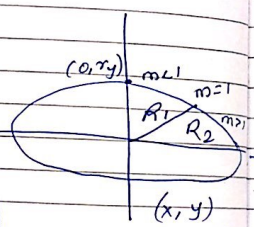
P_{10} → decision parameter for r_1

P_{20} → decision parameter for r_2



~~$a = r_x$~~ $2a = 2r_x$

~~so~~, $2b = 2r_y$



$$P_{10} = r_y^2 - r_x^2 r_y + \tfrac{1}{4} r_x^2$$

$P_{1k} < 0$ (x_{k+1}, y_k)

$$P_{1k+1} = P_{1k} + 2r_y x_{k+1} + r_y^2$$

else

x_{k+1}, y_{k-1}

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

(N.P) Digitize ellipse with $r_x = 8$, $r_y = 6$

Here,

$a = 8$

$b = 6$

$$P_{10} = r_y^2 - r_x^2 r_y + \tfrac{1}{4} r_x^2$$
$$= -332$$

$$P_{11} = P_{1k} + 2r_y^2 x_{k+1} + r_y^2$$
$$= -332 + 72 + 36 = -224$$

$P_{12} =$

$$-224 + 144 + 36 = -44$$

$P_{13} = -44 + 216 + 36$

$P_{14} = 208 - 640 + 288 + 36$

$P_{15} = -108 + 360 + 36$

$P_{16} = 288 + 432 - 512 + 36 = 244$

k	P_{1k}	x_{k+1}	y_{k+1}	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-332	1	6	72	768
1	-224	2	6	144	768
2	-44	3	6	216	768
3	208	4	5	288	640
4	-108	5	5	360	640
5	288	6	4	432	512
6	244	7	3	504	$\geq$ 384 (true)

Region 1 complete

for region 2

$$P_{20} = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

if $p^2_k > 0$, $(x_k, y_k - 1)$

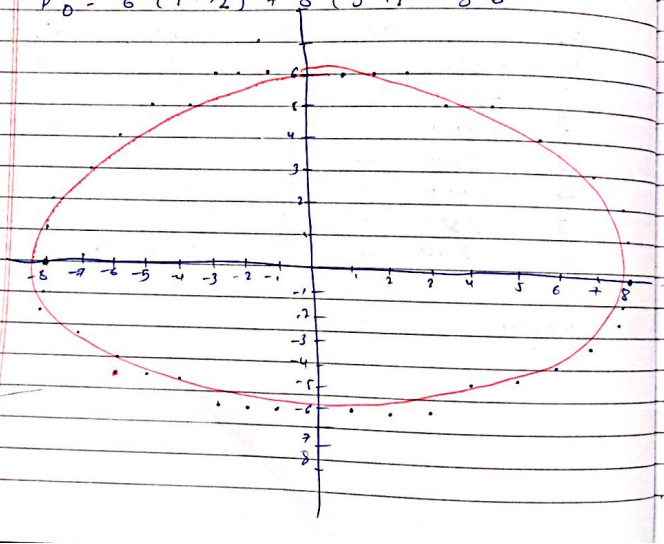
$$P_{2k+1} = P_{2k} - 2r_x^2 y_{k+1} + r_x^2$$

else x_{k+1}, y_{k+1}

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

k	p^2_k	x_{k+1}	y_{k+1}	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-23	8	2	576	256
1	361	8	1	576	128
2	297	8	0	—	—

$$p^2_0 = 6^2(7 + 1/2)^2 + 8^2(3-1)^2 - 8^2 6^2$$



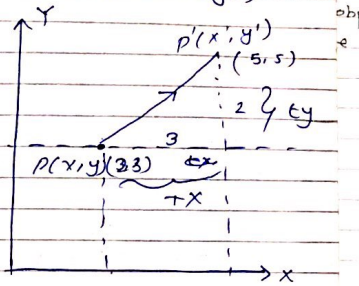
Chapter 4

4. Transformation

Rigid body Transformation

A. Basic Transformation

1) Translation (Position change)
2) Scaling
3) Rotation



1) Translation

$x' = x + tx$

$y' = y + ty$

where

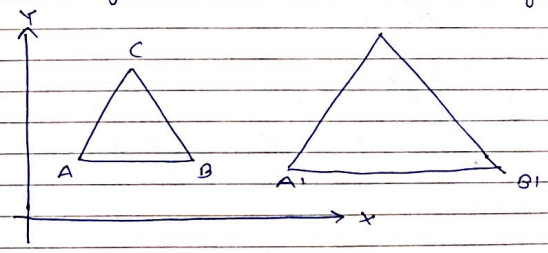
tx & ty are translating factor

Transforming

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

2) Scaling (Size change)

(inlarge or reduce the size of object)



$X' = X \cdot Sx$ $\quad x' = 2 \cdot 3$

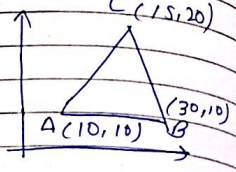
$Y' = Y \cdot Sx$ $\quad y' = 3 \cdot 4$

-ve → clockwise
+ve → Anticlockwise

If $S_x = S_y$ and S_x and $S_y > 1$
→ uniformly enlarge the object

If $S_x \neq S_y$ and S_x and $S_y \neq 1$
→ differential scaling

Example



$S_x = S_y = 2$
A'(20,20) B'(60,20) C'(30,40)

$S_x = S_y = 0.5$
A'(5,5) B'(15,5) C'(7.5,10)

$S_x = 2$, $S_x = 3$
A'(20,30) B'(60,30) C'(30,60)

* $S_x = 0.5$ $S_y = 0.25$
A'(5, 2.5) B'(15, 2.5)
C'(7.5, 5)

In matrix form

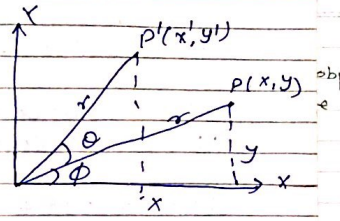
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation :

Degree
Direction (CW or ACW) — -ve, +ve

$x = r\cos\phi$ } — (i)
$y = r\sin\phi$ }

$x' = r\cos(\theta + \phi)$ } — (ii)
$y' = r\sin(\theta + \phi)$ }



Expanding equation (ii)

$x' = r\cos\theta \cdot \cos\phi - r\sin\theta \cdot \sin\phi$
$y' = r\sin\theta \cdot \cos\phi + r\cos\theta \cdot \sin\phi$

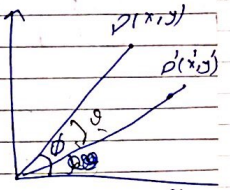
Substituting values of x and y from eqn (i)

$x' = x\cos\theta - y\sin\theta$ [$\because r\cos\phi = x$]
$y' = x\sin\theta + r\cos\theta$

In matrix form

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

For clockwise



$x = r\cos\phi$ } — (i)
$y = r\sin\phi$ }

$x' = r\cos(\phi - \theta)$ } — (ii)
$y' = r\sin(\phi - \theta)$ }

Expanding equation (ii)

$x' = r\cos\theta \cdot \cos\phi + r\sin\theta \cdot \sin\phi$

$y' = r\sin\theta \cdot \cos\phi - r\cos\theta \cdot \sin\phi$

$x' = r\cos\phi \cdot \cos\theta - r\sin\phi \cdot \sin\theta$

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

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

$x' = r\cos\theta \cdot \cos\phi + r\sin\theta \cdot \sin\phi$

$y' = r\sin\phi \cdot \cos\theta - r\cos\phi \cdot \sin\theta$

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

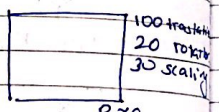
$y' = -x\sin\theta + y\cos\theta$

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

Homogenous cordinate system

$(x, y) \longleftrightarrow (x, y, 1)$

$2\times2 \qquad 3\times3$



① Translation

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

② Scaling :

$x' = x \cdot s_x$

$y' = y \cdot s_y$

$1 = 1$

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

③ Rotation

Anticlockwise

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

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

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

clockwise

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

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

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

or Acw

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

Things to remember :

$$\begin{bmatrix} 1 & 0 & +x \\ 0 & 1 & +y \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

2016 fall

(3/b) A point (5, 3) is required to be rotated about origin by 45° in cw direction and then scaled by a factor of 3. what will be final transformed position after applying these transformation

→ solution:

T_1 = Rotate by 45° in cw (−45°)

T_2 = scale by factor 3 ie S(3, 3)

Net transformation matrix (T) =

$$T = T_2 \cdot T_1$$

$$T = S(3,3) \cdot R(-45)$$

$$T = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

3×3 3×3

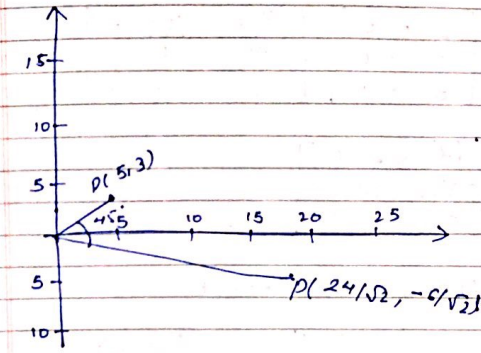
$$= \begin{bmatrix} 3/\sqrt{2} & 3/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So final transformed points

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 3/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$

3×3 3×1

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 15/\sqrt{2} + 9/\sqrt{2} \\ -15/\sqrt{2} + 9/\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 24/\sqrt{2} \\ -6/\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 17 \\ -5 \\ 1 \end{bmatrix}$$

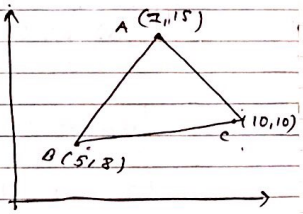


Rotate the triangle ABC 45° cw about Origin and scaling scale it by (2, 3) about origin

solution:

T_1 = Rotate ABC by 45° cw (−45°)

T_2 = Scale by factor 2 and 3 ie S(2, 3)



Net transformation matrix for A $T_A = T_2 \cdot T_1$

$$T_A = T_2 \cdot T_1$$

$$= S(2,3) \cdot R(-45)$$

$$=$$

$$T_A = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

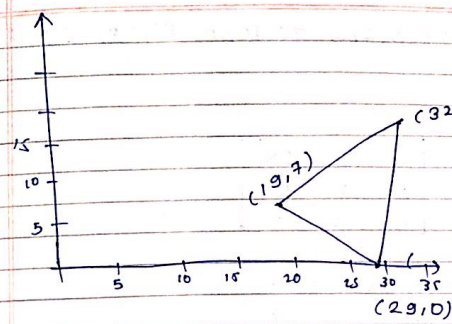
$$= \begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A'e Transformed Points

$$A' = \begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 1 \end{bmatrix} = \begin{bmatrix} 22\sqrt{2} \\ 12\sqrt{2} \\ 1 \end{bmatrix}$$

$$B' = \begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} 13\sqrt{2} \\ 9/\sqrt{2} \\ 1 \end{bmatrix}$$

$$C' = \begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 20\sqrt{2} \\ 0 \\ 1 \end{bmatrix}$$



OR

$$\begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} A & B & C \\ \end{matrix} \begin{bmatrix} 7 & 5 & 10 \\ 15 & 8 & 10 \\ 1 & 1 & 1 \end{bmatrix} = \begin{matrix} A & B' & C' \\ \end{matrix} \begin{bmatrix} 22\sqrt{2} & 13\sqrt{2} & 20\sqrt{2} \\ 12\sqrt{2} & 9\sqrt{2} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Rotate the triangle ABC by 90° anticlockwise about (5,8) and scale it by (2,2) about (10,10)

→ Solution

let:

Rotate:
- T_1 = translate (5,8) to origin ie T(-5,-8)
- T_2 = Rotate 90° ACW about origin ie R(90)
- T_3 = Inverse translation ie T(5,8)

Scaling:
- T_4 : translate (10,10) to origin ie T(-10,-10)
- T_5 : scale it by (2,2) about origin ie S(2,2)
- T_6 : inverse translation ie T(10,10)

Net translation (T) = $T_6\ T_5\ T_4\ T_3\ T_2\ T_1$

$= T(10,10)\ S(2,2)\ T(-10,10)\ T(5,8)\ R(90)\ T(-5,-7)$

we know,

$$T(Tx,Ty) = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(Sx,Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$AC\ R() = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -8 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(X)}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 8 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 13 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & -7 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -2 & 6 \\ 2 & 0 & -14 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -2 & 16 \\ 2 & 0 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

Net transformation matrix (T)

$$= \begin{bmatrix} 0 & -2 & 16 \\ 2 & 0 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

The final transformed cordinates are

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 0 & -2 & 16 \\ 2 & 0 & -4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 5 & 10 \\ 15 & 8 & 10 \\ 1 & 0 & 10 \end{bmatrix}$$

$$= \begin{bmatrix} -14 & -20 & -4 \\ 10 & 6 & 16 \\ 1 & 1 & 1 \end{bmatrix}$$

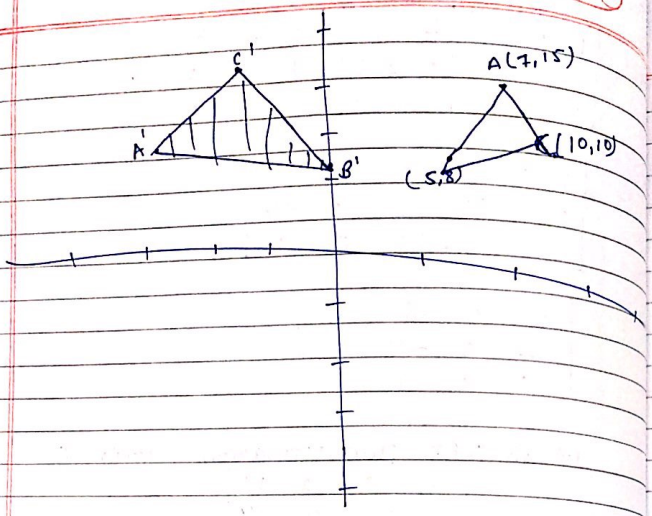


fig: original & transformed objects.

Inverse Transformation

$T^{-1}(tx, ty) = T(-tx, -ty)$

$R^{-1}(\theta) = R(-\theta)$

$S^{-1}(Sx, Sy) = S\left(\frac{1}{Sx}, \frac{1}{Sy}\right)$

Composite Translation

① Translations: (successive matrix)

If two successive translation vectors (tx, ty) and (tx_2, ty_2) are applied to a cordinate position P, the final transformed location p' is calculated as

$p' = T(tx_2, ty_2)\{T(tx_1, ty_1)\cdot p\}$

$p' = \{T(tx_2, ty_2)\cdot T(tx_1, ty_1)\}\, p$

$$p' = \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & tx_2 + tx_1 \\ 0 & 1 & ty_2 + ty_1 \\ 0 & 0 & 1 \end{bmatrix}$$

hence, successive translation are additive

Rotations

Taking two clockwise direction (rotation)

$p' = R(\theta_1)\cdot\{R(\theta_2)\cdot P\}$

$= \{R(\theta_1)\cdot R\theta_2\}\, P$

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

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

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

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

Scaling

$$P' = S(Sx_1, Sy_1)\{S(Sx_2, Sy_2) \cdot P\}$$
$$= \{S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)\} \cdot P$$
$$= \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

① Fixed point Rotation :

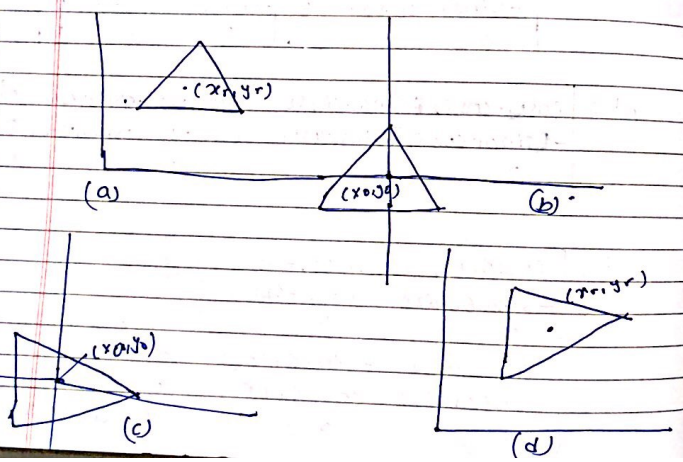


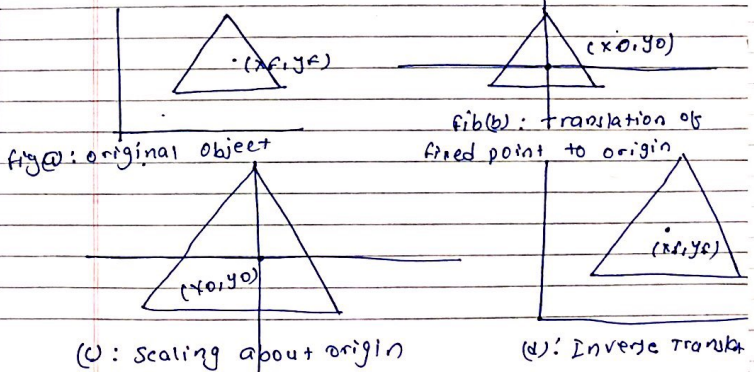
fig Description

a = original object x.
b - Translation of fixed point to origin
c = rotation about origin
d = Translate back to its original position

T_1 : $T(-x_r, -y_r)$ fig (b)
T_2 : $R(\theta)$ Fig (c)
T_3 : $T(x_r, y_r)$ fig (d)

$$T = T(x_r, y_r)\ R(\theta)\ T(-x_r, -y_r)$$
$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) + x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

② Fixed point scaling



T_2 : $S(s_x, s_y)$

T_3 : $T(x_f, y_f)$

$T = T(x_f, y_f)\ S(s_x, s_y)\ T(-x_f, -y_f)$

$$= \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$=$

$$= \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection

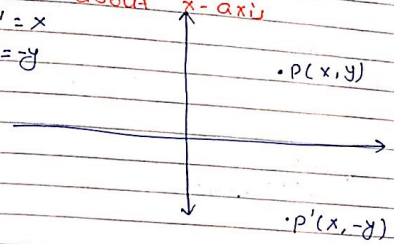
1> Reflection about x-axis

2> Reflection about y-axis

1) Reflection about x-axis

1> $x' = x$

$y' = -y$

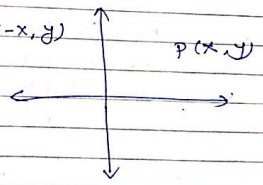


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2) Reflection about y-axis

$x' = -x$

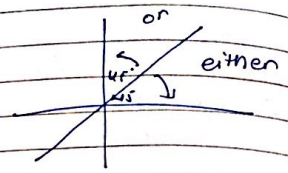
$y' = y$



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

Find the transformation matrix responsible for reflection about

1) $y = x$
2) $y = -x$
3) $y = mx + b$
4) $y = 2x + 3$



$R(45)$
R_{fy}
$R(-45)$ → R_{-45} R_{fx} $R(45)$

① About x

Net Transformation matrix

$T_1 = R(-45)$

$T_2 = R_{fx}$

$T_3 = R(45)$

$T = R(45)\ R_{fx}\ R(-45)$

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

$$T = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$R(\theta)$
$R(45)$ R_{fy} $R(-45)$

About Y

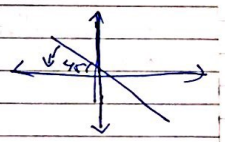
$T = R(-45)\ R_{fy}\ R(45)$

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

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

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

② 2) $y = -x$



$T_2 = R(-45)\ R_{fx}\ R(45)$

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

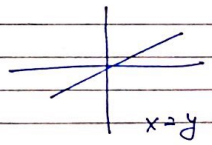
$$= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Trick: $y = -x$

$x' = -y$

$y' = -x$

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

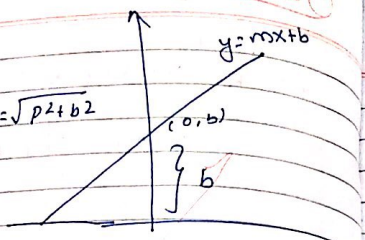


$x' = y$

$y' = x$

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

$y = mx + b$



$\tan\theta = m = \frac{p}{b}$ $\quad h = \sqrt{p^2+b^2}$

$\sin\theta = \frac{p}{h} = \frac{m}{\sqrt{1+m^2}}$

$\cos\theta = \frac{1}{\sqrt{1+m^2}}$

$T = T(0,b)\ R(\theta)\ Rfx\ R(-\theta)\ T(0,-b)$

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

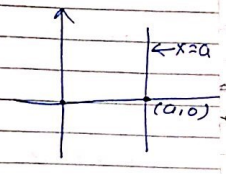
$$= \begin{bmatrix}1&0&0\\0&1&b\\0&0&1\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{1+m^2}}&\frac{m}{\sqrt{1+m^2}}&0\\\frac{m}{\sqrt{1+m^2}}&\frac{1}{\sqrt{1+m^2}}&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{1+m^2}}&\frac{m}{\sqrt{1+m^2}}&0\\\frac{-m}{\sqrt{1+m^2}}&\frac{1}{\sqrt{1+m^2}}&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&-b\\0&0&1\end{bmatrix}$$

$$= \begin{bmatrix}\frac{1-m^2}{1+m^2}&\frac{2m}{1+m^2}&\frac{-2bm}{1+m^2}\\\frac{2m}{1+m^2}&\frac{m^2-1}{m^2+1}&\frac{-bm^2+b}{1+m^2}\\0&0&1\end{bmatrix}$$

$x = a, \quad y = b$

① $x = a$

$T(a,0)\ Rfy\ T(-a,0)$



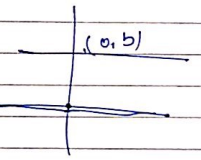
$$\begin{bmatrix}1&0&a\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}-1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&-a\\0&1&0\\0&0&1\end{bmatrix}$$

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

$$\begin{bmatrix}-1&0&2a\\0&1&0\\0&0&1\end{bmatrix}$$

② $y = b$

$T(0,b)\ Rfx\ T(0,-b)$



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

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

$$\begin{bmatrix}1&0&0\\0&-1&2b\\0&0&1\end{bmatrix}$$

2011 F

A(2,2) B(4,1) and C(5,3 Along the line x=3

For x=0

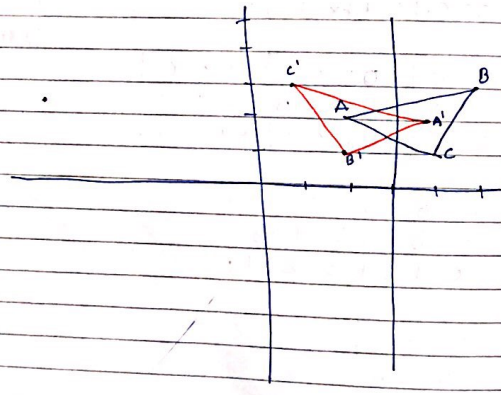
$$\begin{bmatrix} -1 & 0 & 20 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For x=3

$$\begin{bmatrix} -1 & 0 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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



Given a diamond shaped polygon with vertices $V_1(5,5)$, $V_2(3,3)$, $V_3(5,1)$ and $V_4(7,3)$ reflected about a line $y=x+2$

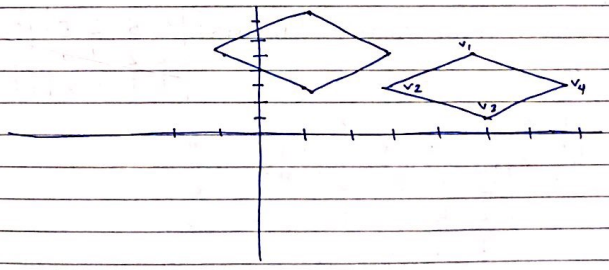
→ $y = mx+b$ $\therefore m=1$ $\therefore b=2$ $[y=x+2]$

$$T = \begin{bmatrix} \frac{1-m^2}{1+m^2} & \frac{2m}{1+m^2} & \frac{-2bm}{1+m^2} \\ \frac{2m}{1+m^2} & \frac{m^2-1}{m^2+1} & \frac{-bm^2+b}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation Matrix $= \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$[V_1' \ V_2' \ V_3' \ V_4'] = \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 3 & 5 & 7 \\ 5 & 3 & 1 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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



Reflect a rectangle A(2,2), B(5,2), C(2,4) D(5,4) about a line x=y

$x' = y$

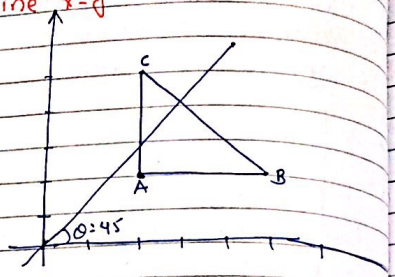
$y' = x$

For $x = y$

$T_1 = R(\theta = 45)$

$T_2 = Rfy$

$T_3 = R(-45^\circ)$



$\therefore T = R(-45)\ Rfy\ R(45)$

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

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

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

Net Transformation Matrix: $\begin{bmatrix} 2 & 5 & 2 & 5 \\ 2 & 2 & 4 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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

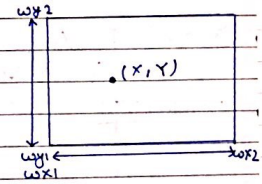
3x4

Windowing and clipping

The primary use of clipping in computer Graphics is to remove objects, lines or line segment that are outside the viewing pane. The viewing transformation is insensitive to the position of points relative to the viewing volume - especially those points behind the viewier - and it is necessary to remove these points before generating the view

Point Clipping:

Clipping a point from a given window is very easy. consider the following figure, where the rectangle indicates the window. point clipping tells us whether the given point (x,y) is within the given window or not; and decides whether we will use minimum and maximum cordinates of the window



The x- cordinate of the given point is inside the window, if x lies in between $wx1 \le x \le wx2$
same way, Y cordinate of the given point is inside the window if Y lies in between $wy1 \le Y \le wy2$

Line Clipping:

The concept of line clipping is the same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only

the portion that is inside the window

Viewing Pipeline :

A world-cordinate area selected for display is called a window. An area on a display device to which a window is mapped is called viewport. The window define what is to be viewed and viewport define where it is to be displayed. Often, windows and viewports are rectangles, with the rectangular edges parallel to the coordinate axes.

In general, The mapping of a part of a world-coordinate scene to ~~def~~ device coordinates is referred to as a viewing transformation. or window to viewport transformation or windowing transformation.

Window to viewport coordinate transformation

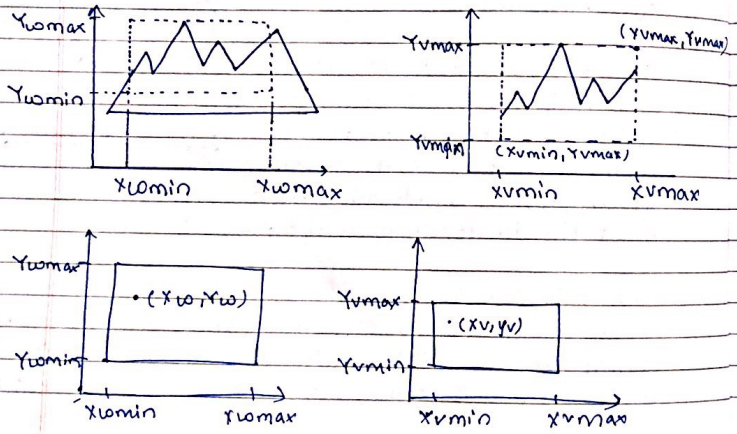


fig: A point at position (x_w, y_w) in a designated window is mapped to viewport cordinate (x_v, w_v) so that relative position in the two areas are the same

Once object descriptions have been transferred to the viewing references frame, we chose the window extent in viewing coordinates and select the viewport limits in normalize coordinates. we ~~do~~ transferred object description to normalized device coordinate, using a transformation that maintain the same relative placement of objects in normalized space as they had in viewing coordinate. If a coordinate position is at the centre of the viewing window, it will be displayed at the centre of the viewport.

Net Transformation Matrix

T1 : Translate lower left corner to origin
ie $T(-x_{wmin}, -y_{min})$

T2 : Scaling ie $S(s_x, s_y)$

T3 : Translate lower left corner of viewport
ie $T(x_{vmin}, y_{vmin})$

where, $s_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmax}}$

$s_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmax}}$

$T = T(x_{vmin}, y_{vmin})\ S(s_x, s_y)\ T(-x_{wmin}, -y_{wmin})$

Define window to viewport transformation for the following

	Dimension of window	viewport
to viewport		
lower left corner	(5,10)	(8,12)
upper right corner	(15,20)	(12,18)

window		device	
xwmax = 15	ywmax = 20	xvmax = 12	yvmax = 18
xwmin = 5	ywmin = 10	xvmin = 8	yvmin = 12

where

$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}} = \frac{12-8}{15-5} = \frac{2}{5}$$

$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}} = \frac{18-12}{20-10} = \frac{3}{5}$$

$$T = T(x_{wmin}, y_{wmin})\ S(2/5, 3/5)\ T(-x_{wmin}, -y_{wmin})$$

$$= T(8,12)\ S(2/5, 3/5)\ T(-5,-10)$$

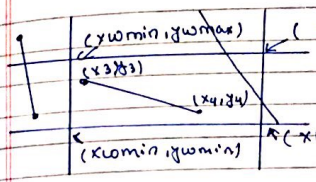
$$= \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2/5 & 0 & 0 \\ 0 & 3/5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2/5 & 0 & -2 \\ 0 & 3/5 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2/5 & 0 & 6 \\ 0 & 3/5 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

Cohen Sutherland Line Clipping:

The algorithm uses the clipping a window as shown in the following figure. The minimum coordinate for the clipping region is (xwmin, ywmin) and the maximum cordinate for the clipping region is (xwmax, ywmax).



We will use 4-bits to divide the entire region. These 4 bit represent the Above, Below, Right, Left (ABRL)

A	B	R	L

1001 (Top, Left)	1000	1010
0001	0000 window	0010
0101	0100	0110

There are three posibilities for the line

→ Line can be completely inside the window (This will be completely shown from the region)

→ Line can be completely outside the window (This line will be completely removed from the region)

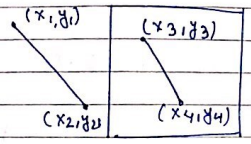
→ Line can be partially inside the window we will find intersection point and draw only portion of that line that is inside region)

Algorithm :

Step 1: Assign a region code for each endpoint.
Step 2: If both endpoints have a region code 0000, then accept this line.
Step 3: Else, perform the logical AND operation for both region codes
3.1: If the result is not 0000, reject this line
3.2: else you need clipping
3.2.1: Choose the endpoint of the line that is outside the window
3.2.2: Find the intesection point at the window boundary (base on region code
3.2.3: ~~Fit~~ Replace endpoint with the intersection point and update the region code.
3.2.4 Repeat Step 2 until we find a clipped line either trivially accepted or trivially rejected
Step 4: Repeat Step 1 for other lines.

cases :

$x_1 < x_{wmin}$
$y_1 < y_{wmax}$
$x_2 < x_{wmin}$
$y_2 > y_{wmin}$



(x_3, y_3)
$x_{wmin} \leq x_3 \leq x_{wmax}$ $\quad y_{wmin} \leq y_3 \leq y_{wmax}$
$x_{wmin} \leq x_4 \leq x_{wmax}$ $\quad y_{wmin} \leq y_4 \leq y_{wmax}$

Trivial cases

> line completely inside
> line completely outside

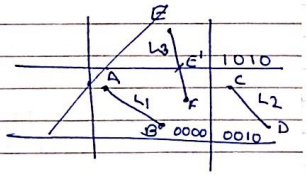
Finding the boundary intersection

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

where value of x is either
x_{wmin} or x_{wmax}

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

where value of y is either
y_{wmin} or y_{wmax}

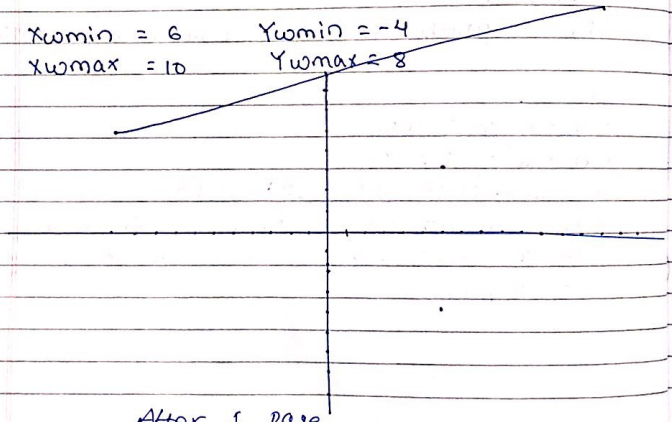


Line L_1
outcode of A : 0000
outcode of B : 0000
Logical AND of
end points = 0000
Line segment AB completely lies inside the window.

Line L2
outcode of C = 0010
outcode of D = 0010
Logical AND of endpoints = 0010
Hence discard line segment CD from displaying whereas save line segment AB completely for display

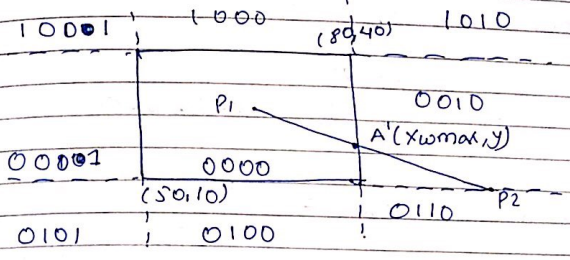
Line L3
outcode of E : 1000
outcode of F : 0000
logical AND of endpoints = 0000
Now, this is a non trivial case ie line segment EF partialy inside and partially outside.
Hence we find boundary intersection of line segment ie E' as 0000
outcode of E' = 0000
outcode of F = 0000
logical AND of E' and F = 0000
Now we save the line segment ie E'F for displaying.

Q) clip a line starting from (-11,5) and ending at (15,11) against the window having its lower left corner at (-6,-4) and upper right corner at (10,8)



After 1 page

Use the Cohen sutherLand algorithm to clip P1(70,20) P2(100,10) against a window lower left corner (50,10) and upper right hand cornerner (80,40)

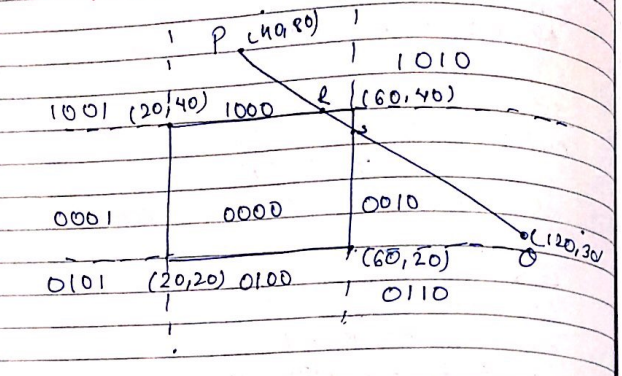


Soln:
outcode of P1 = 0000
outcode of P2 = 0010
performing logical AND P1 and P2 = 0000
This is non trivial case ie line P1P2 is partly inside and partly outside
So dividing line segment P1P2 into P1A' and A'P2
Now
outcode of P1 = 0000
A1' = 0000
The logical AND of P1 & A' = 0000
Hence we save line segment P1A' for displaying

slope of a line = $\frac{y_2-y_1}{x_2-x_1}$ $\therefore m = -1/3$

$m = -1/3$

line P(40,80) and Q(120,30)



Outcode of P = 1000
outcode of Q = 0010
P AND Q = 0000
This is non-trival case

Shearing

A transformation that distorts the shape of an object such that the transformed shape appears as if the object.

i) shearing along x-direction. force

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

In homogeneous form

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

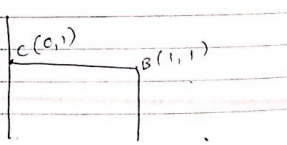
ii) shearing along y-direction

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

In homogeneous form

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

Q

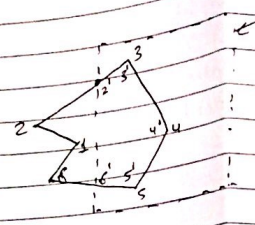


$sh_x = 2$

4

~~Sutherland Hodgeman~~

5. clip the the following:



output vertex list: {} ⇒ ϕ

Edge 2 - 3 (out → in)

output vertex list: {2', 3'}

Edge 3 - 4 (in → in)

output vertex list: {2', 3', 4'}

Edge 4 - 5 (in → in)

output vertex list: (2', 3', 4', 5')

edge 5 → 6 (in → out)

output vertex list (2', 3', 4', 5', 6', 2)

Edge 6 - 1 out to out

output vertex list: {2', 3', 4', 5', 6'}

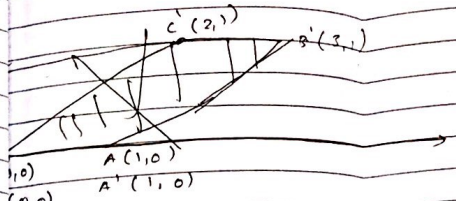
edge 1 - 2 (out - out)

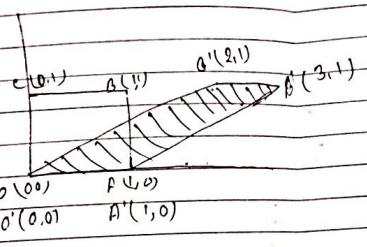
output vertex list: {2', 3', 4', 5', 6'}

2

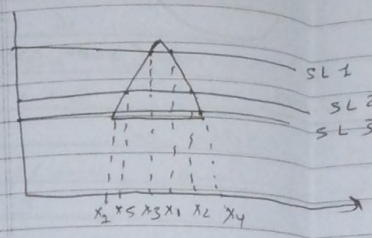
$$= \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} D' & A' & B' & C' \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$





scan line Algorithm (color filling)



$x_3 - x_1$ SL1

$x_5 - x_2$ SL2

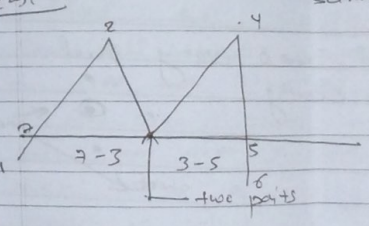
$x_7 - x_4$ SL3

special case

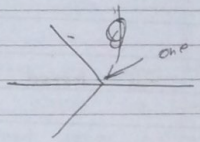
1) Edges lying on same line

2) Edges on opposite side.

case-1 (*) If two vertex are same side



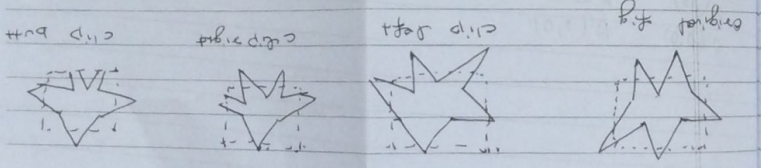
(*) If vertex are different



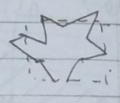
Through sutherland o
Inp same
Sutherland Hodgeman
classmate
Date
Page

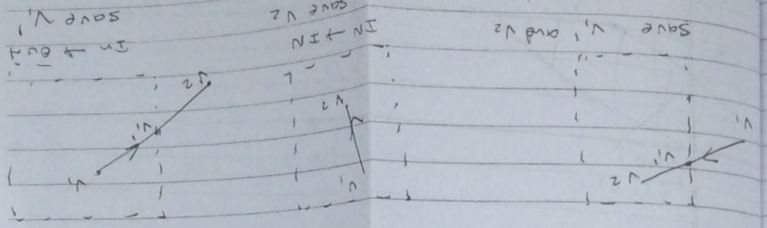
(Line mode closed figure)

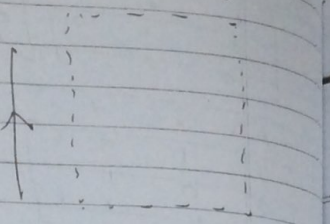
Sutherland Hodgeman polygon clipping algorithm. (DN: 25

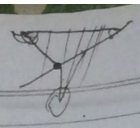


clip top





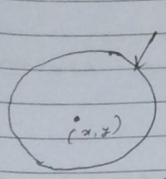




i) Boundary fill Algorithm

- 4-way connected
- 8-way connect

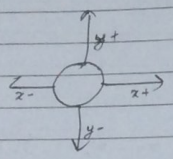
should be apply



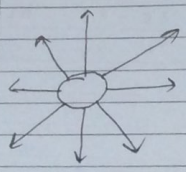
$x^2+y^2-r^2=0$

< 0 (inside circumferen

4-way connected



8-way connected (Faster)



Input Given

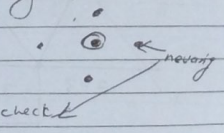
co-ordinate (x,y) (start point

Boundary Color

Fill color

* Two method proceed ~~neva~~ nevaring pixel

1) 4 connecting



Q. write down the algorithm for 4-way boundary fill algorithm

```
void boundar-fill4 (int x, int y, int b-color, int fill-colo
{
    int value = getpixel(x,y);
if (value != b-color && value != fill-color)
{
    putpixel(x,y, fill_color);
    boundary-fill4(x-1,y, b-color, fill-color);
    boundary-fill4(x+1, y, b-color, fill-color)
    boundary-fill4(x, y-1, b-color, fill-colo)
    boundary-fill4(x, y+1, b-color, fill-color);
}
}
```

① boundary
② already painted

Flood fill Algorithm:

Flood fill algorithm is applicable when we want to fil an area that is not defined with single color boundary If fill area is bounded with different color, we can paint that area by replacing a specified interior color instead of searching boundary color value This approach is called flood fill algorithm.

We start from specified interior pixel (x,y) and reassig all pixel values that are currently set to a given interior color with desired fill color.

Algorithm

```
void flood_fill (int x, int y, int fcolor, int ocolor)
{
    int current = getpixel(x,y);
    if(current == ocolor)
    {
        putpixel(x,y,fcolor);
        flood_fill(x-1, y, fcolor, ocolor);
        flood_fill(x,y-1, fcolor, ocolor);
        flood_fill(x+1, y, fcolor, ocolor);
    }
}
```

putpix(x-y, y, fcolor, ocolor)

(IFA is used if there is filled already with ocolor

Flood fill Algorithm : (ocolor=old color)

flood fill algorithm is applicable when we want to fill an area that is not defined with the single color boundary. If fill area is bounded with different color, we can paint that area by replacing a specified interior color instead of searching boundary color value. This approach is called flood fill algorithm.

we start from specified interior pixel (x,y) and reassign all pixel values that are currently set to a given interior color with desired fill color.

Algorithm :

```
Void flood-fill (int x, int y, int fcolor, int ocolor)
{   int current = getpixel(x,y);
    if(current == ocolor){
      putpixel(x, y, fcolor);
      flood-fill (x-1, y, fcolor, ocolor);
      flood-fill (x, y-1, fcolor, ocolor);
      flood-fill (x, y+1, fcolor, ocolor);
      flood-fill (x+1, y, fcolor, ocolor
}
```

Chapter-five 3D

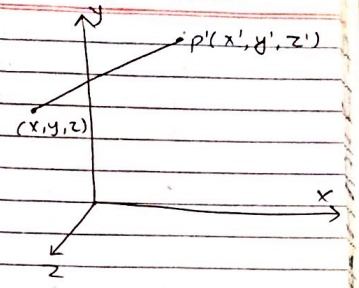
1. 3D Transformation

(i) Translation

$x' = x + tx$

$y' = y + ty$

$z' = z + tz$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

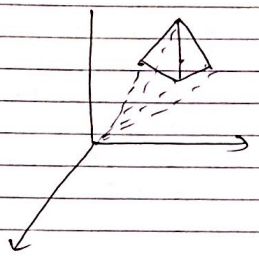
(ii) Scaling :

11.1 scaling about origin

$x' = x \cdot sx$

$y' = y \cdot sy$

$z' = z \cdot sz$



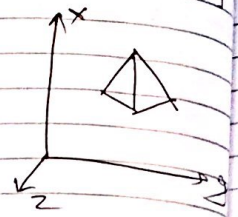
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

11.2 scaling about fixed point

$T = T(xf, yf, zf)\,(sx, sy, sz)\,T(-xf, -yf, -zf)$

$$= \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & 0 & x_f(1-s_x) \\ 0 & s_y & 0 & y_f(1-s_y) \\ 0 & 0 & s_z & z_f(1-s_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



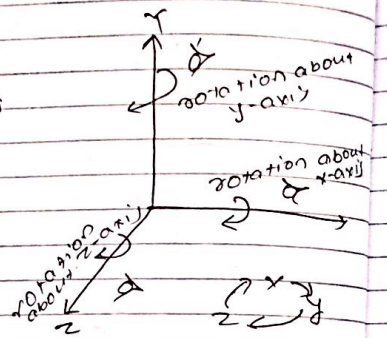
* Rotation :

① Rotation about z-axis

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

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

$z' = z$



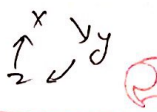
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

② Rotation about x-axis

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

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

$x' = x$



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

Rotation about y-axis

② $z = z\cos\theta - x\sin\theta$

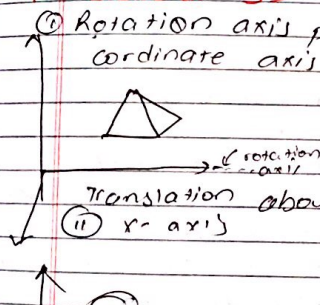
① $x = z\sin\theta + x\cos\theta$

$y' = y$

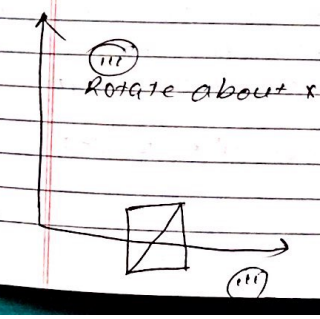
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

\# General 3D rotation :

① Rotation axis paralle to cordinate axis.



Translation about to ② x- axis



(x, y)

(x, -y)

ii) Rotation axis not parallel to coordinate axis

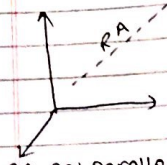


fig: RA not parallel to any cordinate axes

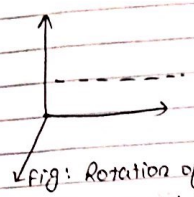


fig: Rotation of axis such that parallel to x-axis

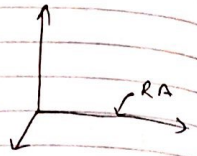
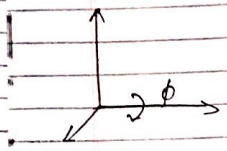


fig: Translating about to x-axis



Rotation by angle ϕ

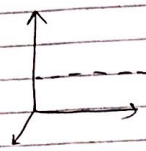


fig: Inverse Translation

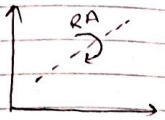
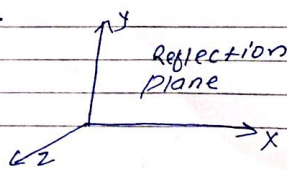
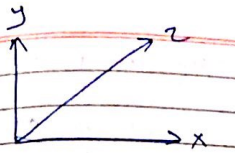


fig: Inverse rotation

Reflection in 3D (Transformation)

A three dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane. In general 3D reflection matrices are setup similar to those for two dimension Reflection relative to given axis are equivalent to 180° rotation about that axis.





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

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

Shear in 3D Transformation

Shearing Transformation can be used to modify object shapes. They are also useful in three dimensional viewing for obtaining general projection transformat in 2D we discussed transformation relative to the ~~second~~ x and y axes. to produce distortions in the shape of a object.

sharing along ~~object~~ z

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

$$SHx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad SHy = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

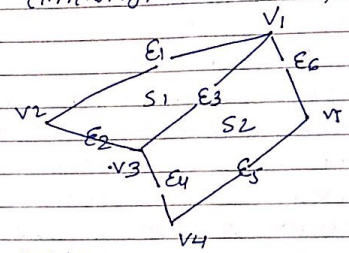
where a and b are shearing force.

3D object Representation:

3D object made up of a polygon are stored in a computer using

a) polygon table: contain vertex, edge and surface list.

b) Attribute table: properties of a surface (intensity, color, transparency)



Polygon Table

Vertex table	Edge table	surface table
$V_1 : x_1\ y_1\ z_1$	$E_1 : V_1, V_2$	$S_1 = E_1, E_2, E_3$
$V_2 : x_2\ y_2\ z_2$	$E_2 : V_2, V_3$	$S_2 = E_3, E_4, E_5, E_6$
$V_3 : x_3\ y_3\ z_3$	$E_3 : V_3, V_1$	
$V_4 : x_4\ y_4\ z_4$	$E_4 : V_3, V_4$	
$V_5 : x_5\ y_5\ z_5$	$E_5 : V_4, V_5$	
	$E_6 : V_5, V_1$	

One may ask: 3D object represent
(i) polygon table.
(ii) plane equation method.

(ii) plane equation method:

To produce display of a three dimensional object, we must process the input representation for the object through several procedures. These processing steps include transformation of modeling and world coordinate descriptions to viewing coordinates, finally to devices coordinate during the steps we need to know the transformation about spatial orientation of the individual surface component of the object.

The equation of plane surface can be represented as:

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

where (x, y, z) is any point on the plane, and the coefficients A, B, C and D are constants describing the spatial properties of the plane. we can obtain the values of A, B, C and D by solving a set of three plane equations using the coordinate values for three non-collinear points of the plane. we can select (x_1, y_1, z_1) & (x_3, y_3, z_3) as three non-collinear points

$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1$

where $k = 1, 2, 3$.

The solution for this set of equation can be obtained using cramer's rule.

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

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

Expanding the determinants, we can write the calculation for plane equation coefficient can be written as:

$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$

$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$

$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$

$D = x_1(y_2 z_3 - y_3 z_2) + x_2(y_3 z_1 - y_1 z_3) + x_3(y_1 z_2 - y_2 z_1)$

As vertex value and other information are entered into the polygon data structure values for A, B, C, D are computed for each polygon and stored with other polygon data.

polygon meshes : (Triangle meshes)

some graphic packages (for example PHIGS) provides several functions for modeling objects. A single plane surface can be specified with a function such as fillArea. But when object surface is to be tiled, it is more convenient to specify surface facets with mesh function.

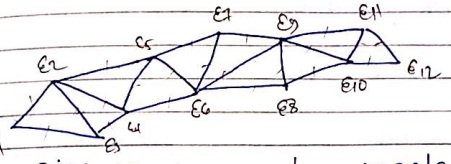
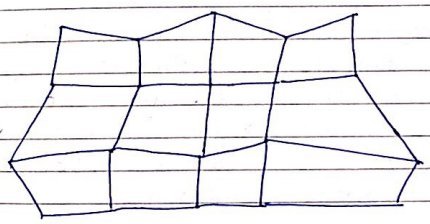


Fig: 10 connected triangle with 12 vertices ie n-2 triangle using n verticle

This function produces n-2 connected triangle using vertices.

Quadrilateral Meshes :



$(p-1)(q-1)$ = quadrilati

Fig: A quadrilateral mesh containing 12 quadrilateral constructed from 5x4 (Px9) vertices

projection :

The technique of representing n-dimensional object into (n-1) ~~dimension~~ plane is known as projection. The object in nature are 3D and the projection plane or the viewplane are always 2D hence we need to take help of projection to ~~take~~ project image of 3D object on to 2D plane.

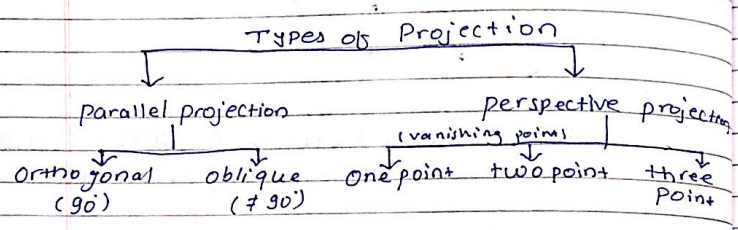


Fig: types of projection.

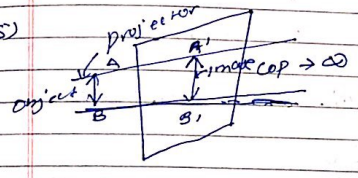
Difference between Parallel and perspective projection.

Parallel projection	perspective proj
→ light from object's edge are parallel.	→ light from object edge are not parallel.
→ The projected image is unrealistic.	→ The projected image is realistic.
→ ~~The~~ pop & cop → ∞	pop & cop finite
→ view	
→ Fig:	

Parallel Projection

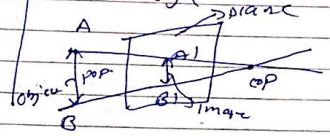
1) light rays traveling from object to viewplane are parallel to each other
2) Distance between centre of project (cop) & plane of projection (pop) is at ∞.
3) Type of parallel projection are orthoganal and oblique.
4) cannot display realistic image

(S)



Perspective projection

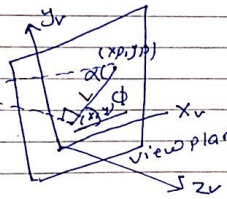
1) light rays traveling from object are inclined at some angle ie. not parallel to each other
2) Distance between cop & pop is at some fixed distance
3) Types are one point, two point & three point
4) can display realistic as viewing from eye



Oblique projection

→ Oblique projection vectors are specified by two angles α & ϕ.
→ (x, y) are also the orthographic cordinate on view plane.

Projection Matrix calculation



project matrix calculation :

$x_p = x + L \cdot \cos\phi$ | cordinates (x, y, z) are viewing coordinate
$y_p = y + L \cdot \sin\phi$

$\tan\alpha = z/L$

$L = z/\tan\alpha = z \cdot L_1$ where $L_1 = 1/\tan\alpha$

$x_p = x + z \cdot (L_1 \cos\phi)$
$y_p = y + z(L_1 \cdot \sin\phi)$

□ Final projection matrix will be

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: orthographic projection: when $L_1 = 0$ ($\phi \neq 0$).
$x_p = x$, $y_p = y$
cavallier projection when $\tan\alpha = 1$ (ie $\alpha = 45°$)
$\phi = 30°$ or $45°$
cabinet projection: when $\tan\alpha = 2$ (ie $\alpha = 63.4°$)
$\phi = 30°$ or $45°$

Perspective projection :

To obtain a perspective projection of a three dimensional object, we transforms point along projection lines that meet at the perspective reference point

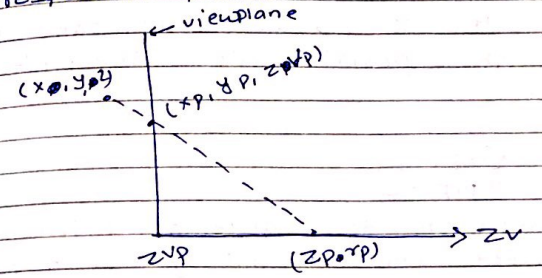


fig: perspective projection

Suppose, we set the projection reference point at z_{prp} along the z_v axis and we place the view plane at z_{vp} we can write equation describing coordinate position along this perspective projection line in parametric form as:

$x' = x - x_u$
$y' = y - y_u$
$z' = z - (z - z_{prp})_u$

where parameter u takes values from 0 to 1.
Coordinate position (x', y', z) represents any point along the projection line. When $u = 0$, we are at position $p = (x, y, z)$. At the other end of the

starting u=1
u=0 (parameter)

line u=1 and we have projection reference point coordinate ~~(0,0,zprp)~~ (0,0,zprp).

On the viewplate z' = zvp and we can solve the z' equation for parameter u at this position along the project line.

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

Substituting the value of u into the equation for x' and y' we obtain perspective transmotion equations.

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

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

where $d_p = z_{prp} - z_{vp}$ is the distance of viewplane from the projection reference point.

Homogenous coordinate representation

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In this representation homogenous factor is

$$h = \frac{z - z_{prp}}{d_p}$$

and the projection coordinate on the view plane are calculated as:

$$x_p = \frac{x_h}{h}, \quad y_p = \frac{y_h}{h}$$

where the original z-coordination value would be retained in projection coordinate for visible surface detection.

Parallel ~~Perspective~~ projection:

A parallel projection is ~~faced~~ formed by extending parallel lines from each vertex of the object until they intersect the plane of the screen. A parallel projection preserves relative properties of object in x and y direction.

In a general parallel projection, we may select any direction for the lines of projection suppose that the direction vector is $[x_p, y_p, z_p]$ and that the image is to be projected on xy plane. If we have a point on the object (x_1, y_1, z_1) we wish to determine where the projected point (x_2, y_2) will lie.

We will write equation for a line passing through the point (x_1, y_1, z_1) and the direction of projection with the use of parametric form.

$$x = x_1 + x_p u \quad \text{---} (i)$$
$$y = y_1 + y_p v \quad \text{---} (ii)$$
$$z = z_1 + z_p v \quad \text{---} (iii)$$

If $z=0$, then value of u from equation (iii)

$$u = -\frac{z_1}{z_p}$$

Substituting the value of u in equation (i) and (ii)

$$x_2 = x_1 - z_1 (x_p / z_p)$$
$$y_2 = y_1 - z_1 (y_p / z_p)$$

the projection formula may be written in matrix in homogenous form as:

~~Let us write an algorithm~~

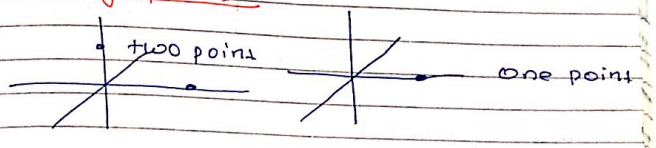
$$[x_2\ y_2\ z_2\ 1] = [x_1\ y_1\ z_1\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -x_p/z_p & -y_p/z_p & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

let us write an algorithm for performing parallel projection

1:40 → 2:20

Perspective projection

① Vanishing point



The vanishing point for any set of lines that are parallel to one of the principal axis of an object is referred to as a principal vanishing point or axis vanishing point. We control the number of vanishing point with the orientation of projection plane and perspective projection are classified as

* one point projection.
* Two point projection
* Three point projection

single point perspective projection:

One point perspective projection occurs when only one principal axis intersect the plane of projection. There are three type of single point ~~per~~ perspective transformations.

* when the projectors are located at X-axis, it is given by

one point :

$$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \; y \; z \; (px+1)]$$

where $px+1 \neq 1$

or $[x' \; y' \; z' \; 1] = \left[\frac{x}{px+1} \; \frac{y}{px+1} \; \frac{z}{px+1} \; 1\right]$

* when the projectors are located at y-axis, it is given by:

$$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \; y \; z \; (qy+1)]$$

where $qy+1 \neq 1$

or, $[x' \; y' \; z' \; 1] = \left[\frac{x}{qy+1} \; \frac{y}{qy+1} \; \frac{z}{qy+1} \; 1\right]$

it has centre of projection at $[0 \; -1/q \; 0 \; 1]$ and vanishing point is located at y-axis at $[0 \; 1/q \; 0 \; 1]$

It has centre of projection at $[-1/p \; 0 \; 0 \; 1]$ and vanishing point is located at x-axis at $[1/p \; 0 \; 0 \; 1]$

* when the projectors are located at z-axis, it is given by

$$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \; y \; z \; (rz+1)]$$

where $rz+1 \neq 1$

or

$$[x' \; y' \; z' \; 1] = \left[\frac{x}{rz+1} \; \frac{y}{rz+1} \; \frac{z}{rz+1} \; 1\right]$$

it has centre of projection at $[0 \; 0 \; -1/r \; 1]$ and vanishing point is located at $[0 \; 0 \; 1/r \; 1]$

For two point : perspective transformation

Two point of two principal vanishing point perspective projection occur when the plane of projection intersection exactly two of the principal axis xy-plane

$$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \; y \; z \; px+qy+1]$$

$$[x' \; y' \; z' \; 1] = \left[\frac{x}{px+qy+1} \; \frac{y}{px+qy+1} \; \frac{z}{px+qy+1} \; 1\right]$$

it has two centre of projection one on x-axis $[-1/p \; 0 \; 0 \; 1]$ and y-axis $[0 \; 1/q \; 0 \; 1]$

It has two vanishing point one on x-axis $[1/p \ 0 \ 0 \ 1]$ one on y-axis $[0 \ 1/q \ 0 \ 1]$

Three point perspective Projections :

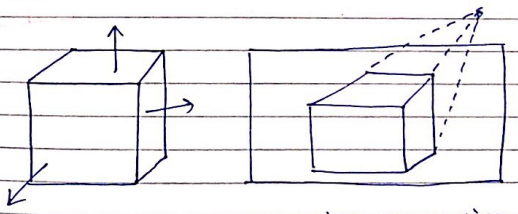
Three point projection occurs when the projection plane intersects all three of the principal axis ie. none of the principal axis is parallel to projection plane. The matrix representation for three point perspective transformation is

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [x \ \ y \ \ z \ \ px+qy+rz+1]$$

$$[x' \ y' \ z' \ 1] = \left[\frac{x}{px+qy+rz+1} \ \ \frac{y}{px+qy+rz+1} \ \ \frac{z}{px+qy+rz+1} \ \ 1 \right]$$

- It has three centre of projection on x-axis at $[-1/p \ 0 \ 0 \ 1]$, y-axis at $[0 \ -1/q \ 0 \ 1]$ and z-axis at $[0 \ 0 \ -1/r \ 1]$
- It has three vanishing point one on x-axis at $[1/p \ 0 \ 0 \ 1]$, y-axis at $[0 \ 1/q \ 0 \ 1]$ and z-axis at $[0 \ 0 \ 1/r \ 1]$



cordinate description

one point perspective projection

Color Model:

Subtracting color model – CMYK

RGB color model: (Addaptive Color model)

Based on the tristimulus, theory of vision, our eyes perceive colour through the stimulation of three visual pigments in the cones of retina. Three visual pigment have a peak sensitivity at wavelength of about 630 nm (Red), 530 nm (Green) and 450 nm (Blue). By comparing the intensities in a light source, we perceive the color of the light. This theory of vision is the the basis of displaying color output on a video monitor using the three color primaries red, green and blue refreed to as the RGB color model.

We can represent this model with unit cube defined on

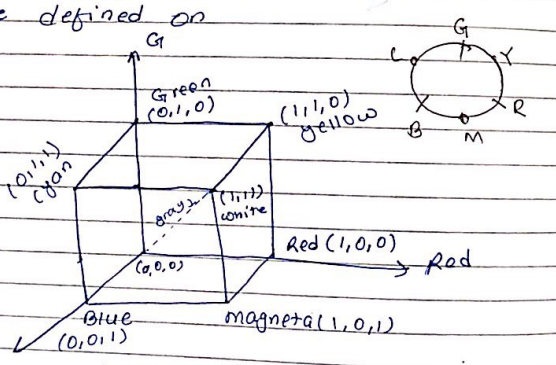


Fig: RGB Color cube model.

The origin represents black and the vertex with coordinate (1,1,1) is white. vertices of the cube on the axes represents the primary colors, and the remaining vertices represents the complentary color for each of color primaries.

As with XYZ color system, the RGB color scheme is an additive color model intensities of the primary colours are added to produce other colors. Each colour point within the bound of the cube be represented by triplet (R, G, B)

$C_\lambda = rR + gG + bB$

where C_λ = wavelength of color

CMYK or subtractive Color model)

- A color model developed with primary colors cyan, magenta, yellow ~~and black~~ is useful for describing color output to hard copy devices. Unlike video monitors, which produce a color pattern by combining light from the screen phosphors, hard copy devices, such as plotters produce a color picture by coating a paper with color pigment. We see the color by reflected light, a subtractive process.

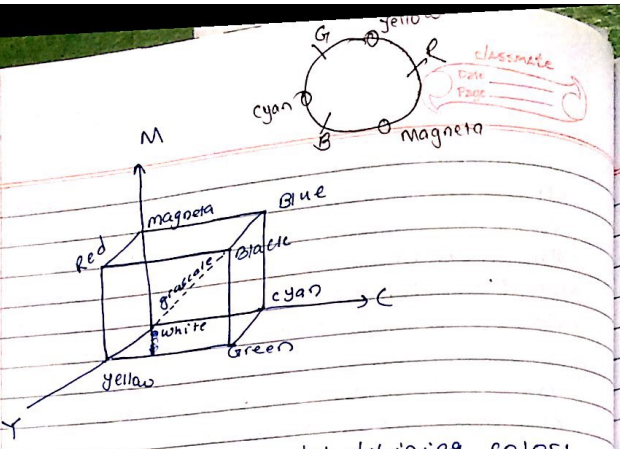


fig: CMY color model defining colors with a subtractive process inside a unit cube.

cyan = green + blue

Therefore when white light is reflected from cyan. cyan colored ink, the reflected light component must have no red component. That is red light is absorbed or subtracted by the ink. similarly magenta ink subtracts the green component from incident light and yellow subtracts the blue component.

Conversion from CMY - RGB :

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

Conversion from RGB to CMY

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

Process model to develop graphical software.
Machine independent graphical language
Graphics file format, TIFF, BMP, JPEG
Graphical package: openGL, GKS, PHIGS

openGL → open Graphics Language.
its a third party library which is used to render
GKS → Graphics Kernel System
PHIGS → programmer Higher Interface G.S.

cycle layer layer ~~से~~ store ~~हुई~~ /

Curve generation :
i) curve generation algorithm
ii) Interpolation

i) Interpolation method of curve generation
In practice, we have to deal with complex curves for which no simple mathematical definition in available such curves can be drawn using approximations methods. We can draw an approximation to a such curve if we have an array of sample points. we can then guess

an approximation to such curve should look like between the sample points. If the sample points are close, the curves look to be smoother and so on.

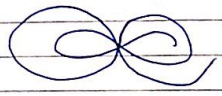
The polynomial equation in parametric form is expressed as:

$$x = f_x(u)$$
$$y = f_y(u)$$
$$z = f_z(u)$$

Why to use parametric form?

i) It treats all direction equally.

ii) It allow multiple value (several y or z values for given x) so that curves can double back or even cross.



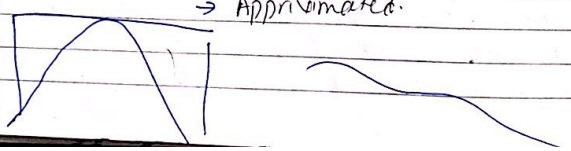
Spline representation

In computer graphics, the term spline curve refers to any composite curve formed with polynomial section satisfying specified boundary condition at the section endpoints.

Types of spline

→ Interpolated spline

→ Approximated.



Interpolated Spline

We specify a spline curve by giving a set of coordinate positions, called control points, which indicates the general shape of curve these control points are then fitted with piecewise continous parametric polynomial function.

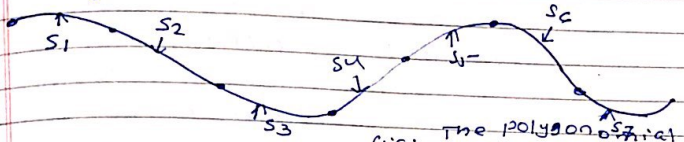


fig: The polygon control curve section S_1, S_2, S_3, S_4 are interpolated net control points

Approximated spline

When polynomial are so fitted that without necessarily passing through any of the control points, it is approximated with the shape of control polygon; the resulting curve is said to have approximated the set of control points

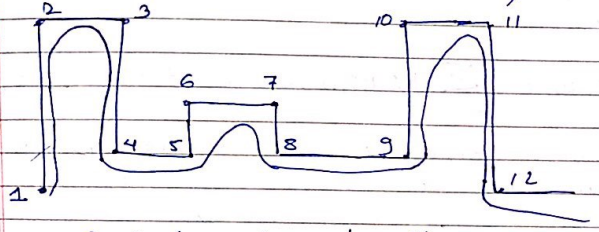


fig: Approximated spline

Cubic Bezier curve:

Unlike the piecewise spline curve, a Bezier spline curve can be fitted to any number of control points without tangent vector specification at any of the control points. A set of characteristics polynomial approximating function called Bezier function, called Bezier blending functions, blends the control points to produce a Bezier curve segment. However degree of Bezier curve segment is determined by number of control points to be fitted with that curve segment.

Given a set of n+1 control points $P_0, P_1 \cdots P_n$, a parametric Bezier curve segment that will fit to those points is mathematically defined by

$$p(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \quad \text{where } 0 \le u \le 1$$

Where $B_{i,n}(u)$ are Bezier blending functions, also known as Bernstein Basis function defined as

$$B_{i,n}(u) = C(n,i)\, u^i (1-u)^{n-i}$$

and $C(n,i)$ is the Binomial coefficients

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

degree = n+1
n = n+1

Expanding the equation, we get

$$p(u) = P_0 B_{0,n}(u) + P_1 B_{1,n}(u) + \cdots P_{n-1} B_{n-1,n}(u) + P_n B_{n,n}(u)$$

$$p(u) = P_0 \{C(n,0)\, u^0 (1-u)^n\} + P_1 \{C(n,1)\, u^1 (1-u)^{n-1}\}$$

$$\cdots + P_{n-1} \{C(n,n-1)\, u^{n-1} (1-u)^1\} + P_n \{C(n,n) u^n (1-u)^0\}$$

From this equation we discover ~~two~~ very important facts about Bezier curves. If we minutely observe the blending function in curly brackets.

i) n, the degree of polynomial blending functions and thus of Bezier curve segment is one less than no. of control points used

ii) The Bezier curve is general passes through only the first control point P_0 and the last control point P_n.

For Bezier curve of degree 3 (n = 3) we find four (n+1) control points are required. Thus for a parametric cubic Bezier curve:

$$p(u) = \sum_{i=0}^{3} P_i B_{i,3}(u) \quad 0 \le u \le 1$$

$$p(u) = P_0 B_{0,3}(u) + P_1 B_{1,3}(u) + P_2 B_{2,3}(u) + P_3 B_{3,3}(u)$$

where, $B_{0,3}(u) = \left\{\frac{3!}{0!3!}\right\} u^0 (1-u)^3 = (1-u)^3$

$B_{1,3}(u) = \left\{\frac{3!}{1!2!}\right\} u^1 (1-u)^2 = 3u(1-u)^2$

$B_{2,3}(u) = \frac{3!}{2!1!} u^2 (1-u) = 3u^2(1-u)$

$B_{3,3}(u) = \frac{3!}{3!0!} u^3 (1-u)^0 = u^3$

Using these value:

$p(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u) P_2 + u^3 P_3$

$p(u) = (-u^3 + 3u^2 - 3u + 1) P_0 + (3u^3 - 6u^2 + 3u) P_1 + (-3u^3 + 3u^2) P_2 + u^3 P_3$

This expression can be pressed in matrix form ie.

$p(u) = [F][B]$

$p(u) = [T][M][B]$

Here the Basis function matrix $[F] = [B_{0,3}(u), B_{1,3}(u), B_{2,3}(u), B_{3,3}(u)]$

The Bezier geometry matrix

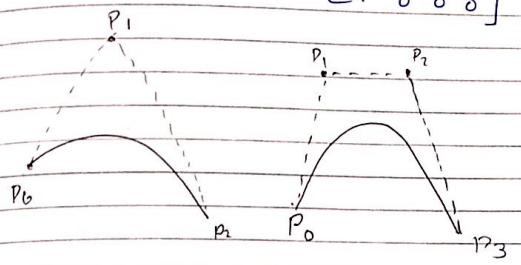
$$[B] = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad [T] = [u^3 \; u^2 \; u \; 1]$$

$$[M] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

such that $[F] = [T][M]$

Therefore a cubic bezier curve controlled by point P_0, P_1, P_2, P_3

$$p(u) = [u^3 \; u^2 \; u \; 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$



Example: 2D Bezier curves

Bezier curves:

Bezier curves can be generated

Visible surface detection methods

A major consideration in the generation of realistic graphics display is identifying those parts of a scene that are visible from a chosen viewing position. There are many approaches we can take to solve this problem
- Some methods require more memory
- Some involve more processing time.
- Some apply only to special types of object

- Hidden surface elimination methods.
- visible surface detection methods.

Two approaches:
- Object space method
- image space method.

(a) **Object space method**

Compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible

(b) **Image space method**

visibility is decided point by point at each pixel position on the projection plane.
- commonly used

- Most visible surface detection algorithms use sorting and coherence method to improve performance. Sorting is used to facilitate depth comparision by ordering the individual surface in a scene acc. to their distance from the view plane

- Line Display algorithm generally uses object space method to identify visible lines in wire frame display.

Back-face Detection: [osm]

A fast and simple object space method for identifying the backfaces of poly-hedron is based on the 'inside-outside' test. A point (x, y, z) is 'inside' a polygon surface with plane parameters A, B, C and D if

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

When an inside point is along the line of sight to polygon surface then the polygon must be a back face (we are inside that face and cannot see the front of it from our viewing position

Consider the Normal vector N to a polygon surface which has Cartesian components (A, B, C). In general if V is a vector in the viewing direction then polygon is a backface if

$$V \cdot N > 0$$

Furthermore, if object descriptions have been converted to projection coordinates and our viewing direction is parallel to the viewing z_v axis, then $V = (0, 0, V_z)$ and

$$V \cdot N = V_z C$$

so that we only need to consider the sign of C, the Z component of the normal vector N

~~$V \cdot N = V_z C$~~

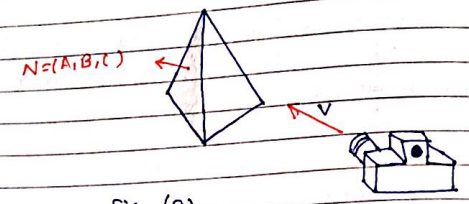


fig (a)

In a right handed viewing system with viewing direction along the negative Z_v axis, the polygon is back face if

$C \leq 0$ and $C < 0$ [ie $z \leq 0$]

Thus, in general, we can label any polygon as a back face if its normal vector has z-component value.

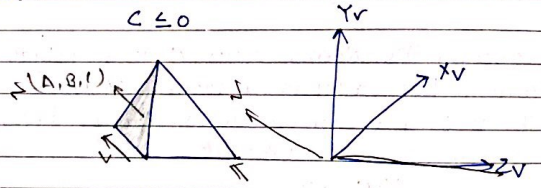
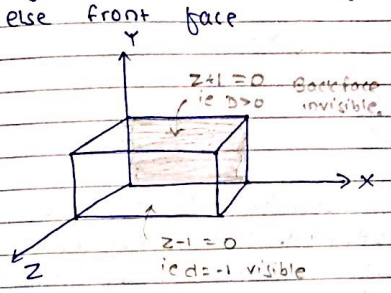


fig: A polygon surface with plane parameter $C < 0$ in a right handed co-ordinate system is identified as a back-face when the viewing direction is along the -ve Z_v axis

object is centered at origin if $D > 0$ (backface else front face



Depth Buffer Method : (z-buffer)

A commonly used image-space approach for detecting visible surface is the depth buffer method which compares surface depth at each pixel position on the projection plane.

- The process is also referred to z-buffer method, since object depth is usually measured from plane along the z-axis of a viewing ~~plane~~ system.
- This methods is usually applied to scenes containing only polygon surface because depth values can be computed very quickly and the method is easy to implement.
- with object description converted to a projection coordinate each (x,y) position on a polygon surface corresponds to the orthographic projection point (x,y) on the view plane
- Therefore for each pixel position (x,y) on the viewplane object depth can be compared by comparing z-values.

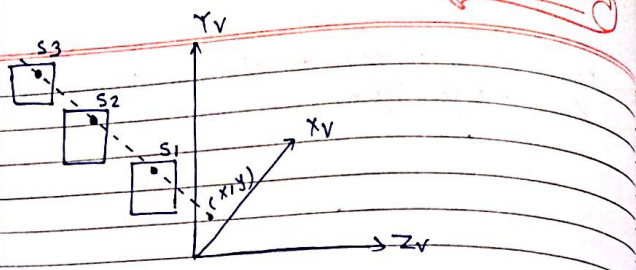


fig: At view plane position (x,y), Surface S_1 has the smallest depth from view plane and so is vible to the position ie its surface intensity value(x,y) is saved.

- We can implement the depth buffer algorithm in normalize co-ordinate so that the z-values range from zero (0) at the back clipping plane to zmax at the front clipping plane
- The value of zmax can be set to either one(unit) or to the largest value that can be stored in the system

In this method two buffer areas are required
- Depth Buffer: To store the depth value for each (x,y) position as Surface S are processed
- Refresh Buffer: To store the intensity value for each (x,y) position.

Initially, depth buffer set to 0 and refresh buffer to bk intensity.

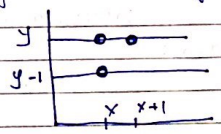
Each surface listed in the polygon table is then processed one scan line at a times.

- The calculated depth is compared to the value previously stored in the depth buffer at that position.
- If The calculated depth is greater than the value store in the depth buffer, the new value is stored and the surface intensity at that position is determined and placed in the same xy location in the refresh buffer

Algorithm
1. Initialize the depth buffer and refresh buffer so that for all buffer position (x,y),
   depth(x,y) = 0 , refresh(x,y) = I_{bk}
2. for
2. For each pixel on each polygon surface, compare depth value (z-value) to previously stored value in depth buffer (x,y).
   - If z > depth(x,y) then set
     depth(x,y) = z;
     refresh(x,y) = I_{surf}(x,y);

3. plot the point (x,y) and z-value at each depth buffer with corresponding values in refresh buffer.

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



The next position (x+1,y) along the scan line is obtained as

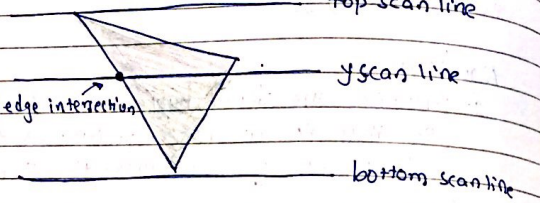
$$z' = -\frac{A(x+1) - By - D}{C}$$

or,

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

The ratio $\frac{-A}{C}$ is constant for each surface

On each scan line, we start by calculating depth on a left edge polygon that intersect that ~~intersect~~ line



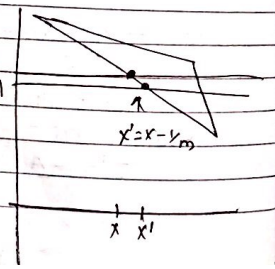
Scan line intersecting a polygon surface

Starting at a top vertex we can recursively calculate x position as $x' = x - ym$ where m is a slope of edge. Depth values down the edge are the obtained recursibly ~~at~~ for $(x - ym, y-1)$ ie (x, x')

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

If we processing down a vertical edge, slope is infinite then:

$$z' = z + B/C$$



A- Buffer method

A buffer method represents an antialiased, area-averaged, accumulation-buffer method developed by Lucasfilm for implementation in the surface rendering system called REYES (Renders Everything You Ever Saw). It is an extension of the ideas in depth buffer method.

While z buffer methods deals only with opaque object / surface and cannot accumulate intensity values more than one surface, as is necessary if transparent surfaces are to be displayed.

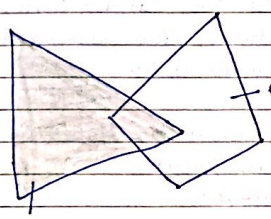
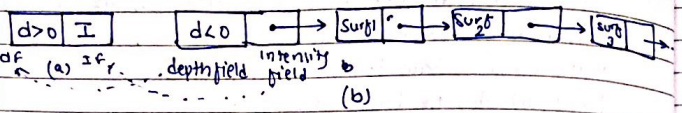


Fig: viewing an opaque surface through transparent surface

A-buffer method expands the depth buffer so that each position in the buffer can be reference a linked list of surface. Thus more than one surface intensity can be taken into consideration at each pixel position, object edges can be antialiased.

Each position in the A-buffer has two fields

- depth field - stores a positive or negative real no.
- intensity field - stores surface intensity information or a pointer value.



If the depth field is positive, single surface overlapping the corresponding pixel area. It stores RGB components of surface color and percent of pixel coverage.

If depth field is -ve, multiple surface contributions to the pixel intensity. Intensity field stores a address of next linked list of surface data.

Data for each surface in the linked list includes
- RGB intensity components
- percent of transparency
- depth
- percent of area covered
- pointer to next surface
- surface identifier.

Scan-Line Method

The image space method for removing hidden surface is an extension of scan line algorithm for filling polygon interiors. Instead of filling just one surface, deals with multiple surface. All polygon surfaces intersecting the scan line are examined which are visible. Depth calculation are made for each overlapping surface to determine which is nearest to the view plan. When the visible surface determined, intensity value entered in refresh buffer.

It request edge table, polygon table :
- Edge table contents
  - Coordinate end points for each scan line
  - Pointer in polygon table to identify the surfaces bounded by each line
  - inverse slope of each line.

- Polygon table contents:
  - Coefficient of plane eqn for each surface
  - pointer into edge table
  - intensity information of the surface

- Active edge list contains edges that cross the current scan line, sorted inorder of increasing X.

- Flag is defined for each surface that is set 'ON' or 'OFF' to indication if a position alone a scan line inside or outside.

Any number of overlaping polygon Surfaces can be processed with thin scan-line method. Flags for the Surfaces are set to indicate whether a position is Inside or Outside, depth calculation are performed when surfaces overlap

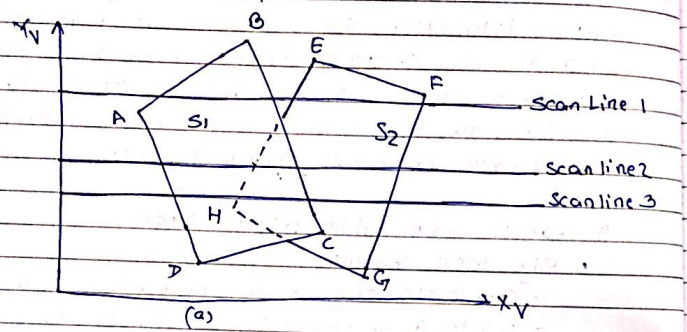


Fig: Scan lines Crossing the projection of two Surfaces S_1 and S_2, in the view plane. Dashed lines indicate the boundaries of hidden surfaces.

Example (a):

The active list of scan line 1 are:
AB, BC, EH and FG

Only the S_1 is on for AB and BC because single surface

Only the S_2 is on for EH and FG. No other position alons scan line 1 intersect surface.

For scan line 2 and 3, active list are:
AD, EH, BC, FG

AD to EH, S_1 is ON

EH to BC, S_1 & S_2 are ON, depth is caclulated for two surface. depth of S_1 is less than S_2 so intensities of 1 are loaded into the refresh buffer until BC is

Algorithm

1. Established data structure
   a. Edge table, polygon table, active edge, flag
2. Repeat for each scan Line.
   a. Update AEL (active edge list) for scan line y
   b. Scan line across light using bk color until flag goes on
   c. When one flag is ON, enter intensity of that surface to refresh buffer
   d. when two or more flags are ON, do depth sorting and stored the intensity of pixel nearest to the view plane.

Depth Sorting Method : [Painter's Algorithm]

Using both image space and object-space approach, the depth-sorting method performs the following basic functions

1. Surfaces are sorted in descending depth.
2. Surfaces are scan converted in order, starting with the surface of greatest depth.

Sorting operation are carried out in both image and object space, the scan conversion of the polygon surfaces is performed in image space.

In creating an oil painting, an artist first paints the background colors. Next, the most distant objects are added, then nearer objects and so forth. At final step foreground objects are painted on the canvas over the bk and other objects that have been painted on the canvas. Each layer of paint Covers up previous layer

Using same technique, we first sort the surfaces according to their distance from their view plane. The intensity values for the farthest surface are then entered into the refresh buffer. Taking each surface succeeding in turn (decreasing depth) we paint the surface intensities onto the frame buffer over the intensities of the previously processed surface.

Painting polygon surfaces onto the frame buffer according to depth is carried out in several steps. Assuming we are viewing along -z direction, surfaces are ordered on the first pass acc to smallest z-value on each surface.

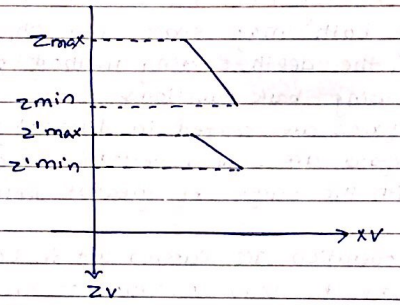


fig: two surfaces with no depth overlap

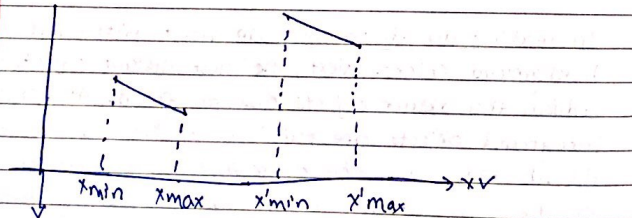


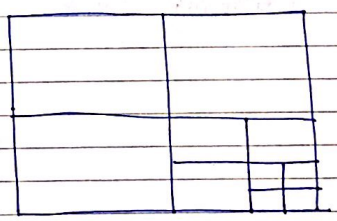
fig: Two surface with depth overlap but no overlap in the x-direction.

Area Subdivision method

This technique for hidden surface removal is essentially an image-space method, but object-space operations can be used to accomplish depth ordering of surfaces. The area sub-division method takes advantage of area coherence in a scene by locating those view areas that represent part of single surface.

We apply this method by successively dividing the total viewing area into smaller and smaller rectangles until each small to area is the projection of part of single visible surface or no surface at all.

To implement this method, we need to establish tests that quickly identify the area as a part of single surface or complex. If test indicate view is complex, we sub-divide it. Now we apply test for each smaller area and subdividing these if the tests that indicate the visibility of a single surface is still uncertain.

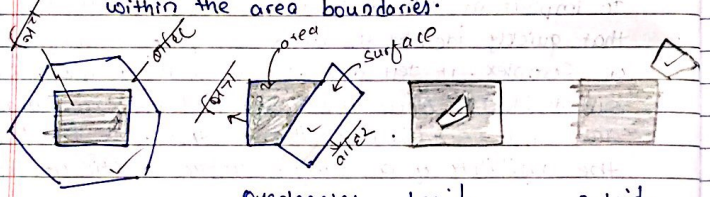


Dividing a square sized area into equal sized quadrants at each step.

we can describe these relative surface charac. as:

Surrounding surface: One that completely encloses the area.
Overlapping surface: One that is partialy inside and partialy outside the area
Inside surface: One that is completely inside.
Outside surface: One that is completely outside.

The tests for ~~defining~~ determing surface visibility within an area
- All surfaces are outside surfaces wrt area.
- Only one inside, overlapping or surrounding surface is in the area.
- A surrounding surface obscures all other surfaces within the area boundaries.



Surrounding Overlapping Inside outside

Unit 7: illumination model and shading

- illumination Theory
- Ambient ~~Theory~~ Light
- Reflections: Diffuse, Specular
- Surface shading methods
  - Constant shading
  - Gouraud shading
  - Phong shading
  - Fast phong shading
- Color models: RGB, CMYK

Realistic displays of a scene are obtained by generating perspective projections of objects and by applying natural lighting effects to the visible surfaces. An illumination model, also called a lighting model and sometimes referred to as a shading model, is used to calculate the intensity of light that we should see at a given point on the surface of an object. A surface-rendering algorithm uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.

Surface-rendering procedures also called surface shading method, we will refer to the model for calculating light intensity at a single surface point as an illumination model or lighting model, we will use the term surface rendering to mean a procedure for applying a lighting model to obtain pixel intensities for all the ~~objects~~ projected surface position in a scene.

Lighting effects include light reflections, transparency, surface texture and shadows.

Light Sources

The total reflected light is the sum of the contribution from light sources and other reflecting surfaces in the scene. Thus a surface that is not directly exposed to a light source may still be visible if nearby objects are illuminated.

light source mean an object that is emitting radiant energy, such as light bulb and the sun. A luminous object can be both a reflector and source

Point source: The simplest model for a light emitter
distributed light source: fluroscent light (long)

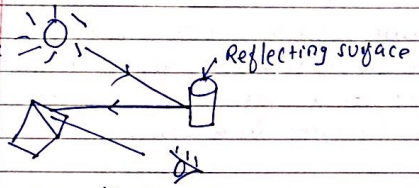
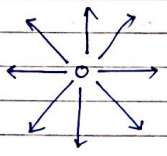
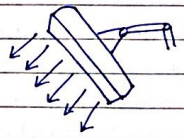


fig: illuminous



point source



distributed light source.

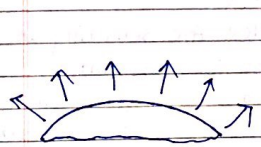
Note: Shiny materials reflect more of the incident light, and dull surfaces absorb more of the incident light. Similarly, for an illuminated transparent surface, some of the incident light will be reflected and ~~source~~ some will be transmited through the material.

Diffuse Reflection

Surfaces that are rough or grainy, tend to scatter the reflected light in all directions. This scattered light is called diffuse reflection. shown in fig (a)

Specular Reflection:

The light sources create highlights, or bright spots, called specular reflection. The highlighting effect is more pronouced on shiny surface than on dull ~~object~~ surfaces. An illustration of specular reflection is shown in fig (b)



fig(a): Diffwe reflections from a surface

fig(b): Specular reflection superimposed on diffuse reflection vectors.

Basic illumination models:

We discuss simplified methods for calculating light intensities. ~~The empirical models deter~~.

1) Ambient Light

A surface that is not exposed directly to a light source still will be visible if nearby objects are illuminated. In our basic illumination model, we can set a general level of brightness for a scene. This is a simple way to model the combination of light reflections from various surfaces to produce a uniform illumination called the ambient light, or a background light.

Ambient light has no spatial or directional characteristics. The amount of ambient light incident on each object is a constant for all surfaces and over all direction.

Surface Shading Method (Intensity interpolation)

(i) Constant-Intensity shading:

A fast and simple method for rendering an object with polygon surfaces is constant-intensity shading, also called flat shading. In this method, single intensity is calculate for each polygon. All points over the surface of the polygon are then displayed with the same intensity value.

- Useful for quickly displaying the general appearance of a curved surface.

In general flat-shading of polygon facets provides an accurate rendering for an object if all of the following assumptions are valid:

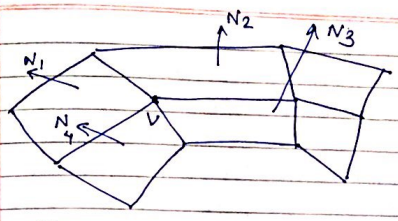


Fig:

The normal vector at vertex V is calculated as the average of the surface normals for each polygon sharing that vertex.

- The object is a polyhedron and is not an approximation of an object with a curved surface.
- All light sources illuminating the object are sufficiently far from the surface so that $N \cdot L$ and the attenuation function are constant over the surface.
- The viewing position is sufficiently far from the surface so that $V \cdot R$ is constant over the surface.

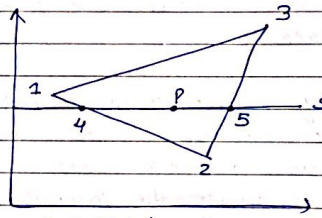
Even if all of these conditions are not true, we can still reasonably approximate surface-lighting effects using small polygon facets with flat shading and calculate the intensity for each ~~layer for ea~~ facet, say at the centre of polygon.

(11) **Gouraud shading**

The intensity interpolation scheme developed by Goroud and generally refferred to as Gouraud shading, renders a polygon surface by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud shading by performing the following calculations.

- Determine the average unit normal vector at each polygon vertex.
- Apply an illumination model to each vertex to calculate the vertex intensity.
- Linearly interpolate the vertex intensities over the surface of the polygon.



(a) Fig: ~~tear~~ linear interpolation for Goroud

~~Example:~~

At each polygon vertex we obtain a normal vector by averaging the surface normals of all polygon sharing that vertex.

Thus, for any vertex position V, unit vertex normal with the calculation:

$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|} \quad \text{— (1)}$$

we can determine intensity at point 4 is:

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \quad \text{— (11)}$$

Similarly, intensity for 5 is interpolated from intensity values at vertices 2 and 3. intensities at u and s as:

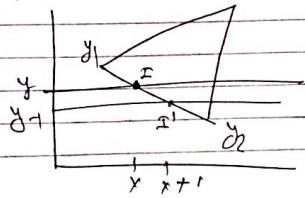
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5 \quad \text{— (12)}$$

The intensity at edge position (x, y) is interpolated as:

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

Then we can calculate intensity along this edge for next scan like, y-1 as

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



Note:

Gouraud shading removes the intensity discontinuities associated with the constant shading model, but it has some other dificiency. Highlights on the surface are sometimes displayed with anomalous shapes.

The linear intensity interpolation can cause bright or dark spot intensity streaks, called Mach bands, to appear on the surface.

These effects can be reduced by dividing the surface into a greater number of polygon faces or by using phong shading, that require more calculation.

(iii) Phong Shading [normal vector interpolation shading]

A more accurate method for rendering a polygon surface is to interpolate normal vectors and then apply the illumination model to each surface point. It displays more realistic highlights on a surface and greatly reduces the mach-band effect.

A polygon surface is rendered using phong shading by carrying out the following steps:

(a) - Determine the average unit normal vector at each polygon vertex.

(b) - Linearly interpolate the vertex normals over the surface of the polygon.

(c) - Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

for c

$$I = K_a I_a + K_a I_a (\vec{N} \cdot \vec{L}) + K_s I_L (\vec{V} \vec{R})^{n_s}$$

for a :

$$\vec{N}_{avg} = \frac{\Sigma \vec{N_i}}{|\Sigma \vec{N_i}|}$$

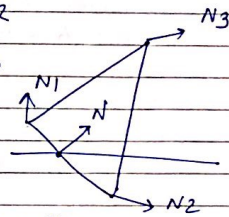
calculation for b

$\vec{N}$ can be obtained by interpolating ~~(1) and (2)~~ N_1 and N_2

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

similarly for next scan line

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



similarly for horizontal scan line, N_1 can be obtained from interpolation between $\bar{N}$ and N_2 given by

$$N_p = \frac{x_2 - x_p}{x_2 - x} \cdot \bar{N} + \frac{x_p - x}{x_2 - x} \bar{N_2}$$

similarly for next scan line

$$N_p' = \bar{N_p} + \frac{\bar{N_2} - \bar{N}}{x_2 - x}$$

Comments:
- Requires more calculation
- time consuming
- Produces more realities.

(iv) Fast phong method:

Phong method can be speeded up by using approximation in the illumination model calculations of normal vectors. Fast Phong shading approximates the intensity calculations using a Taylor-series expansion and triangular surfaces patches.

Since phong shading interpolates normal vectors from vertex normal, we can express the surface normal N at any point (x,y) over a triangle as:

$N = Ax + By + C$ — (1)

where vectors A, B and C are determined from the three vertex equations:

$N_k = Ax_k + By_k + C, \quad k = 1,2,3$ — (2)

with (x_k, y_k) denoting a vertex position.

$$I_{diff}(x,y) = \frac{L \cdot N}{|L||N|}$$

$$= \frac{L \cdot (Ax+By+C)}{|L|\,|Ax+By+C|}$$

$$= \frac{(L \cdot A)x + (L \cdot B)y + (L \cdot C)}{|L|\,|Ax+By+C|}$$

$$= \frac{ax + by + c}{(dx^2 + exy + fy^2 + gx + hy + i)^{1/2}}$$

$a = L \cdot A$

by Taylor Series expansion

$$I_{diff} = T_5x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0$$

Shear:

A transformation that distorts the shape of an object such that the tranformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called shear

$x' = x + sh_x$

$y' = y$

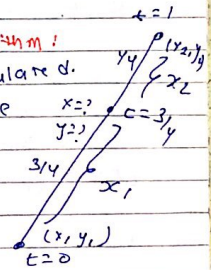
$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Liang Barsky Line clipping algorithm:

At time t, (x,y) should calculated.

- Parametric Equation of the line
- Line (x_1,y_1) to (x_2,y_2)

Consider t: range from 0 to 1



At 3/4th of line pathe

$t = 3/4$

The location is

$x = y_4x_1 + 3/4\,x_2$

$y = y_4y_1 + 3/4\,y_2$

At time = t

$x = (1-t)x_1 + t \cdot x_2$

$y = (1-t)y_1 + t \cdot y_2$

$x = x_1 - x_1 \cdot t + t \cdot x_2$

$= x_1 + t(x_2 - x_1)$ ∴ $x = x_1 + t\Delta x$

Parametric line equation is:

$x = x_1 + t\Delta x$

$y = y_1 + t\Delta y$

Wcordinate

$x_{wmin} \le x \le x_{wmax}$

$y_{wmin} \le y \le y_{wmax}$

$x_{wmin} \le x_1 + t\Delta x \le x_{wmax}$

$y_{wmin} \le y_1 + t\Delta y \le y_{wmax}$

$x_1 + t\Delta x \ge x_{wmin}$

$x_1 + t\Delta x \le x_{wmax}$

$y_1 + t\Delta y \ge y_{min}$

$y_1 + t\Delta y \le y_{wmax}$

$t\Delta x \ge x_{wmin} - x_1$

$t\Delta x \le x_{wmax} - x_1$

$t\Delta y \ge y_{wmin} - y_1$

$t\Delta y \le y_{wmax} - y_1$

make all equalities to $\le$

$-t\Delta x \le x_1 - x_{wmin}$

$t\Delta x \le x_{wmax} - x_1$

$-t\Delta y \le y_1 - y_{wmin}$

$t\Delta y \le y_{wmax} - y_1$

$tp_k \le q_k \quad [k = 1, 2, 3, 4$

$k = 1$ (is the line inside left boundary ?)

$k = 2$ right

$k = 3$ bottom

$k = 4$ top

$p_1 = -\Delta x \quad q_1 = x_1 - x_{wmin}$

$p_2 = \Delta x \quad q_2 = x_{wmax} - x_1$

$p_3 = -\Delta y \quad q_3 = y_1 - y_{wmin}$

$p_4 = \Delta y \quad q_4 = y_{wmax} - y_1$

Algorithm

step1: get the endpoints (x_1, y_1) to (x_2, y_2)

step2: find $\Delta x, \Delta y, p_1, p_2, p_3, p_4, q_1, q_2, q_3, q_4$

step3: assign $t_1 = 0$, $t_2 = 1$

(i) if $p_k = 0 \quad k = 1, 2, 3, 4$

then line is parallel to the window

(ii) if $q_k < 0 \quad (k = 1, 2, 3, 4)$

then line is outside the window

(iii) for non zero value of p_k

if $p_k < 0$ then find t_1

$t_1 = \max(0, q_k/p_k)$

else $p_k > 0$ find t_2

$t_2 = \min(q_k/p_k, 1, q_k/p_k)$

→ if $t_1 > t_2$, line completely outside - reject.

else find new set of (x, y) if t_1 exchanged

$x = x_1 + t\Delta x$

$y = y_1 + t\Delta y$