1. Introduction

* Computer Graphics:

Computer graphics field is related to the generation of graphics or images using computers which includes the creation, storage and manipulation of image objects. Thus, computer graphics is an integral part of all computer user interface and is necessary visualising 2-D, 3-D objects in all most all areas such as education, science, engineering, medicine, commerce, research, advertising and entertainment.

* Components of Computer graphics:

Interactive computer graphics consists of three components, such as:

1) Digital memory buffer / Frame buffer
2) Video monitor / TV monitor
3) Display controller

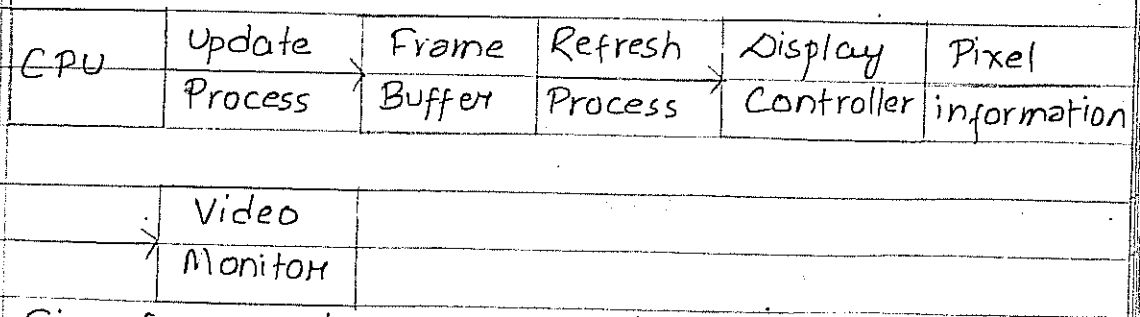


Fig:- Components of Computer graphics

Using these components, we are able to see the output on the screen in the form of pixels.

1) Digital memory buffer / Frame buffer:

This is the place where image or pictures are stored as an array (matrix of 0 and 1) where 0 represents darkness and 1 represents image or pictures. Frame buffer is the Video RAM (V-RAM) i.e. used to hold or map the image displayed on the screen. The amount of memory required to hold the image depends primarly on the resolution of the screen image and also the color depth used per pixel.

* Note:
The formula to calculate how much video memory is required at a given resolution and bit depth is:

Memory in MB = (X-Resolution) x (Y-Resolution x bits per pixel

[* 360 [illegible] = 640 x 480]

2) Video monitor / TV monitor :

Monitor helps us to view the display and make use of CRT (Cathode Ray Tube).

3) Display Controller :

It is an interface between digital memory buffer and video monitor. The main function of this is to pass the contents of frame buffer to the monitor. The display controller reads each successive bytes of data from the frame buffer memory and converts 0's and 1's into corresponding video signals. These signals is then fed to the video monitor to produce a black and white pictures on the screen.

* Advantages of Computer Graphics :

The main advantages of computer graphics are as follows :

→ It provides tools for producing pictures not only of concrete real world objects but also of abstract objects such as : mathematical surface.

→ It has the ability to show moving pictures and

Jain-

Thus, it is possible to produce animation with computer graphics.

→ With computer graphics, user can also control the animation speed, portion of the view, the geometric relationship between the objects in the scene.

→ The computer graphics also provides facility called update dynamics. With update dynamics it is possible to change the shape, colour or other properties of objects being viewed.

→ The computer graphics provides tool called motion dynamics. With these tools, user can move and tumble objects with respect to a stationary observer, or we can make objects stationary and the viewer moving around them.

2014-03-24th

* Applications of computer graphics:

Computer graphics is used today in many different areas of science, engineering, industry, business, education, medicine, art and training.

1) User interface:
Most application have user interfaces that rely on desktop, window system to manage multiple

simultaneous activities, and on point and click facilities to allow user to select menu items/ icons and objects on the screen. These activities falls under computer graphics.

2) Plotting:
Plotting 2-D, 3-D graphics of mathematical, physical and economics function use computer graphics extensively. The histogram, bar and pi-chart, the task scheduling charts are the most commonly used plotting. These all are used to present meaningful patterns of complex data.

3) Computer aided drafting and design (CAD):
One of the major uses of computer graphics is to design components and systems of mechanical, electrical and electronics devices including structure such as building, automobile bodies, air plane, very large scale integrated chips (VLSI) and computer networks. These designs are more frequently used to test the structural, electrical and thermal properties of the system.

4) Entertainment:
Disney movies such as Lion kings and the Beauty and the Beast and the other scientific movies like startrek and highly animated film like Avatar are the best examples of the application

of computer graphics in the field of entertainment. Computer and video games such as FIFA worldcup, Formula Super Bike and Motto are few games where computer graphics is extensively used.

5) Art and commerce:
Here, computer graphics is used to produce pictures that express a message and attract attention such as a new model of a car moving around along the ring of a saturn. These pictures are frequently seen at transpotation terminals, supermarket, hotels, etc. The slide production of commercial, scientific or educational, presentation is another cost effective use of computer graphics. One of such graphics package is power point.

6) Cartography:
Cartography is a subject, which deals with making maps and charts. Computer graphics is used to produce both accurate and schematic representation of geographical and other natural phenemenon from measurement data. Eg: includes geographic maps, whether maps, global information system (GIS) and Global Positioning System (GPS) are strictly based on such type of geographical chart. Surfer is one such graphic package which is extensively used for cartography.

7) Medical:

Computer graphics is now widely used in medical sector. CG becames a powerful tool of diagnosis and treatment in the hand of doctors. Computerized axial Tomography (CAT) is a system through which two dimensional images of cross-section of human body or specific organs are produced. Tomography, is a technique of X-ray photographic that allows cross sectional views of physiological system to be displayed. CT scan technology is another example where CG is used.

8) Internet:

Internet is the computer network system where computer graphic is maximum used. Internet would be nothing without graphics. We can listen high quality streaming music or watch movies on the web. So, without CG on the web it would be just like a notepod file which contains only text but no images.

9) Simulation:

* Characteristics / Importance of Computer Graphics (CG):

1) The interactive nature and preview capability of computer processing is extremely useful in CG. One can create a drawing without resorting drawing instruments and without back breaking physical effort from various angles and in different styles.

2) CG is not only pretty and expressive to look but also very user friendly making its operation adventure.

3) With most computer applications, it must be remembered that invaribly it is not the computer that makes mistakes, it is the person entering the data or using the command.

2. Graphics Hardware

2

* Hardware concepts:

2.1 Interractive input devices:

1) Keyboard

Keyboard is a device primarily used for entering text string i.e. keyboard is an efficient device for inputing non graphics data. It usually consists of alphanumeric keys, function keys, cursor keys and a separate numeric key pad. These keys are used to move the cursor, to select the menu items, predefined functions.

2) Mouse

Mouse is a small hand held device used to position the cursor on the screen. They can be picked up, moved in space and then put down again without any change in the reported position. It's types are - mechanical and optical mouse.

3) Tablet or Digitizer

A tablet is a digitizer. In general, digitizer is a locator device which is used for drawing, painting, or interactively selecting coordinate positions on an object, and to input a set of discrete coordinate position. These positions can be joined with straight line segments to approximate the original shape of an object. Tablet digitizers

an object by detecting the position of movable stylus (pencil shaped device) held in user hand. A tablet consist of flat surface and its size varies from 6 by 6 inches upto 48 by 72 inches or more. It's types are:- The image generally does not appear on the tablet itself but rather is displayed on computer monitor

i) Electrical tablet
ii) Sonic tablet
iii) Resistive tablet

2014-03-25

Working principle:

When pressure is applied to a point on a tablet using a stylus, the horizontal wire and a verticle wire associated with the corresponding grid point meet each other causing an electric current to flow into each of the wire. Since an electric current is only present in the two wires that meet, the unique coordinate for the stylus can be retrieved. The coordinate returned are tablet coordinates which are converted to the user or screen coordinates by an imaging software.

i) Electric tablet:

A grid of wires on one by fourth $(1/4)^{th}$ to one by two $(1/2)^{th}$ inch centre is embeded in the tablet surface. And electromagnetic signals generated by

the electric pulses is applied in the sequence to the wire in the grid to generate electrical signals in a wire coil in the stylus. The strength of the signal generated by each pulse is used to determine the position of the stylus. The signal strength is also used to determine roughly how far the stylus is from the tablet.

ii) Sonic tablet:

The sonic tablet uses sound waves to couple the stylus to microphone positioned on the periphery of the digitizing area. An electric spark at the tip of the stylus creates sound bursts. The position of the stylus or coordinate values is calculated using the delay between when the sparks occurs and when its sound arives at each microphones.

iii) Resistive tablet:

The tablet is just a piece of glass coated with thin layer of conducting material. When a battery powered stylus is activated at certain position, it emits high frequency radio signals, which generate the radio signal on the conducting layers. The strength of signal received at the edge of the tablet is used to calculate the position of the stylus.

Touch panel [Touch screen]:

A touch screen is a computer display screen that is sensitive to human touch, allowing a user to interact with a computer by touching pictures or words on the screen. Touch screen system accept input directly through the monitor. It uses sensors to detect the touch of a finger and are useful where environmental condition prohibit the use of keyboard or mouse. It is also useful for selecting options from menus.

In a typical optical touch panel, light emitting diodes (LED) are mounted in adjacent edges - one vertical and one horizontal. The opposite vertical and horizontal edges contain light detectors (photo diode). These detectors instantly identifies which two orthogonal light beams emitted by the LEDs are blocked by a finger or other pointing device and thereby records the (X,Y) coordinates of the screen position touched for selection. This is low resolution touch panel having 100x50 positions in each direction.

Types are:

- Electric touch panel:

An electric touch panel is constructed with two transparent plates separated by a small distance. One of the plate is coated with a conducting

material and the other plate is coated with a resistive material. When the outer plate is touched, it is forced into contact with the inner plate. This contact creates a voltage drop across a resistive plate which is converted to the coordinate values of the selected screen position.

• Sonic touch panel: Burst of high frequency sound waves travelling alternatively horizontally & vertically are generated at the edge of the panel. In this panel, sonic beams are generated from the horizontal and vertical edges of the screen. Touching the screen causes part of the wave to be reflected back to its source. The sonic beam is constructed or reflected back by putting a finger in the designed location on the screen. The screen position at the point of contact is then calculated using the time elasped between the waves is emmitted and when it arrives back at the source. This is high resolution Touch panel having 500 positions in each direction.

• Light pen: working principle (08-S)

The light pen is a pointing device shaped like a pen and is connected to the computer. The tip of the light pen contains a light sensitive element (photoelectric cell) which when placed against the screen detects the light from the screen enabling the computer to identify the location of the pen on the screen. A light pen can work with any CRT based monitor but not with LCD screens.

It is a pencil shaped device to determine the coordinates of a point on the screen where it is activated by pressing the buttons. In Raster display, Y is set at y-max and x-changes from

0 to x-max for the first scanning. For the second line and x again changes from 0 to x-max and so on. When the activated light pen sees burst of light at certain position as the electron beam hits the phosphor coating at that position, it generates an electric pulse which is used to save the x and y coordinate values and interrupt the computer. By reading the saved values, the graphics package can determine the coordinates of the position seen by the light pen.

* Drawbacks of the light pen:

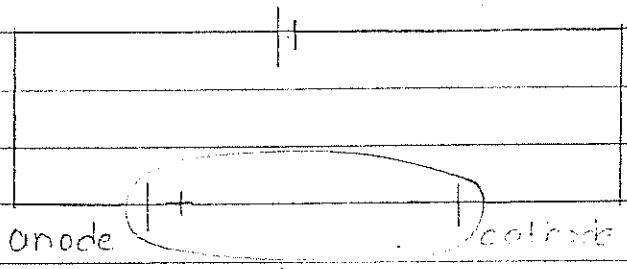
i) It sometimes give false reading due to background light in a room.
ii) It cannot report the coordinates of a point that is completely black.
iii) Prolonged use of a light pen can cause arm fatigue.
iv) Light pen hides/obscures the screen image as it is pointed to a require spot.

2014-05-26

* Cathode ray tube (CRT):

The simplest version of a cathode ray tube consists of a gas filled glass tube in which two metal

plates, one negatively charged (the cathode) and the other positively charged (the anode) have been placed. when a very large voltage is placed across the electrodes, the neutral gas inside the tube will ionize into conducting plasma and a current will flow as electrons travel from the cathode to the other side.



A CRT is a type of analog display device. They are special, electronic vaccum tubes that use focused electrons beams to display image. They are most famous for their use in television, computer, radar display and ATM and are also used in video game equipment. A CRT has a cathode or negatively charged terminal which is a heated filament, much like the filament seen in the light bulb. The filament is contained inside a vaccum with a glass tube. Inside the tube a beam of electrons is allowed to flow from the filament through the vaccum to the anode.

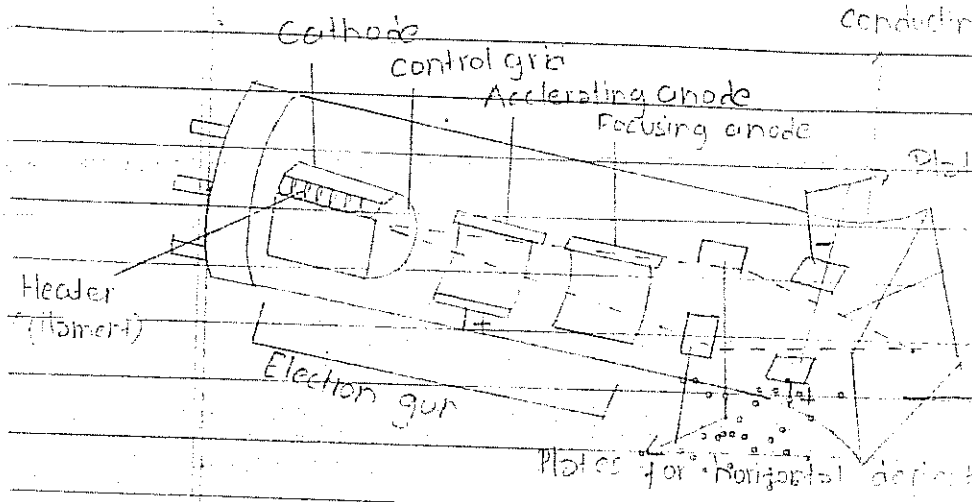


Fig: Schematic diagram of principle elements of CRT / Cross sectional view of monochromatic CRT

Figure given above shows the schematic diagram of principle elements of CRT. There are two types of CRT:

i) Monochromatic CRT
ii) Coloured CRT
   a) Beam penetration
   b) Shadow mask
      - Delta-Delta shadow mask
      - Precision in line.

i) Monochromatic CRT:

The cathode at the left end in the figure is raised to a high temperature by the heater, and the electrons evaporate from the surface of the surface of the cathode. The accelerating anode, with a small hole at it's centre is maintained at a high potential of the order 1-20kV.

relative to the cathode. This potential difference gives rise to an electric field directed from right to left in the region between accelerating anode and the cathode. Electrons passing through the hole in the anode form an arrow beam and travel with constant horizontal velocity from the anode to the screen. The control grid regulates the number of electrons that reach the anode and hence the brightness of the spot on the screen. The focusing anode ensures that electrons leaving the cathode inslightly different directions are focused down to a narrow beam and all arrive at the same spot on the screen. The assembly of cathode, control grid, focusing anode and accelerating electrode is called the electron gun. So, the beam of electron gun emits a stream of electrons that is accelerated towards phosphor coated screen by a high positive voltage applied in the inner side of tube near the screen. The beam of electrons passes between two pairs of deflecting plates. An electric field between the first pair of plates deflects the electrons horizontally and an electric field between the second pair deflects them vertically. If no deflecting field are present the electrons travel in a straight line from the hole in the accelerating anode to the centre of the screen where they

produce a bright spot.

ii) Coloured CRT:

a) The

a) The beam penetration CRT:

The normal CRT can generate image of only a single color due to limitations of its phosphor. A coloured CRT devices uses multi layers namely red and green in the display screen. controlled by modulating a normally constant parameter called beam accelerating potential. The screen is coated with a layer of green phosphor over which a layer of red phosphor is deposited. When a low potential electron beam strikes the screen only the red phosphor is excited thus, producing a red display. A higher velocity beam will penetrate into the green phosphor increasing the green component of the light output. And the electron beam with an intermediate speed penetrates the middle of the screen and gives the combination of red and green such as orange, yellow, etc.

The speed of the electrons and hence the screen color at any point is controlled by the beam accelerating voltage/potential and a display depends upon how far a electron beam penetrates into phosphor layer.

Advantage:
The bigest advantage is that it is half cost of shadow mask and its resolution is better.

Disadvantage:
The bigest disadvantage is that the change of color takes time. which doesn't suit interactive graphics at all.

2014-03-27th

* Resolution:
Resolution is defined as a maximum number of points that can be displayed horizontally and vertically without overlap on the display device.

1024 × 1024 → scan line number
↳ no. of pixel in each scan line

* Refresh rate:
Refresh rate is the number of times per second the image is redrawn to give a feeling of unflickering pictures and it is usually 60 frames per second. As the refresh rate decreases flickering develops because the eye can no longer integrate the individual light impulse coming from a pixel.

Numericals:
Consider three different raster system with resolution of 640 × 480, 1280 × 1024 and 2560 × 2048. What size of frame buffer (in bytes) is needed for

Resolution:

each of these system, to store 12 bits per pixel? How much storage is required for each system if 24 bits per pixel are to be stored?

⇒ Solution:

Case I: For 640 x 480

Total no. of pixel required for 640x480 resolution = 640 x 480 pixels = 307200 pixels.

Given,

1 pixel can store 12 bits.

Therefore, size of frame buffer (bytes)

$$= \frac{640 \times 480 \times 12}{8} \quad [1 \text{ byte} = 8 \text{ bits}]$$

$$= 460800 \text{ byte}.$$

Given,

1 pixel can store 24 bits.

Therefore, size of frame buffer (bytes)

$$= \frac{640 \times 480 \times 24}{8}$$

$$= 921600 \text{ byte}.$$

Case II: For 1280 x 1024

Total no. of pixel required for 1280 x1024 resolution = 1280 x 1024 = 1310720 pixels.

Given,

1 pixel can store 12 bits.

Therefore, size of frame buffer (bytes)

$= \frac{1310720 \times 12}{8}$

$= 1966080$ byte

Given,

1 pixel can store 24 bits.

∴ size of frame buffer (bytes) $= \frac{1310720 \times 24}{8}$

$= 3932160$ byte

Case III : For 2560 × 2048

Total no. of pixel required for 2560 × 2048 is ; $2560 \times 2048 = 5242880$.

Given,

1 pixel can store 12 bits

∴ Size of frame buffer (bytes) $= \frac{5242880 \times 12}{8}$

$= 7864320$ byte.

Given,

1 pixel can store 24 bits.

∴ size of frame buffer $= \frac{5242880 \times 24}{8}$

$= 15728640$ byte

Consider a raster system with a resolution of 1024 x 1024. What is the size of a raster (in byte) needed to store 4-bits per pixel. How much storage is required, if 8-bits per pixel are stored.

=> Solution:

Case I: For 1024 x 1024

Total no. of pixel required for 1024 x 1024 resolution = 1024 x 1024 pixels

= 1048576 pixels.

Given,

1 pixel can store 4-bits

Therefore, size of storage (bytes)

$$= \frac{1024 \times 1024 \times 4}{8}$$

= 524288 byte.

Given,

1 pixel can store 8-bits

Therefore, size of storage (bytes)

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

= 1048576 byte.

Consider two raster systems with resolution of 640 x 480 and 1280 x 1024. How many pixels could be accessed per second in each of these system.

by a display controller that refreshes the screen at a rate of 60 frames per second? What is the access time per pixel in each system.

⇒ Solution:

Case I: For 640 × 480

Total no. of pixel required for 640×480 resolution = 640×480 pixels

= 307200 pixels.

which is the no. of pixels contained by a frame

Now, controller can access 60 frames per second

∴ Total no. of pixels accessed = 60 × 307200 persec

= 18432000

Now,

Access time per pixel = $\frac{1}{\text{total pixel accessed persec}}$

= $\frac{1}{18432000}$

= 5.43×10^{-8} sec

Case II: For 1280 × 1024

Total no. of pixel required for 1280×1024 resolution = 1280×1024 pixels

= 1310720 pixels.

Since, controller can access 60 frames per second,

∴ Total no. of pixels accessed = 60 × 1310720

= 78643200

Then,

Access time per pixel = $\frac{1}{78643200}$

$= 1.27 \times 10^{-3}$ seconds/pixels.

How many kilo bytes does a frame buffer need in a 600x400 pixel?

=> Solution,

Suppose 1 pixel can store n bits. then the size of frame buffer = $n \times 600 \times 400$

$= 240000n$ bits.

$= \frac{240000n}{8}$ bytes

$= 30000n$ bytes

$= \frac{30000n}{1024}$

$= 29.29nKB$ //

How much time is spend to scanning across each row of pixels during screen refresh on a raster system with resolution of 1280x1024 and a refresh rate of 60 frames per second.

=> Solution,

Resolution = 1280x1024 (Given)

That means system contains 1024 scan lines and each scan line contains a 1280 pixels

and, refresh rate = 60 frame / seconds.

That means, 1 frame takes $\frac{1}{60}$ seconds

Therefore, one frame consists of 1024 scan lines
Therefore, 1024 scan lines takes $\frac{1}{60}$ seconds.

∴ one scan lines takes $\frac{1}{60} \times \frac{1}{1024}$

$= 1.63 \times 10^{-5}$ secs.

How long would it take to load a 640×480 frame buffer with 12 bits per pixels, if 10^5 bits can be transffered per second? How long would it take to load a 24 bits per pixel frame buffer with a resolution of 1280×1024 using the same transfer rate.

⇒ Solution;

Case I:

b) Shadow Mask:

Shadow mask method are commonly used in raster scan system (including color TV) because they produce a much wider range of colors than the beam penetration method. 256 colors can be produced by this method. A shadow mask CRT has three phosphor color dots (i.e. R, G, B) at each pixel position called triad. One phosphor dot emits red light, another emits a green light and the third emits a blue light. This type of CRT has three electron guns, one for each (phosphor) color dot and a shadow mask grid just behind the phosphor coated screen (whose holes are same as no of pixels). Before the stream of electrons produced by the CRT cathode reaches the phosphor coated plate, it encounters the shadow mask which is a sheet of metal engraved with a pattern of holes. The mask is positioned in the glass funnel of the CRT during manufacture and the phosphor is coated on to the screen so that electrons coming from the red, green, blue gun position only strikes the appropriate phosphor dot.

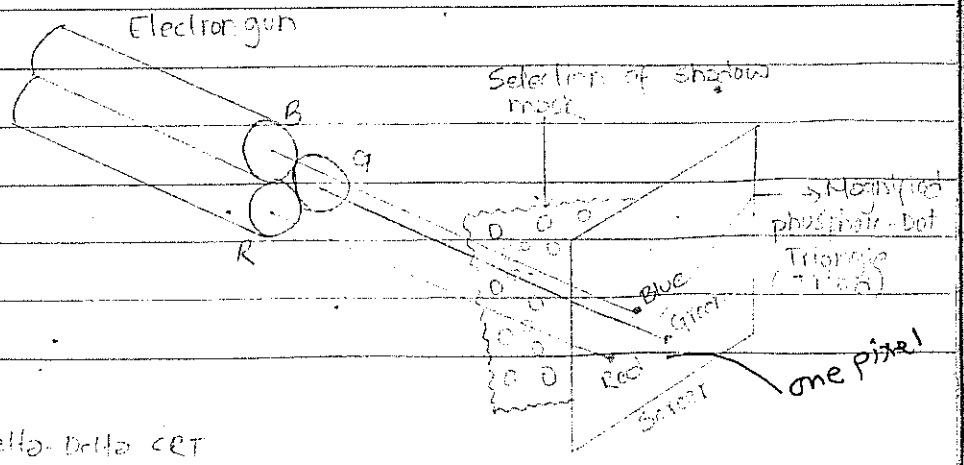


Fig: Delta-Delta CRT

* Advantage:

∅

+ Delta - Delta shadow mask:

The beam of electrons are deflected and focused onto the shadow mask which contains the series of holes. When electron hit phosphor they activate a dot triangle. The phosphor dots triangle are arranged such that electron beam activate only its corresponding color dots when it passes through the shadow.

* Drawback:

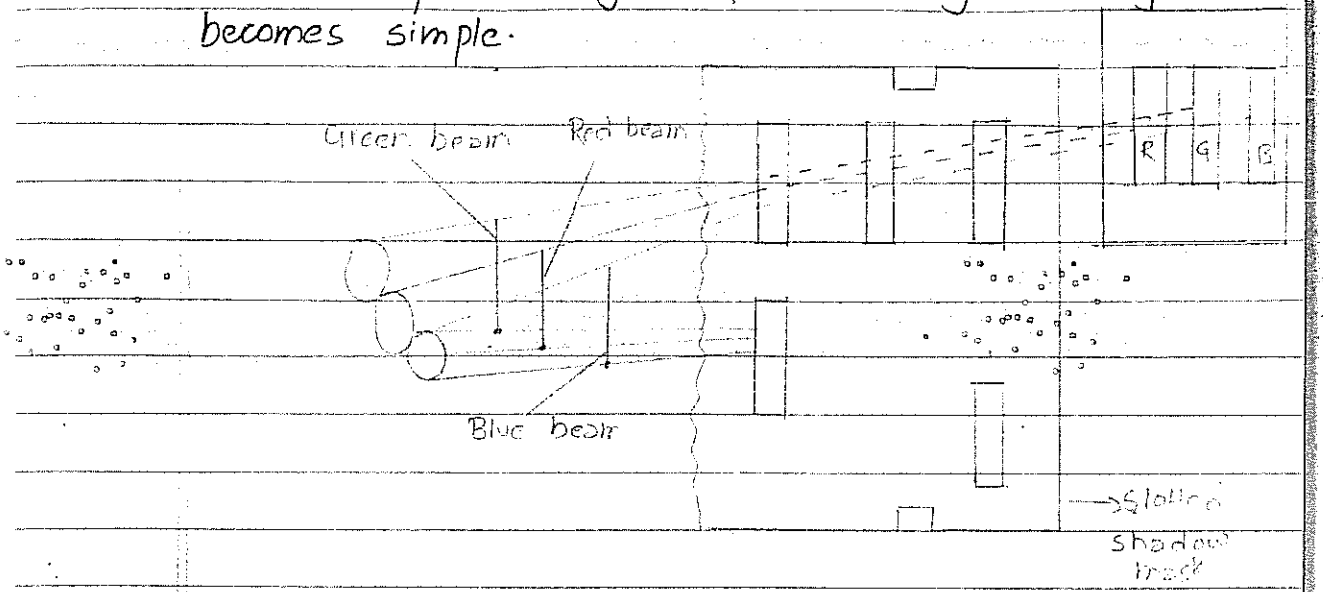
- The drawback of this CRT is to achieve high precision due to alingment of shadow mask holes and triads.
- The focus is not sharp over the complete screen.

+ Precision in Line:

A triad has a line pattern as are three electron guns. It eliminates the drawback of delta-delta CRT. As theire is continuous strip from top to bottom of the screen for each color. They are placed one after another on screen. But, it is slightly reduces the image sharpness at the edge of the tubes.

* Advantages:

Because of inline guns, the convergence adjustment becomes simple.



2019 04 - 9th

* Raster Display Technology:

Raster display technology is developed in early 70's. based on television technology; common CRT monitor In this System, The electron beam in raster scan display is swept across the screen, one row at a time from top to bottom and when it reaches bottom, scan starts again from top and produce zig-zag line. As the electron beam moves across each row, the beam intensity is turned ON as one & OFF as 0 to create a pattern of brightening light or spot. Picture definition is stored in a memory area called refresh buffer or frame buffer. It holds the set of intensity values for all screen points (pixels). The stored intensity values are retrived from frame buffer and displayed or painted on the screen one row at a time. (Each screen point is reffered as a pixel.)

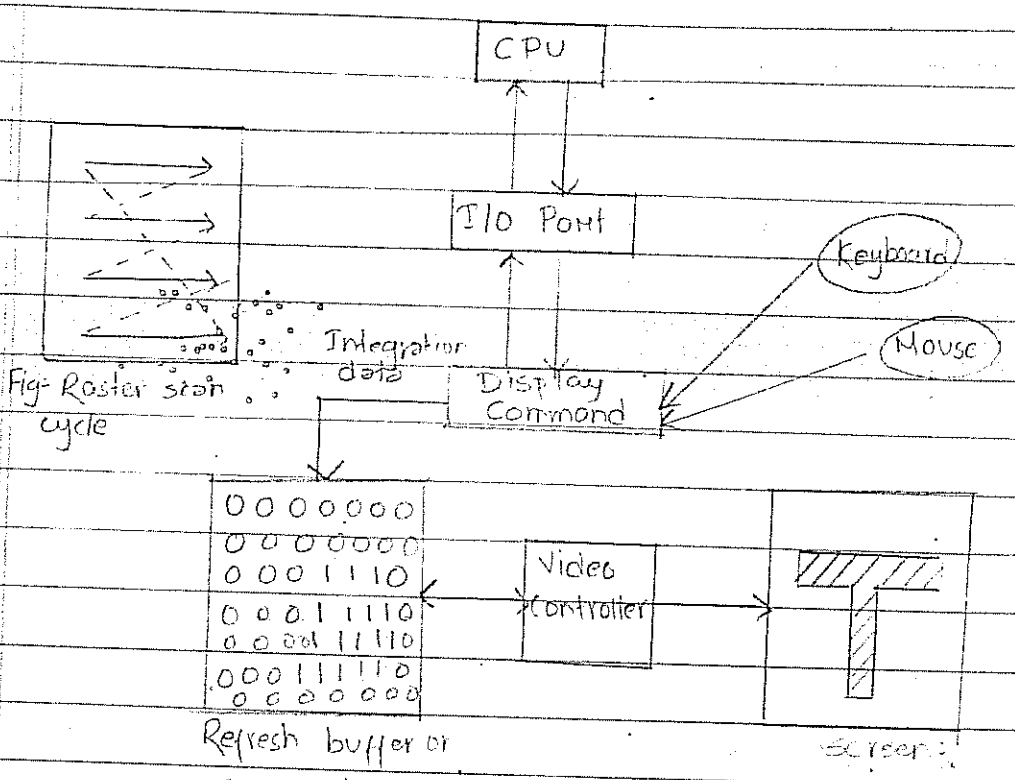


Fig: Raster scan cycle

Fig: Architecture of raster display

The architecture of raster display consists of video controller, CPU, refresh buffer, CRT (screen), keyboard, mouse as shown in figure. The picture definition or display image is stored in a form of 0's and 1's in the refresh buffer or frame buffer. The video controller read this display buffer and produce the actual image on the screen. It controls the operations of display device and access the frame buffer to refresh the screen.

The raster scan proceeds as follows:
Starting from top left corner of the screen, the

electron gun scans horizontally from left to right one scan line (row) at a time jumping to the left end of the next lower row until the bottom right corner is reached. Then it jumps again to the top left corner and starts again finishing one complete refresh cycle. When the beam is moved from left to the right it is ON and when it is moved from left right to the left it is OFF. Positive x-valve increases to the right and positive y-valve increases from bottom to top. Scan line are labelled from 0 to Y_{max} and from Y_{max} at the top of screen to zero at the bottom.

* Random or Vector display technology:

In this display technology pictures lines, charts are created on the tube surface in any random order or direction given in vectorial way. For this reason, this type of device was also known as vector calligraphic or stroke. For eg: if we want a line connecting a point A with point B, we simply drive the beam deflection circuit which will cause beam to go directly from point A to point B.

Random display architecture

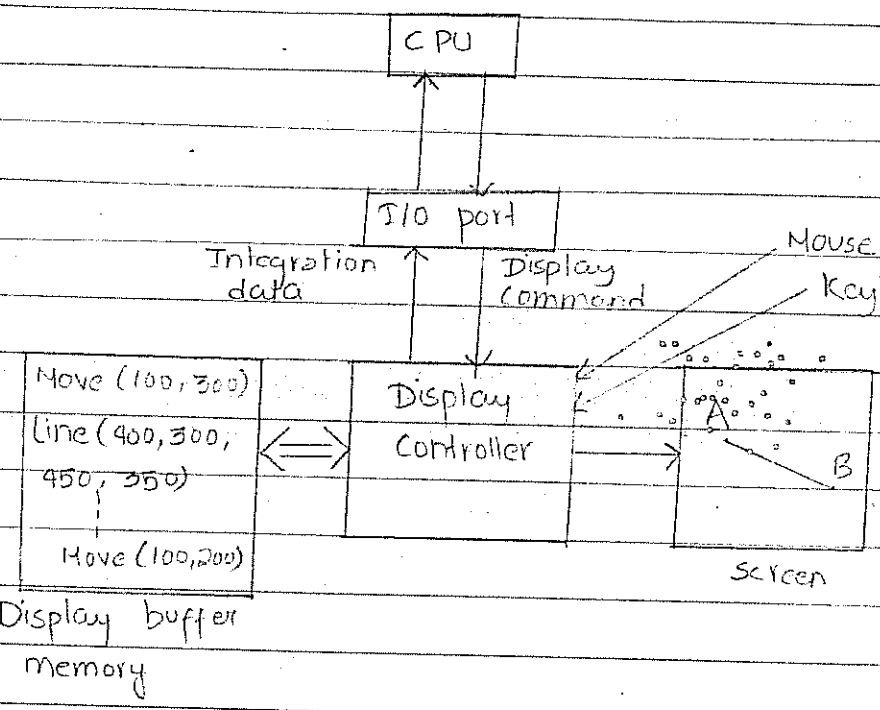


Fig: Architecture of Random display

Random display architecture consists of display (processor) controller, display buffer memory, CPU and CRT. A display controller is connected to a CPU. The display buffer ~~or~~ stores the computer produced display list or display program which contains line plotting commands with (x,y) end points co-ordinate as well as character plotting command; The display controller interprets commands for plotting points, lines and characters and sends digital and point coordinate to a vector generator. The vector generator then converts the digital coordinate values. The vector generate to analog deflection voltage for beam deflection circuit that move an electron beam writing on the CRT phosphor coated

screen. Thus, beam sweapt across the screen doesnot follow any fixed pattern, the direction is varbitary as given by display command.

* Differences between Raster and Random display technology :

Raster	Random
1) The line produced are zig-zac as the plotted values are discrete points sets.	1) Smooth and continvous lines cure produced as the electron beam directly follows the line path because CRT beam follows the line path only.
2) In this case the electron beam is sweapt across the screen one row at a time from top to bottom.	2) Here, CRT has the electron beam directly only to the ~~only the spot~~ parts of the screen, where a picture is to be drawn.
3) Picture definition is stored as a set of intensity values for all screen points or pixel in a memory area called refresh buffer or frame buffer.	3) Picture definition is stored as a set of line drawing commands in a memory area called display buffer memory.
4) It has less Resolution.	4) It has high resolution.

5) Graphic primitives are specified in terms of their end points and must be scan converted into their corresponding pixels or points in the frame buffer.	5) Scan conversion is not required.
6) Refreshing on Raster scan display is carried out at the rate of 60 to 80 frames per second.	6) Random scan system carries out the refresh cycle at the rate of 30 to 60 times per second.
7) It is less costly than Random display technology.	7) It is generally costlier than Raster display technology.
8) Raster display has ability to display areas with certain patterns.	8) It displays or draws only lines and characters.
9) Editing is difficult.	9) Editing is easy.

* DDA algorithm:

10) Use interlacing	10) Does not use interlacing
11) Refresh rate is independent of picture complexity.	11) Refresh rate depends directly on picture complexity or upon the number of lines to be displayed
12) Solid/pattern fill is easy.	12) Solid/pattern fill is difficult.
13) Use monochrome or shadow mask type	13) Use monochrome or beam penetration type.

3. Two Dimensional Algorithms

Date: 04/04/11th

* DDA algorithm: (Digital Differential Analyzer)

DDA algorithm is a line drawing algorithm based on obtaining successive pixel value. that is: Δx and Δy. It is a scan conversion line algorithm based on calculating either Δx or Δy. The basic of DDA method is to take unit step along one of the coordinate assume x coordinate and compute the corresponding values along the other coordinates, lets say y coordinate.

Eg: if we have $dx = 11$ and $dy = 6$ then we would take unit steps along x-coordinate and compute the steps m along y

$$\text{slope}(m) = \frac{dy}{dx}$$

If $(m <= 1.0)$ then

Let x_step = 1

{$dx = 1$, $dy = 0$ or 1}

else

$(m > 1.0)$

Let Y_step = 1

{$dy = 1$, Y_step = 0 or 1}

Consider a line with positive slope.

If slope(m) ≤ 1, then we sample at unit x interval ($\Delta x = 1$) and compute each successive y values as: i.e $\Delta x = 1$

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

$$m = \frac{dy}{dx} \qquad \Delta y = m$$

$$= \frac{y_{k+1} - y_k}{1} \rightarrow y_{k+1} - y_k = m$$

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

For a line with positive slope if $m > 1$, then we sample at unit y-interval and calculate each successive x-values as: ie $\Delta y = 1$.

$$m = \frac{dy}{dx} \qquad \Delta x = \frac{\Delta y}{m} = \frac{1}{m}$$

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

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

Algorithm for DDA method of line drawing:

1) Input two end points (x_0, y_0) as left end point and (x_n, y_n) as right end point.
2) Calculate the horizontal deflection $(dx) = x_n - x_0$. and verticle deflection $(dy) = y_n - y_0$.
3) Test if (absolute(dx) > absolute(dy))
   steps = absolute(dx)
   else
   steps = absolute(dy)
4) Calculate $\Delta x = \frac{dx}{steps}$ and $\Delta y = \frac{dy}{steps}$

5) Start with (x_0, y_0) to determine offset needed at each step to generate the next pixel position along the line steps times and plot (x_0, y_0) as (x, y).

6) Do

For ($k = 1$ to steps)

{

$x = x + dx$

$y = y + dy$

}

End do

8) Using a DDA algorithm digitize a line from point P(3, 6) to (12, 13).

⇒ Here,

$(x_0, y_0) = (3, 6)$

$(x_n, y_n) = (12, 13)$

$dx = x_n - x_0 = 12 - 3 = 9$

$dy = y_n - y_0 = 13 - 6 = 7$

Here, $dx > dy$

$\therefore$ steps $= 9$

Now,

$$\Delta x = \frac{dx}{\text{steps}} = \frac{9}{9} = 1 \quad \& \quad \Delta y = \frac{dy}{\text{steps}} = \frac{7}{9} = 0.78$$

$$m = \frac{\Delta y}{\Delta x} = \frac{0.78}{1} = < 1$$

x_k	y_k	$x_{k+1}(x+\Delta x)$	$y_{k+1}(y+\Delta y)$	Rounded
3	6	4	6.78	(4, 7)
4	6.78	5	7.56	(5, 8)
5	7.56	6	8.34	(6, 8)
6	8.34	7	9.12	(7, 9)
7	9.12	8	9.90	(8, 10)
8	9.90	9	10.68	(9, 11)
9	10.68	10	11.46	(10, 11)
10	11.46	11	12.24	(11, 12)
11	12.24	12	13.02	(12, 13)

Hence, the line between (3,6) and (12,13) has been plotted using DDA algorithm.

Q) Plot the line between the points (16,18) to (10,10)

=> Here,

$(x_0, y_0) = (16, 18)$

$(x_n, y_n) = (10, 10)$

$dx = x_n - x_0 = 10 - 16 = |-6| = 6$

$dy = y_n - y_0 = 10 - 18 = |-8| = 8$

Here, $dy > dx$

$\therefore$ Steps $= 8$

Now,

$$\Delta x = \frac{dx}{\text{Steps}} = \frac{-6}{8} = -0.75$$

& $$\Delta y = \frac{dy}{\text{Steps}} = \frac{-8}{8} = -1$$

x_k	y_k	x_{k+1}	y_{k+1}	Rounded
16	18	15.25	17	(15,17)
15.25	17	14.5	16	(15,16)
14.5	16	13.75	15	(14,15)
13.75	15	13	14	(13,14)
13	14	12.25	13	(12,13)
12.25	13	11.5	12	(12,12)
11.5	12	10.75	11	(11,11)
10.75	11	10	10	(10,10)

Hence, the line between (16,18) to (10,10) has been plotted using DDA algorithm.

Advantages and disadvantages of DDA:

Advantages:

1) Faster than the direct use of line equation and it does not do any floating point multiplication.

Disadvantages:

1) Cummulative error due to precesion in the floating point representation which may cause calculated point to drift away from their actual position.

014 04 16th

* Bresenham's Line Drawing algorithm:

It is an improvement upon DDA algorithm to use integer arithmetic only. The bresenham algorithm is incremental scan conversion algorithm. The big advantage of this algorithm is that it uses only integer calculations. The basic idea behind this algorithm is to find the closest pixel position to the line path.

For $m < 1$

In Bresenham's Line algorithm we test the sign of an integer parameter P_k whose value is proportional to the difference between the separation of the pixel position from the actual line $y = mx + c$ i.e.

$$P_k \propto (d_1 - d_2)$$

where,

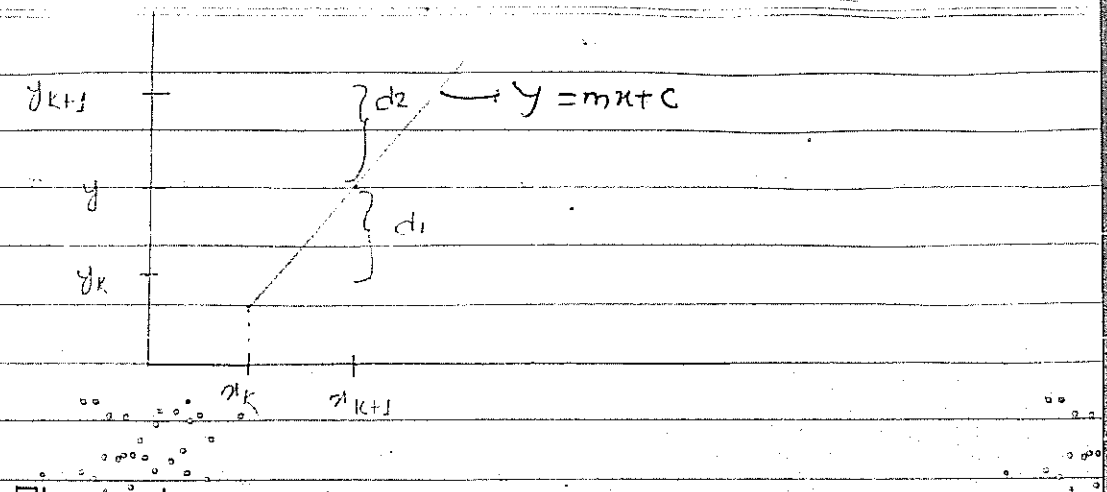
P_k = integer value

d_1, d_2 = separation of two pixels p

Let, us consider a line with positive slope less than one ($m < 1$) starting from (x_0, y_0), we take sampling at unit x-direction (interval) in each successive column (x-position) and plot the pixel whose scan line and plot the pixel whose scan line y value is closest to the line path. If d_1 and d_2 are the vertical separation of pixels from the line then the value of y at x_{k+1} or at sampling position $(x_k + 1)$ is $y = mx_{k+1} + c$ ——— (i)

P. processing Left to right

Suppose x_k & y_k has just plotted, then two conducted pixels to be plotted are (x_{k+1}, y_k) or (x_{k+1}, y_{k+1})



Then, $d_1 = y - y_k$

$= (mx_{k+1} + c) - y_k$

and $d_2 = y_{k+1} - y$ $\quad \because [y_{k+1} = y_k + 1]$

$= y_k + 1 - x_{k+1} m - c$

Now, the difference between two separation is;

$d_1 - d_2$

$= mx_{k+1} + c - y_k - y_k - 1 + x_{k+1} m + c$

$d_1 - d_2 = 2mx_{k+1} - 2y_k + 2c - 1$ ——— (A)

Now, multiplying Δx on both sides, we get;

$\Delta x (d_1 - d_2) = 2\Delta y x_{k+1} - 2y_k \Delta x + 2c\Delta x - \Delta x$ $\quad [\because m = \frac{\Delta y}{\Delta x}]$

$P_k = \Delta x (d_1 - d_2) = 2\Delta y x_{k+1} - 2y_k \Delta x + 2c\Delta x - \Delta x$

$P_k = 2\Delta y (x_k + 1) - 2y_k \Delta x + 2c\Delta x - \Delta x$ $\quad [\because x_{k+1} = x_k + 1]$

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

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

where,

$B = 2\Delta y + 2c\Delta x - \Delta x$ is constant term and is independent of any pixel position.

If $P_k < 0$ then, $d_1 - d_2 < 0 \Rightarrow d_1 < d_2$

So, y_k is more closer than y_{k+1} from the line path and we should plot (x_{k+1}, y_k).

Else, we should plot (x_{k+1}, y_{k+1}).

Now, coordinate changes along the line occur in unique unit interval then we can obtain successive decision parameter P_{k+1} using incremental integer calculation. So, decision parameter for next step $(k+1)$ is

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

Now, substracting eq^n (ii) by eq^n (iii);

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

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

If $P_k < 0$ then, we plot lower pixel

$$x_{k+1} = x_k + 1 \quad \& \quad (x_{k+1}, y_k)$$

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

Then; (Putting above value in eq^n 2)

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

Else, $P_k \geq 0$, then we plot upper pixel

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

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

Then,

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

For initial decision:

To start from initial point, we need initial point with its initial decision parameter (P_0) which can be derived from equation (A)

$$d_1 - d_2 = 2mx_{k+1} - 2y_k + 2c - 1$$

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2c - 1 \quad [\because x_{k+1} = x_k + 1]$$

$$d_1 - d_2 = 2mx_k + 2m - 2y_k + 2c - 1$$

$$d_1 - d_2 = 2(mx_k - y_k + c) + 2m - 1$$

$$d_1 - d_2 = 2m - 1 \quad [\because y_k = mx_k + c]$$

$$d_1 - d_2 = 2\frac{\Delta y}{\Delta x} - 1$$

$$d_1 - d_2 = \frac{2\Delta y - \Delta x}{\Delta x}$$

$$\Delta x(d_1 - d_2) = 2\Delta y - \Delta x$$

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

Numerical

1) Digitize a line with end points (20,10) and (30,18) using Bresenham's line drawing algorithm

=> Solution,

Given;

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

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

Now;

Slope (m) $= \frac{x_2 - x_1}{y_2 - y_1} = \frac{30-20}{18-10} = \frac{10}{8} = 0.8 < 1$

For $m < 1$

$P_0 = 2\Delta y - \Delta x = 2 \times 8 - 10 = 16 - 10 = 6$

Initial point to plot is: (20,10)

K	P_K	x_{K+1}, y_{K+1} (next pixel value)	P_{K+1}	$P_{K+1} = P_K + 2\Delta y$
1	6	21, 11	2	
2	2	22, 12	−2	
3	−2	23, 12	−6 14	$P_{K+1} = P_K + 2\Delta y$
4	−6 14	24, 13	10	
5	10	25, 12	~~14~~ 6	
6	~~14~~ 6	26, 13	~~8~~ 2	
	2	27, 16	−2	
	−2	28, 15	14	
	14	29, 17	10	
	10	30, 18		

2) Draw a line using Bresenham algorithm between points (1,1) to (8,5).

=> Solution;

Given;

$(x_1, y_1) = (1,1)$

$(x_2, y_2) = (8,5)$

Now,

Slope $(m) = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5-1}{8-1} = \frac{4}{7} = 0.57 < 1$

For $m < 1$; $P_0 = 2\Delta y - \Delta x = 2 \times 4 - 7 = 8 - 7 = 1$

Initial point to plot is (1,1)

K	P_K	x_{k+1}, y_{k+1}	P_{K+1}
1.	1	2, 2	-5
2.	-5	3, 2	3
3.	3	4, 3	-3
4.	-3	5, 3	5
5.	5	6, 4	-1
6.	-1	7, 4	7
7.	7	8, 5	

2014-04-18th

* Bresenham's Line drawing algorithm :-

1) Input the two line end points (x_1, y_1) & (x_2, y_2) and store the left end point in (x_0, y_0).

2) Load (x_0, y_0) into the frame buffer i.e. plot the first point.

3) Calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - \Delta x$ and obtain the starting value for the decision parameter as $P_0 = 2\Delta y - \Delta x$.

4) At each x_k along the line starting at $k=0$ perform the following test.

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

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

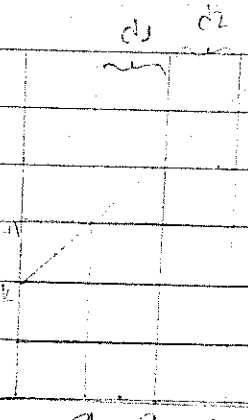
otherwise,
The next point to plot is $(x_k + 1, y_k + 1)$ and

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

5) Repeat step 4 Δx times.

* Bresenham's line drawing algorithm (1m1 ≤ 1)

Suppose x_k and y_k are the initial coordinate of the line which has been plotted in figure. Then two candidate pixels to be plotted are $(x_k, y_k + 1)$ or (x_{k+1}, y_{k+1})



If d_1 and d_2 are vertical separations of pixels from the line then the value of x at y_{k+1} or at sampling position y_{k+2} is:

$$y = mx + c$$

$$x = \frac{y - c}{m}$$

$$x = \frac{y_{k+1}}{m} - \frac{c}{m} \quad \text{---(1)}$$

Then,

$$d_1 = x - x_k$$

$$d_1 = \frac{y_{k+1}}{m} - \frac{c}{m} - x_k$$

$2c\Delta x - 2\Delta y x_k - \Delta y$ | $\frac{2(y_k+1)\Delta y}{m}$ | $\frac{2\Delta y \cdot \Delta x y_k + 2\Delta y}{\Delta y}$ | $= 2\Delta x y_k + 2\Delta y$

and,

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

$$d_2 = x_{k+1} - \frac{y_{k+1}}{m} + \frac{c}{m}$$

$$d_1 - d_2 = \frac{y_{k+1}}{m} - \frac{c}{m} - x_k - x_{k+1} + \frac{y_{k+1}}{m} - \frac{c}{m}$$

or, $d_1 - d_2 = 2\frac{(y_k+1)}{m} - \frac{2c}{m} - 2x_k - 1$ —— (ii)

Multiplying Δy on both sides;

$$\Delta y (d_1 - d_2) = 2\Delta x y_k + 2\Delta x - 2c\Delta x - 2\Delta y x_k - \Delta y$$

Let us define a decision $P_k = \Delta y (d_1 - d_2)$

$$P_k = 2\Delta x y_k - 2\Delta y x_k + B \quad \text{—— (iii)}$$

where,

$$B = 2\Delta x - 2c\Delta x - \Delta y$$

If $P_k < 0$ then $d_1 - d_2 < 0$

$\Rightarrow d_1 < d_2$

So, x_k is more closure than x_{k+1} from the line and we should plot (x_k, y_{k+1}) else we should plot (x_{k+1}, y_{k+1})

Now,

Coordinate changes along the line occur in unique unit interval. Then, we can obtain successive decision parameter using incremental integer i.e.

$$P_{k+1} = 2\Delta x y_{k+1} - 2\Delta y x_{k+1} + B \quad \text{—— (iv)}$$

We get,

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

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

If $P_k \geq 0$, then $y_{k+1} = y_k + 1$

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

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

For initial decision parameter from eqⁿ (11)

$$d_1 - d_2 = \frac{2(y_k + 1)}{m} - \frac{2c}{m} - 2x_k - 1$$

$$d_1 - d_2 = \frac{2y_k}{m} + \frac{2}{m} - \frac{2c}{m} - 2x_k - 1$$

$$d_1 - d_2 = \frac{2}{m} + 2\left(\frac{y_k}{m} - \frac{c}{m} - x_k\right) - 1$$

$$d_1 - d_2 = \frac{2}{m} - 1 \quad \left[\because x_k = \frac{y_k}{m} - \frac{c}{m}\right]$$

$$\Delta y (d_2 - d_1) = 2\Delta x - \Delta y$$

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

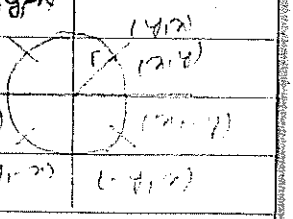
* Circle:

A circle is defined as a state of points that are all at a given distance 'r' from the center position (x_c, y_c).

DN 04 21

⇒ Mid point Algorithm:

It is based on incremental calculation of decision parameter. It is used to find closest pixel to the circumference at each sampling step. The idea in this approach is to test half way position between two pixel to determine if this mid-point is inside or outside the circle boundary. We use symmetric property to find the points over an octant. We can take unit step in positive x-direction and by using decision parameter we find the closest y positioned in the circular path.



Consider a circle section for $x=0, x=y$ where slope of the curve varies from 0 to 1. Calculation of circle point (x,y) in one octane gives the circle point shown for other seven octanes. To apply the mid point, we define a circle function as:

$$F_c(x,y) = x^2 + y^2 - r^2$$

Suppose, if pt (x,y) lies inside the circle Boundary then $F_c(x,y) < 0$.

If pt (x,y) lies on the circle boundary, then $F_c(x,y) = 0$

If pt (x,y) lies outside the circle boundary then $F_c(x,y) > 0$

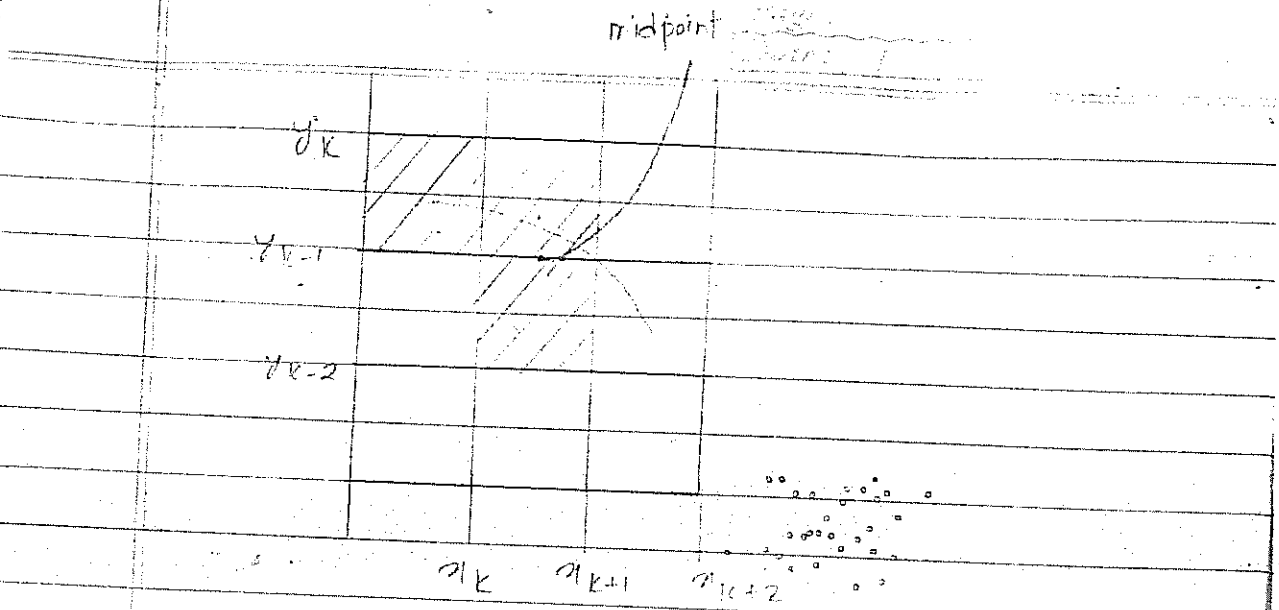


Fig: mid point between candidate pixels at sampling position x_{k+1} along a circular path

Here, assume that we have just plotted the pixels (x_k, y_k). We next need to determine whether the pixel at position (x_k+1, y_k) or the one at position (x_k+1, y_k-1).

The decision parameter is the circle function which we have evaluated in equation (1) at the mid point between these two pixels.

$$P_k = F_c\left(x_k + 1, y_k - \frac{1}{2}\right)$$
$$= (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad \text{——(11)}$$

Successive decision parameter are obtained by using incremental parameter.

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

$P_k < x^2+y^2<r^2$

$$P_{k+1} = f_c(x_{k+1}+1, y_{k+1}-\tfrac{1}{2})$$
$$= (x_{k+1}+1)^2 + (y_{k+1}-\tfrac{1}{2})^2 - r^2$$
$$= (x_{k+1}+1)^2 + (y_{k+1})^2 - y_{k+1} + (\tfrac{1}{2})^2 - r^2$$
$$= (x_{k+1})^2 + 2x_{k+1} + 1 + (y_{k+1})^2 - y_{k+1} + (\tfrac{1}{2})^2 - r^2$$
$$= (x_{k+1})^2 + y_k^2 - y_k + (\tfrac{1}{2})^2 - r^2 - y_k^2 + y_k + 2(x_{k+1}) + 1 + (y_{k+1})^2 - y_{k+1}$$

[∵ adding and substracting y_k^2 and y_k]

$$= (x_{k+1})^2 + (y_k - \tfrac{1}{2})^2 - r^2 - y_k^2 + y_k + 2(x_{k+1}) + 1 + (y_{k+1})^2 - y_{k+1}$$
$$= P_k - y_k^2 + y_k + 2(x_{k+1}) + 1 + (y_{k+1})^2 - y_{k+1}$$
$$= P_k + 2(x_{k+1}) + ((y_{k+1})^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

If $P_k < 0$, then, $y_{k+1} = y_k$ and the mid point is inside the circle and the pixel on scan line y_k is closer to circle boundary.

$$\therefore P_{k+1} = P_k + 2(x_{k+1}) + 1$$

Else; $y_{k+1} = y_k$ and the mid point is outside the circle boundary. and we select the pixel at $y_k - 1$.

$$\therefore P_{k+1} = P_k + 2(x_{k+1}) + 1 - 2(y_{k+1})$$

Initial decision parameter P_0 at $(x_0, y_0) = (0, r)$ is evaluated as $P_0 = f_c(x_0+1, y_0 - \frac{1}{2})$

$$= F(0+1, r - \tfrac{1}{2})$$
$$= (1, r - \tfrac{1}{2})$$
$$= 1 + (r - \tfrac{1}{2})^2 - r^2$$

$$= 1 + x^2 - r + \frac{1}{4} - x^2$$

$$= \frac{1}{4} + 1 - r$$

$$= \frac{5}{4} - r$$

$$= \approx 1 - r$$

$\therefore P_0 = 1 - r$

Midpoint circle algorithm:

1. Input radius 'r' at circle centre (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$ i.e. initialising starting point as $x_0 = 0$ and $y_0 = r$.

2. Calculate the initial value of the decision parameter as $P_0 = 1 - r$.

3. At each 'x_k' position, 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 $P_{k+1} = P_k + 2(x_{k+1}) + 1$.
- Otherwise, the next point along the circle is $(x_{k+1}, y_k - 1)$ and $P_{k+1} = P_k + 2(x_{k+1}) + 1 - 2(y_{k+1})$

4. Determine symmetric points in the other seven octanes.

5. Move each calculated pixel position (x, y) into the circular path centered on (x_c, y_c) and plot the coordinate values $x = x + x_c$ and $y = y + y_c$.

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

Numerical

Digitize a circle center at origin and radius 10 by using mid point circle algorithm.

⇒ Given,

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

radius $(r) = 10$

Here,

$(x_0, y_0) = (0, r) = (0, 10)$

Now,

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

The value of $P_0 = -9$ i.e. < 0.

So,

K	P_K	x_{K+1}, y_{K+1}	P_{K+1}
1	-9	1, 10	-6
2	-6	2, 10	-1
3	-1	3, 10	6
4	6	4, 9	-3
5	-3	5, 9	8
6	8	6, 8	5
7	5	7, 7	6

Assingment

Digitize a circle $(x-2)^2 + (y-3)^2 = 25$.

2014-04-23

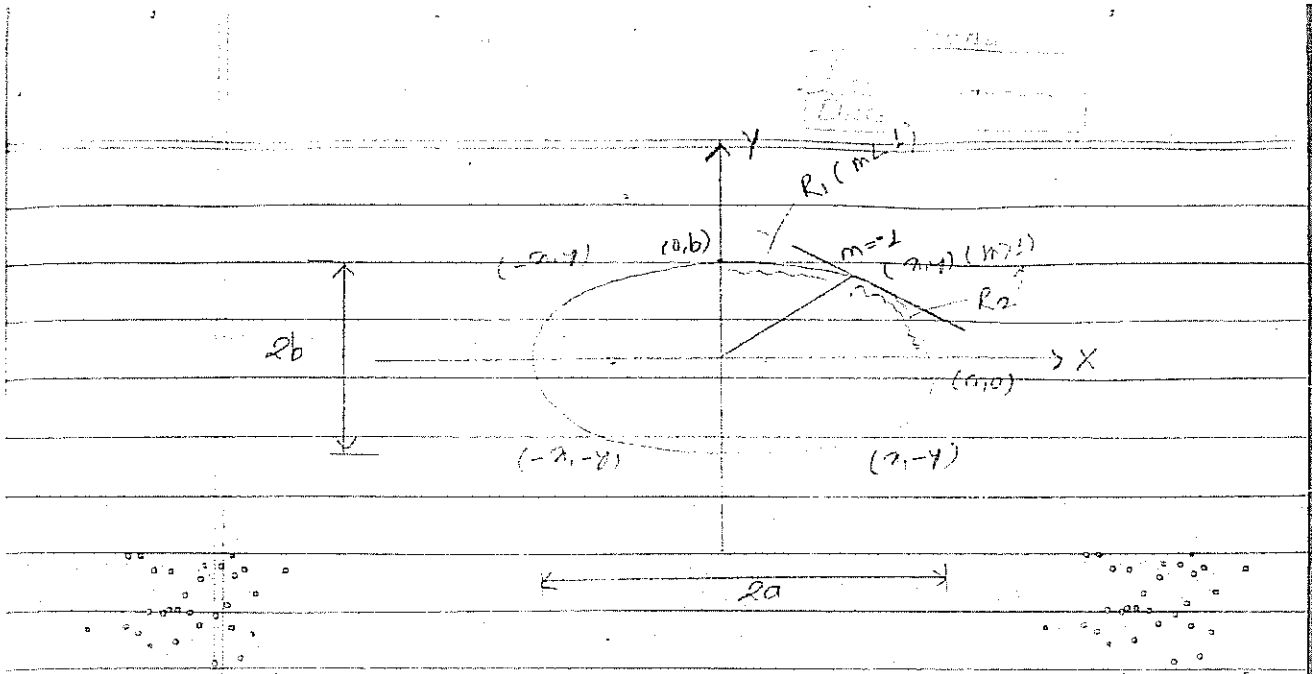
* Ellipse:

Ellipse is an elongated circle where electrical curves can be generated by modifying circle drawing procedure to take into account the different dimension of an ellipse along the major and minor axis. The ~~equatio~~

The equation of the ellipse in standard form is given by

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

This algorithm is used to display the ellipse in standard form. The algorithm is applied throughout the first quadrant according to the slop of the ellipse. For the region (R_1) where the slope of the curve is less than one, we process by taking unit step in x-direction and for the region (R_2) where the slop is greater than one we take unit steps in y direction.



Let us define an ellipse function centered at origin by $F_{ellip}(x,y) = b^2x^2 + a^2y^2 - a^2b^2$ → If $F(x,y) < 0$ then the point lies inside the ellipse.

→ If $F(x,y) = 0$ the point lies on the ellipse.

→ If $F(x,y) > 0$ the point lies outside the ellipse.

Starting at (0,b), we take unit steps in x-direction until we reach the boundary between region 1 (R_1) and region 2 (R_2). Then we switch to unit step in y-direction over the remainder of curve in first quadrant. At each step, we need to test the value of the slope of the curve. The ellipse slope is calculated from eqn:

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

Differentiating both side w.r.t. 'x'

$$\frac{2x}{a^2} + \frac{2y}{b^2}\frac{dy}{dx} = 0$$

or, $$\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$$

At the boundary region 1 and region 2, $dy/dx = -1$

$$2b^2x = 2a^2y$$

Therefore, we move out of region 1 (R_1) when

$$2b^2x \geq 2a^2y.$$

For region 1 (R_1)

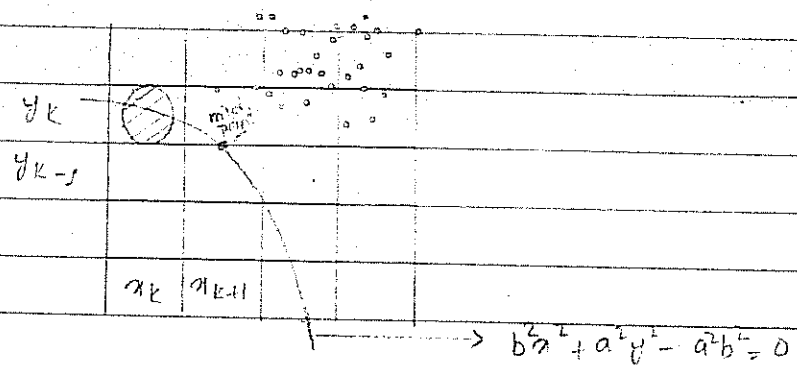


Fig: Midpoint between candidate pixel at sampling position x_{k+1} along in elliptical path.

If (x_k, y_k) is the pixel first plotted then the candidate pixel are $(x_k + 1, y_k)$ and $(x_k + 1, y_k - 1)$.

Let us define decision parameter at the mid point $y_k - \frac{1}{2}$ at sampling position x_{k+1} by

$$P_{1k} = f(x_k + 1, y_k - \tfrac{1}{2})$$
$$= b^2(x_{1k} + 1)^2 + a^2(y_k - \tfrac{1}{2})^2 - a^2b^2 \quad \text{---(i)}$$

$$P_{k+1} = b^2(x_{k+1} + 1)^2 + a^2(y_{k+1} - \tfrac{1}{2})^2 - a^2b^2 \quad \text{---(ii)}$$

Now, substracting eqn (i) from (ii);

$$P_{k+1} - P_k = b^2\{(x_{k+1} + 1)^2 - (x_k + 1)^2\} + a^2\{(y_{k+1} - \tfrac{1}{2})^2 - (y_k - \tfrac{1}{2})^2\}$$

If $P_k < 0$, midpoint lies inside the ellipse, so we plot (x_k+1, y_k) i.e. $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$

Therefore, $P_{k+1} = P_k + b^2\{(x_k+1)^2 + 2x_{k+1} + 1 - (x_k+1)^2\} + a^2\{(y_k - \frac{1}{2})^2 - (y_k - \frac{1}{2})^2\}$ $[\because x_{k+1} = x_k + 1]$

or, $P_{k+1} = P_k + 2b^2 x_{k+1} + b^2$ ——— (iii)

If $P_k \geq 0$, the midpoint lies outside the ellipse

So, we plot: (x_k+1, y_k-1) i.e. $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

$\therefore P_{k+1} = P_k + b^2\{(x_k+1)^2 + 2x_{k+1} + 1 - (x_k+1)^2\} + a^2\{(y_k - 1 - \frac{1}{2})^2 - (y_k - \frac{1}{2})^2\}$

$= P_k + b^2(2x_{k+1} + 1) + a^2\{(y_k - \frac{1}{2})^2 - 2(y_k - \frac{1}{2}) + 1 - (y_k - \frac{1}{2})^2\}$

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

$[-2a^2(y_k - \frac{1}{2}) + a^2$
$-2a^2 y_k + a^2 + a^2$
$-2a^2 y_k + 2a^2$
$-2a^2(y_k - 1)$
$-2a^2 y_{k+1}]$

Now, initial decision parameter is given by; $[\because y_{k-1} = y_{k+1}]$

$P_0 = F(x_0 + 1, y_0 - \frac{1}{2})$

$= F(0+1, b - \frac{1}{2})$

$= b^2 1^2 + a^2(b - \frac{1}{2})^2 - a^2 b^2$ $[\because$ [illegible]$]$

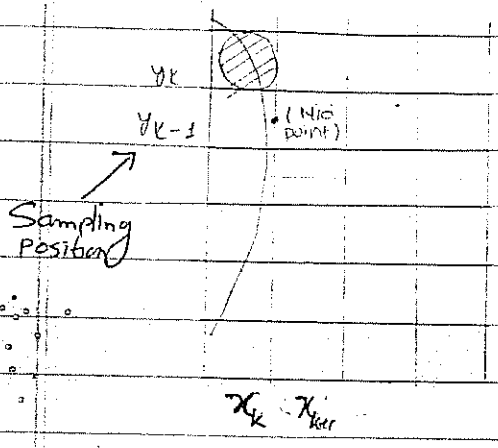
$= b^2 + a^2(b^2 - b + \frac{1}{4}) - a^2 b^2$

$= b^2 + a^2 b^2 - a^2 b + \frac{1}{4} a^2 - a^2 b^2$

$= b^2 - a^2 b + \frac{a^2}{4}$ ——— (v)

Page
Date 04 05 014

For region 2 (R_2) [$m > 1$]



$b^2x^2 + a^2y^2 - a^2b^2 = 0$

Fig: Midpoint between candidate pixel at sampling position y_{k-1} along an elliptical path

If (x_k, y_k) be the pixel just plotted then the candidate pixel are (x_k, y_{k-1}) and (x_{k+1}, y_{k-1}). So, let us define the decision parameter at the mid point $(x_k + \frac{1}{2})$ at sampling position y_{k-1} by P_{2k},

$$P_{2k} = F(x_k + \tfrac{1}{2}, y_k - 1)$$
$$= b^2(x_k + \tfrac{1}{2})^2 + a^2(y_k - 1)^2 - a^2b^2 \quad \text{---(i)}$$

Now, successive decision parameter for region (2) is:

$$P_{2k+1} = F(x_{k+1} + \tfrac{1}{2}, y_{k+1} - 1)$$
$$= b^2(x_{k+1} + \tfrac{1}{2})^2 + a^2(y_{k+1} - 1)^2 - a^2b^2 \quad \text{---(ii)}$$

Substracting (i) from (ii)

$$P_{2k+1} - P_{2k} = b^2\{(x_{k+1} + \tfrac{1}{2})^2 - (x_k + \tfrac{1}{2})^2\} + a^2\{(y_{k+1} - 1)^2 - (y_k - 1)^2\} \quad \text{---(iii)}$$

If $P_{2k} > 0$, the mid point lies outside the ellipse, so we plot $(x_k, y_k - 1)$

i.e.

$x_{k+1} = x_k$

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

$\frac{1+2}{2}$ $\frac{3}{2}$ (15)

$P_{2k+1} = P_{2k} + a^2\{(y_{k-1}+1)^2 - (y_k-1)^2\}$

$= P_{2k} + a^2\{y_k^2 - 4y_k + 4 - (y_k^2 - 2y_k + 1)\}$

$= P_{2k} + a^2\{y_k^2 - 4y_k + 4 - y_k^2 + 2y_k - 1\}$

$= P_{2k} + a^2(3 - 2y_k)$

OR,

$P_{2k+1} = P_{2k} + b^2\{(x_k + \frac{1}{2})^2 - (x_k + \frac{1}{2})^2\} + a^2\{(y_{k-1}-1)^2 - (y_k-1)^2\}$

$= P_{2k} + a^2\{(y_k-1)^2 - 2(y_k-1) + 1 - (y_k-1)^2\}$

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

$= P_{2k} - 2a^2y_{k+1} + a^2$ ——(iv)

If $P_{2k} < 0$, the mid point lies inside the ellipse so we plot (x_k+1, y_k-1). i.e.

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

Now,

$P_{2k+1} = P_{2k} + b^2\{(x_k + 1 + \frac{1}{2})^2 - (x_k + \frac{1}{2})^2\} + a^2\{(y_k - 1 - 1)^2 - (y_k-1)^2\}$

$= P_{2k} + b^2\{(x_k + \frac{1}{2})^2 + 2(x_k + \frac{1}{2}) + 1 - (x_k + \frac{1}{2})^2\} + a^2\{(y_k-1)^2 - 2(y_k-1) + 1 - (y_k-1)^2\}$

$= P_{2k} + b^2(1 + 2x_k) + 1 + a^2\{-2(y_k-1) + 1\}$

$= P_{2k} + 2b^2x_k + 2b^2 + a^2(-2y_k + 2 + 1)$

$= P_{2k} + 2b^2(x_{k+1}) - 2a^2(y_{k+1}) + a^2$ $[\because y_{k+1} = y_k - 1]$

$= P_{2k} + 2b^2(x_{k+1}) - 2a^2(y_{k+1}) + a^2$

$-2a^2y_k + 2a^2 + a^2$

$-2a^2(y_k-1) + a^2$

$-2a^2y_{k+1} + a^2$

Now, initial decision parameter is given by;

$P_{20} = f(x_{20} + \frac{1}{2}, y_{20} - 1)$

$= b^2(x_0 + \frac{1}{2})^2 + a^2(y_0 - 1)^2 - a^2b^2$ ——(vi)

NOTE: (x_0, y_0) for region 2 in the last point for region 1. The pixel for other quadrant are determined by symmetry

Date 04 05 014

Midpoint Ellipse Algorithm: (05-1)

1. Input R_x, R_y and ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin $(x_0, y_0) = (0, r_y)$.

2. Calculate the initial value of the decision parameter in region I as $P_{10} = b^2 - a^2 b + \frac{1}{4}a^2$.

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

If $P_{1k} < 0$, the next point along the ellipse centered on $(0,0)$ is (x_{k+1}, y_k) and

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

Otherwise

the next point along the ellipse circle is $(x_k + 1, y_k - 1)$ and $P_{1k+1} = P_{1k} + 2b^2 x_{k+1} - 2a^2 y_{k+1} + b^2$

and continue until $2b^2 x \geq 2a^2 y$

4. Calculate the initial value of the decision parameter in region II using the last point (x_0, y_0) calculated in region I as

$$P_{20} = b^2 (x_0 + \tfrac{1}{2})^2 + a^2 (y_0 - 1)^2 - a^2 b^2$$

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

If $P_{2K} > 0$, the next point along the ellipse centered on $(0,0)$ is $(x_K, y_K - 1)$ and

$$P_{2K+1} = P_{2K} - 2a^2 y_{K+1} + a^2$$

Otherwise

the next point is $(x_K + 1, y_K - 1)$ and

$$P_{2K+1} = P_{2K} + 2b^2 x_{K+1} - 2a^2 y_{K+1} + a^2$$

using the same incremental calculations for x and y as in region I.

6. Determine symmetric points in the other three quadrants.

7. Move each calculated pixel position (x, y) onto the elleptical path centered on (x_c, y_c) and plot the coordinate values as

$$x = x + x_c$$

$$y = y + y_c$$

8. Repeat the steps for region I until $y \geq 0$

Q) Realise an ellipse with $r_x = 8$ and $r_y = 6$ centered at $(0,0)$.

Region I

$\Rightarrow$ $(x_0, y_0) = (0, r_y)$

Now, the corresponding decision parameter at region I can be as:

K	P_{1K}	x_{K+1}, y_{K-1}	$2b^2 x_{K+1}$	$2a^2 y_{K+1}$	P_{1K+1}
	P_{2K}	x_{K+1}, y_{K+1}			P_{2K+1}

Date: 07/05/014

Polygon Filling:

Filling is the process of colouring in a fixed area or region. There are two basic approaches to area filling on raster system. One way to fill an area is to determine the overlap interval for scan lines that cross the area i.e. scan line fill algorithm.

Another method for area filling is to start from a given interior position and point outward from this point, until we encounter the specified boundary conditions.

Area or region may be defined at pixel level or geometric level. When the regions are defined at pixel level, we are having different algorithms like:

- Boundary fill
- Float fill
- Edge fill
- Fence fill

In case of geometric level we are having scan line fill algorithm.

4- connected and 8- connected pixel:

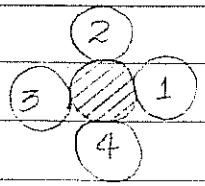


Fig:- 4- connected

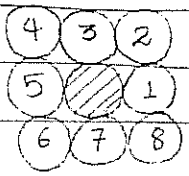


Fig:- 8-connected

These are the techniques in which pixels are considered as connected to each other. In 4- connected method, the pixels may have upto 4- neighbouring pixels as right, above, left and below of the current pixel as shown in figure above. Similarly, in 8- connected method, the pixel may have upto 8- neighbouring pixels as shown in figure.

By using any one of these techniques we can fill the interior of the polygon.

Boundary fill Algorithm:

This is a recursive algorithm that begins with a starting pixel called a seed, inside a region and continuous painting towards the boundary. The algorithm checks to see if this pixel is a boundary pixel or has already fill. If answer is no it fills the pixel and makes a recursive call to itself using each and

every neighbouring pixel as a new seed. If the answer is yes, the algorithm simply return to its caller.

A boundary fill procedure accepts as input the coordinates of an interior points (x,y), a fill color and a boundary color. starting from (x,y), the procedure test neighbouring positions to determine whether they are of boundary color, if not they are painted with the fill color and their neighbours are tested. This process continues until all pixels upto the boundary color for the area have been tested. There may be two methods for proceeding to neighbouring pixel from the seed point.

i) 4- connected
ii) 8- connected

The following procedure illustrates a recursive method for filling a 4-connected area. The boundary fill method requires the coordinate of a starting point (x,y), a fill color and boundary color as argument.

```
void boundaryFill(int x, int y, int Fill, int boundary)
{
    int Current = getpixel(x,y);
```

```
if ((current != boundary) && current != Fill))
{
    setcolor (Fill);
    setpixel (x,y);
    boundaryFill (X+1, Y, Fill, boundary);
    boundaryFill (X-1, Y, Fill, boundary);
    boundary Fill (X, Y+1, Fill, boundary);
    boundary Fill (X, Y-1, Fill, boundary);
}
}
```

Recursive boundary fill algorithm may not fill regions correctly if some interior pixels are already displayed in the fill color. This occurs because the algorithm checks next pixel both for boundary color and for fill color. Encountering a pixel with the fill color can cause a recursive branch to terminate, leaving other interior pixel unfilled. To avoid this we can first change the color of any interior pixels that are initially set to the fill color before applying the boundary fill procedure.

The boundary fill algorithm is having limitations. It fills the polygon having unique boundary color. If the polygon is having boundaries with different colors then, this algorithm fails.

Date: 09 05 014

Flood fill algorithm

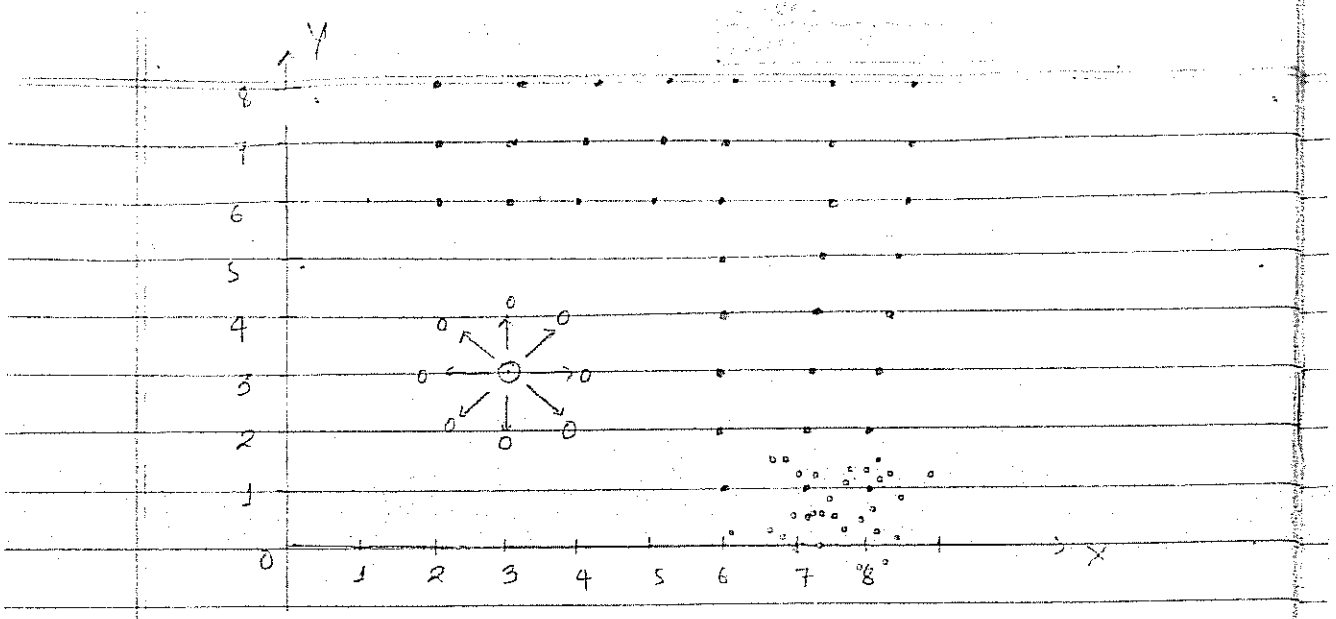
This algorithm also begins with a seed(starting pixel) inside the region. It checks to see if the pixels has the regions original color. If the answer is yes, it fills the pixel with a new color and uses each of the pixels neighbour as a new seed in a recursive call. If answer is no, it returns to the caller.

This method shares the great similarities in its operating principle with the boundary fill algorithm. It is particularly useful when the region to be filled has no uniformely coloured boundary i.e.. fill in an area that is not defined within a single color boundary. So, in this case we can point such areas by replacing a specified interior color instead of searching for a boundary color value. If the area we want to paint has more than one interior color, we can first reassign pixel values, so that all interior points have the same color. Using either a 4-connected or 8-connected approach we then step through pixel position until all interior points have been repainted.

The following procedure flood fills a 4-connected region, recursively starting from the input position.

```
void Floodfill (int x, int y, int Fill, int old color)
{
    if (getpixel (x, y) == old color)
    {
        setcolor (Fill);
        setpixel (x,y);
        Flood Fill (x+1, y, Fill, old color);
        Flood Fill (x-1, y, Fill, old color);
        Floud Fill (x, y+1, Fill, old color);
        Floud Fill (x, y-1, Fill, old color);
    }
}
```

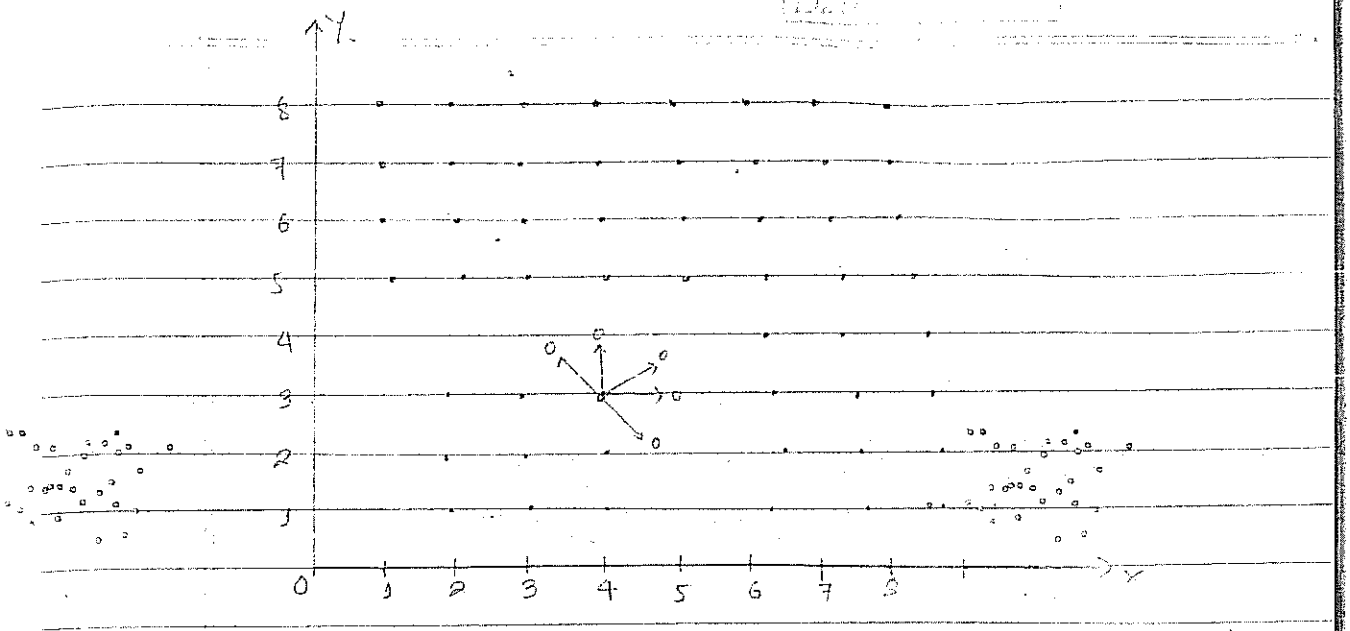
We can modify procedure FloodFill to reduce the storage requirement of the stack by filling horizontal pixels spans as boundary fill algorithm. In this approach, we stack only the begining position for those pixel spans having the value old color. The steps in this modified algorithm are similar to those for boundary fill. Starting at the first position of each span, the pixel values are replaced until a value other than old color is encountered.



How would a flood fill algorithm fill the region as shown in the figure using the 8-connected for region pixel?

⇒ Assume that a seed pixel is (3,3). The flood fill algorithm will inspect the eight points surrounding the seed i.e. (4,4), (3,4), (2,4), (2,3), (2,2), (3,2), (4,2) & (4,3). Since all the points surrounding the seed have the region's original colors each point will be filled.

Now, each of eight points become a new seed. and the points surrounding each new seed are inspected and filled. This process continues until all the points surrounding all the seeds are red of the regions



2019-05-12

Scan Line polygon Fill Algorithm

In contrast to the boundary fill and flood fill algorithm that fills regions defined at the pixel level in the image space, this algorithm handles polygonal regions that are geometrically defined by the coordinates of the vertices. Although such regions can be filled by first scan converting edges to get the boundary pixels and then applying a boundary fill algorithm to finish the job, the following is much more efficient approach that make the information regarding the edges that are available during scan conversion to facilitate the filling of interior pixel.

Figure shows the scanline procedure for said filling of polygon area. For each scan line crossing a polygon, the area filling algorithm locates the intersection points of the scanline with the polygon edges. These intersection points are then stored from

left to right and the corresponding frame buffer positions between each intersection pair that are set to the specified fill colour.

Figure shown four intersection point such as $x=10$ to $x=14$ and from $x=18$ to $x=24$.

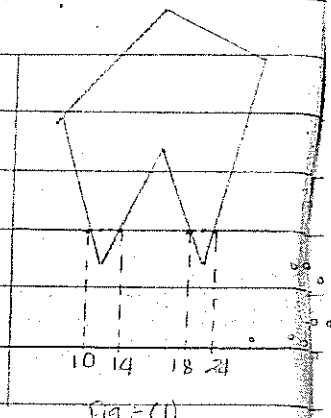


Fig - (i)

A scan line passing through a vertex intersect two polygon edges at that position.

Figure (ii) shows that two scan line at positions y and y' that intersect edge end points. Scan line y intersect five polygon edges. However, scan line y' intersects an even number (4) of edges although it also passes through a vertex. The difference between scanline y and scanline y' is for scanline y, the two intersecting edges sharing a vertex are on opposite side of the scan line. But for scanline y', the two intersecting edges are both above the scanline. In scan line y, we need to do some additional processing, the vertices that require additional processing the vertices that require additional processing are those that have connecting edges in opposite side of the scan line. If the two intersecting edges sharing a vertex are on opposite side of scan line then we store only one intersection point for that vertex otherwise for second case we store two intersection points.

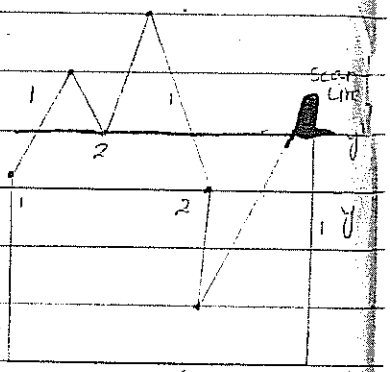


Fig (ii)

It is necessary to calculate x-intersection point for scan line with every polygon sides. We can simplify these calculations by using coherence property. A coherence property of a scene is a preoperty by which we can relate one part of a scene with the other part of a scene. Here, we can use slope of an edge as a coherence property. By using this property, we can determine the x-intersection value on the lower scan line if the intersection value for current scan line is known.

$$x_{i+1} = x_i - \frac{1}{m}$$ where, m is the slope of the ~~line~~ edge

As we can scan from top to bottom value of y-coordinate between the two scan line changes by one.

$$y_{i+1} = y_i - 1$$

Sometime it is necessary to compute the x-intersection for scan line with every polygon side. We need to consider only the polygon sides with end points crossing the current scan line.

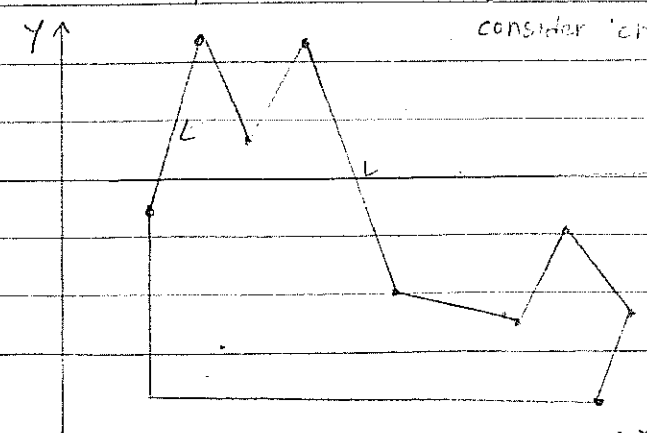
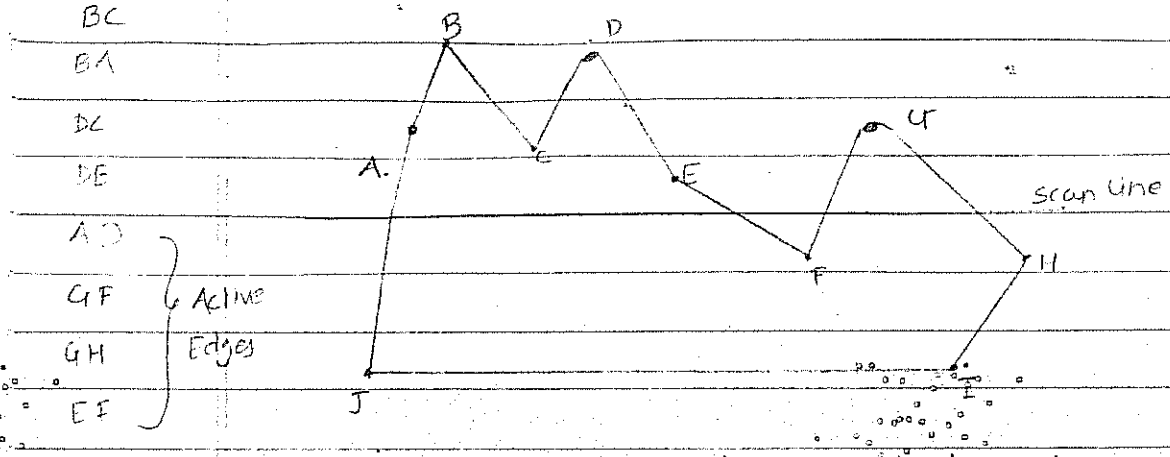


Fig: Consider only the sides which intersect the scan line



HI

JI

Sorted list of edges

It will be easier to identify which polygon sides should be tested for x-intersection, if we first sort the sides in order of their maximum y-value.

Transformation:

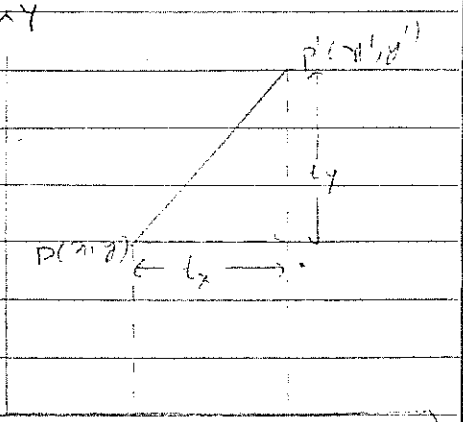
Changing coordinate description of an object is called transformation. There are three basic transformation:

1) Translation
2) Scaling
3) Rotation

1) Translation

Changing the coordinate position along a straight line is called translation.

If point $P(x,y)$ is translated to a position by $P'(x',y')$ by t_x units parallel to x-axis and t_y units parallel to Y-axis.



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

In matrix form

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

i.e. $P' = P + T$ where T is transformation matrix.

2) Scaling

The coordinate position of an object by multiplying a constant factor (scaling factor) is called scaling.

Scaling techniques:

i) About origin

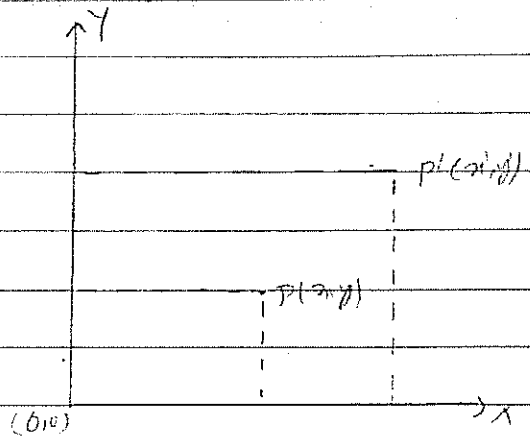
ii) About Fixed point

i) About origin

If $P(x,y)$ be scalled to a point $P'(x',y')$ coordinate by scaling factor S_x times in x units and S_y times in y units then,

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$



In matrix form;

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

i.e. $P' = S(S_x, S_y) \cdot P(x,y)$

where,

$S(S_x, S_y)$ is a scaling matrix.

If $S_x = S_y$, this type of scaling is uniform scaling.

If $S_x \neq S_y$, this type of scaling is differential "

ii) About fixed point

If $P(x,y)$ be scalled to a point $P'(x',y')$ by S_x times in x-units and S_y times in y-units about (x_F, y_F) then

$$x' - x_F = S_x (x - x_F) \quad , \quad x' = S_x \cdot x + x_F (1 - S_x)$$

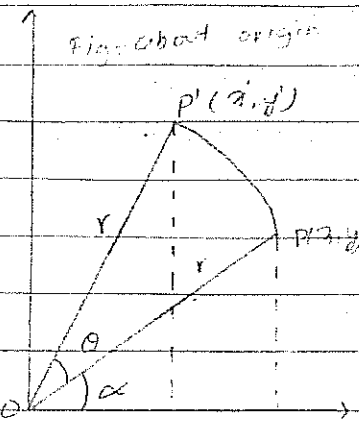
$$y' - y_F = S_y (y - y_F) \quad , \quad y' = S_y \cdot y + y_F (1 - S_y)$$

In matrix form:-

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

3) Rotation

Changing the coordinate position along a circular path is called rotation.



Types of rotation:

i) About origin

ii) About any point

i) About origin

If $p(x,y)$ is rotated to a new position $p'(x',y')$ in anticlockwise direction by an angle θ then (x',y') can be calculated as following.

Let the angle made by the line OP with x-axis is α and the radius of circular path is r, then

$x = r\cos\alpha$

$y = r\sin\alpha$

Also,

$x' = r\cos(\theta+\alpha)$

$y' = r\sin(\theta+\alpha)$

$x' = r(\cos\theta.\cos\alpha - \sin\theta.\sin\alpha)$

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

$y' = r(\sin\theta.\cos\alpha + \cos\theta.\sin\alpha)$

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

In matrix form;

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

For clockwise;

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

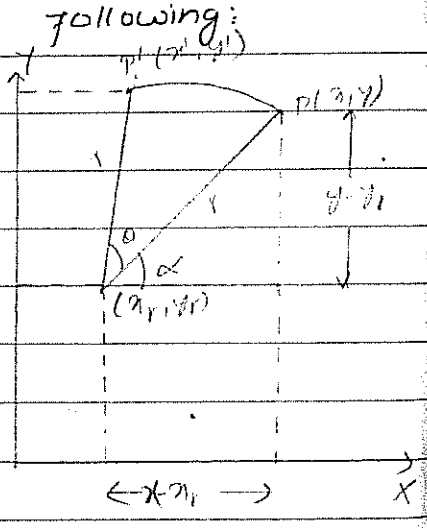
(ii) About any point

If $P(x, y)$ is rotated to a new position $P'(x', y')$ coordinate by an angle θ, then (x', y') can be calculated by the following:

Then,

$$X - X_r = r\cos\alpha$$

$$Y - Y_r = r\sin(\theta + \alpha)$$



$$X' - X_r = r\cos(\alpha + \theta)$$

$$y' - y_r = r\sin(\alpha + \theta)$$

$$x' - x_r = r(\cos\alpha\cos\theta - \sin\alpha\sin\theta)$$

$$= (x - x_r)\cos\theta - (y - y_r)\sin\theta \quad -(i)$$

$$y' - y_r = r(\sin\alpha\cos\theta + \cos\alpha\sin\theta)$$

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

Date 01-05-2154

Matrix representation and homogeneous coordinate.

To treat all the transformation in the same manner homogeneous coordinate are used for representation. Each point (x, y) is represented by triple (x, y, h) and (x', y', h') represent same point, if one is multiple of other. Let $x'/x = y'/y = h'/h$. So, $(x/h, y/h, h'/h)$ and $(x/h, y/h, 1)$ also represents the same point. For simplicity we use $h = 1$.
Now, 2×2 matrix is changed into 3×3 matrix.

Basic transformation in general form

$$P' = M_1 P + M_2 \quad \text{——} \quad (i)$$

For translation, M_1 = Identity matrix.

For rotation or scaling:
M_2 contains the translation terms associated with pivot point or scaleng fixed point.

From the above eqn (i), we produce the sequence of transformation (Scaling → rotation → translation) one step at a time. So, efficient approach will be to combine the transformation so that final coordinate position are obtain directly from the initial coordinates by eliminating the calculation of intermediate coordinate values.

For this, we need to reformulate eqn (i) to eliminate the matrix addition. We can combine

the multiplicative and translation term for 2D geometric transformation into a single matrix representation by expanding the 2x2 matrix into 3x3 matrix. Then only matrix multiplication is required to express all transformation eqn. For this we represent each cartesian coordinate (x, y) into homogeneous coordinate triple as (x, y, h) where we use $h = 1$.

def: The process of calculating the product of matrix of a number of different transformation in a sequence is known as concatenation or combination of transformation and the resultant product matrix is reffered to as composite or concatenated transformation matrix. Application: Application of composite transformation matrix on the object coordinate eliminates the calculation of intermediate coordinate values after each successive transformation.

2x2 matrix changed into 3x3 matrix

- For translation:

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

i.e. $P' = T(t_x, t_y) \cdot P$

Inverse translation

$$P' = T(-t_x, -t_y) \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

- For Rotation:

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

Inverse rotation

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

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

- For scaling:

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

Inverse scaling:

$$S^{-}(S_x, S_y) = S\left(\frac{1}{S_x}, \frac{1}{S_y}\right)$$

$$P' = S\left(\frac{1}{S_x}, \frac{1}{S_y}\right) \cdot P$$

Composite matrix:

1) Two successive translation are additive

$T_1(t_{x_1}, t_y)$ and $T_2(t_{x_2}, t_{y_2})$ are two successive translation then, composite matrix,

$$M = T_1(t_{x1}, t_{y1}) \cdot T_2(t_{x2}, t_{y2})$$

$$= \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

2) Two successive scaling are multiplicative.

If $S_1(s_{x1}, s_{y1})$ and $S_2(s_{x2}, s_{y2})$ are two successive scaling then, composite matrix,

$$M = S_1(s_{x1}, s_{y1}) \cdot S_2(s_{x2}, s_{y2})$$

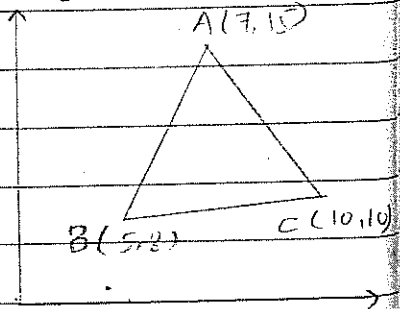
$$= \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_{x1} \cdot s_{x2} & 0 & 0 \\ 0 & s_{y1} \cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Numerical:

1) Rotate the $\triangle ABC$ by 45° clockwise about origin and scale it by (2,3) about origin.

⇒ Soln



$\Rightarrow$ Sol^n

Traingle ABC in matrix form is:

$$\begin{bmatrix} 7 & 5 & 10 \\ 15 & 8 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

Step I: Rotate by 45° (clockwise)

Step II: Scaling by (2, 3)

$\therefore$ Net transformation $= S(2,3) \cdot R(-45) \rightarrow$ clockwise

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

Now;

The transformation points are:

$$\begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 5 & 10 \\ 15 & 8 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 31.11 & 18.38 & 28.28 \\ 16.97 & 6.36 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore$ $A' = (31.11, 16.97)$

$B' = (18.38, 6.36)$

$C' = (28.28, 0)$ //

General pivot point rotation: / rotation about any orbitary point

We can rotate any selected point (x_r, y_r) by following the sequence of translate – rotate – translate operations.

Steps:

1. Translate the object so that the pivot point (x_r, y_r) coincide with coordinate origin. i.e. $T(x_r, y_r)$ to the origin.

2. Rotate the object about the origin with the given angle θ i.e. $R(\theta)$.

3. Perform the inverse translation so that the object will move to the original position i.e. $T(-x_r, -y_r)$.

Then composite matrix or Net transformation is;

$$C_m \text{ or } T = T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r)$$

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

General Fixed point scaling:

Steps:

1. Translate the object so that the fixed point coincides with the coordinate origin.

2. Scale the object with respect to coordinate origin

3. Use the inverse translation of step1 to return the object to its original position.

Then, composite matrix is;

$$Cm \text{ or } T = T(x_s, y_s) \cdot S(S_x, S_y) \cdot T(-x_s, -y_s)$$

$$= \begin{bmatrix} 1 & 0 & x_s \\ 0 & 1 & y_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_s \\ 0 & 1 & -y_s \\ 0 & 0 & 1 \end{bmatrix}$$

Numerical:

Perform a 45° rotation of $\triangle A(0,0), B(1,1)$ and $C(5,2)$ a) about the origin b) about point $P(-1,-1)$.

2014-05-22nd

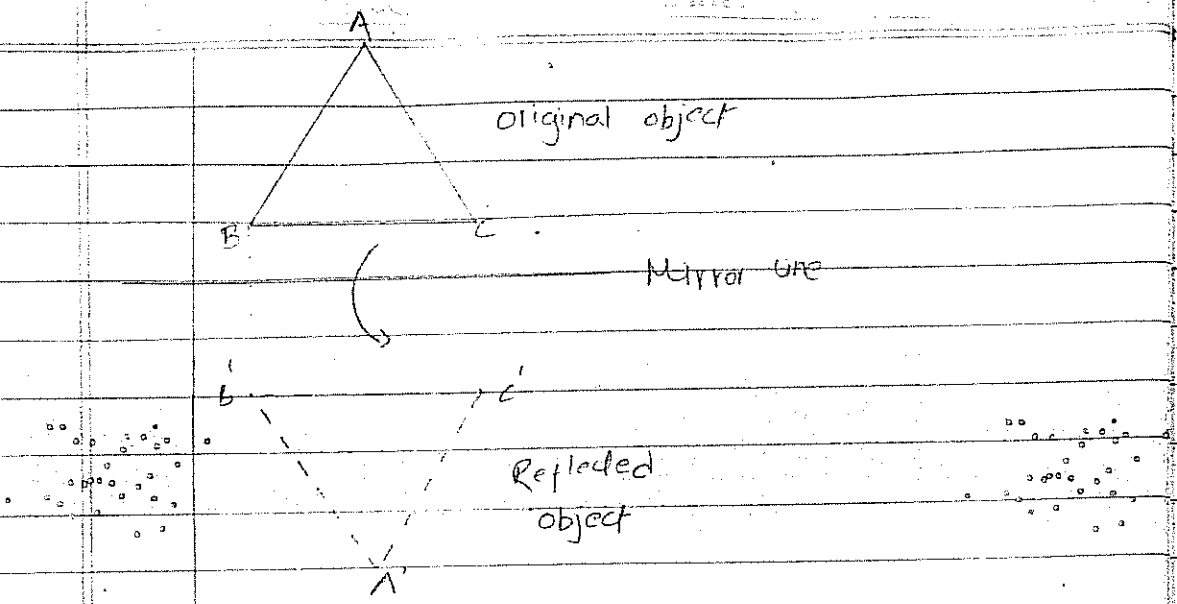
Other Transformation:

1) Reflection
2) Shearing

1) Reflection

Reflection is a transformation that produces a mirror image of an object. In 2-D reflection, we consider any line in 2-D plane as the mirror.

a) Reflection about x-axis



Basic principle behind reflection transformation are:

i) The image of an object is formed on the opposite side to ~~wear~~ where object is lying with respect to mirror line. on the opposite side.

ii) The perpendicular distance of the object from the mirror lying is same to the distance of the reflected image.

iii) For reflection about x-axis, x-coordinate is not changed and sign of y-coordinate is changed. Thus, we reflect point (x, y) in the x-axis, we get $(x, -y)$ i.e.

$$x' = x$$

$$y' = -y$$

So, the transformation matrix for reflection about

x-axis or $y=0$ axis is

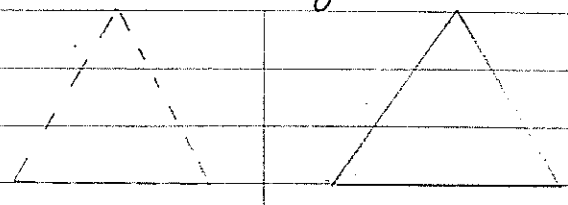
$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

and the transformation is represented as:

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

$\therefore P' = R_{fx} \cdot P$

b) Reflection about y-axis:



For reflection about y-axis, y-coordinate is not changed and sign of x-coordinate is changed. Thus, we reflect point (x, y) in the y-axis, we get $(-x, y)$ i.e.

$x' = -x$

$y' = y$.

So, the transformation matrix for reflection about y-axis or $x=0$ axis is:

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

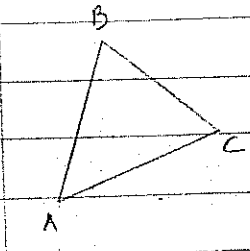
and the transformation is represented as:

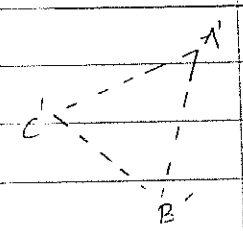
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$\therefore P' = R_{fy} \cdot P$.

c) Reflection about origin:

In this case, we actually choose the mirror line as the axis perpendicular to the xy plane and passing through the origin. After reflection both the x and y coordinate of the object is flipped i.e. $x' = -x$ & $y' = -y$. This can be represented in matrix form. Thus, the transformation matrix for reflection about origin is:





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

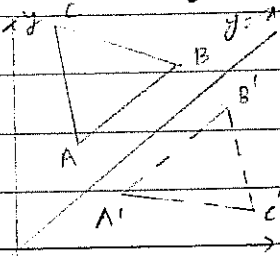
d) Reflection about the straight line $y = x$

If we reflect (x, y) point about the line $y = x$, we get (y, x) i.e.

$x' = y$

$y' = y$ x

This can be represented by:



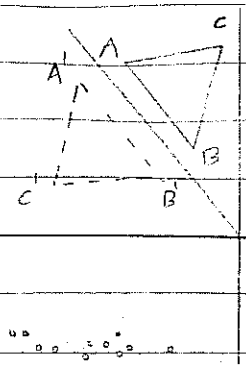
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$C \cdot M = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

e) Reflection about the straight line $y = -x$.

If we reflect point (x, y) about the line $y = -x$ we get $(-y, -x)$ i.e.

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



This can be represented in matrix form:

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

Numerical:

1. Determine transformation matrix responsible for reflection of object about the line $y = x$.

2014-05-26th

Reflection about the line parallel to x-axis:

1. Translate the line to the origin so that it will overlap to the x-axis.

2. Reflect the object about x-axis. $T(0, c)$

3. Perform inverse translation so that the line moves to its original position. i.e. $T^{-1}(0, c)$.

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

$$P' = M \cdot P = M \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

g) Reflection about the line $y=mx+c$

Steps:

1. Translate the line and the object so that the line passes through the origin.

2. Rotate the line and the object about the origin until the line coincide with one of the coordinate axis to perform reflection.

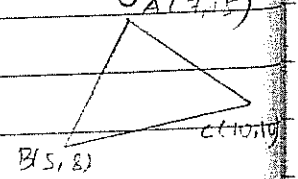
3. Reflect the object through anyone of the coordinate axis.

4. Apply the inverse rotation about the origin to shift the line at translated position.

5. Apply inverse translation to send back the object to its original position.

Numerical:

Rotate the ΔABC by 90° anticlockwise about origin and scale it by coordinate (2,2) about origin (5,8)



1 Translate 2 Rotation 3 R 4 [Scaling matrix * [ ]

⇒ Sol^n

i) T(-5,-8)

ii) R(90°)

iii) T(5,8)

iv) S(2,2)

Scaling about T⁻¹(10,10)

Solution:

Step: 1

$T(-5,-8)$

Step: 2

$R(90^\circ)$

Step: 3

$T(5,8)$

Step: 4

$T(-10,-10)$

Step: 5

$S(2,2)$

Step: 6

$T(10,10)$

Net transformation:

$T(-5,-8) \cdot R(90^\circ) \cdot T(5,8) \cdot T(-10,-10) \cdot S(2,2) \cdot T(10,10)$

$$= \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -8 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & -5 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -13 \\ 1 & 0 & -3 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

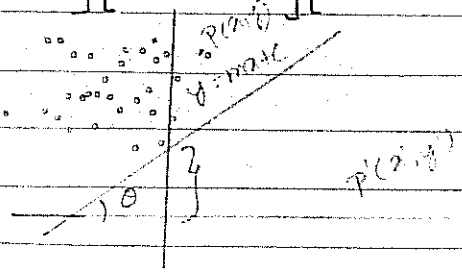
Date: 2019 05 28

Cont...

For $y = mx + c$

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

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



Numerical:

Q) Determine the 2x transformation matrix responsible for reflection about $y = x + 3$.

⇒ Here,

$m = 1$

& $\theta = 45$

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

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

Q) Reflect the $\triangle ABC$ about the line $3x - 4y + 2 = 0$. The position vector of the coordinate ABC is given as A(4,1), B(5,2), C(4,3).

⇒ Here,

$m = 3/4$

$\tan\theta = 3/4$

$\sin\theta = 3/5$

$\cos\theta = 4/5$

The intersection of the line $3x - 4y + 8 = 0$ with

$x = 0$ is $y = 2$ $(0, 2)$

$y = 0$ is $x = -\frac{8}{3}$ $(-\frac{8}{3}, 0)$

$$C_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$= 5$

Numerical:

Q) Magnify the triangle with vertices A(0,0), B(1,1) and C(5,2) to twice its size while keeping C(5,2) fixed.

⇒ Soln

The ΔABC can be represented in matrix form as:

$$\begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Now, Here, $S_x = S_y = 2$

Composite matrix is:

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

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

Now, New coordinates are:

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

$B' =$

$C' =$

$\therefore$

$A' = (-5, -2)$

$B' = (-3, 0)$

$C' = (5, 2)$

Q) Consider the square A(1,0), B(0,0), C(0,1) and D(1,1). Rotate the square ABCD by 45° about A(1,0)

⇒ Soln

Square ABCD can be represented in matrix form as:

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

We know,

Rotation about any fixed point is;

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

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

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

Now,

New coordinates are:

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

Page:
Date:

8) Reflect the square ABCD with coordinate A(2,0), B(4,0), C(2,-2), D(4,-2) at line $y=2$.

⇒ Sol^n square ABCD can be represented in matrix form as: $\begin{bmatrix} 2 & 4 & 2 & 4 \\ 0 & 0 & -2 & -2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Now,

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

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

Now,

New coordinates are:

$$A'B'C' = \begin{bmatrix} 2 & 4 & 2 & 4 \\ 0 & 0 & -2 & -2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

=

Windowing concept:

When drawing are too complex, they become difficult to read. In such situation it is useful to display only those portions of the drawing that are of intrest. This gives the effect of looking at the image through a window. Further more, it is desirable to enlarge thise portions to take full advantage of display surface. The method for selecting and enlarging the portion of a drawing is called windowing. The technique for not showing that part of the drawing which one is not intrested in is called clipping.

The viewing transformation:

Displaying an image of a picture involves mapping the coordinates of a point and lines that form the picture into the appropriate coordinates on the device or workstation where the image is to be displayed. This is done through the use of coordinate transformation known as viewing transformation. The mapping of a part of a world coordinate scene or system to device coordinate is reffered to as a viewing transformation. Sometime the 2D viewing transformation is simply called as windows-to-viewport transformation.

A world coordinate area selected for display is called a window. An area on a display device to which a window is mapped is called a viewport. The window defines what is to be viewed and the viewport defines where it is to be displayed.

2014-05-21

The mapping of a part of a world coordinate scene to device coordinate is reffered to as a viewing transformation or

The viewing transformation is a process of three steps:

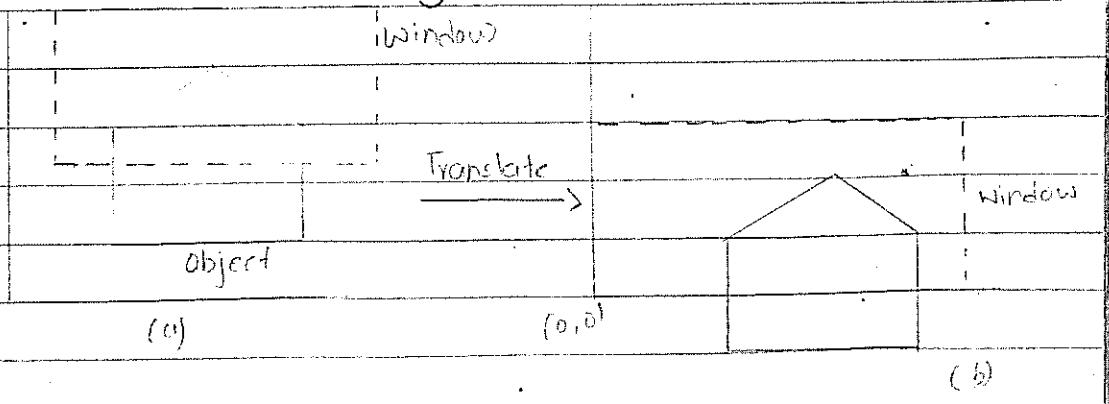
i) Translation

ii) Scaling

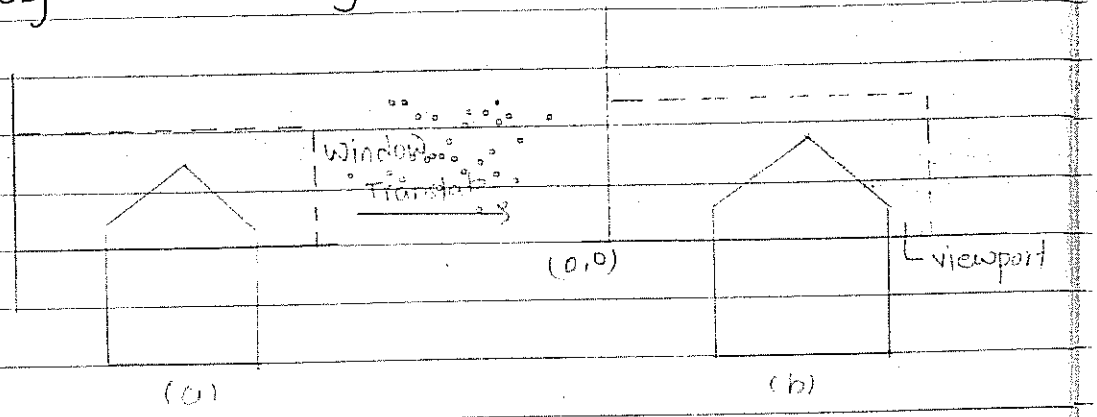
iii) The inverse translation:

Step 1:

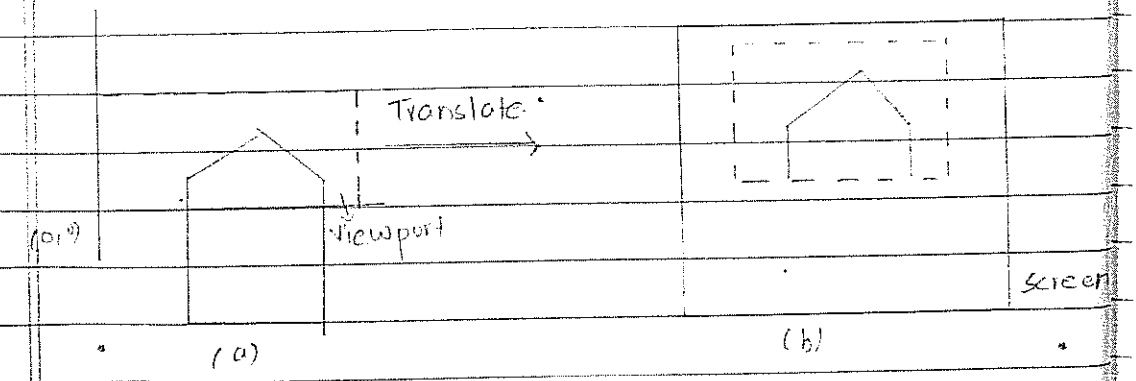
The object together with its window is translated until the lower left corner of the window is at the origin.



Step 2:
The object and the window are now scalled until the window has the dimension same as viewport. In other words we are converting the object into image and the window in viewport.



Step 3:
The final transformation state is another translation to move the viewport to its correct position on the screen.



This transformation maintains the same relative placement of objects in normalized space as they have in viewing coordinates. If a coordinate

position is at the center of the viewing window, for instance it will be displayed at the center of the viewport.

Figure below illustrates the window to viewport mapping. A point at position (x_w, y_w) in the window is mapped into the position (x_v, y_v) in the associated viewport. To maintain the same relative placement in the viewport as in the window, we require that

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$\frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - x_{wmax}}$$

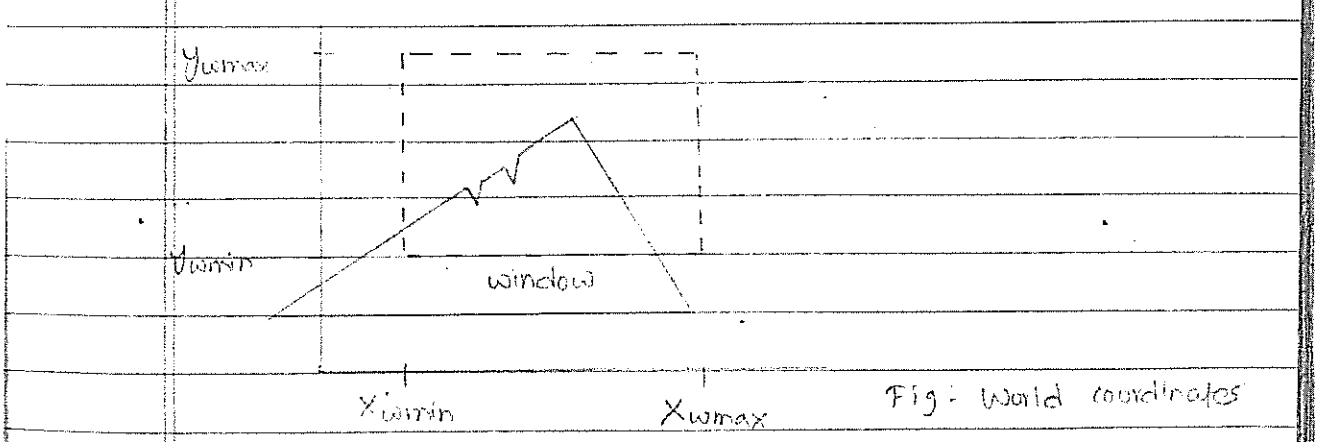


Fig: World coordinates

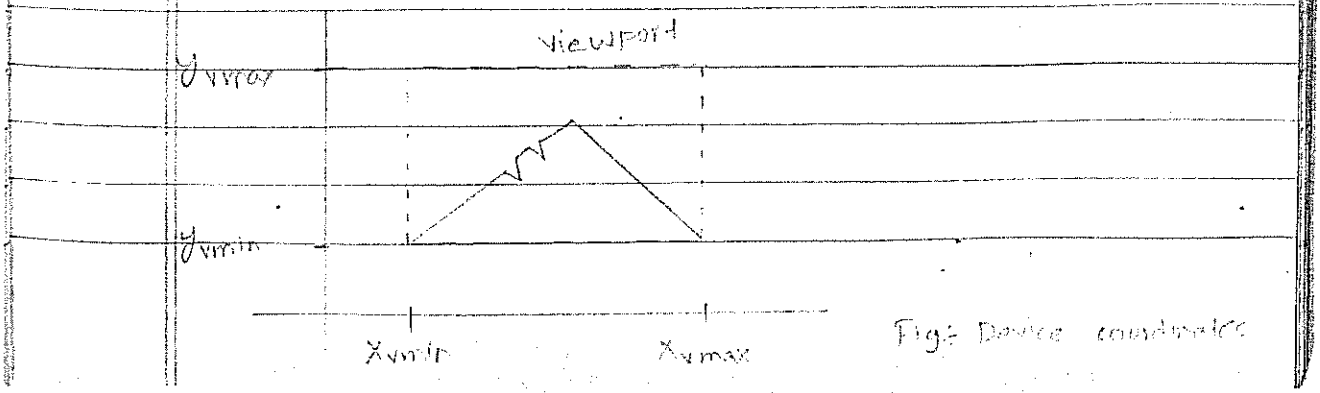


Fig: Device coordinates

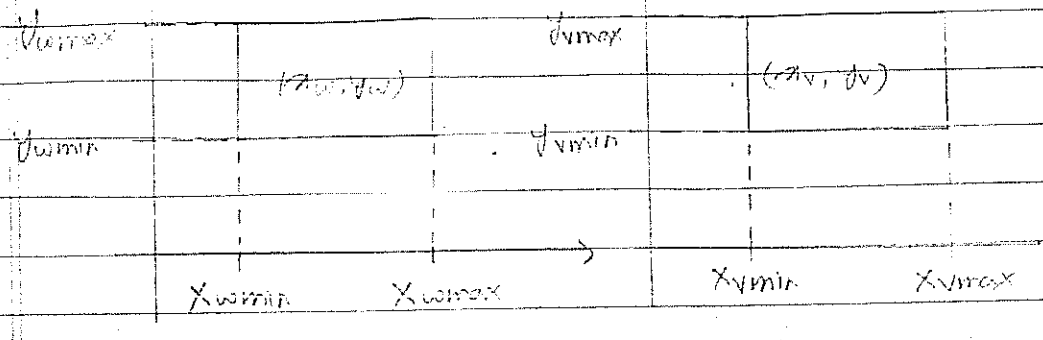


Fig: Window to viewport transformation.

Solving above expression for the viewport position (x_v, y_v), we get;

$$x_v = x_{vmin} + (x_w - x_{wmin}) \cdot S_x$$

$$y_v = y_{vmin} + (y_w - y_{wmin}) \cdot S_y$$

where;

$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

We can express these two formulas for computing (x_v, y_v) from (x_w, y_w) in terms of translate, scale - translate transformation. In a matrix form:

$$\begin{bmatrix} V_x \\ V_y \\ 1 \end{bmatrix} = N \begin{bmatrix} W_x \\ W_y \\ 1 \end{bmatrix}$$

$S_x = (X_{vmax} - X_{vmin}) / (X_{wmax} - X_{wmin})$

$S_y = (Y_{vmax} - Y_{vmin}) / (Y_{wmax} - Y_{wmin})$

where,

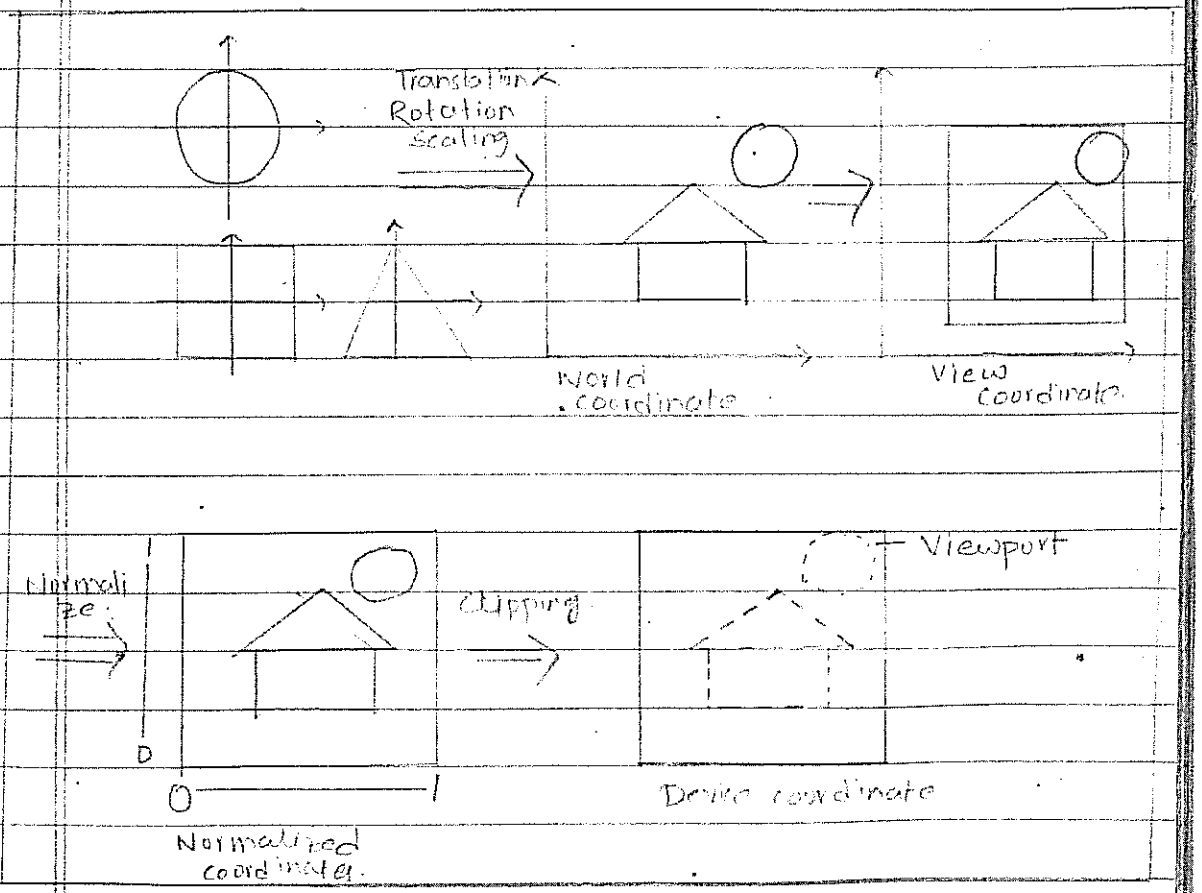
$$N = T \begin{bmatrix} 1 & 0 & x_{vmin} \\ 0 & 1 & y_{vmin} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{wm} \\ 0 & 1 & -y_{wm} \\ 0 & 0 & 1 \end{bmatrix}$$

∴ Net windows to viewport transformation is:

$N = T(x_{vmin}, y_{vmin}) \cdot S(S_x, S_y) \cdot T(-x_{wmin}, -y_{wmin})$

2-D viewing pipelining: Explain (08-M)

Some graphics package that provide window and viewport operation allow only standard rectangle, but a more general approach is to allow the rectangular window to have any orientation. In this case, we carry out the viewing transformation in several steps as indicated below:



First we construct the scene in the world coordinate (Wc) using the output primitives. Next, to obtain a particular orientation for the window, we can setup a 2-D viewing coordinate system in the world coordinate plane, and define a window in the viewing coordinate system. The viewing coordinate reference frame is used to provide a method for setting up arbitary orientations for rectangular windows. Once, the viewing reference frame is established we can transform descriptions in world coordinate to viewing coordinates.

We then define a viewport in normalized coordinates (in the range from 0 to 1) and map the viewing coordinate descriptions of the scene to normalized coordinates.

At the final step, all parts of the picture that are outside of the viewport are clipped and the contains of the viewport are transformed to the device coordinate.

	Construct		Convert		Map viewing coordi-
NC →	World-coordinate scene using Modelling coordinate Transformation	WC →	World coordinate to viewing coordinate	VC →	nate to Normalized viewing coordinate Using window-viewport specification

	Map Normalized	
NVC →	Viewport to Device coordinate	DC →

2-D viewing pipeline can be achieved by following steps:

i) Construct world coordinate scene using modelling coordinate transformation.

ii) Convert world coordinate to viewing coordinate.

iii) Transform viewing coordinate to Normalized coordinate (between 0 and 1 or between -1 & 1)

iv) Map normalized coordinate to device coordinate.

Modeling coordinate →	Construction of objects & scene	World coordinate →	definition of viewing area & orientation	Viewing coordinate →	Transformation of normalized viewing frame

Normalized coordinate →	Mapping to device dependent value	Device coordinate →

04/05/03

We need Clipping to extract part of a defined scene for viewing; identifying visible surface in 3-D views, antialiasing line segments and object boundaries, creating objects using solid modelling procedures, displaying a multiwindow environment, and drawing and painting operations that allows parts of a picture to be selected for copying, moving, erasing or duplicating

Clipping:

Any procedure that identifies those portions of the picture that are either inside or outside of a specified region of space is called clipping algorithm and the region against which an object is to be clipped is called clipwindow.

Line clipping:

Test the line segment to know the status of line with respect to clipwindow i.e.

a) Either line lies completely inside the window.

b) or it lies outside the window completely.

c) or it is a clipping candidate.

Cohen Sutherland Algorithm:

~~It is Line Clipping algorithm The algorithm divides 2D space into 9 parts using the four linear boundaries of the window.~~

The Cohen Sutherland Algorithm uses an divide and conquer strategy. It is one of the oldest and most popular line clipping algorithm. Generally the method speeds up the processing of the line segment by performing initial tests that reduce the number of intersection that must be calculated.

Every line end points in a picture is assigned a four digit binary code called a region code (out code), that identifies the location of the point

relative to the boundary of the clipping rectangle. We will use the order left, right, bottom, top (LRBT) for four bits.

Coordinate regions can be corelated with the bit position in the region code as one through 4 from right to left as:-

bit(1) = Left
bit (2) = Right
bit (3) = Bottom
bit (4) = Top

Bit values in the region code are determined by comparing end point coordinate values (x,y) to the clip boundaries.

1001	1000	1010
	Clip	
0001	Window	0010
	0000	
0101	0100	0110

TBRL

Condition:

If P(x,y) is a point then R can be calculated as:

if ($x < x_{wmin}$)
    1st bit = 1
else
    1st bit = 0

if ($x > x_{wmax}$)

2^{nd} bit = 1

else

2^{nd} bit = 0

if ($y < y_{wmin}$)

3^{rd} bit = 1

else

3^{rd} bit = 0

if ($y > y_{wmax}$)

4^{th} bit = 1

else

4^{th} bit = 0

Let, R_1 and R_2 are the two region codes then,

Case I:

If R_1 OR $R_2 == 0$ then line is visible. Both end points of the line segment lie within the window

Case II:

If R_1 AND $R_2 != 0$, then the line is invisible. When line definitely lies outside the window.

Case III:

If R_1 AND $R_2 == 0$, then line is clipping candidate OR partially visible

If a line is a clipping candidate then intersecting points are calculated by using slope intercept form of the line.

i) For left windowedge:

Intersection point will be (x_L, y) where;

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

ii) For right window edge:

Intersection point will be (x_R, y) where;

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

iii) For top window edge:

Intersection point will be (x, y_T) where;

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

iv) For bottom window edge:

Intersection point will be (x, y_B) where;

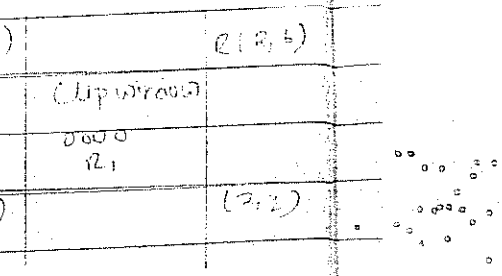
$x = x_1 + \frac{1}{m}(y_B - y_1)$

In all above equation, x_L is x-value of left edge of clipping window. x_R is x-value of right edge of clipping ($x_R = x_{wmax}$), y_B is y-value of bottom edge of clipping window ($y_B = y_{wmin}$), y_T is y-value of top edge of clipping window ($y_T = y_{wmax}$) and acceptable value of x & y are:

$x_L \leq x \leq x_R$

$y_B \leq y \leq y_T$

9) Using the Cohen Sutherland algorithm, clip the line segment A(-1,5) and B(3,8) defined by the vertices lower left hand corner 'L' (-3,1) and upper right hand corner R(2,6).



→ Here;

$x_{wmin} = -3$

$x_{wmax} = 2$

$y_{wmin} = 1$

$y_{wmax} = 6$

Given;

$x_1 = -1 \quad x_2 = 3$

$y_1 = 5 \quad y_2 = 8$

$$\text{Slope } (m) = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8-5}{3+1} = \frac{3}{4}$$

Now, region code of A(-1,5) is assigned as:

if $(x_1 < x_{wmin})$; $-1 < -3$; False ; 0 (1st bit)

if $(x_1 > x_{wmax})$; $-1 > 2$; False ; 0 (2nd bit)

if $(y_1 < y_{wmin})$; $5 < 1$; False ; 0 (3rd bit)

if $(y_1 > y_{wmax})$; $5 > 6$; False ; 0 (4th bit)

4th bit 3rd bit 2nd bit 1st bit

Region code = 0 0 0 0 = R_1

4th	3rd	2nd	1st
1	0	1	0
T	B	R	L

Now, region code of $P(3,8)$ is assigned as:

Here,

$x_2 = 3$ & $y_2 = 8$

1st bit	$x_2 < x_{wmin}$	$3 < -3$	False	0 (1st bit)
2nd bit	$x_2 > x_{wmax}$	$3 > 2$	True	1 (2nd bit)
3rd bit	$y_2 < y_{wmin}$	$8 < 1$	False	0 (3rd bit)
4th bit	$y_2 > y_{wmax}$	$8 > 6$	True	1 (4th bit)

Region code $(R_2) = 1010$

1) ORing R_1 & R_2	2) ANDing R_1 & R_2	3) AND
0000	0000	$R_1 \&\& R_2 == 0$
1010	1010	So, the line
1010	0000	is clipping
False	False	candidate.

For intersection point, we use top point;

$$X = x_1 + \frac{1}{m}(y_t - y_1)$$

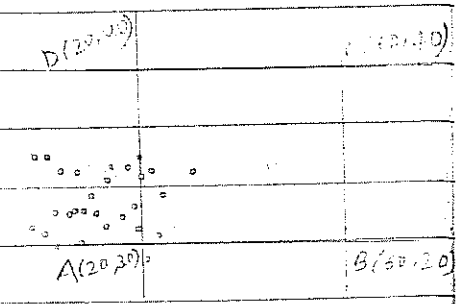
$$= -1 + \frac{1}{m}(6-5)$$

$$= -1 + \frac{1}{3/4}(1)$$

$$= \frac{1}{3}$$

∴ Line segment $AB'(\frac{1}{3}, 6)$ is saved for display.

Q) Given a clipping window A(20,20), B(60,20), C(60,40), D(20,40). Using Sutherland point algorithm find the visible portion of the line segment joining the points P(40,80), Q(120,30).



⇒ Sol^n

Given;

$x_L = x_{wmin} = 20$

$x_R = x_{wmax} = 60$

$y_B = y_{wmin} = 20$

$y_T = y_{wmax} = 40$

$x_1 = 40 \quad x_2 = 120$

$y_1 = 80 \quad y_2 = 30$

$$\text{Slope } (m) = \frac{y_2 - y_1}{x_2 - x_1} = \frac{30-80}{120-40} = \frac{-50}{80} = \frac{-5}{8}$$

Now, region code of P(40,80) is assigned as:

if ($x_1 < x_{wmin}$) ; $40 < 20$; False ; 0 ; 1st bit

if ($x_1 > x_{wmax}$) ; $40 > 60$; False ; 0 ; 2nd bit

if ($y_1 < y_{wmin}$) ; $80 < 20$; False ; 0 ; 3rd bit

if ($y_1 > y_{wmax}$) ; $80 > 40$; True ; 1 ; 4th bit

∴ Region code for $R_1 = 1000$

Now, region code for $Q(120,30)$ is;

if $(x_2 < x_{wmin})$; $120 < 20$; False ; 0 ; 1st bit

if $(x_2 > x_{wmax})$; $120 > 60$; True ; 1 ; 2nd bit

if $(y_2 < y_{wmin})$; $30 < 20$; False ; 0 ; 3rd bit

if $(y_2 > y_{wmax})$; $30 > 40$; False ; 0 ; 4th bit

Region code of $R_2 = 0010$

i) Oring R_1 & R_2

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

ii) Anding R_1 & R_2

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

Both endpoints codes are not zero and since, $R_1 \& \& R_2 = 0$, so the line is clipping candidate and hence line cannot be rejected as invisible

Now, the intersecting points are calculated by using slope intercept form of the line:

i) For left window edge

$$Y = m(x_L - x_1) + y_1$$

$$= -\frac{5}{8}[(20-40)] + 80$$

$= 92.5$ which is greater than y_T and hence rejected

ii) For right window edge:

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

$$= \frac{-5}{8}(60-40)+80$$

$$= 67.75$$ // which is greater than y_T so, rejected

iii) For top window edge:

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

$$= 40 + \left(-\frac{8}{5}\right) \times (20-80)$$

$$= 136$$ // which is greater than x_R and hence rejected

iv) For bottom window edge:

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

$$= 40 + \left(-\frac{8}{5}\right) \times (40-80)$$

$$= 104$$ which is greater than x_R and hence rejected

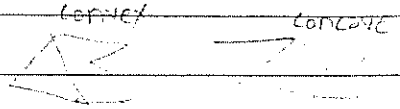
Since both value of y are greater than y_T and both values of x are greater than x_R. Therefore ~~Here we can see the intersecting points~~ Line segments PQ are completely outside the window. //

Polygon clipping:

Line clipping algorithm cannot be used on a polygon because we must generate new edges along the window boundaries as well as clip the original edges.

Types of polygon:

A polygon is said to be convex if the line joining any two interior points of the polygon lies completely inside the polygon. Otherwise, a non-convex polygon is said to be concave.



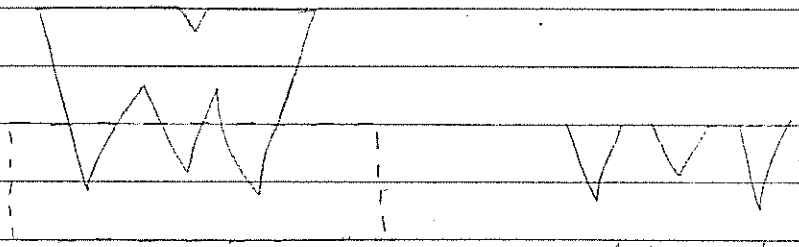
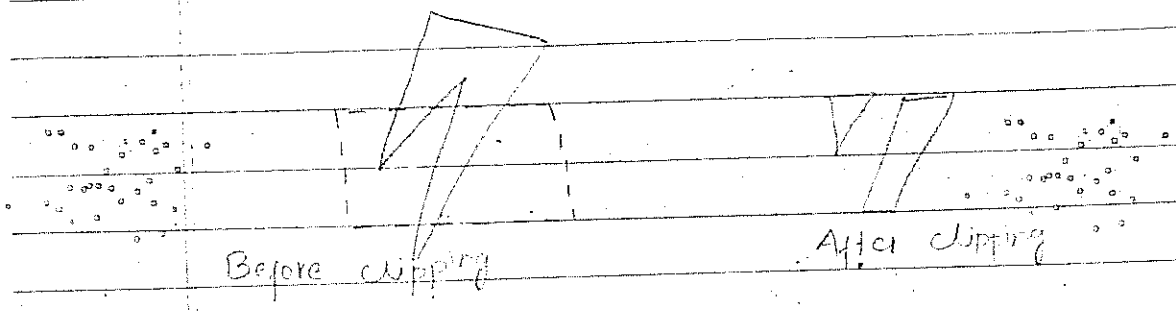


Fig: Before clipping

Fig: After clipping

For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. Thus, the output of the polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries. In polygon clipping, each edges of the polygon must be tested against each edges of the clipped rect-

angle, new edges must be added and existing edges must be discarded, retained or divided. Multiple polygons may result from clipping a single polygon.



The Sutherland Hodgman algorithm:

Sutherland Hodgman polygon clipping algorithm uses a divide and conquer strategy. It serves a series of simple and identical problems that when combined solve the overall problem. The difference between this strategy for a polygon and a line clipping are: the polygon clipper clips against four edges in succession. whereas, the line clipper tests the outcode to see which edge is crossed and clips only when necessary. In Sutherland Hodgman algorithm a polygon consists of an ordered sequence of vertices. Let, $P_1, P_2, \ldots, P_n$ be the vertex list of the polygon to be clipped. Let, edge 'e' be anyone of the edges of a convex clipping polygon which will be processed by the clipping algorithm in the order of the vertex pair.

The clipping algorithm will be called once for each vertex of the polygon. For each call, the algorithm will return either no vertex at all, the original vertex without change or one or more vertices.

When we are clipping a polygon with respect to any particular edge of the window, at that time we have to consider following four different cases:

Case I:

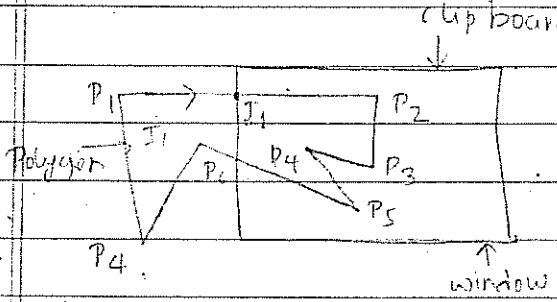


Fig: store $I_1 P_2$ in output vertex list

If the 1st vertex is outside the window boundary and the 2nd vertex is inside the window boundary, then the intersection point of polygon with boundary edge of window & the vertex which is inside the window is stored in a output vertex list i.e. $P_1 P_2$ edges.

Case II:

If both i.e. 1st & 2nd vertex are inside the window boundary then we have to store the 2nd vertex only in output vertex list i.e. $P_2 P_3$ edges.

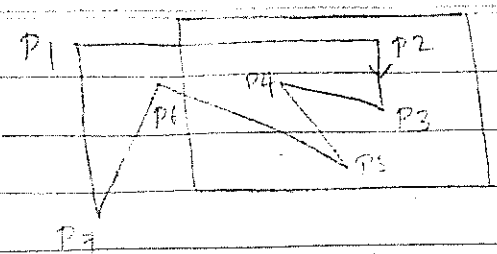


Fig: Store only P3 in output vertex

Case III:

If the 1st vertex is inside the window & the 2nd vertex is outside the window boundary then we have to store intersection point in output vertex list.

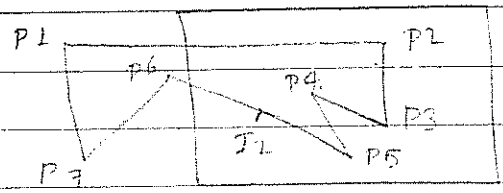


Fig: Store only I1 in output vertex list.

Case IV:

If both 1st and 2nd vertex of a polygon are lying outside the window boundary then no vertex is stored in output vertex list i.e. P6P7.

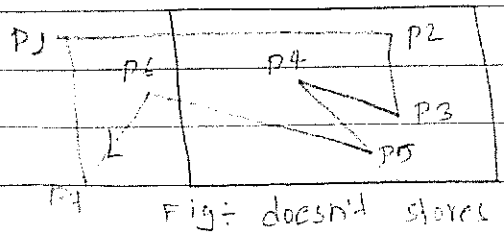
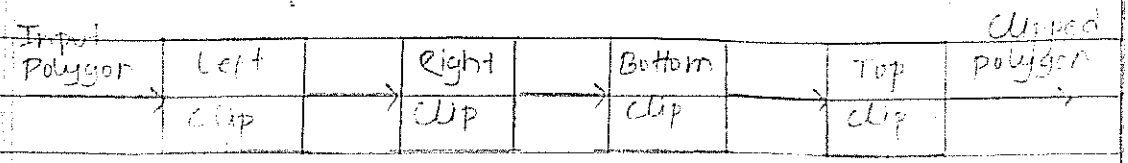
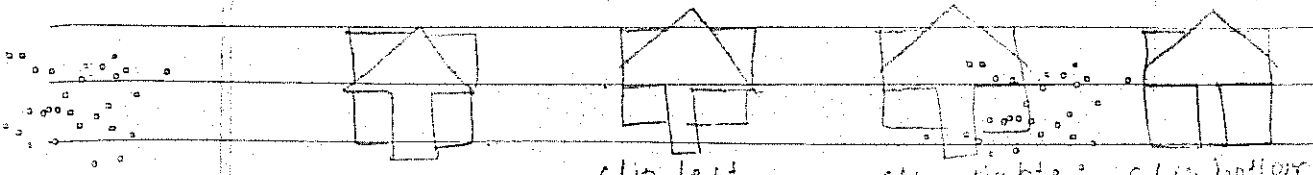


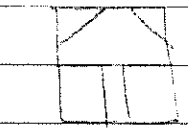
Fig: doesn't stores anything

Once all vertices have been considered for one clip window boundary, the output list of vertices is clipped against the next window boundary. The block diagram for this algorithm is as follows:

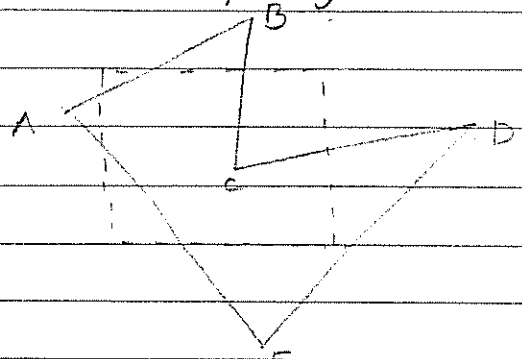


For example:





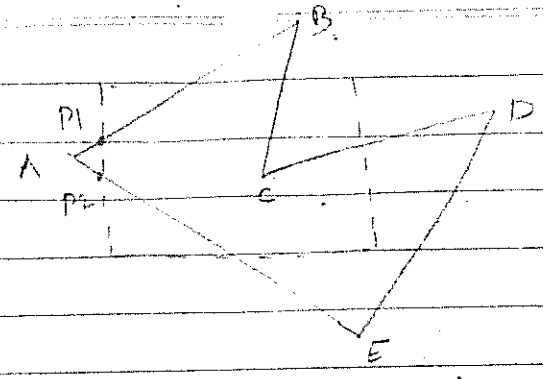
Let, us take an example to understand the polygon clipping by Sutherland Hodgman algorithm. Suppose, we have a polygon ABCDE which we want to clip against the rectangular window.



Here, vertex list will be A, B, C, D, E and edges will be AB, BC, CD, DE, EA.

Step I: Clip left

We will consider all edges of polygon with respect to left boundary of window. Diagrammatic representation of left clip polygon is:



Left clip:

AB → P1B

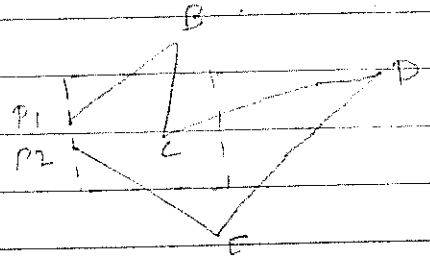
BC → C

CD → D

DE → E

EA → P2

So, after left clipping our output vertex list will become P1, B, C, D, E, P2.



Step II: Clip Right

Now, modified list of vertices is passed to clip right procedure.

Right clip:

P1B → B

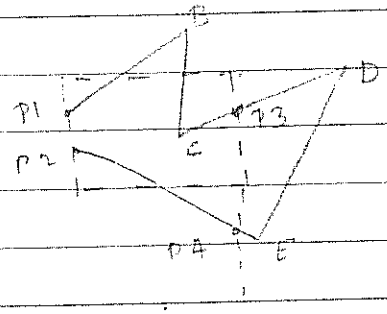
BC → C

CD → P3

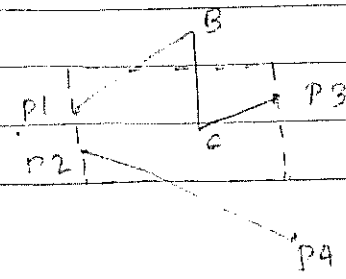
DE → No vertex

EP2 → P4P2

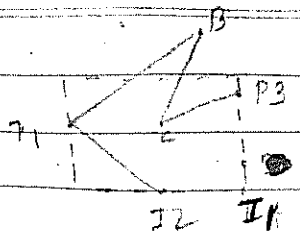
P2P1 → P1



So, after right clipping our output vertex list will become B, C, P3, P4, P2, P1



Step III : Clip Bottom



Bottom clip:

$P_2 P_1 \rightarrow P_1$

$P_1 B \rightarrow B$

$BC \rightarrow C$

$CP_3 \rightarrow P_3$

$P_3 P_4 \rightarrow I_1$

$P_4 P_2 \rightarrow I_2 P_2$

After bottom clipping: the output vertex list are: $P_1, B, C, P_3, I_1, I_2, I_2$.

Step IV : Clip Top

The modified list of vertices is passed to the top clip procedure.

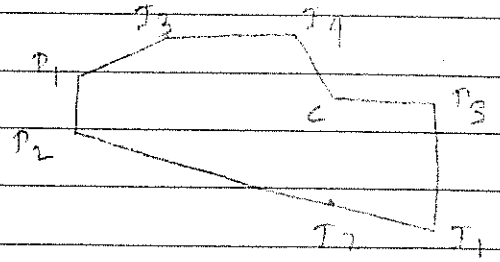
Top clip:

$P_1 B \rightarrow I_3$ $\quad P_3 I_1 \rightarrow I_1$ $\quad P_2 P_1 \rightarrow P_1$

$BC \rightarrow I_4$ $\quad I_1 I_2 \rightarrow I_2$

$CP_3 \rightarrow P_3$ $\quad I_2 P_2 \rightarrow P_2$

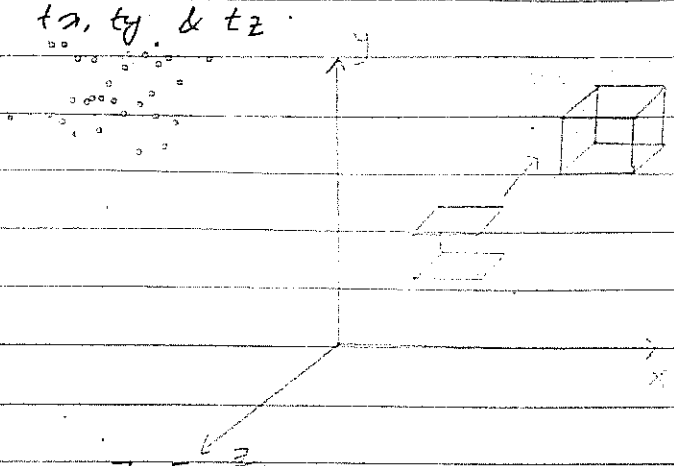
The final clipping becomes:



5.1 3D Co-ordinate System and 3D Transformation

1. Translation

In a three-dimensional homogeneous coordinate representation, a point is translated from position $P(x, y, z)$ to position $P'(x', y', z')$. It requires translation distances t_x, t_y & t_z.



Equations:

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

In matrix form;

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

$\therefore P' = T \cdot P$

2. Scaling

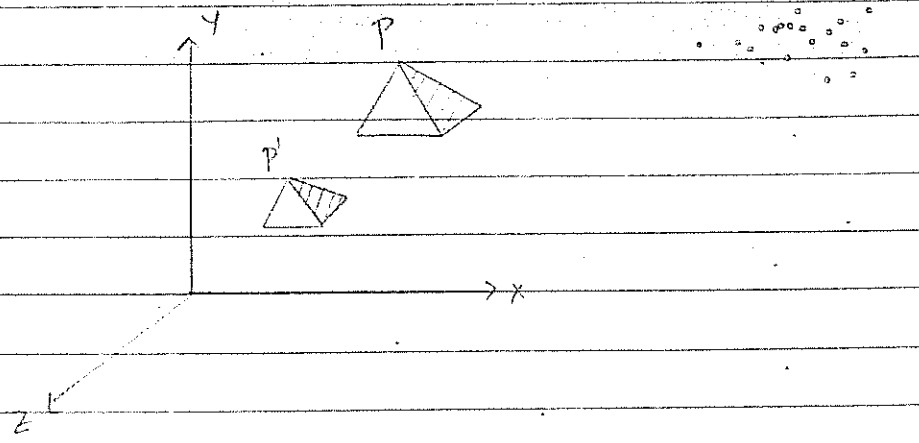
→ Scaling changes the size of an object and repositi-on of object relative to coordinate origin.

→ We can preserve the original shape of object with uniform scaling ($S_x = S_y = S_z$)

Matrix representation of scaling transformation

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

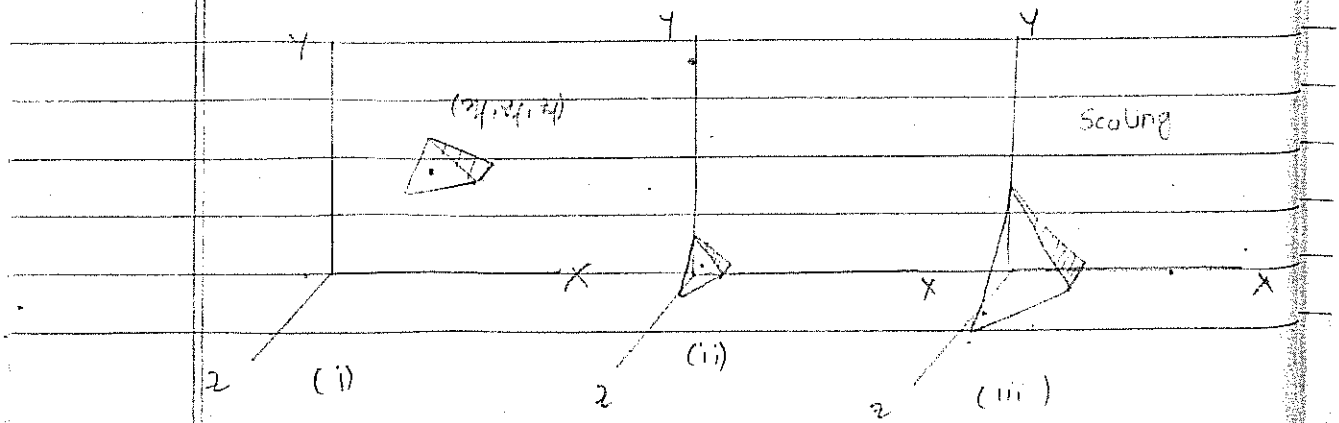
$$P' = S \cdot P$$

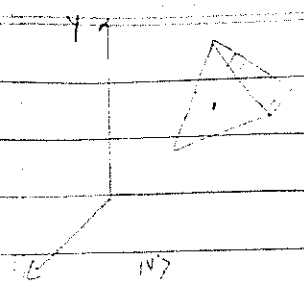


Scaling about any fixed point

Scaling with respect to selected fixed position (x_f, y_f, z_f) can be represented with

1) Translate fixed point to origin.
2) Scale the object relative to coordinate origin.
3) Translate fixed point back to original position.





Composite matrix is:

$$C_m = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & 0 & (1-S_x) \\ 0 & S_y & 0 & (1-S_y) \\ 0 & 0 & S_z & (1-S_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3 Rotation

To generate a rotation transformation for an object, we must designate an axis of rotation (about which the object is to be rotated) and the amount of angular rotation.

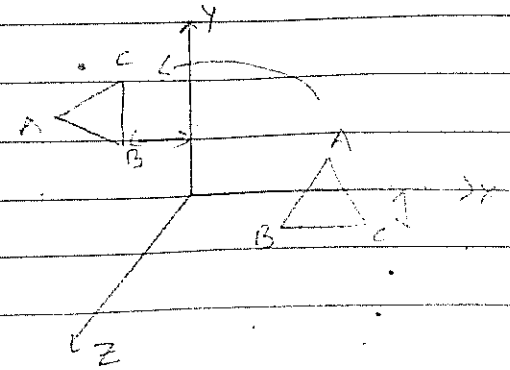
1) 3D rotation about z-axis:

Equations:

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

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

$z' = z$



In matrix form:

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

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

→ Here, z component doesnot change.

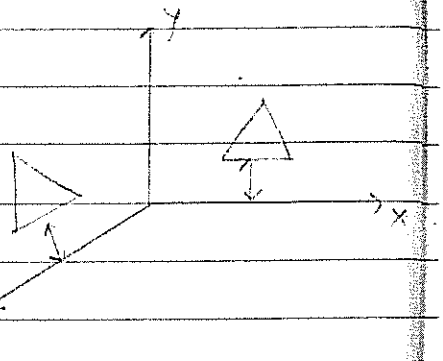
ii) Rotation about y-axis:

Equations:

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

$y' = y$

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



In matrix form:

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

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

→ Here, y component doesnot change.

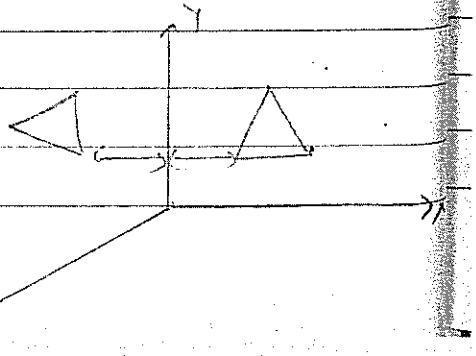
iii) Rotation about x-axis:

Equations:

$x' = x$

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

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



In matrix form:

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

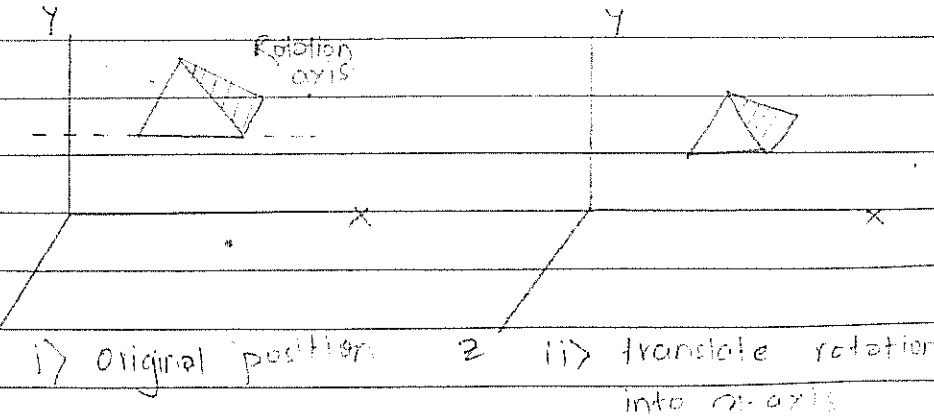
iv) Rotation about an axis parallel to one of the coordinate axis:

For this we need to perform some series of transformation:

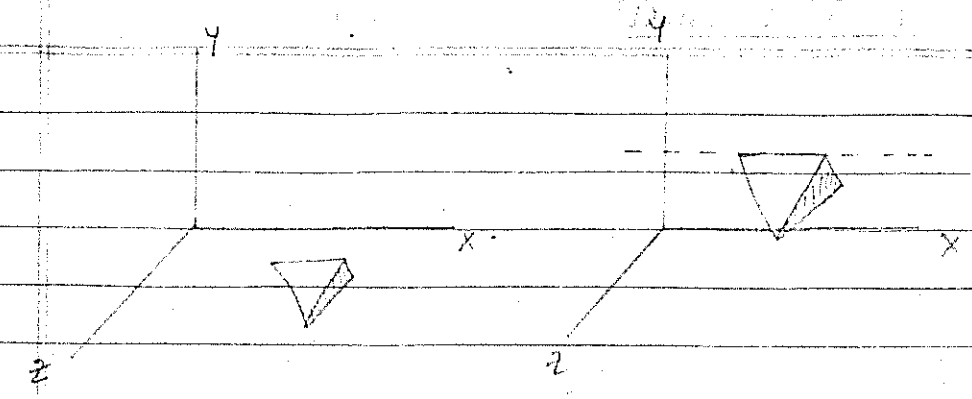
i) Translate object to perform some series of transformation with parallel coordinate axis $T(-t_x, -t_y, -t_z)$.

ii) Perform specified rotation about that axis. $R_x(\theta)$.

iii) Translate object back to its original position. $T(t_x, t_y, t_z)$.



i) Original position ii) translate rotation axis into z-axis



iii) Rotate object through an angle θ

iv) Translate rotation axis to original axis

Composite matrix is;

$$C_m = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$C_m = T \cdot R_x(\theta) \cdot T^{-1}$

$\therefore P' = C_m \cdot P$

v) Rotation about an arbitary axis:

We need some series of transformation.

i) Translate the object such that rotation axis passes through co-ordinate origin.

ii) Perform rotation to make axis of rotation coincide with one of the coordinate axis (for eg:- coincide with z-axis).

iii) Rotate about z-axis by angle θ.

iv) Perform inverse rotation to bring rotation axis back to its original orientation.

v) Perform inverse translation to bring rotation axis back to its original position.

$$C_m = T(-t_x, -t_y, -t_z) \cdot R_x^{-1} \cdot R_y^{-1} \cdot R_z R_y R_x \cdot T(-t_x, -t_y, -t_z)$$

where,

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ is translation matrix.

$$R_{x\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & -\cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{y\beta} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{z\gamma} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, $R_x(\alpha)$, $R_y(\beta)$ & $R_z(\gamma)$ are 3D-coordinate axis rotation matrix.