* Definition Of Computer Graphics :->

Computer graphics deals with:

- Geometric modeling: creating mathematical models of 2D & 3D objects.
- Rendering: producing images given these models.
- Animation: defining/representing the time dependent behavior of objects.

* History Of Computer Graphics :->

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

- Crude plotting on hardcopy devices such as teletypes and line printers dates from early days of computing.
- The whirlwind computer developed in 1950 at MIT and computer-drive CRT displays for o/p, both for operator use and for cameras producing hard copy.
- The SAGE air-defence system developed in the middle 1950's was ~~the~~ the first to use ~~computer~~ command & control CRT displays console on which operators identified targets with the light pens.
- The begining of modern interactive graphics however were found in IVAN SUTHERLAND's seminal doctoral work on the sketchpad drawing systems which uses keyboard and light p[illegible]
- By the mid-sixties, a no. of research projects all[illegible] commercial products had appeared as the be[illegible]g of of CAD activities in computer, automobile & aer[illegible] d other

- Due to high cost of graphics hardware, expensive computing resources, difficulty in writing large interactive program and due to many reasons the human computer interaction was still done primarily in both mode using punched cards. After the advent of graphics based personal computers such as Apple Macintosh and IBM PC, the costs of both hardware and software drown down. Thus, the interactive graphics (GUI) as "the window on the computer" became an integral part of PC featuring graphical interaction.
- Another major hardware advence of late sixties was attaching a display is a minicomputer, relieving heavy demands of refreshed display devices.
- In 1968, another such devices was invented. The refresh display hardware for geometric transformations could scale, rotate and translate points and lines on the screen at real time; perform the 2D & 3D clipping and could produce parallel and perspective projections. The development of inexpensive raster graphics based on television in early seventies contributed more to the growth of the field.

The development of graphics can not be standoff without the study of graphics input technology. Now computer graphics have become integral and infamous technology in the computer system.

* Application of computer Graphics :→

Computer graphics started with the display of data or hard copy plotter and CRT screens had grown include the creation, storage and manipulation of mode is of images of objects. These models come from a diverse set of fields and include physical mathematical, engineering, architectural, natural phenomena and so on. There is virtually no area in which graphical displays can not be used to some advantages.

⟨1⟩ Computer-Aided Design for Engineering and architectural system. (CAD) :→

In CAD, interactive graphics is used to design components and systems of mechanical, electrical electromechanical and electrical devices including structures such as buildings, automobiles bodies, airplane, VLSI chips, optical systems and telephone and computer networks. The emphasis is on interacting with a computer based model of component or system being designed in order to test, for eg its structural, electrical or thermal properties. This model is interpreted by a simulator that feeds back the behavior of the system to use for further interactive design and test cycles.

Architects are interactive graphs method to layout floor plans that shows positioning of rooms, doors, windows, stairs, shelves and other

try out arrangement for wiring, electrical outlets and other system to determine space utilization on a building. The realistic displays then allows architects and their clients to study appearance of building and even go for a simulated "walk" through rooms or around building.

2) Presentation Graphics :-

Another major application areas of computer graphics is the "Presentation Graphics". Presentation Graphics are used to provide illustrations for reports or to generate transparencies for use with graphics.

Presentation Graphics is commonly used to summarize financial, statical, mathematical, scientific and economic data for research reports, mangerial reports and other types of report.

The 3D graphics are usually used simply for effects they can provide a more digramatic or more attractive presentation of data relationship

3) Computer Art :-

Computer graphics is used to generate arts. They are widely used in both fine art and commercial art applications. Artist use a variety of computer method including special purpose hardware, artist paints brush program, other paint package, etc.

fine artists uses a variety of computer techniques to produce images. These images are created using a combination of 3D modeling package, CAD softwares. These technique for generating electronic images are also applied in commercial art for logos and other design, page layout, etc.

A common graphics method exployed in many commercial is morphing, where one object is transformed into another.

<4> Education and training :

computer graphics is used in education and training for making it more effective and more illustrative. Eg. if a teacher is to teach bonding of molecules or electrons jump from one state (higher state) to lower energy state or the structure of gene. Then he can demonstrate these concepts using computer graphics software or presentations.

<5> Entertainment :

Computer graphics methods are new commonly used in making motion picture, music videos and TV shows. Images are drawn in wire-frame and will be shaded with rendering methods to produce solid surfaces. computer graphics are also used to introduce virtual characters to movies like characters in "Lord of the Ring".

<6> Visualization :-

Some method generates very large amount of data/information for eg a survey of one lakh people

choice for using different toothpaste generates large amount of data. Analysing the property of whole amount of data is very difficult. Therefore to visualize large amount of information graphical computer systems are used.

(7) Image Processing:
Image can be created using simple point program or can be fed into computer by scaning the image. These pictures need to be changed to improve the quality,

(8) Graphical User Interface
Multiple window, icons, menus allows a computer setup to be utilized more efficiently. GUI use graphical objects called gizmos to represent certain objects or process involved in human computer communication for virtual purpose

CHAPTER#2 GRAPHICS HARDWARE

↳ Input devices are used to feed data or info. into a computer system. They are usually used to provide input to the computer upon which reaction, outputs are generated. Data input devices like keyboards are used to provide additional data to computers whereas pointing and selection devices like mouse, light pens, touch panels are used to provide visual and indication- input to the application.

*Keyboard ⇒
Keyboard is a primary serial input device for each keypress, a keyboard sense a specific ASCII code to the computers. Keyboards are also capable of sending/coding combinations of key press & special keys like function keys. The coding for the combination of key pressing is called "chording" Cursor controlled keys and function keys are used for doing operations in a single keystroke.

Several different technologies are used to detect a key depression including mechanical contact closure, change in capacitance and magnetic coupling. Chroding is not possible with standard coded keyboard which returns only an ASCII code per keystrokes and returns nothing if two keys are pressed simultaneously. An uncoded keyboard can identify of all key press simultaneously allowing chroding.

* Mouse (Mechanical and Optical) :-

Mouse is a serial input device used to select object movement in graphics system. It converts horizontal movement into vertical movement. Primary principle of cursor movement is to detect the amount of displacement of mouse in horizontal plane. This displacement is maped into screen. Generally the mapping ratio is 1:8.

According to their working mechanism two type of mouse are there. They are:

- Mechanical mouse &
- Optical mouse

Mechanical mouse contains two rollers. One roller count horizontal displacement and other roller counts vertical displacement. The motion of the roller in the base of mechanical mouse is converted into digital values that are used to determine the direction and magnetic of movement.

Optical mouse uses a special mouse pad with a matrix of Black and white grid of lines. The matrix reflects light from the LED on the bottom of the course. The intensity/quality of reflected light varies as Black & white lines reflects it alternatively. The alternate reflections are used to calculate the relative position change.

Advantages :- (i) None of comp' screen is obscure (ii) movements become easier as hand rests at a desk for movement (iii) Direct - point & - click (iv) Easy to use

* Light Pens :-

Light pens are pencil shaped devices used to select screen positions by detecting the light coming from points on the CRT screen. They are sensitive to short burst of light emitted from the phosphor coating as the instant electron beam strikes a particular point. An active light pen pointed at a spot on the screen as the electron beam light. Some of the spot generates an electrical pulse that causes the co-ordinate position of the electron beam to be recorded.

Disadvantages:

i) When used for long time, leads to an arm pain.
ii) Some part of the screen will be obscure by hand.
iii) Cannot read black pixel position.
iv) Reading might be errorous if background light has very high intensity.

* Touch-Panel :-

Touch-panels are a sensitive surface i.e. used to point directly. The panel can be touched by finger or any other objects like stylus. Transparent touch panels are integrated with computer monitor for manipulation of information display.

A basic touch panel sense voltage drop when a user touches the panel. It knows where the voltage has dropped and accordingly calculate the positions.

- Resistance based touch panel uses two substances of glass or plastic separated by insulators.

Scanned by CamScanner

When a user touches the panel these two substances are connected and the resistance at that point will drop down. The resistance drop is sensed and the touch position is calculated.

- In capacitance based touch panel, some charge is spread on the screen. When user touches the panel the charge is drawn by finger from each side proportionally. The sensor calculates the charge in frequency of charge/voltage and find out the touch position.
- In Acoustic (sound) based touch panel uses very high frequency (5 MHz). The sound waves are generated from X-axis & Y-axis sides are reflected from opposite side of either axis. When user touches the panel, the sound waves are interrupted and reflected back from their mid wave. The sensors on the X-axis & Y-axis sides measured the time the sound waves took to get reflected back and estimate the touch position.
- An optical touch panel uses Infra red (IR) emitters and sensor on X- & Y- ~~axis~~ axis sides, ~~sides measured~~ emitter are placed and on the opposite sides sensors are placed when a user touches the screen, the point will interrupt two IR beam from X- and Y-axis. This is detected by IR sensors and touch position is calculated.

Advantages:
i) Touch panel can be mounted an display screen leaving more space on desktop but mouse, joystick etc. take some space.

* TABLETS (Electronic, Sonic and Resistive):
A common device for drawing, painting or interactively selecting co-ordinate position on an object is digitizer. Typically a digitizer is used to scan over a drawing on object and to input a set of discrete co-ordinate position which can be joined with straight line segments to approximate the curve surface shape.
One type of digitizer is Graphical table (data tablets), which is used to input 2D co-ordinate by activity a hand cursor or stylus as selected position, flat surface. A hand cursor contains cross hairs for ~~selecting~~ sighting position. While a stylus is a pencil shaped device i.e. pointed to the position of the tablet. Many graphics tablets are constructed with a rectangular grid of wires embeded on ~~sequence along the wires~~ the table surface.
Electromagnetic pulses are generated in sequence along the wires and electric signal is induced in a wire coil. In an activated stylus or hand cursor to record tablet position.
Acoustic (Sonic) tablets use sound waves to detect a stylus position. Either strip microphones or point microphones are used to detect the sound emitted by an electrical spark from a stylus tip.

The position of the stylus is calculated by calculating the arrival time of generated sound at different microphone position.
Application ⇒ Digitizing a drawing in a book.

* Joysticks ⇒

A joystick consists of a small, vertical lever called the stick) mounted on a base i.e. used to steer the screen cursor around. Most joysticks select screen position with actual stick movement; others respond to pressur on the stick. The distance that the stick is moved in any ~~screen~~ direction from its center position corresponds to screen-cursor movement in that direction. Potentiometers mounted at the base of the joystick measure the amount of movement, and spring return the stick to the center position when it is released. One or more buttons can be programmed to act as input switches to signal certain actions once a screen position has been selected.

In another type of movable joystick the stick is used to activate switches that cause the screen cursor to move at a constant rate in the direction selected. Pressure-sensitive joysticks also called isometric joysticks, have a movable stick. Pressure on the stick is measured with strain gauges and converted to movement of the cursor in the direction specified.

* Cathod Ray Tubes (CRT) ⇒

The operation of most video monitors is based the standard cathod-ray tube (CRT) design.

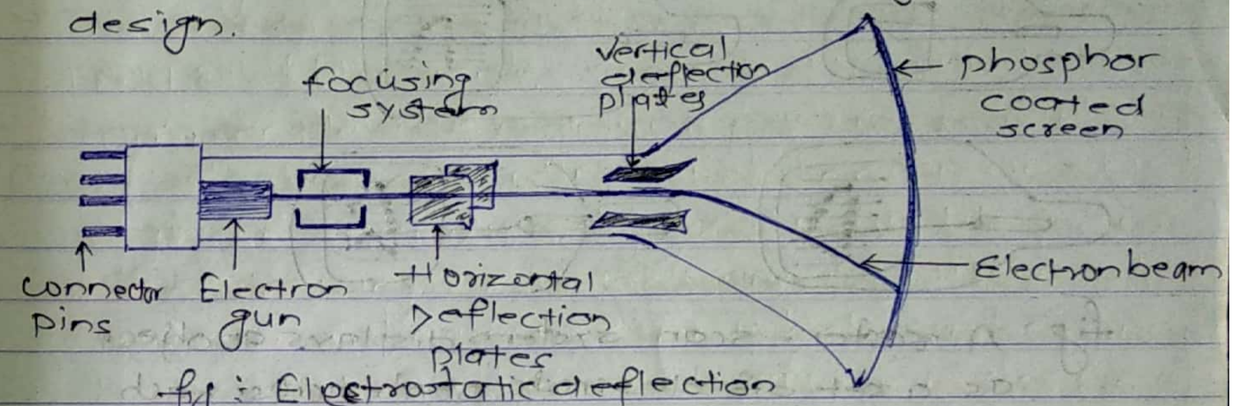


fig: Electrostatic deflection of the electron beam in CRT.

An electron gun emits a beam of electrons, which passes through focusing and deflection system and hits on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. The number of points displayed on a CRT is referred to as the resolution (eg. 1024×768). Different phosphors emit small light spot of different colors, which can combined to form a range of colors. A common methodology for color CRT display is the shadow-mask method.

The light emitted by phosphor fades very rapidly, so it needs to redraw the picture rapidly

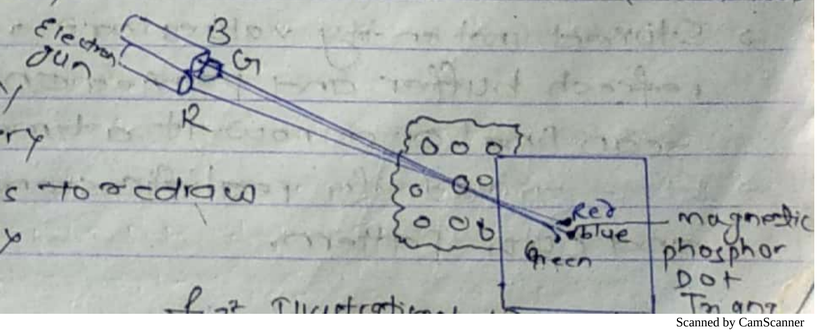


fig: Illustration

* Raster and Vector Display Architecture

* RASTER DISPLAY :-

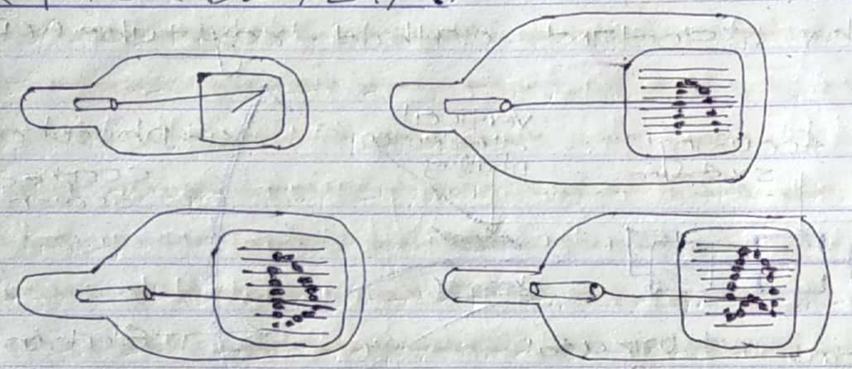


fig: A raster-scan system displays an object as a set of descrete points across each scan line.

- The raster graphics developed in early 70's
- It is common CRT monitors.
- It is based on TV technology.
- In this system, electron beam swaps across the screen one row at a time from top to bottom.
- Beam intensity vary (on or off) by the movement of electron beam across each room.

→ Picture definitions are stored in refresh buffer or frame buffer which holds the setup intensity values for all the screen point also known as pixel. (Pel)

→ Stored intensity values are retrieved from the refresh buffer and pointed on the screen on scan line (one row at a time).

→ It is suited for realistic scenes containing shading and color pattern.

→ In monochromatic monitor, frame buffer consists of one bit per each pixel and for color monitor frame buffer consists of 24 bits for each pixel.

→ The refresh rate for this system is at the rate of 60 to 80 frame per second i.e. refresh rate are described in units of cycle per second.

Raster Scan Display :-

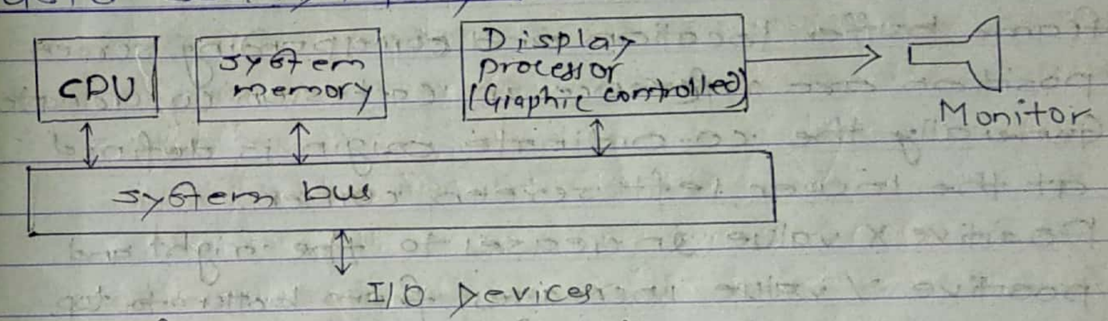


fig:- Raster scan display

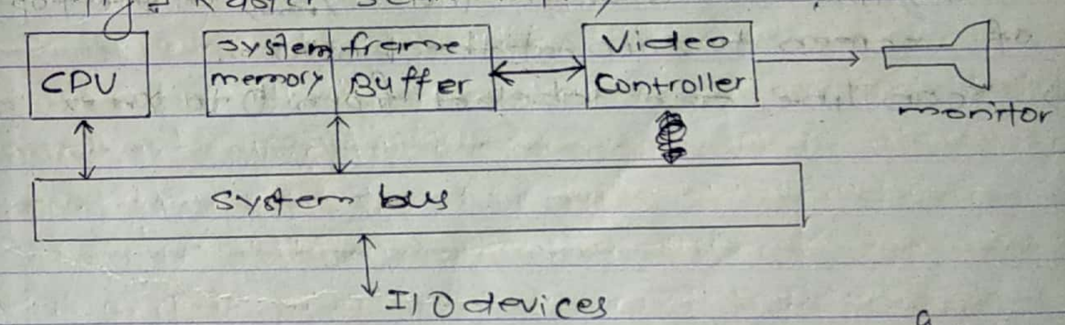


fig:- Architecture of raster system with a fixed position of the system memory required for the frame buffer.

⇒ → Interactive raster graphic generally employ several processing units.

→ In addition to the CPU a special buffer processor called video controller or display controller is used to control the operations of display devices.

→ frame buffer can be anywhere in the system memory and the video controller access the frame buffer to refresh the screen.

→ A fixed area of system memory is reserved for the frame buffer.

→ Video controller direct access the the frame buffer.

→ frame buffer location and corresponding screen position are reference in cartesian co-ordinate.

→ Generally the co-ordinate origin is defined at the lower left screen corner.

→ Positive X value increases to the right and positive Y value increases from bottom to top.

→ Scanline are labeled from Y_{max} at the top of screen to zero at the bottom.

→ Scan line are labeled from 0 to X_{max}.

* Random Scan (or Vector display system or stoke writing or calligraphic systems) :-

→ In random scan display, an image on the screen is drawn with one line at time and for this reason it is also called vector display.

→ A typical organization a vector system is shown below.

→ An application program is input and stored in the system memory along with the graphic package.

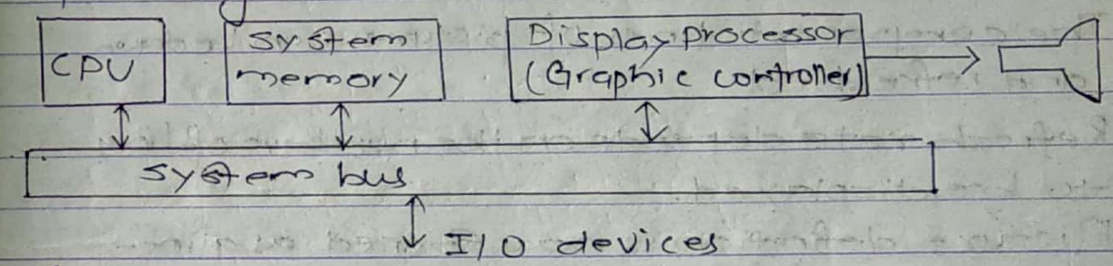


fig :- Architecture of simple random scan system

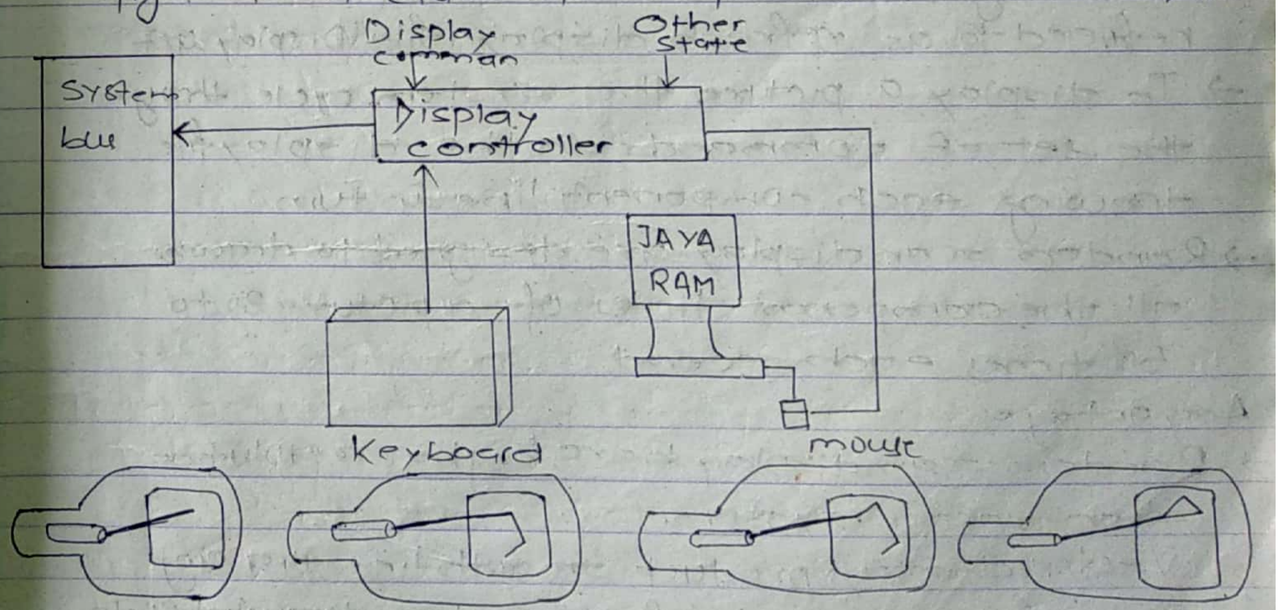


fig : Random scan system drawing through the component line of an object in ...

→ Random scan display is the use of geometrical primitives such as points, lines, curves, and polygons, which are all based upon mathematical equation.

~~Working~~:

→ In random scan display, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn.

→ Random scan sometimes draw a picture one line at a time

→ The component lines of picture can be drawn and refreshed

→ Refresh rate depends on the number of lines to be displayed.

→ Picture definition is now stored as a line-drawing commands an area of memory referred to as refresh display file (Display list)

→ To display a picture, the system cycle through the set of commands in the display file drawing each component line in turn

→ Random scan display are designed to draw all the component lines of a picture 30 to 60 times each second

Advantages

→ Random scan display have higher resolution than raster system.

→ Vector displays product smooth line drawing

→ The minimal amount of information translates to a much smaller file size.

→ On zooming in, and it remains smooth.

→ The parameters of objects are stored and can be later modified

Disadvantages:

- when the number of commands in the buffer goes high the system take a long time to process and draw pictures.
- cannot apply shading features.

* Comparision Between Raster and Random Graphics

S. N.	Raster	Random
1.	In this system, electron beam	
	* Low resolution picture	* high resolution picture.
	* zigzag picture is obtained	* smoothness
	* 60-80 frame per second refresh rate	* 30-60 frame per second refresh rate.
	* It draws picture from top to buttom one line at a time.	* It draw where picture is to be obtained directly.
	*	

* Architecture Of Graphical Display Terminals:

Colored CRT Monitor:

The CRT display color picture by using the combination of phosphorous that emits different color lights. By combining the emitted light from the different phosphorous range of colour can be generated. Two basic technique for producing colour display with CRT are :

1) Beam penetration method
2) Shadow-mask method (Raster)

Beam penetration method :

This is a random scan method, in this method a line is created. CRT beam is adjusted in such a way that electron beam only hits spots where picture is to be drawn. Two phosphorous layer (Red & Green) are adjusted such that outer layer should red and inner layer should be green. The color picture depends on how far the electron beam penetrates. The slow electron beam only excite outer red layer. The ~~speed~~ high speed electron beam excite inner green layer. At the intermediate speed, the combination of red and green lights are emitted to show two additional colors orange or yellow. This method is used with random scan monitors. Quality of picture is not as good as with other methods.

Shadow Mask Method

Operation of delta-delta CRT.

Three electron guns aligned with the triangular color dot pattern on the screen, are directed to each triangle by a shadow mask.

It is used in raster scan system object is the set of discrete point across each scanline. Three types of phosphorous dot is coated in each pixel i.e. red, green and blue. Just behind the CRT screen shadow mask grid is placed whose number is same as a number of pixel.

The three electron gun is adjusted for each phosphorous. The electron beam is deflected & focused to the shadow mask hole. The passed from the hole activate the ~~hole~~ dot triangle and produce the color spot on the screen. The color of pixel is controlled by light of intensity. By combining the color, 17 million color can be obtained. for true color each pixel has 24 bits in the frame buffer.

Advantages :-

i. Used in raster scan system (including color TV)
ii. Designed as RGB monitors.
iii. High quality raster graphic system have 24 bits each pixel in the frame buffer (a full color system or a true color system)

Scanned by CamScanner

* Advanced raster Graphic Architecture:

Scanned by CamScanner

CHAPTER#03

TWO-DIMENSIONAL ALGORITHM AND TRANSFORMATION

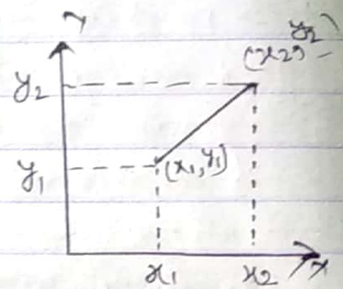
* Line Drawing Algorithm:

Scan Line Algorithm:

The cartesian slope equation of a st. line is, $y = mx + b$ — i)

where m represents the slope of the line and b is the y-intercept.

Suppose two end points of a line segment at positions (x_1, y_1) and (x_2, y_2) are shown in fig, then



$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{— ii)}$$

$$b = y - mx \quad \text{— iii)}$$

for any given x-interval Δx along the line, we compute the corresponding y-intercept Δy from eqn ii)

$$\Delta y = m\Delta x \quad \text{— iv)}$$

$$\therefore \Delta x = \frac{\Delta y}{m} \quad \text{— v)}$$

These equations form the basis of determing deflection voltage in analog devices.

Case I

for $|m| < 1$

Then Δx can be proportional to a small horizontal deflection voltage and the corresponding voltage deflection is set to y as calculated from eqn (iv)

Case II

for $|m| > 1$

Then Δy can be set proportional to small vertical deflection voltage with the corresponding horizontal voltage set proportional to Δx calculated from eqn v

Case III

for $|m| = 1$ then $\Delta x = \Delta y$ and then horizontal and vertical voltages equal.

Components

→ It uses multiplication

→ It uses float operation.

* DDA (Digital Differential Analyzer):

→ It is scan conversion algorithm.

→ It is based on calculating either Δx and Δy

→ It is faster for calculating pixel position than direct use.

→ Eliminates the multiplication and float point operation arithmatic.

Derivation:

This algorithm samples the line at unit interval in one co-ordinate and determines corresponding integer value nearest the line path for other co-ordinates.

The eqn of line is,

$$y = mx + b \quad \text{— i)}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{— ii)}$$

for any interval Δx, corresponding interval is given by $\Delta y = m\Delta x$

Case I:

$|m| < 1$, we sample at unit x interval i.e $\Delta x = 1$

$$x_{k+1} = x_k + 1 \quad \text{— iii)}$$

Then we compute each successive y-values

$$\Delta y = m \qquad [\because \Delta x = 1]$$

$$\therefore y_{k+1} = y_k + m \quad \text{— iv)}$$

Scanned by CamScanner

case II :

$|m|>1$, we sample at unit y-interval i.e. $\Delta y = 1$ and compute each successive x-values

$$\therefore \quad 1 = m\Delta x$$

$$\Delta x = \frac{1}{m}$$

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{—— v)}$$

$$\therefore y_{k+1} = y_k + 1 \quad \text{—— vi)}$$

Above eqns hold for the lines processed from left end to right end. for the reverse process i.e. if the line is to be processed from right to left then,

for $|m|<1$, $\Delta x = -1$

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

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

for $|m|>1$, $\Delta y = -1$

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

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

eg: Digitize a line with two end points (10,15) & (15,30)

soln:-

$$m = \frac{30-15}{15-10} = 3$$

$\therefore |m|>1$

so, we sample at y-interval,

$$x_{k+1} = x_k + 1/m \quad , \quad y_{k+1} = y_k + 1$$

S.N	x	y
1	10	15
2	10	16
3	11	17
4	11	18

Comments :

→ It is faster to calculate the position

→ Due to propagation of round off error due to successive addition the calculated pixel may shift among from the true line path.

Algorithm :-

```
// DDA routine for restoring a line, end position are (x1, y1)
and (x2, y2) assumed not equal
  If abs(x2-x1) ≥ abs(y2-y1) then
      length := abs(x2-x1);
  else
      length := abs(y2-y1);
  end if
    select the larger Δx or Δy to be one raster unit
    Δx := (x2-x1)/length ;
    Δy := (y2-y1)/length;
// round the values rather than truncate using sign function
// makes the algorithm working in all quadrature
    x := x1 + 0.5 * sign(Δx) ;
    y := y1 + 0.5 * sign(Δy) ;
//begin main loop
    i := 1
    while (i ≤ length)
      Plot (Integer(x), Integer(y));
        x := x + Δx;
        y := y + Δy;
        i := i+1;
    end while
// finish
```

Scanned by CamScanner

case II :

$|m|>1$, we sample at unit y-interval i.e. $\Delta y = 1$ and compute each successive x-values

$$\therefore 1 = m\Delta x$$

$$\Delta x = \frac{1}{m}$$

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{--- v)}$$

$$\therefore y_{k+1} = y_k + 1 \quad \text{--- vi)}$$

Above eqns hold for the lines processed from left end to right end. for the reverse process i.e. if the line is to be processed from right to left then,

for $|m|<1$, $\Delta x = -1$

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

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

for $|m|>1$, $\Delta y = -1$

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

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

eg: Digitize a line with two end points (10,15) & (15,30)

soln:

$$m = \frac{30-15}{15-10} = 3$$

$\therefore |m|>1$

so, we sample at y-interval,

$$x_{k+1} = x_k + 1/m \quad , \quad y_{k+1} = y_k + 1$$

S.N	x	y
1	10	15
2	10	16
3	14	17
4	11	18

Comments :

→ It is faster to calculate the position

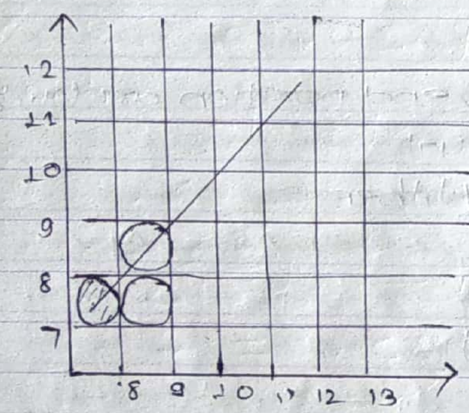
→ Due to propagation of round off error due to successive addition the calculated pixel may shift among from the true line path.

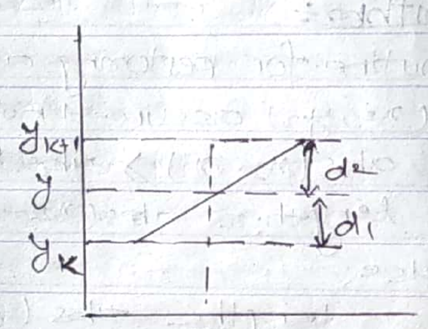
Algorithm :

```
// DDA routine for restoring a line, end position are (x1, y1)
and (x2, y2) assumed not equal
  If abs(x2 - x1) ≥ abs(y2 - y1) then
      length := abs(x2 - x1);
  else
      length := abs(y2 - y1);
  end if
  select the larger Δx or Δy to be one raster unit
  Δx := (x2 - x1)/length ;
  Δy := (y2 - y1)/length;
// round the values rather than truncate using sign function
// makes the algorithm working in all quadrature
  x := x1 + 0.5 * sign(Δx) ;
  y := y1 + 0.5 * sign(Δy) ;
// begin main loop
  i := 1
  while (i ≤ length)
      plot (Integer(x), Integer(y));
      x := x + Δx;
      y := y + Δy;
      i := i+1;
  end while
// finish
```

* Bresenham's line Algorithm (BLA):

DDA includes calculation related to m and 1/m which is little complicated. Bresenham's improves DDA algorithm by only involving integer calculation.





Bresenham's algorithm selects the best pixel co-ordinate by testing the sign of an integer parameter whose value is proportional to the difference between the separation of two pixel actual line path.

Case I

$|m| < 1$, $m > 0$

Let (x_k, y_k) be the pixel position determined then the next pixel to be plotted is either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1})

Let d_1 & d_2 be the separation of pixel position (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}) from the actual line path.

$$y = mx + b \quad \text{— i)}$$

Then at the sampling position (x_{k+1})

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

from fig

$$d_1 = y - y_k$$

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

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

Let us define a decision parameter P_k for the kth step by

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

$\Delta x > 0$

$\therefore P_k < 0$ if $d_1 < d_2$

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

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

$$= \Delta x\{y - y_k - y_k - 1 + y\} \qquad \boxed{\text{Since } y_{k+1} = y_k + 1}$$

$$= \Delta x\{2y - 2y_k - 1\} \qquad [\text{using eq (ii)}]$$

$$= \Delta x[\{2m(x_k + 1) + b\} - 2y_k - 1] \qquad [\because x_{k+1} = x_k + 1]$$

$$= \Delta x[2mx_k + 2m + 2b - 2y_k - 1]$$

$$P_k = 2mx_k\Delta x - 2y_k\Delta x + (2m + 2b - 1)\Delta x$$

$$= 2\left(\frac{\Delta y}{\Delta x}\right)x_k.\Delta x - 2y_k\Delta x + (2m + 2b - 1)\Delta x$$

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

where $c = (2m + 2b - 1)\Delta x$ is a const.

for next step

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

Scanned by CamScanner

from eqns iii) & iv)

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

i.e.

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

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

where

$$y_{k+1} - y_k = 0 \text{ or } 1$$

<u>If $P_k < 0$</u> then we plot lower pixel

$$y_{k+1} = y_k \quad \text{—(v)}$$

$$\therefore P_{k+1} = P_k + 2\Delta y \quad \text{—(vi)}$$

<u>If $P_k \geq 0$</u> then we plot upper pixel

$$\therefore y_{k+1} = y_k + 1 \quad \text{— vii}$$

$$\therefore P_{k+1} = P_k + 2\Delta y - 2\Delta x \quad \text{— viii}$$

Therefore line decision parameter is

$$P_0 = 2\Delta y x_0 - 2\Delta x y_0 + c \quad \text{(using the value of c)}$$

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

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

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\cdot\left(\frac{\Delta y}{\Delta x}\right)\cdot\Delta x + 2(y_0 - m x_0)\Delta x - \Delta x$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\cdot\frac{\Delta y}{\Delta x}\cdot\Delta x - \Delta x$$

$$\boxed{\therefore P_0 = 2\Delta y - \Delta x}$$

Q. Digitize a line with end points (15,18) & (10,15) using BLA.

solution:

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

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

$|m| < 1$, we sample at x direction

first pixel, (10,15)

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

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

We know that if $P_k < 0$

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

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

if $P \geq 0$

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

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

We have $P_k \geq 0$

$$P_1 = P_0 + 2\Delta y - 2\Delta x$$

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

$$= -3$$

$$P_2 = P_1 + 2\Delta y$$

$$= -3 + 2\times 3 = 3$$

$$\therefore P_2 = 3$$

$$P_3 = P_2 + 2\Delta y - 2\Delta x$$

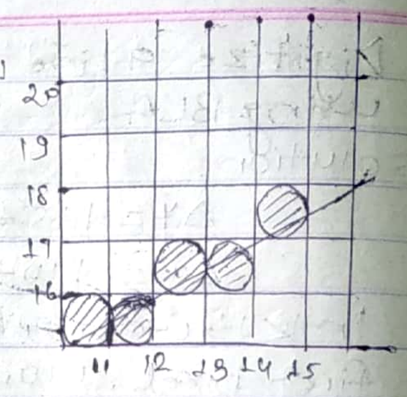
$$= 3 + 2\times 3 - 2\times 5$$

$$P_3 = -1$$

$$P_4 = P_3 + 2\Delta y = -1 + 2\times 3 = 5$$

Scanned by CamScanner

K	P_K	x_{K+1}	y_{K+1}
0	1	11	16
1	-3	12	16
2	3	13	17
3	-1	14	17
4	5	15	18



Q. Digitize a line with end points (15, 15) & (10, 18) use BLE and DDA

Solution:

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

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

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

$|m|<1$, we sample at x direction.

first pixel (10, 15).

$P_0 = 2\Delta y - \Delta x = 2\times3 - 5 = 1$

we know that, If $P_K \geq 0$

$P_1 = P_0 + 2\Delta y - 2\Delta x$

$= 1 + 2\times3 - 2\times5 = -3$

$P_K < 0$

$P_2 = P_1 + 2\Delta y$

$= -3 + 2\times3 = 3$

$P_3 = P_2 + 2\Delta y - \Delta x$

$= 3 + 2\times3 - 2\times5 = -1$

$P_K < 0$

$P_4 = P_3 + 2\Delta y = -1 + 2\times3 = 5$

Note:

for $P_K \geq 0$

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

$y_{K+1} = y_K - 1$

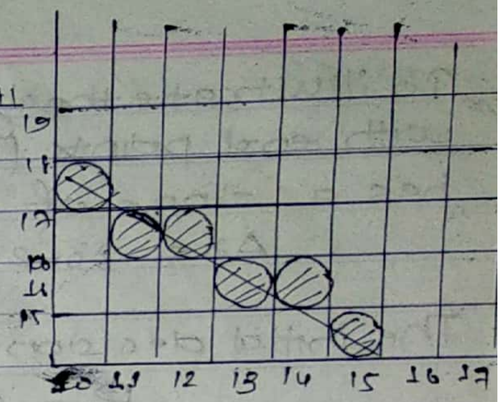
when slope is -ve

$P_K < 0$

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

$y_{K+1} = y_K$

K	P_K	x_{K+1}	y_{K+1}
0	1	11	17
1	-3	12	17
2	3	13	16
3	-1	14	16
4	5	15	16



* Bresenham's Line-Drawing Algorithm for $|m|<1$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0)
2. Load (x_0, y_0) into the frame buffer; that is plot the first point.
3. Calculate the constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as
$$P_0 = 2\Delta y - \Delta x$$
4. At each x_K along the line, starting at $K=0$, perform the following test:
If $P_K < 0$, the next point to plot is (x_K+1, y_K) and
$$P_{K+1} = P_K + 2\Delta y$$
Otherwise, 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.

Scanned by CamScanner

To illustrate the algorithm we digitize the line with end points (20, 10) and (30, 18). This line has a slope of 0.8 with

$$\Delta x = 30-20 = 10, \quad \Delta y = 18-10 = 8$$

The initial decision parameter has the value

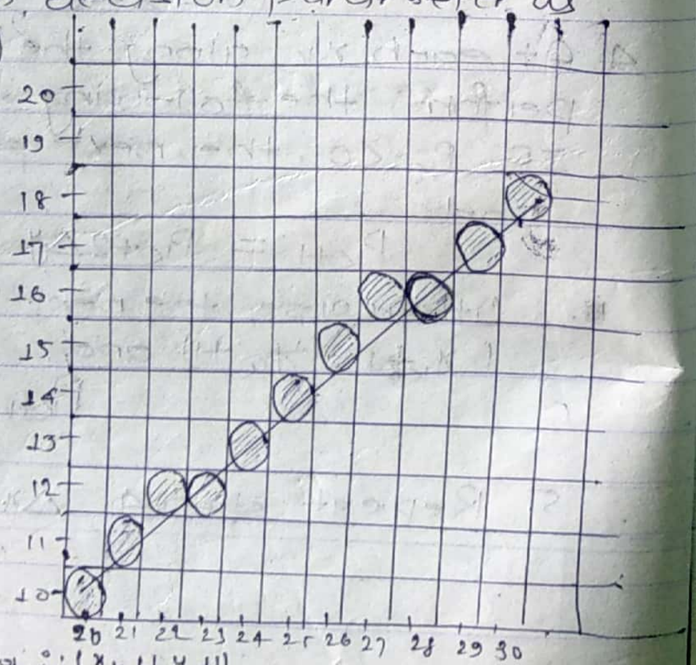
$$P_0 = 2\Delta y - \Delta x$$
$$= 2\times 8 - 10$$
$$P_0 = 6$$

and the increments for calculating successive decision parameters are

$$2\Delta y = 2\times 8 = 16$$
$$2\Delta y - 2\Delta x = 2\times 8 - 2\times 10 = -4$$

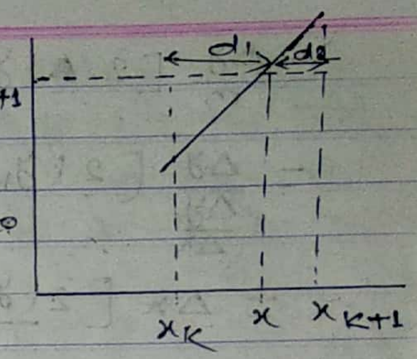
We plot the initial point $(x_0, y_0) = (20, 10)$ and determine successive pixel positions along the line path from decision parameter as

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



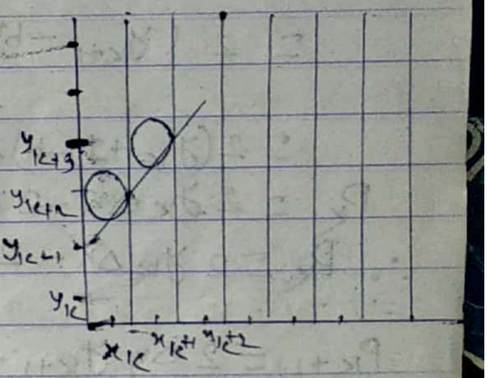
Note: if $P_k < 0$, pt to plot is (x_{k+1}, y_k)
& $P_{k+1} = P_k + 2\Delta y$
otherwise $P_{k+1} = P_k + 2\Delta y - 2\Delta x$ & pt to plot is (x_{k+1}, y_{k+1})

Case II : $|m| > 1$



Let (x_k, y_k) be the pixel position determined then the next pexel to the plotted is either (x_{k+1}, y_{k+1}) or (x_k, y_{k+1}).

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



$$y = mx + b$$

The actual value of x is given by

$$x = (y-b)/m$$

sampling position at y_{k+1}

from fig,

$$d_1 = x - x_k \quad \&$$
$$d_2 = x_{k+1} - x$$

Let us define a decision parameter P_k and P_k is defined as,

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

$\Delta y > 0$

$$P_k < 0 \text{ if } (d_1 < d_2)$$
$$P_k \geq 0 \text{ if } (d_1 \geq d_2)$$

$$P_k = \Delta y\,(d_1 - d_2) = \Delta y\,(x - x_k - x_{k+1} + x) = \Delta y\,(2x - x_{k+1} - x_k)$$
$$= \Delta y\,(2x - x_k - 1 - x_k) = \Delta y\,(2x - 2x_k - 1)$$
$$= \Delta y\left[2\cdot\frac{(y_{k+1} - b)}{m} - 2x_k - 1\right]$$

Scanned by CamScanner

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

$$= \frac{\Delta y}{\frac{\Delta y}{\Delta x}}\left[2(y_{k+1}-b) - 2x_k\cdot\left(\frac{\Delta y}{\Delta x}\right) - \Delta y/\Delta x\right]$$

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

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

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

$$P_k = 2y_k\Delta x - 2x_k\Delta y + 2(1-b)\Delta x - \Delta y \quad \text{—} (*)$$

$$\therefore P_k = 2y_k\Delta x - 2x_k\Delta y + c \quad \text{—(i)} \quad \text{where } c = 2(1-b)\Delta x - \Delta y$$

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

from eqn (i) & (ii)

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

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

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

<u>If $P_k \geq 0$</u>

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

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

<u>If $P_k < 0$</u>

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

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

for initial parameter

from eqn (*),

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

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

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

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

$$= 2\Delta x\, y_0 - 2\Delta y\, x_0 + 2\Delta x - 2\left(y_0 - \frac{\Delta y}{\Delta x}x\right)\Delta x - \Delta y$$

$$= 2\Delta x\, y_0 - 2\Delta y\, x_0 + 2\Delta x - (2\Delta x\, y_0 - \Delta y\, x_0)\frac{\Delta x}{\Delta x} - \Delta y$$

$$\therefore \boxed{P_0 = 2\Delta x - \Delta y}$$

Note: $|m| < 1$, x-direction sample, $x_{k+1} = x_k + 1$

$|m| > 1$, y-direction sample, $y_{k+1} = y_k + 1$

* Bresenham's Complete Algorithm:

1. Input two end points (x_1, y_1) and (x_2, y_2)
2. Compute $dx = |x_2 - x_1|$ and $dy = |y_2 - y_1|$
3. If $x_2 - x_1 < 0$ and $y_2 - y_1 > 0$ or $x_2 - x_1 > 0$ & $y_2 - y_1 < 0$
   Then set $a = -1$ else $a = 1$
4. If $dy < dx$ then
   i. if $x_1 > x_2$ then $t := x_1$; $x := x_2$; $x_2 := t$;
      $t := y_1$; $y := y_2$; $y_2 := t$
   ii. find initial decision parameter $P = 2dy - dx$
   iii. plot the first pixel (x_1, y_1)
   iv. Repeat the following till $|x_1| < |x_2|$
      a. If $P < 0$ then,
         $P := P + 2dy$;
         else, $P := P + 2dy - 2dx$
         $y := y + a$;

Scanned by CamScanner

b. Increase x_1 by 1 i.e. $x_1 = x_1 + 1$;

c. plot (x_1, y_1)

5. Else $|m| > 1$

i. check if $(y_1 > y_2)$ then

$t := x_1$; $x_1 = x_2$; $x_2 := t$;

$t := y_1$; $y_1 = y_2$; $y_2 := t$;

ii. find the initial decision parameter, $P_0 = 2dx - dy$

iii. Plot the first point (x_1, y_1)

iv. Repeat the following till $y_1 < y_2$

a. If $P < 0$ then, $P = P + 2dx$;

else, $P = P + 2dx - 2dy$;

$x_1 = x_1 + a$;

b. Increase y_1 by 1 i.e. $y_1 = y_1 + 1$;

c. plot (x_1, y_1)

* CIRCLE:

A circle is defined as the locus of the point (i.e. set of points) which moves in a plane so that its distance from a point is constant, the fixed point is centre and constant distance is called the radius of the circle.

In other words, a circle is defined as a set of points that are all at a given distance 'r' from the centre position (x_c, y_c). This distance relationship is expressed by the Pythagorean theorem in cartesian co-ordinate as

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

Methods To Draw Circle:

a.) Direct Method

b) Trigonometric Method

c) Midpoint circle method

(a) Direct Method

$$x^2 + y^2 = r^2 \quad \text{(centre at } (0,0))$$

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

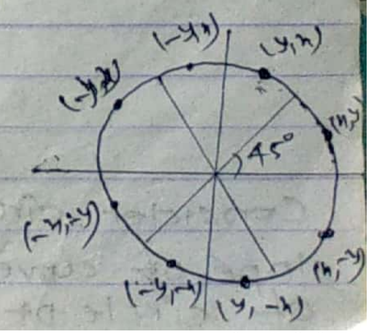
$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad \text{center at } (x_c, y_c)$$

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

(b) Trignometric Method:

$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta$$



Symmetry of a circle. Calculation of a circle pt (x,y) in one octant yields the circle pts shown for the other seven octants.

* Bresenham's Mid point Algorithm:

This is also a scan converting algorithm.
The equation of the circle with centre (h,k) is given by,

$$(x-h)^2+(y-k)^2=r^2 \quad \text{— i)}$$

when $h=k=0$, then eqn of the circle at origin is

$$x^2+y^2=r^2 \quad \text{— ii)}$$

Differentiating both sides,

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

or, $\frac{dy}{dx}=-\frac{x}{y}$, where $\frac{dy}{dx}$ = slope

Now, if $\frac{dy}{dx}=0$, then $x=0$

If $\frac{dy}{dx}=-1$ then $x=y$

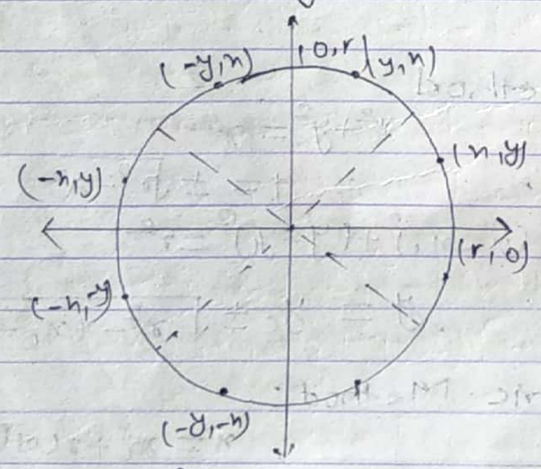


fig: Symmetry of circle

Consider circle section for $x=0$, $x=y$ where slope of curve varies from 0 to 1. Calculation of circle pt (x,y) in one octant gives the

circle point shown for other seven octants. To apply the mid point we define a circle function as

$$f_{circle}(x,y)=x^2+y^2-r^2 \quad \text{— iii)}$$

Suppose

$$f(x,y)=\begin{cases} <0, & \text{if } (x,y) \text{ is inside the circle boundary} \\ =0, & \text{if } f(x,y) \text{ is on the circle boundary} \\ >0, & \text{if } f(x,y) \text{ is outside the circle boundary} \end{cases}$$

The circle function tests are performed for the mid positions between pixels near the circle path at each sampling step.

Here we assume that we have just plotted the pixel (x_k, y_k). We next need to determine whether the pixel at position (x_{k+1}, y_k) or the one at the position (x_{k+1}, y_k-1) is closer to the circle.

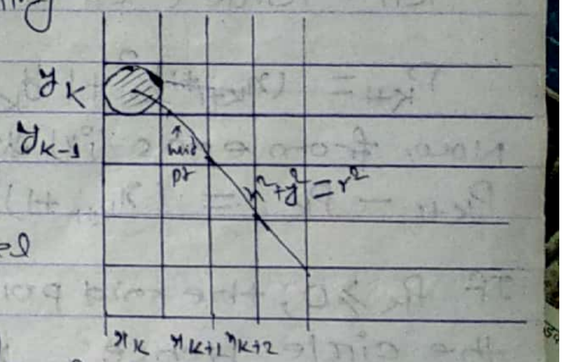


fig: Midpoint between co-ordinate pixels at sampling position x_{k+1} along a circular path.

The decision parameter is the circle function which we have evaluated in eqn (iii) at mid point between these two pixel.

i.e.

$$P_k=f_{circle}(x_{k+1}, y_k-\tfrac{1}{2})$$

$$P_k=(x_k+1)^2+(y_k-\tfrac{1}{2})^2-r^2 \quad \text{— (iv)}$$

If $P_k < 0$, this mid pixel is inside the circle and the ~~circle~~ pixel on the scan line y_k is closer to the circle boundary. Otherwise, the mid position is outside or on the circle boundary, and we select the pixel on scanline $y_k - 1$, successive decision parameters can be obtained using incremental calculation.

We obtained a recursive expression for the next decision parameter by evaluating the circle function at sampling position.

$$x_{k+1} + 1 = x_k + 2$$

$$P_{k+1} = f_{circle}(x_{k+1}+1, y_{k+1} - \tfrac{1}{2})$$

$$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1} - \tfrac{1}{2})^2 - r^2 \quad \text{— (v)}$$

Now, from eqns (iv) & (v)

$$P_{k+1} - P_k = (x_{k+1}+1)^2 + (y_{k+1} - \tfrac{1}{2})^2 - (x_k+1)^2 - (y_k - \tfrac{1}{2})^2 \quad \text{— (vi)}$$

If $P_k \geq 0$, the mid point lies on the outside the circle hence, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

from eqn (vi)

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

$$= P_k + (x_k+1)^2 + 2(x_k+1)\cdot 1 + 1 + (y_k - \tfrac{1}{2})^2 - 2(y_k - \tfrac{1}{2})\cdot 1 + 1 - (x_k+1)^2 - (y_k - \tfrac{1}{2})^2$$

$$\boxed{P_{k+1} = P_k + 2(x_{k+1} - y_{k+1}) + 1} \quad \text{— (vii)}$$

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

value of y_{k+1} = either y_k or y_{k-1} depends on P_k

If $P_k < 0$, the mid point lies inside the circle.

$$x_{k+1} = x_k + 1, \quad y_{k+1} = y_k$$

so eqn (vi) will becomes

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

$$= P_k + (x_k+1)^2 + 2(x_k+1) + 1 - (x_k+1)^2$$

$$\boxed{P_{k+1} = P_k + 2(x_k+1) + 1} \quad \text{— (viii)}$$

for initial decision parameter $(x_0, y_0) = (0, r)$

$$\therefore P_0 = f(x_0+1, y_0 - 1/2)$$

$$= f(1, r - 1/2)$$

$$= 1 + (r - 1/2)^2 - r^2$$

$$= 1 + r^2 - r + \tfrac{1}{4} - r^2 = \tfrac{5}{4} - r$$

$$\therefore P_0 = 1 - r \quad \text{— ix}$$

Since 5 & 4 are integer values so ~~that~~ can be rendered off. or $\frac{5}{4} = 1.25$ taking integer only

Midpoint Circle Algorithm:

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as
$(x_0, y_0) = (0, r)$
2. Calculate the initial value of 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
$$P_{k+1} = P_k + 2x_{k+1} + 1$$
Otherwise, the next point along the circle is (x_{k+1}, y_{k-1}) and
$$P_{k+1} = P_k + 2(x_{k+1} - y_{k+1}) + 1$$
where
$$2x_{k+1} = 2x_k + 2 \quad \& \quad 2y_{k+1} = 2y_k - 2$$
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and the plot the coordinate values:
$x = x + x_c$, $y = y + y_c$
6. Repeat step 3 through 5 until $x \geq y$

Q- Digitize $x^2 + y^2 = 100$ in first octant.

Soln:-

centre $(0,0)$

Radius, $r = 10$

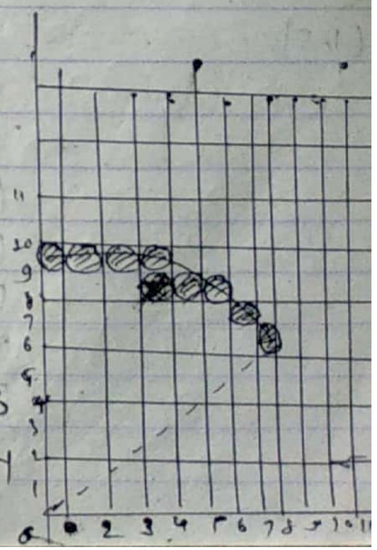
We demonstrate the mid point circle algorithm by determining positions along the circle octant in the first quadrant from $x=0$, to $x=y$.

The initial value of decision parameter is
$$P_0 = 1 - r$$
$$P_0 = 1 - 10 = -9$$

Successive decision parameter values are
Positions along the circle path are calculated using mid-point method as,

$$P_{k+1} = P_k + 2x_{k+1} + 1 \quad (P_k < 0)$$
Set
$$x_{k+1} = x_k + 1, \quad y_{k+1} = y_k$$
$$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1 \quad (P_k > 0)$$
Set $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

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



$(x,y) = (x_1+x_c, y_1+y_c)$

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

soln:

centre $(h,k) = (2,3)$

Radius, $r = 5$

1st pixel $(0,5)$

$P_0 = 1-r = 1-5$

$\therefore P_0 = -4$

if $P_{k+1} = P_k + 2x_{k+1} + 1 \quad (P_k < 0)$

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

if $P_k > 0$, $P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$,

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

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

Other pixel	Actual pixel		$(x = x_c + x, y = y_c + y)$
(1, 5)	(1, 5)	(3, 8)	(x, y)
	(-1, 5)	(1, 8)	$(-x, y)$
	(-1, -5)	(1, -2)	$(-x, -y)$
	(1, -5)	(3, -2)	$(x, -y)$
	(5, 1)	(7, 4)	(y, x)
	(5, -1)	(7, 2)	$(y, -x)$
	(-5, -1)	(-3, 2)	$(-y, -x)$
	(-5, 1)	(-3, 4)	$(-y, x)$

(2, 5)	(2, 5)	(4, 8)
	(-2, 5)	(0, 8)
	(-2, -5)	(0, -2)
	(2, -5)	(4, -2)
	(5, 2)	(7, 5)
	(5, -2)	(7, 1)
	(-5, -2)	(-3, 1)
	(-5, 2)	(-3, 5)
(3, 4)	(3, 4)	(5, 7)
	(-3, 4)	(-1, 7)
	(-3, -4)	(-1, -1)
	(3, -4)	(5, -1)
	(4, 3)	(6, 6)
	(4, -3)	(6, 0)
	(-4, -3)	(-2, 0)
	(-4, 3)	(-2, 6)
(4, 3)	(4, 3)	(6, 6)
	(-4, 3)	(-2, 6)
	(-4, -3)	(-2, 0)
	(4, -3)	(6, 0)
	(3, 4)	(5, 7)
	(3, -4)	(5, -1)
	(-3, -4)	(-1, -1)
	(-3, 4)	(-1, 7)

Scanned by CamScanner

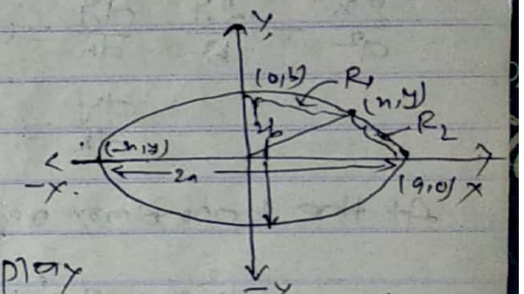
* ELLIPSE :-

Ellipse is an elongated circle. Therefore elliptical curves can be generated by modifying circle-drawing-procedures to ~~keep~~ take into account the different dimensions of an ellipse along the major and minor axes.

The equation of ellipse in standard form is

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

$$\therefore \quad y = \pm b\sqrt{\left(1 - \frac{x^2}{a^2}\right)}$$



The algorithm is used to display the ellipse in standard form. The algorithm is applied throughout the 1st quadrant according to the scope of the ellipse. for the region R_1 where the slope of region is less than 1. We process by taking unit step in x-direction and for region R_2 we take unit step in y-direction.

Let us define an ellipse function centered at origin by,

$$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2$$

If $f(x,y) < 0$, the point lies inside the ellipse
$= 0$, the point lies on the ellipse.
> 0, the point lies outside the ellipse.

Starting at $(0,b)$ we take the unit steps in x-direction until we reach the boundary between region R_1 & R_2

Then we switch to unit steps 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 is calculated from eqn $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

Differentiating both sides w.r.t x

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

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

At the boundary region 1 & region 2, $\frac{dy}{dx} = -1$

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

Therefore, we move out of region 1 (R_1) when $2b^2 x \geq 2a^2 y$ for the region R_1

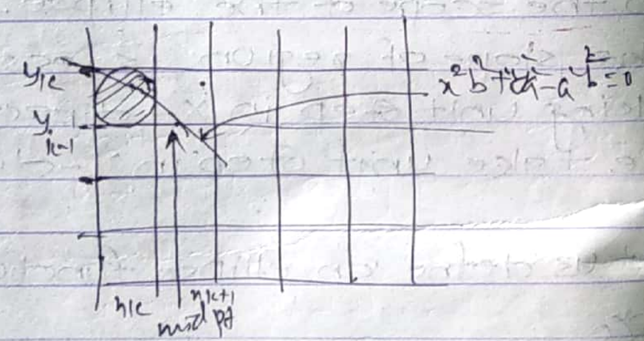


fig: Mid point between candidate pixel at sampling position x_{k+1} along in elliptical path

If (x_k, y_k) is the pixel first plotted then the candidate pixels 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

$$P_k = f(x_{k+1}, y_{k-1/2})$$

$$\therefore P_k = b^2(x_{k+1})^2 + a^2(y_k - \tfrac{1}{2})^2 - a^2b^2 \quad \text{—(i)}$$

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

Substracting (i) & (i)

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

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

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

$$\therefore \boxed{P_{k+1} = P_k + 2b^2 x_{k+1} + b^2} \quad \text{—— iii)}$$

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

If $P_k \geq 0$, the mid point lies outside the ellipse so we plot (x_{k+1}, y_{k-1})

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

So eqn (*) becomes,

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

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

$$\therefore \boxed{P_{k+1} = P_k + 2b^2 x_{k+1} - 2a^2 y_{k+1} + b^2}$$

Scanned by CamScanner

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

Now,

initial decision parameter is

$$P_0 = f(x_0+1, y_0 - 1/2)$$

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

$$= b^2 + a^2(b^2 - b + 1/4) - a^2b^2$$

$$= b^2 + a^2b^2 - a^2b + a^2/4 - a^2b^2$$

$$\boxed{P_0 = b^2 - a^2b + a^2/4} \quad \text{—(v)}$$

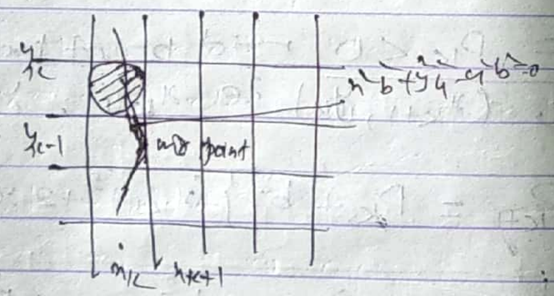


fig: Mid point between candidate pixel at sampling position y_{k-1} alongan elliptical path.

for Region R_2 ($|m| > 1$)

If (x_k, y_k) be the pixel just plotted then the next plotting pixel will be either (x_k, y_{k-1}) or (x_{k+1}, y_{k-1}). So let us define the decision parameter at the midpoint $x_k + \frac{1}{2}$ at sampling position y_{k-1} by

$$P_k = f(x_k + \tfrac{1}{2}, y_{k-1})$$

$$P_k = b^2(x_k + \tfrac{1}{2})^2 + a^2(y_k - 1)^2 - a^2b^2 \quad \text{—(i)}$$

$\therefore$

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

substrating (i) & (ii)

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

<u>If $P_k > 0$</u>, the mid point lies outside the ellipse, so we plot (x_k, y_{k-1}) i.e. $x_{k+1} = x_k$ & $y_{k+1} = y_{k-1}$

so eqn iii will be,

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

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

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

$$\boxed{P_{k+1} = P_k - 2a^2y_{k+1} + a^2} \quad \text{—(v)}$$

<u>If $P_k \le 0$</u>, the mid point lies on inside the ellipse, so we plot $(x_{k+1} = x_k + 1, y_{k+1} = y_k - 1)$

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

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

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

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

Scanned by CamScanner

The initial value of decision parameter is

$$P_0 = f(x_0+1/2, y_0-1)$$

$$P_0 = b^2(x_0+1/2)^2 + a^2(y_0-1)^2 - a^2b^2 \quad \text{--- (vi)}$$

* Mid Point Ellipse Algorithm:

1. Obtain the centre (x_c, y_c), semi-major and semi minor axis lengths as a and b
2. set $x=0, y=b$ i.e. $(x_0, y_0) = (0, b)$
3. ~~plot the p. Calculate the~~
3. plot the point (x,y) and its symmetry point at appropriate positions by $x = x+x_c, y = y+y_c$
4. Calculate the initial value of decision parameter in region 1 (R_1) as,

$$P_0 = b^2 - a^2b + a^2/4$$

5. At each x_k position in region 1, starting at $k=0$ perform the following tests:

if $P_k < 0$, the next point along the ellipse centered on $(0,0)$ is (x_{k+1}, y_k) &

$$P_{k+1} = P_0 + 2b^2x_{k+1} + b^2$$

Otherwise, the next point along the ellipse is (x_{k+1}, y_{k-1}) &

$$P_{k+1} = P_0 + 2b^2x_{k+1} - 2a^2y_{k+1} + b^2$$

with $2b^2x_{k+1} = 2b^2x_k + 2b^2$,

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

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

6. Calculate the initial value of decision parameter in region 2 (R_2) using the last point (x_0, y_0) calculated in region 1 (R_1) as

$$P'_0 = b^2(x+0.5)^2 + a^2(y-1)^2 - a^2b^2$$

7. At each y_k position in R_2, starting at $k=0$ perform the following test:

If $P'_0 > 0$, the next point along the ellipse centered on $(0,0)$ is (x_k, y_k-1) and

$$P'_{k+1} = P'_0 - 2a^2y_{k+1} + a^2$$

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

$$P'_{k+1} = P'_0 - 2a^2y_{k+1} + a^2 + 2b^2x_{k+1}$$

using the same incremental calculation for x and y as in region 1.

8. Determine symmetry points in the other three quadrants.
9. Move each calculated ~~points in the other~~ pixel position (x,y) onto the elliptical path centered (x_c, y_c) & plot the co-ordinate values:

$$x = x + x_0 \qquad y = y + y_c$$

10. Repeat the steps for region 1 until $2b^2x \geq 2a^2y$

Q. Digitize an ellipse $(x-2)^2/64 + (y+5)^2/36 = 1$ using mid point algorithm.

Soln:-

center $= (2, -5)$

$a = 8$ $\quad 2b^2x = 0$ (with $2b^2 = 72$

$b = 6$ $\quad 2a^2y = 2a^2y$ ($-2a^2 = -128$)

for region 1

first pixel is $(0, b) = (0, 6)$

$P_0 = b^2 - a^2b + a^2/4$

$P_0 = -332$

succesive decision parameters values and positions along the ellipse path are calculated using the mid point method as, if $P_k < 0$, $P_{k+1} = P_k + 2b^2x_{k+1} + b^2$, (x_{k+1}, y_k)
if $P_k > 0$, $P_{k+1} = P_k + 2b^2x_{k+1} - 2a^2y_{k+1} + b^2$, (x_{k+1}, y_{k+1})

k	P_k	(x_{k+1}, y_{k+1})	$2b^2x_{k+1}$	$2a^2y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

We now move out of region 1, since $2b^2x_{k+1} > 2a^2y_{k+1}$
$504 > 384$

for region 2, initial point is $(x_0, y_0) = (7, 3)$ and initial decision parameter

$P_0' = b^2(x_0 + 1/2) + a^2(y-1)^2 - a^2b^2$
$= 36(7 + 1/2)^2 + 64(3-1)^2 - 36 \times 64$
$P_0' = -151$

The remaining positions along the ellipse path in the first quadrant are then calculated as

if $P_k > 0$

$P'_{k+1} = P'_k - 2a^2y_{k+1} + a^2$ where $x_{k+1} = x_k$; $y_{k+1} = y_k - 1$

if $P_k \leq 0$

$P'_{k+1} = P'_k + 2b^2x_{k+1} - 2a^2y_{k+1} + a^2$, where $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

k	P'_k	(x_{k+1}, y_{k+1})	$2b^2x_{k+1}$	$2a^2y_{k+1}$
0	-151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	169	(8, 0)	—	—

A plot of the selected positions around the ellipse boundary within the first quadrant is shown in fig below:

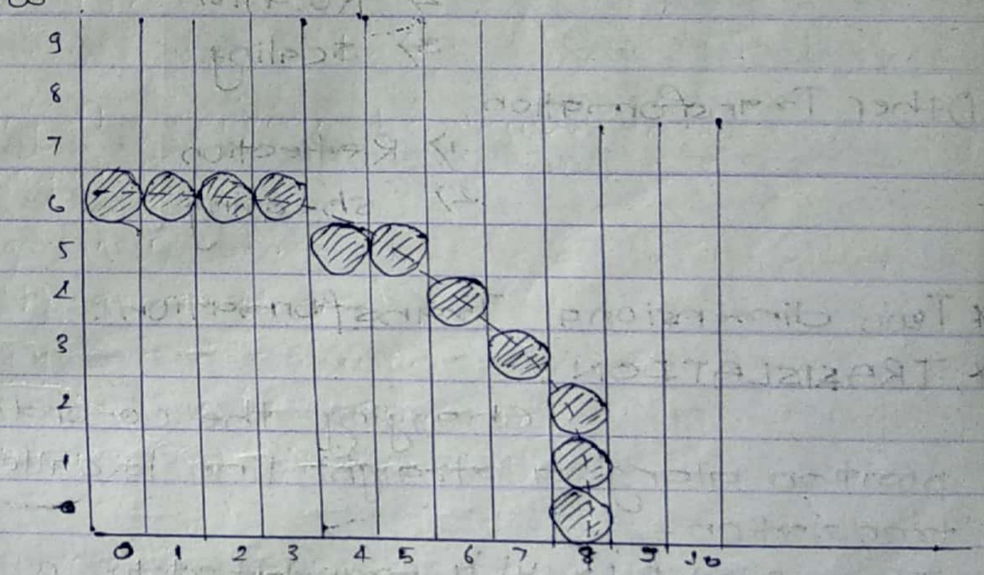


fig: positions along the elliptical path centered on the origin with $b = 6$, $a = 8$ using the mid point algorithm to calculate the pixel addresses in the first quadrant.

Scanned by CamScanner

* Transformation :

Changing co-ordinate description of an object is called transformation.

Types:

- Rigid-body transformation (transformation without deformation in shape i.e. every point on the object is translated by the same amount)
- Non rigid body transformation (transformation with change in shape)

Basic Transformation

1) Translation
2) Rotation
3) Scaling

Other Transformation

1) Reflection
2) shearing

* Two dimensional Transformation:

* TRANSLATION:

Changing the co-ordinate position along a straight line is called translation.

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

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

Translation is a rigid-body transformation.

The translation distance pair (t_x, t_y) is called translation vector or shift vector.

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

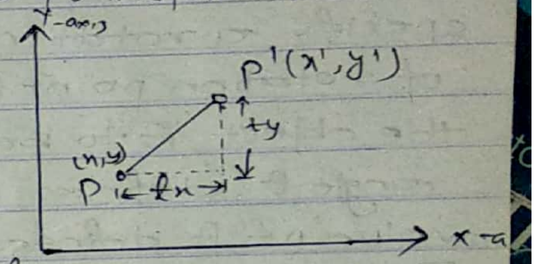


fig: Translation of a pt P from P to P' with translating vector T.

Q. Translate the given point (2,5) by the translating value (3,3)

soln: $P(x, y) = (2, 5)$

$t_x = 3$

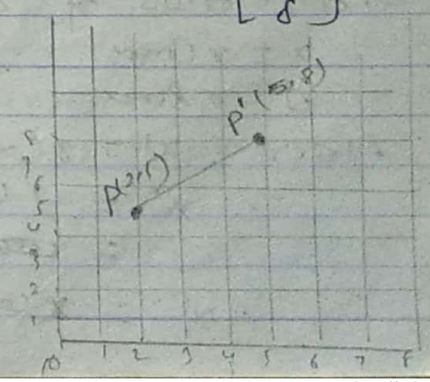
$t_y = 3$

$x' = x + t_x = 2 + 3 = 5$

$y' = y + t_y = 5 + 3 = 8$

or $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$

$\therefore P'(x', y') = (5, 8)$



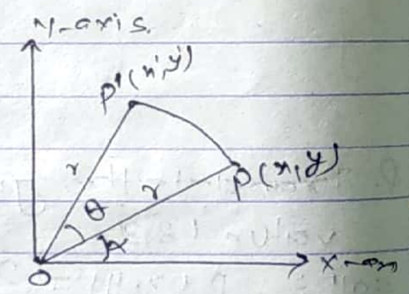
* Rotation :-

A two dimensional rotation is applied to an object by repositioning it along a circular path in xy plane. In other words, changing the co-ordinate position along a circular path is called rotation. To generate a rotation, we specify a rotation angle θ and position (x_r, y_r) of rotation point (or pivot point) about which the object is to be rotated. Positive value of rotation angle θ defines anticlockwise rotation and negative value of θ defines rotation in clockwise direction.

There are two types of rotation

(1) About origin

(2) About any point.



(1) About Origin:

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

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

$$x = r\cos\alpha$$
$$y = r\sin\alpha$$

Also, $x' = r\cos(\theta+\alpha) = r(\cos\theta\cos\alpha - \sin\theta\sin\alpha)$

$$\therefore x' = r\cos\theta\cos\alpha - r\sin\theta\cdot\sin\alpha$$
$$= r\cos\alpha\cos\theta - r\sin\alpha\sin\theta$$
$$\therefore x' = x\cos\theta - y\sin\theta \qquad [\because x = r\cos\alpha,\ y = r\sin\alpha]$$

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

In matrix form

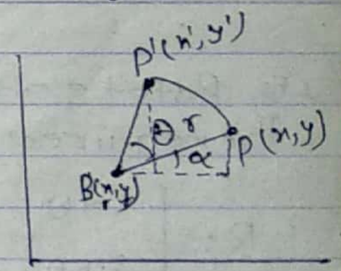
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$\therefore P' = RP$

where $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

(2) About any point:

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



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

Algorithm

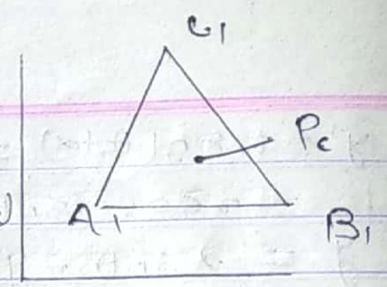
i. Translate the object so that the pivot position is moved to the origin.

ii. Rotate the object about the origin.

iii. Translate the object so that pivot point is returned to original position.

Steps in the above sequence of transformation are as follows:

Scanned by CamScanner

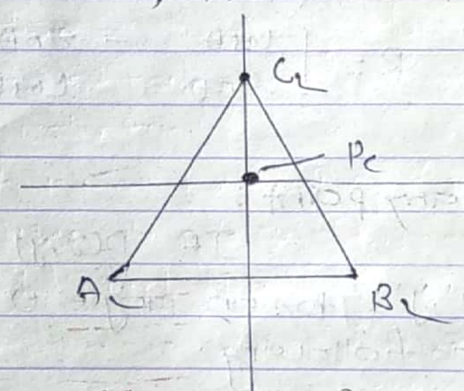


1. Initial position of object A_1, B_1, C_1 & pivot point $P_c(x_c, y_c)$

ii. Translate the $P_c(x_c, y_c)$ to origin. The required translation matrix is,

$$T_1 = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$

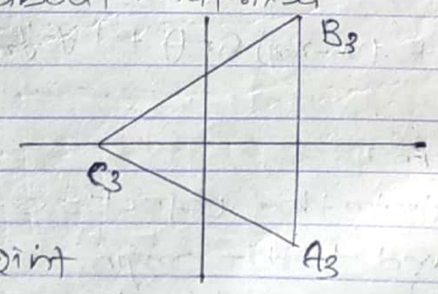
Therefore translate A_1, B_1, C_1 to A_2, B_2, C_2



iii. Rotate about origin by angle θ

The required rotation about matrix is

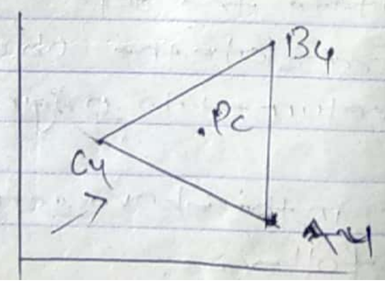
$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



iv. Translate the pivot point to $P_c(x_c, y_c)$.

The required translation matrix is

$$T_2 = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix}$$



The composite transformation matrix is

$$= T_2 R T_1$$

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

Any point P on the object will be transferred to P' such that

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

$$\therefore A_4 = R(\theta) \cdot A_1$$

$$B_4 = R(\theta) \cdot B_1$$

$$C_4 = R(\theta) \cdot C_1$$

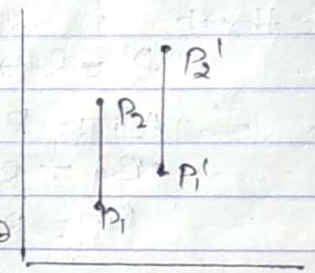
Scanned by CamScanner

* Scaling:

A scaling transformation alters the size of an object. Transformed objects are scaled and repositioned.

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

Consider a line segment P_1P_2 in two-dimensional co-ordinate system.



Scaling operation is obtained by multiplying co-ordinate values (x, y) of each point by scaling factors s_x and s_y to produce the transformed co-ordinates (x', y') such that.

$$x' = x \cdot s_x$$
$$y' = y \cdot s_y$$

Scaling factors s_x scales object in the x-direction while s_y scales in the y direction.

In matrix form

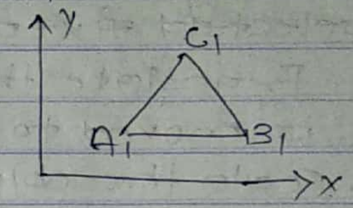
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P_1' = S \cdot P_1$$
$$P_2' = S \cdot P_2$$

where $S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

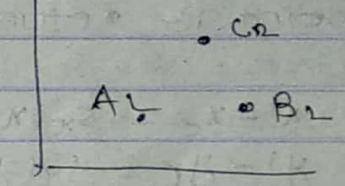
polygones are scaled by applying transformation to each vertex and then regenerating the polygon using the transformed vertices.

i. Consider a polygon $A_1B_1C_1$ (initial position)



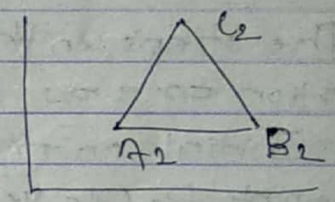
ii. scale A_1 to A_2
B_1 to B_2
C_1 to C_2

such that $A_2 = S \cdot A_1$
$B_2 = S \cdot B_2$
$C_2 = S \cdot A_3$



The required scaling matrix is

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



iii. Draw $A_2B_2C_2$

Now polygon $A_2B_2C_2$ is the result of scaling polygon $A_1B_1C_1$ by factors s_x and s_y in x and y direction resp.

<u>conditions</u>; when $s_x = s_y \leq 1$, reduces the size of object

$s_x = s_y < 1$, moves object closer to the co-ordinate origin

$s_x = s_y > 1$; increase the size of the object

$s_x = s_y > 1$, moves object away from the co-ordinate origin.

Scanned by CamScanner

General fixed point scaling

Algorithm:

// scaling of any object with respect to selected fixed position (x_c, y_c)

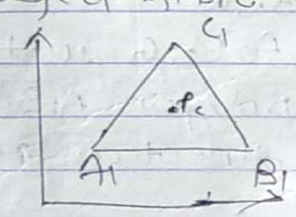
1. Translate the object so that fixed point is moved to the origin.
2. Scale the object w.r.t. the origin.
3. Translate the object so that the fixed point is returned to its original position.

i.e.

$$x' - x_c = S_x(x - x_c)$$
$$y' - y_c = S_y(y - y_c)$$

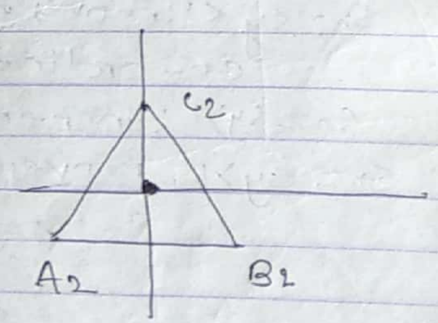
The steps in the above sequence of transformation are as follows:

i. Initial position of object $A_1 B_1 C_1$ and fixed point $P_c(x_c, y_c)$



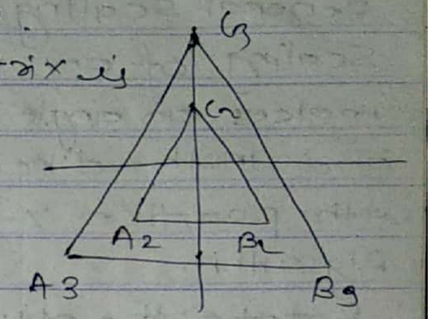
ii. Translate P_c to origin. The required translation matrix is

$$T_1 = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$



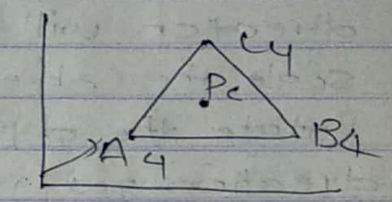
iii. Scale with respect to origin. The required scaling matrix is

$$S_1 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



iv. Translate P_c to (x_c, y_c). The required translation matrix is

$$T_2 = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix}$$



The composite expression for scaling is

$$P' = T_2 S_1 T_1 P$$

The complete transformation matrix is

$$S = (T_2 S_1 T_1)$$

Any point P on the object will get transformed to P' such that,

$$P' = SP$$

$$\therefore A_4 = S \cdot A_1$$
$$B_4 = S \cdot B_1$$
$$C_4 = S \cdot C_1$$

$\therefore$ Net transformation

$$T = T_2 S_1 T_1 = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$

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

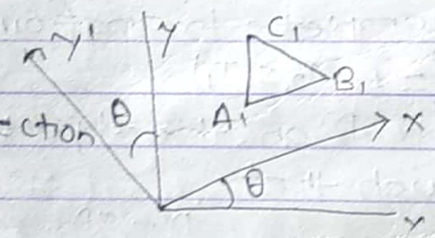
Scanned by CamScanner

General scaling Direction:
Scaling of any object in the direction x' which makes an angle θ with positive x-axis by S_x and in the direction y' that makes angle θ with positive y-axis by S_y.

Algorithm:

i. Rotate the object so that the desired scaling direction will get align with the co-ordinate axis.
ii. Scale the object w.r.t. origin.
iii. Rotate the object so the the desired scaling direction get returned to their original orientation.

Steps in the above sequence of transformation are as follows:

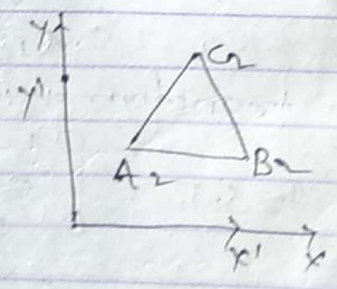


i. Initial position of object $A_1B_1C_1$ and direction

ii. Rotate x' & y' by θ in clockwise direction

The required rotation matrix is

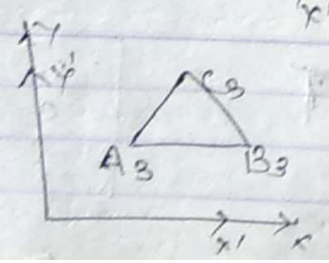
$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



iii. Scale the object

The required scaling matrix is

$$S_1 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



iv. Rotate the x' and y' by an angle θ in anticlockwise direction.

The required rotation matrix is

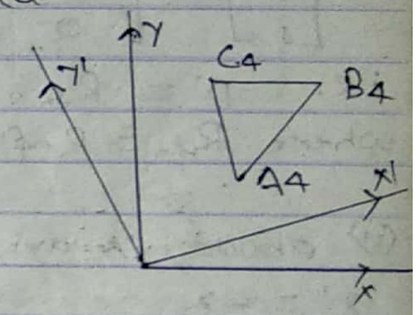
$$R_2 \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The composite transformation matrix is

$$S = R_2 S_1 R_1 = ((R_2 \cdot S_1) R_1)$$

Any point P on the object will get transformed to P' such that,

$$P' = S \cdot P$$
$$\therefore A_4 = S \cdot A_4$$
$$B_4 = S \cdot B_4$$
$$C_4 = S \cdot C_4$$



Now, object $A_4B_4C_4$ is the result of scaling $A_1B_1C_1$ by S_x in x' direction and by S_y in y' - direction.

* Reflection:

A reflection is a transformation that produces a mirror image of an object.

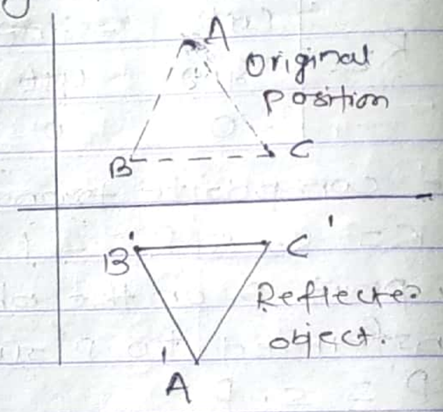
(i) about x-axis ($y=0$)

$x' = x$

$y' = -y$

In matrix form

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



i.e. $P' = R_{fx} \cdot P$

where R_{fx} = Reflection matrix about x-axis.

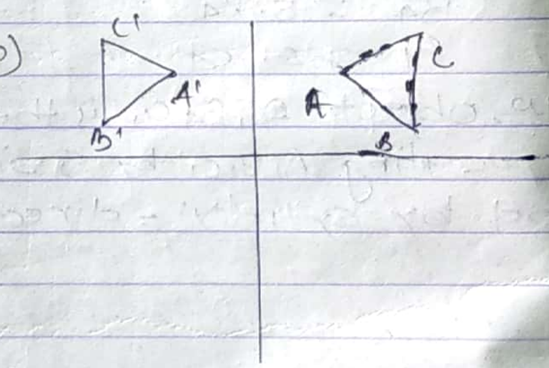
(ii) about y-axis ($x=0$)

$x' = -x$

$y' = y$

In matrix form

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



i.e. $P' = R_{fy} \cdot P$

R_{fy} = Reflection about y-axis.

* Alternative Way for Reflection:

Step I:

Rotate the object so that y-axis concise x-axis i.e. $R(-90)$

Step II

Reflection about y-axis R_{fx}

Step III

Rotate back the object so that y-axis move to

original position. $R(-90)$

Net transformation

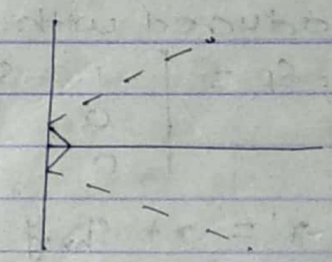
i.e. $R_{fy} = R(90) \cdot R_{fx} \cdot R(-90)$

$$= \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90 & \sin 90 & 0 \\ -\sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

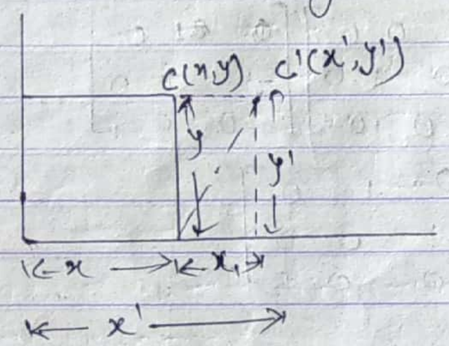
$$= \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Scanned by CamScanner

* Shearing:

~~Shear~~ Transformation that causes deformation in the original object by supposing as if the object is composed of internal layers and these layers are caused to slide over is shearing.



A x-direction shear relative to the x-axis is produced with

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

$x' = x + Sh_x \cdot y$

$y' = y$

In matrix form

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

Similarly about y-axis

$x' = x$ &

$y' = y + Sh_y \cdot x$

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

i.e.

$P' = Sh_y \cdot P$

* Inverse Transformation:

for inverse ~~transformation~~ translation

$T^{-1}(t_x, t_y) = T(-t_x, -t_y)$

for inverse Rotation

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

for inverse scaling

$S^{-1}(S_x, S_y) = S(1/S_x, 1/S_y)$

Q. Prove that two successive translation are additive. i.e. $T(t_{x1}, t_{y1}) \cdot T(t_{x2}, t_{y2}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$

Soln:-

Let P be the poin translated by $T(t_{x1}, t_{x2})$ to point P

then, $P' = T(t_{x1}, t_{x2}) \cdot P$

Let P' be the point translated by $T(t_{x2}, t_{y2})$ to pt P'' the

$P'' = T(t_{x2}, t_{y2}) \cdot P'$

$= T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1}) \cdot P$

Net transformation $= T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1})$.

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

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

$= T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$

$\therefore T(t_{x1}, t_{y2}) \cdot T(t_{x1}, t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$

This proves that two successive translation are additive.

Scanned by CamScanner

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

soln:

Let P be the point rotated by $R(\theta_1)$ to P' then

$P' = R(\theta_1) \cdot P$

Let P' the point rotated by $R(\theta_2)$ to point P'' then

$P'' = R(\theta_2) \cdot P'$

$= R(\theta_2) \cdot R(\theta_1) \cdot P$

$\therefore$ Net transformation $= R(\theta_2) \cdot R(\theta_1)$

$P'' = R(\theta_1+\theta_2) \cdot P$

Q. Prove that two successive scalings are multiplicative i.e. $S(S_{x2}, S_{y2}) \cdot S(S_{x1}, S_{y1}) = S(S_{x1}S_{x2}, S_{y1}S_{y2})$

soln:

Concatening transformation matrices for two successive scaling operations produces the following composite scaling matrix.

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

or $S(S_{x2}, S_{x1}) S(S_{x2}, S_{y2}) = S(S_{x1}S_{x2}, S_{y1}S_{y2})$

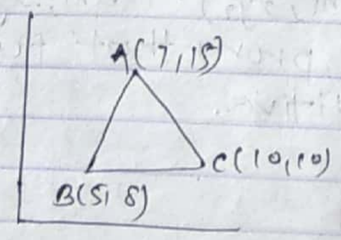
Q. Rotate ΔABC by 45° clockwise about origin & scale it by (2,3) about origin.

soln:

Step I : Rotation by 45° (clockwise)

Step II : scaling by (2,3)

$S(2,3)$



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

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

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

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

The transformation points are : $A' = TA$

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

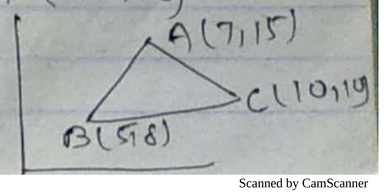
$B' = TB$

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

$C' = TC$

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

Q. Rotate ΔABC by 90° anticlockwise about (5,8) and scale it by (2,2) and about (10,10)



Scanned by CamScanner

Soln:- Step I: ~~Rotate by 90°~~

Step II:

$T(-5,-8)$

Step II

$R(90)$

Step III

$T(5,8)$

Step IV, $T(-10,-10)$

Step V, $S(2,2)$

Step VI, $T(10,10)$

Net transformation:

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

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -8 \\ 0 & 0 & 1 \end{bmatrix}$$

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

(i) $R(-45)$

(ii) R_{fx}

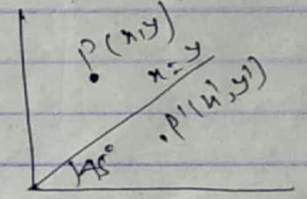
(iii) $R(45)$

Net transformation

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

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

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



i.e. $x' = y$

$y' = x$

Q. Determine transformation matrix responsible for reflection of about the line $y + x = 0$

Soln:-

(i) $R(45^\circ)$

(ii) R_{fx}

(iii) $R(-45^\circ)$

Net transformation $= R(-45^\circ)\,R_{fx}\,R(45^\circ)$

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

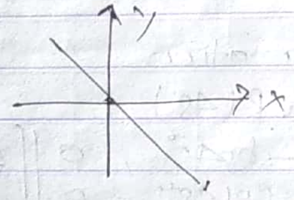
Scanned by CamScanner

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

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

i.e. $x' = -y$

$y' = -x$



(a) Right handed (b) left handed

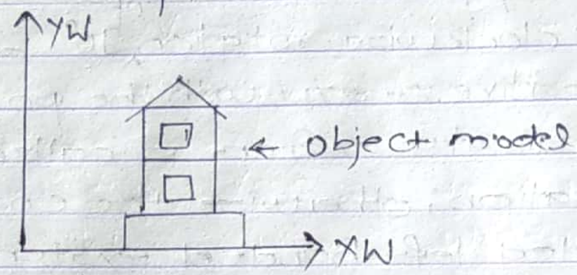
→ The word-co-ordinate system is the right handed co-ordinate system. The object model that resides in object space is described in WCS. This model represents the objects in the physical units of length. The WCS is theoretically infinite in extent.

→ The physical device-co-ordinate system is the co-ordinate system that corresponds to the device or workstation where image or picture of the object model is to be displayed.

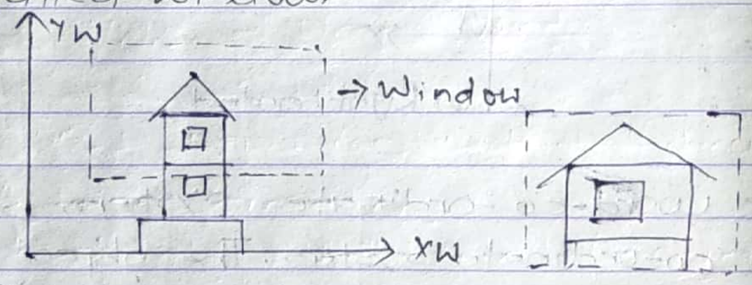
→ The nomalized device co-ordinate (NDC) system is a ~~device~~ right handed co-ordinate system in which the display area of the virtual display device corresponds to the unit square.

Scanned by CamScanner

Windowing Concepts:



A world co-ordinate area selected for display is called window.



The window can be directly mapped onto the display area or onto a subregion of display device. An area on a display device to which a window is mapped is called a viewport. The window defines what is to be viewed, the viewport defines where it is to be displayed.

If a window is changed a different part of the object is displayed at the same position on the screen.

If the viewport is changed, the same portion of the object model is displayed at different position on the display device.

* Window To Viewport Transformation:

The mapping of a word co-ordinate to device co-ordinate is referred to as viewing transformation. This is the transformation from window to viewport.

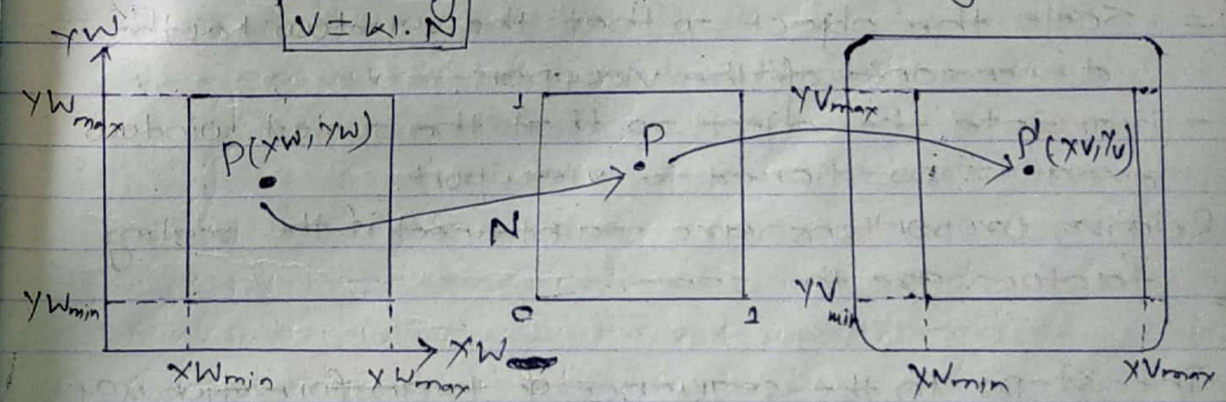
The viewing transformation is formed by the following transformations.

The normalized transformation N, that maps world co-ordinates to Normalized device co-ordinates.

The workstation transformation W, that maps normalized device co-ordinates to ~~Norm~~ device co-ordinates.

The viewing transformation is given as

$$\boxed{V = W \cdot N}$$



consider a point P(XW, YW) in WCS mapped to P' (XV, YV) in the device co-ordinate system then

$$\frac{XV - XV_{min}}{XV_{max} - XV_{min}} = \frac{XW - XW_{min}}{XW_{max} - XW_{min}} \quad \text{—(i)}$$

$$\frac{YV - YV_{min}}{YV_{max} - YV_{min}} = \frac{YW - YW_{min}}{YW_{max} - YW_{min}} \quad \text{—(ii)}$$

Scanned by CamScanner

from eqns (i) & (ii).

$$xV = xV_{min} + (xW - xW_{min})\, S_x \quad \text{—(iii)}$$

$$yV = yV_{min} + (yW - yW_{min})\, S_y \quad \text{—(iv)}$$

where,

$$S_x = \frac{xV_{max} - xV_{min}}{xW_{max} - xW_{min}} = \frac{\text{Viewport x extent}}{\text{Window x extent}}$$

$$S_y = \frac{yV_{max} - yV_{min}}{yW_{max} - yW_{min}} = \frac{\text{Viewport y extent}}{\text{Window y extent}}$$

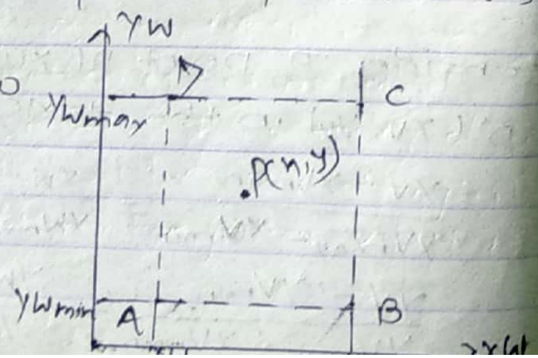
This conversion is performed with the following sequence of transformation.

- Translate the object so that lower left corner of window is moved to origin.
- Scale the object so that the window has the dimensions of the viewport.
- Translate the object so that the scaled window area is positioned to viewport.

Relative proportions are maintained if the scaling factors are the same.

The steps in the sequence of transformation is as follows:

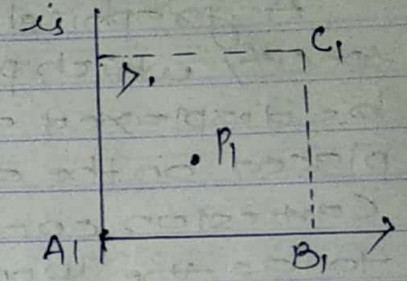
Initial position of window



Step I :

Translate (xW_{min}, yW_{min}) to origin the required translated matrix is

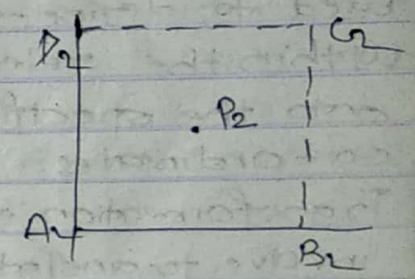
$$T_1 \begin{bmatrix} 1 & 0 & -xW_{min} \\ 0 & 1 & -yW_{min} \\ 0 & 0 & 1 \end{bmatrix}$$



Step II

scale w.r.t. Origin. The required matrix is

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

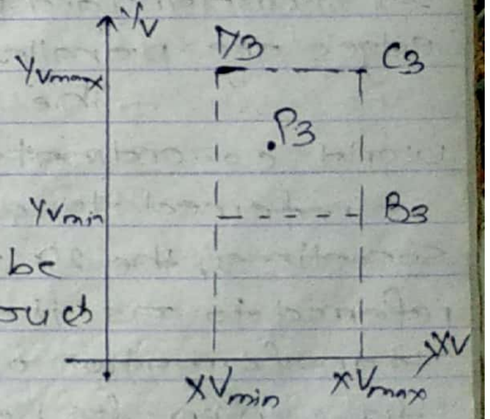


Step III

Translate to (xV_{min}, yV_{min})

The required translation matrix is

$$T_2 \begin{bmatrix} 1 & 0 & xV_{min} \\ 0 & 1 & yV_{min} \\ 0 & 0 & 1 \end{bmatrix}$$



The composite transformation matrix is, $V = ((T_2 \cdot S) \cdot T_1)$

Any point P on the object will be transformed to $P'(x', y', 1)$ such that,

$$P' = V \cdot P.$$

Scanned by CamScanner

* Mechanism for displaying view of a picture on an output device :

A graphical package allows a user to specify which part of the defined picture is to be displayed and where that part is to be placed on the display device. Any convenient Cartesian co-ordinate system is referred to as the world co-ordinate reference frame can be used to define the picture. The picture parts within the selected area are then mapped onto the specified areas of the device co-ordinates.

Transformation from world to device co-ordinates involve translation, rotation and scaling operations as well as clipping.

Often, windows and viewports are rectangles in standard positions, if the rectangle edges are parallel to the co-ordinate axes.

The mapping of a part of a world co-ordinate scene to a device co-ordinate is referred to as viewing transformation. Sometimes, the 2D viewing transformation is simply referred to as the window-to-viewport transformation or the windowing transformation.

fig. given below illustrates the mapping of a picture selections that falls within a rectangular window onto a designated rectangular viewport.

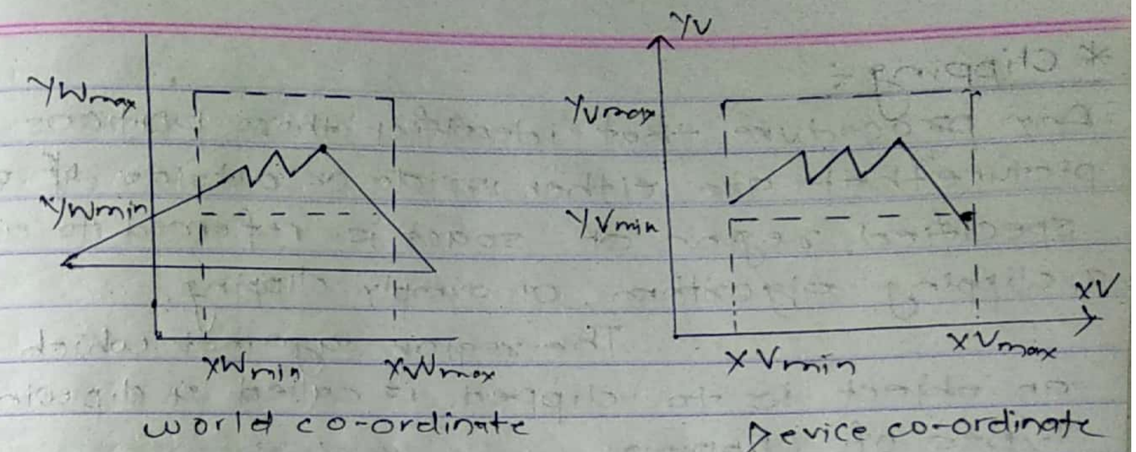


fig: A viewing transformation using standard rectangles for the window and viewport.

Steps to be followed for window to viewport transformation are

- Generate world co-ordinate
- convert world co-ordinate to view co-ordinate.
- Map the ~~world~~ viewing co-ordinate to normalized viewing co-ordinates using window-viewport specification
- Map the normalized viewing co-ordinate to device co-ordinate system.

The steps can be illustrated through a pipeline

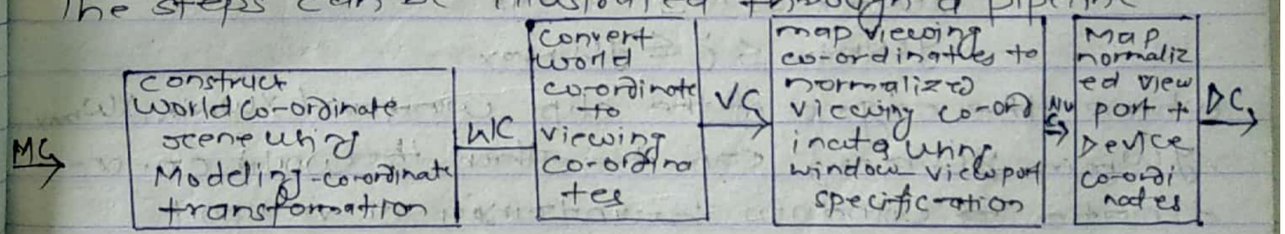
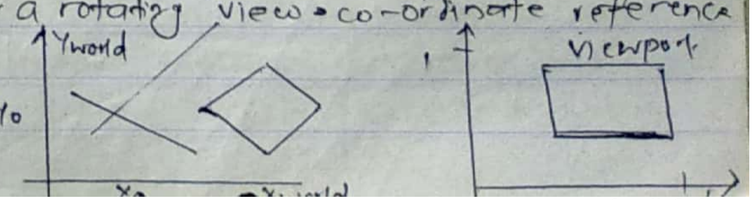
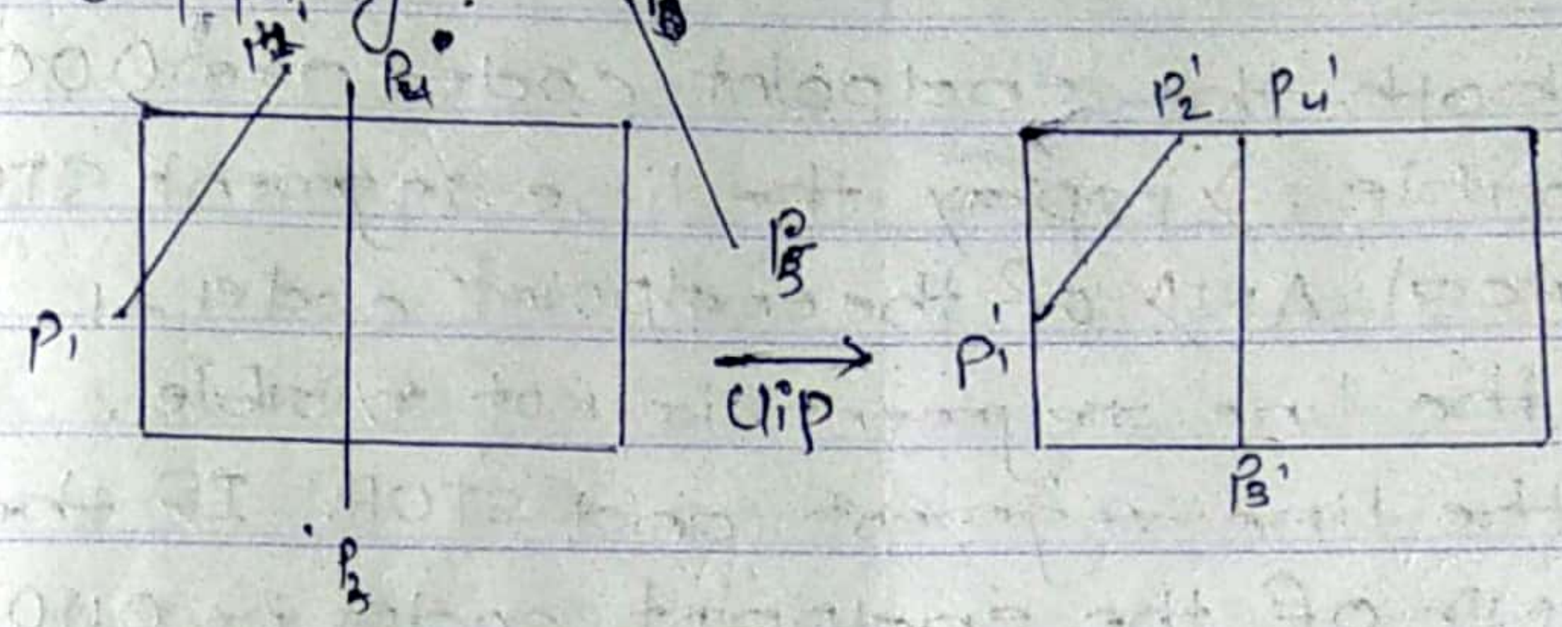


fig: The 2D viewing - transformation pipeline

fig given below show a rotating view-co-ordinate reference frame and mapping to normalized co-ordinates



* Line clipping :



* <u>Cohen Sutherland Line Clipping</u> :

Clipping algorithm

<u>Step I</u>

Accept the end points of line segment and window boundaries i.e. $x_1, y_1, x_2, y_2, x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax}$

<u>Step II</u>

Assign a 4 bit code to each end point of line segments i.e. $B_4 B_3 B_2 B_1$ where

$B_1 = \text{sign}(x_{min} - x)$: ←

$B_2 = \text{sign}(x - x_{max})$: →

$B_3 = \text{sign}(y_{min} - y)$: ↓

$B_4 = \text{sign}(y - y_{max})$: ↑

such that : $\text{sign}\, A = 1$ if $A > 0$

$= 0$ otherwise

The code is determined according to which of the following nine regions of the plane the endpoint lies in

1001	1000	1010
0001	0000	0010
0101	0100	0110

Step III :

If both the endpoint codes are 0000, the line is visible. Display the line segment STOP. If the logical AND of the endpoint codes is not 0000, the line segment is not visible. Discard the line segment and STOP. If the logical AND of the endpoint codes is 0000, the line segment is clipping candidate.

Step IV

Determine the intersecting boundary

If bit B_1 is 1, intersect with $x = xw_{min}$

If bit B_2 is 1, intersect with $x = xw_{max}$

If bit B_3 is 1, intersect with $y = yw_{min}$

If bit B_4 is 1, intersect with $y = yw_{max}$

Step V

Determine the intersecting point coordinates (x', y'). The equation of a line passing through (x_1, y_1) & (x_2, y_2) and (x', y') will be

$$\frac{x' - x_1}{x_2 - x_1} = \frac{y' - y_1}{y_2 - y_1}$$

$$x' - x_1 = \left(\frac{x_2 - x_1}{y_2 - y_1}\right)(y' - y_1) \qquad m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$x' = x_1 + \frac{1}{m}(y' - y_1) \quad \text{—(1)} \qquad y' = yw_{min} \text{ or } y' = yw_{max}$$

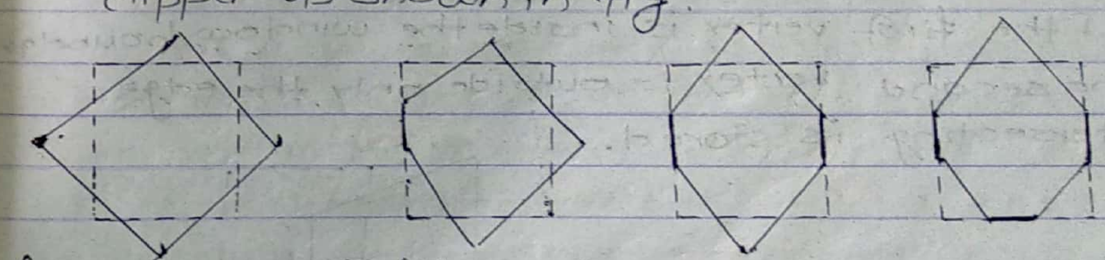
Similarly, $y' - y_1 = m(x' - x_1)$ then $x' = xw_{min}$ or $x' = xw_{max}$

Step VI : Goto Step II

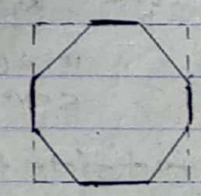
* Sutherland - Hodgman Polygon Clipping :

A polygon is clipped by processing the entire figure against each window boundary. The clipping sequence is as follows;

- The initial set of polygon vertices is clipped against the left rectangle boundary. This produces a new sequence of vertices.
- The set of vertices are passed to a right boundary clipper, a bottom boundary clipper and a top boundary clipper. At each step a new sequence of output vertices is generated and passed to the next window boundary clipper as shown in fig.



fig(a) Original polygon (b) Clip left (c) Clip Right (d) Clip bottom



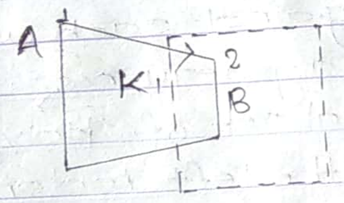
(e) Clip Top

Algorithm to generate net set of output vertices at each boundary clipping.

Each pair of adjacent polygon vertices is passed to a window boundary clipper.

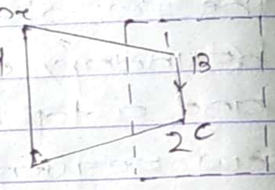
1. If the first vertex is outside the window boundary, and second vertex is inside the

window boundary both the intersecting point of the polygon edge and the second vertex are stored.



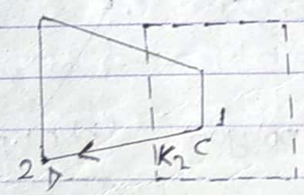
Out → in save K_1, B

2. If both input vertices are inside the window boundary only the second vertices is stored.



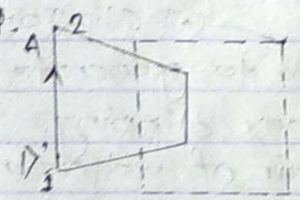
in → in
save C

3. If the first vertex is inside the window boundary and second vertex is outside only the edge intersecting is stored.



in → out
save → K_2

4. If both input vertices are outside the window boundary nothing is stored.

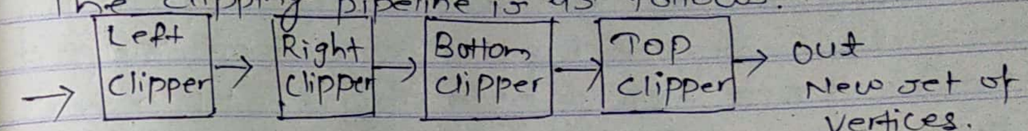


out → out
save none

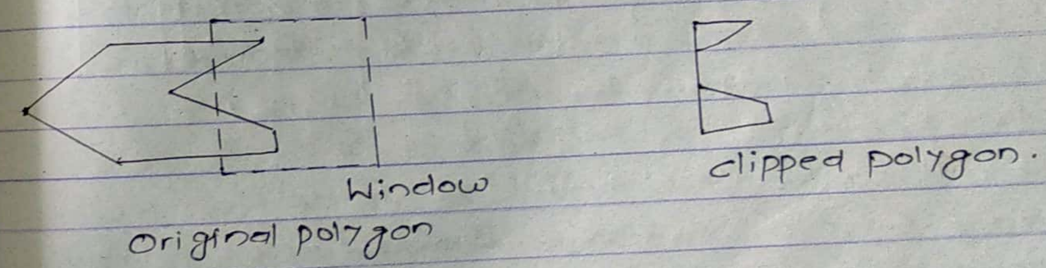
All the stored values give the new set of output vertices

once all vertices has been processed for one clip window boundary the o/p list of vertices is clipped against the next window ~~The clipping~~ o/p vertices

The clipping pipeline is as follows.



convex polygons are correctly clipped by the Sutherland-Hodgman algorithm. But concave polygons with the same ~~poly~~ algorithm produces the connected areas as shown in fig. This occurs when the clipped polygon have two or more separate sections.



Extension of 2D transforms to 3D

Just as 2D - transformation can be represented by 3×3 matrices using homogeneous co-ordinate can be represented by 4×4 matrices, provided we use homogeneous co-ordinate representation of points in 3D space as well.

TRANSLATION:

In 3D co-ordinate representation, a point is translated from position $P(x, y, z)$ to position $P'(x', y', z')$ by adding translation vector $T(t_x, t_y, t_z)$ to point $P(x, y, z)$

Such that

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

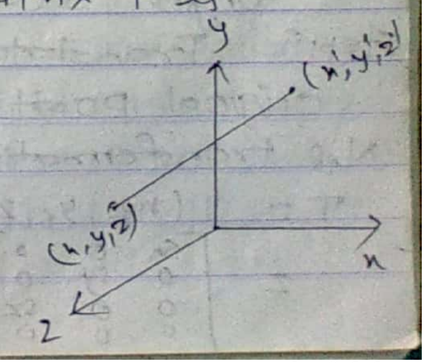
$$\therefore P' = P + T$$

The homogeneous 3D co-ordinate transformation matrix T is given by

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

The inverse transformation matrix T^{-1} is,

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



i.e. $T^{-1}(t_x, t_y, t_z) = T(-t_x, -t_y, -t_z)$

SCALING:

⟨1⟩ About Origin:

Explicit expression

$$x' = x * S_x$$
$$y' = y * S_y$$
$$z' = z * S_z$$

The matrix representation for the scaling x-formation of a position $p(x, y, z)$ relative to the co-ordinate origin.

Matrix form:

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

Where scaling parameters are assigned to positive values.

⟨2⟩ About any fixed point (x_f, y_f, z_f)

Scaling w.r.t to a selected fixed position (x_f, y_f, z_f) can be represented with the following transformation

Step I: Translate the fixed point with the origin

$$T(-x_f, -y_f, -z_f)$$

Step II: scale the object relative to the co-ordinate origin $S(S_x, S_y, S_z)$

Step III: Translate the fixed point back to its original position $T(x_f, y_f, z_f)$

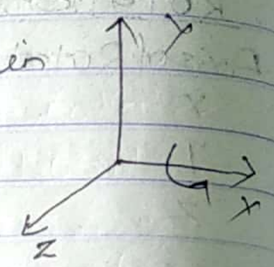
Net transformation

$$T = T(x_f, y_f, z_f) \cdot S(S_x, S_y, S_z) \cdot T(-x_f, -y_f, -z_f)$$

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

Scanned by CamScanner

Rotation about x-axis



Rotation about x-axis by an angle θ in anticlockwise direction will shift point $P(x, y, z)$ to $P'(x', y', z')$

All x-coordinate will remain unchanged. The new co-ordinates will then be

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

The matrix representation for x-axis rotation is

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

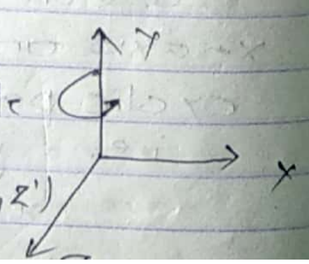
then

$$P' = R_x \cdot P$$

Inverse rotation about x-axis i.e. rotation about x-axis in clockwise direction is obtained by replacing θ with −θ such that

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

Rotation About y-axis



Rotation about y-axis by an angle θ in anticlockwise direction will move point $P(x, y, z)$ to $P'(x', y', z')$

All the y-co-ordinate will remain unchanged. The new co-ordinates will then be,

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

The matrix representation for y-axis rotation is,

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

Rotation About Any co-axis:
(General 3D Rotation)

If the axis of rotation is other than co-ordinate axis then rotation about any given axis is obtained by using the composite transformation matrix. The required transformation matrix is obtained by setting up the transformation sequence that -

- moves the selected rotation axis onto one of the co-ordinate axis.
- rotates about the same co-ordinate axis.
- Returns the selected axis onto its original axis.

<I> Rotation about an axis that parallel to one of the co-ordinate axis:

Algorithm:

i. Translate the object so that the rotation axis coincides with the parallel co-ordinate axis.

Scanned by CamScanner

The composite matrix for the required transformation is

$$R(\theta) = ((T_2 . R_x) T_1)$$

Any point P on the object will get transformed to P' such that

$$P' = R(\theta) . P$$

New transformed object will be the result of rotation about an axis parallel to x-axis.

(II) Rotation about an arbitrary axis (Not parallel to any of the co-axis)

Algorithm:

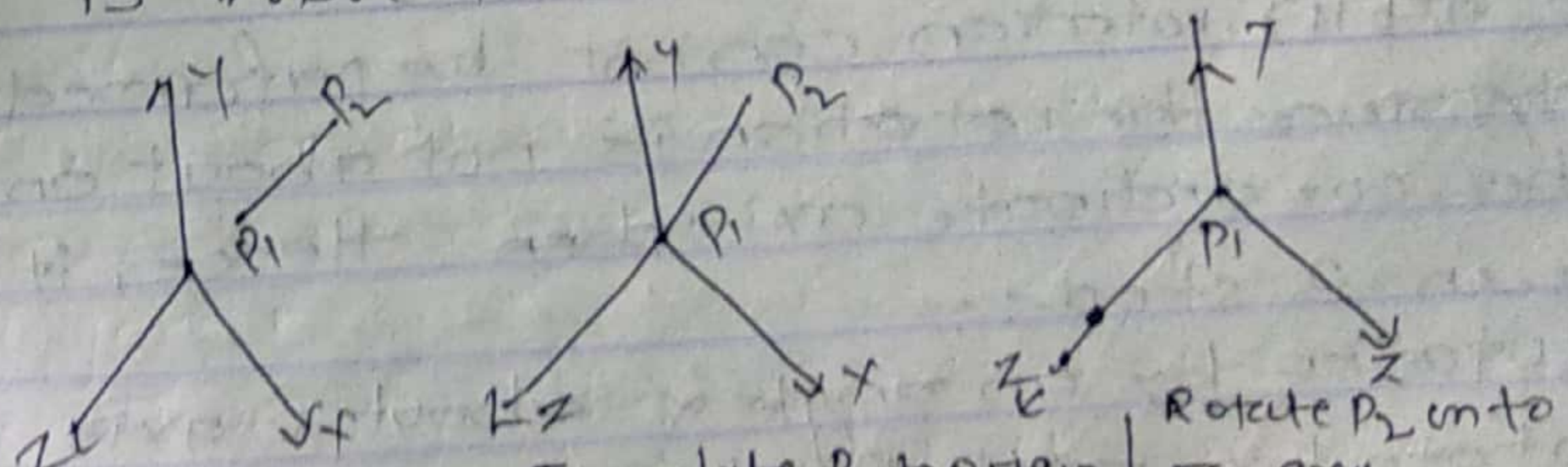
i. Translate the object, so that the rotation axis passes through the co-ordinate origin.

ii. Rotate the object so that axis of rotation coincides with one of co-ordinate axis.

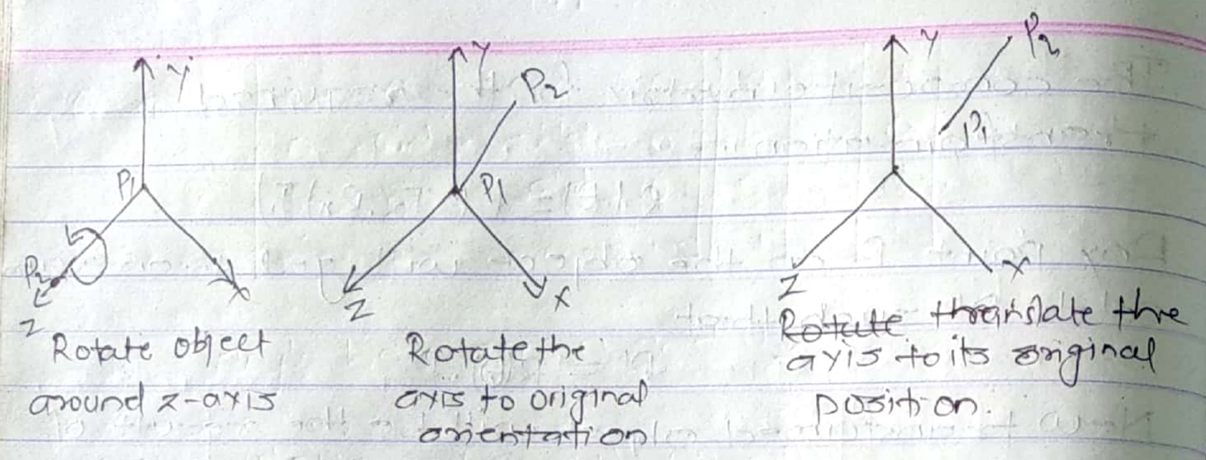
iii. Perform the specified rotation about the same co-ordinate axis.

iv. Rotate the object, so that rotation axis is move back to its original orientation passing through original.

v. Translate the object, so that rotation axis is moved back to its original position.



Initial position Translate P1 to origin



Let P_1, P_2 be the two end points of axis of rotation then the axis vector is

$$V = P_2 - P_1 = \{(x_2, y_2, z_2) - (x_1, y_1, z_1)\}$$
$$= \{x_2 - x_1,\ y_2 - y_1,\ z_2 - z_1\}$$

Then unit vector along rotation axis is

$$u = \frac{V}{|\vec{V}|} = \frac{(x_2 - x_1,\ y_2 - y_1,\ z_2 - z_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}$$

$$= (a, b, c) \quad \text{(let)}$$

where (a, b, c) are direction cosines of rotation axis i.e. $a^2 + b^2 + c^2 = 1$

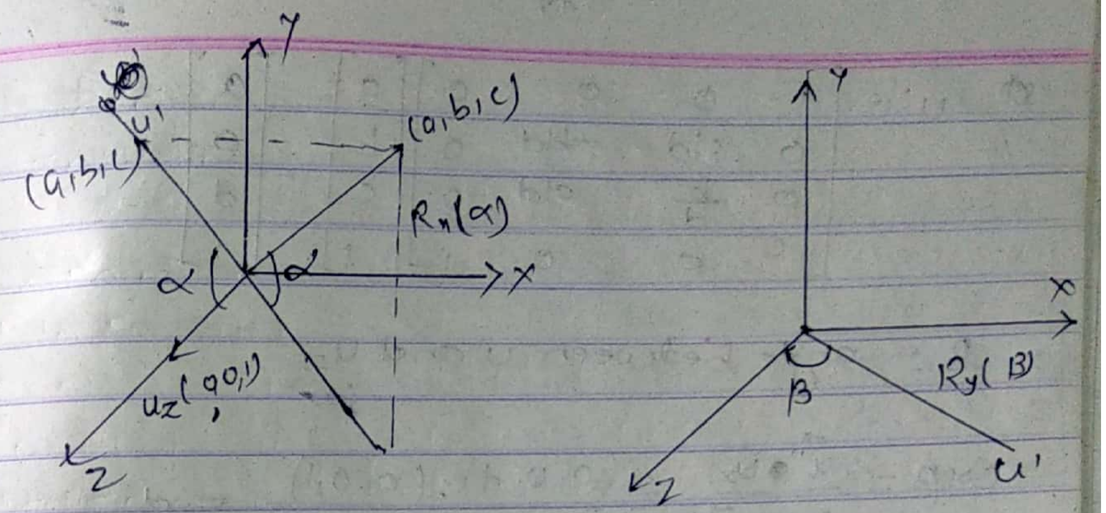
(i).

$$T(-x_1, -y_1, -z_1) = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) for step ii) rotation can not be performed directly since the rotation is not about any of the co-ordinate axis ~~then~~. Hence, it is done in 2 steps:

(a) Rotation by an angle α about x-axis.

(b) Rotate by angle β about



Step(a)

Angle for rotation of u around x-axis into xz plane is equal to angle between u' (projection of u on yz plane and z-axis)

$$\cos\alpha = \frac{\vec{u_n} \cdot \vec{u_z}}{|\vec{u_n}||\vec{u_z}|} = \frac{(0, b, c) \cdot (0, 0, 1)}{\sqrt{b^2 + c^2}} = \frac{c}{d}$$

$$\sin\alpha = \frac{\vec{u_n} \times \vec{u_z}}{|\vec{u_n}||\vec{u_z}|} = \frac{(0, b, c) \times (0, 0, 1)}{\sqrt{b^2 + c^2}} = \frac{b}{d}$$

where $d = \sqrt{b^2 + c^2}$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step(b)

u moves to position u' after being rotated by α about x-axis so the position of u_a is given by

$$u' = R_x(\alpha) \cdot u$$

Scanned by CamScanner

$$u' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & \frac{b}{d} & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ 0 \\ d \\ 1 \end{bmatrix}$$

β = angle between u' and u_z

$$\cos\beta = \frac{\vec{u''} \cdot \vec{u_z}}{|u''||\vec{u_z}|} = \frac{(a,0,d)\cdot(0,0,1)}{\sqrt{a^2+d^2}} = d$$

$$\sin\alpha = \frac{\vec{u''} \times \vec{u_z}}{|\vec{u''}||\vec{u_z}|} = \frac{(a,0,d)\times(0,0,1)}{\sqrt{a^2+d^2}} = -a$$

$$\therefore R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step: 4

(a) $R^{-1}_{y}(\beta)$

(b) $R^{-1}_{x}(\alpha)$

Step 5:

$T(x_1, y_1, z_1)$

$\therefore$ Net transformation

$= T(x_1,y_1,z_1) \cdot R^{-1}_x(\alpha) \cdot R^{-1}_y(\beta) R_z(\theta) \cdot R_y(\beta) \cdot R_z(\alpha) \cdot T$

* Reflection:

(1) About Axis:

Reflection about axis is equivalent to $180°$ rotation about the axis.

About z-axis:

$$R_{fz} = R_z(180°) = \begin{bmatrix} \cos 180° & -\sin 180° & 0 & 0 \\ \sin 180° & \cos 180° & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

About x-axis

$$R_{fx} = R_x(180°)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 180° & -\sin 180° & 0 \\ 0 & \sin 180 & \cos 180 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

About y-axis:

$$R_{fy} = R_y(180°)$$

$$= \begin{bmatrix} \cos 180 & 0 & \sin 180° & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 180° & 0 & \cos 180 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

(ii) About any Axis:

Net transformation:

$T(x_1, y_1, z_1)\, R_x(-\alpha)\, R_y(-\beta) : R_z\, R_y(\beta) \cdot R_x(\alpha) \cdot T(-x_1, -y_1, -z_1)$

(iii) About plane:

$x' = x,$

$y' = y$

$z' = z.$

(a) About xy plane (z - axis)

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

(b) About xz plane (y - axis)

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

(c) About yz plane (x - axis):

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

About any plane:

step1: Translate the plane such that it passes through origin i.e. normal vector passes through origin.

step2: Rotate the plane such that normal vector lies one of the co-ordinate axis.

step3: Perform reflection about the plane whose normal vector is one of the co-ordinate axis.

step4: Rotate back the plane such that normal vector takes its original orientation.

step5: Translate back the plane to its original position.

* SHEARING:

In 2D, shearing about x-axis means x-values changes by amount proportional to y-values and y-values remains same. However, in 3D, z-axis means x & y value change by amount proportional to z-value and z-value remain same i.e.

$x' = x + Sh_x \cdot z$

$y' = y + Sh_y \cdot z$

$z' = z$

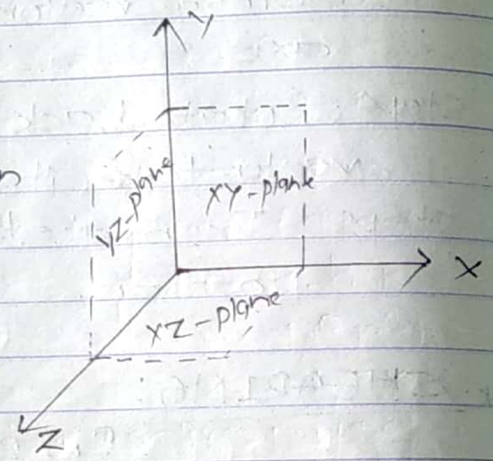
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & Sh_x & 0 \\ 0 & 1 & Sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Similarly,

$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_x & 1 & 0 & 0 \\ Sh_y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad SH_z = \begin{bmatrix} 1 & Sh_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & Sh_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

* Three Dimensional Object to screen perspective viewing Transformation:

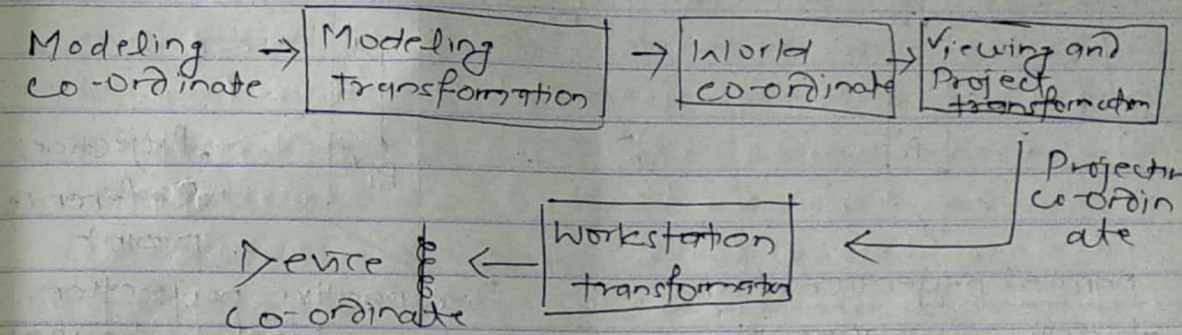


The 3D viewing process is inherently more complex than in the 2D viewing process. In 2D, we simply specify the window on 2D world and a viewport on the 2D-view surface. Conceptually objects in the world are clipped against the window and are then transformed into the viewport for display. The extra complexity in 3D-viewing is caused in the part by the added dimensions and in part by the fact that display device are only 2D.

The solution is mismatch between 3D object and 2D display accomplished by introducing projections which transform 3D objects onto a 2D projection plane.

In 3D-viewing we specify a new volume in world a projection onto a projection plane and a viewport on the viewsurface. Conceptually objects in the 3D new volume and are then projected. The content of the projection of the new volume onto the projection plane, called the window, are then displayed (transferred into the view port for display)

figure below shows the conceptual model of the 3D-viewing process.

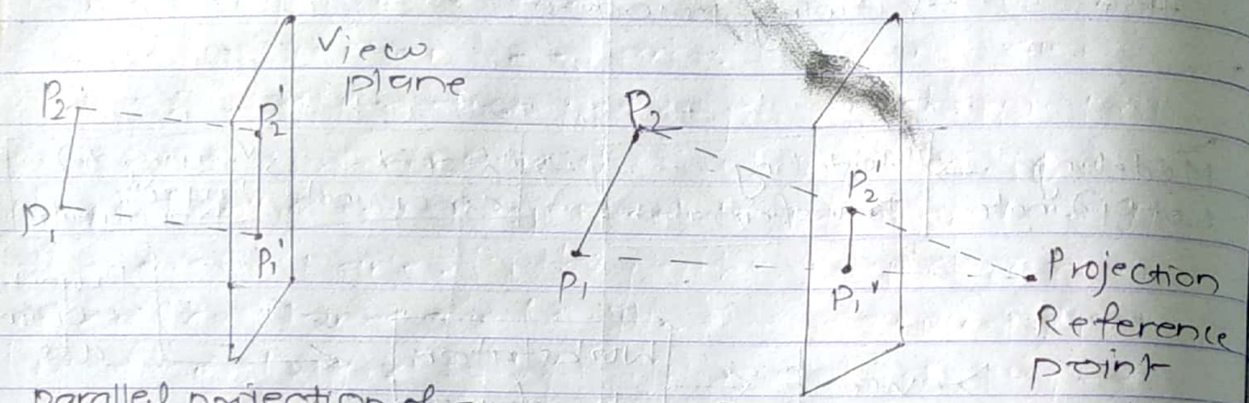


* PROJECTIONS:

Once the world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the 3D objects onto the 2D view plane. There are two ~~types~~ basic projection methods. In a parallel projection, coordinates positions are transformed to view plane along parallel lines, as shown in the eg. fig. for a perspective projection, objects positions are transformed to the view plane along lines that converge to a point called the projection reference point (or center of projection). The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.

Scanned by CamScanner

* Projection:

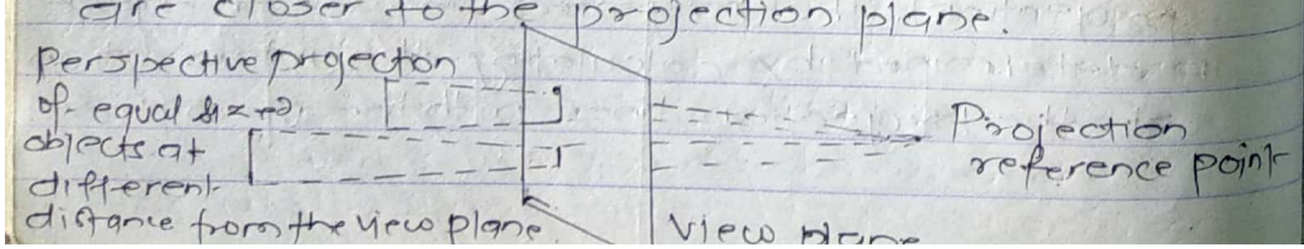


parallel projection of an object to the viewplane

Perspective projection of an object to the view plane

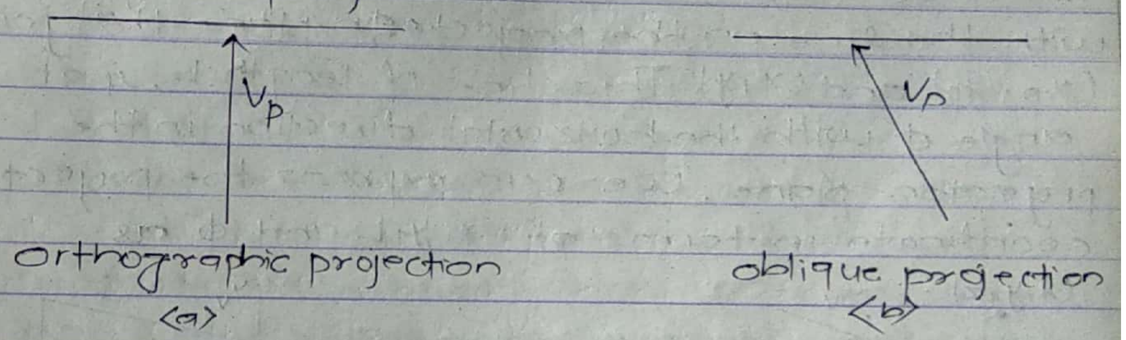
A parallel projection preserves relative proportions of object, and this is the method used in drafting to produce scale drawings of 3D objects. Accurate view of the various sides of an object are obtained with a parallel projection, but this does not give us a realistic representation of the appearance of a 3D object.

A perspective projection, on the other hand produces realistic views but does not preserve relative proportions. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.



* Parallel Projections:

We can specify a parallel projection with a projection vector that defines the direction for the projection line. When the projection is perpendicular to the view plane, we have an orthographic parallel projection. Otherwise, we have an oblique parallel projection. Fig. illustrates the two types of parallel projections.



Orientation of the projection vector V_p produces an orthographic p

Transformation equation for an orthographic projection are straightforward. If the view plane is placed at position z_{vp} along axis as shown in fig below. Then any point (x, y, z) in viewing co-ordinates is transformed to projection co-ordinate as

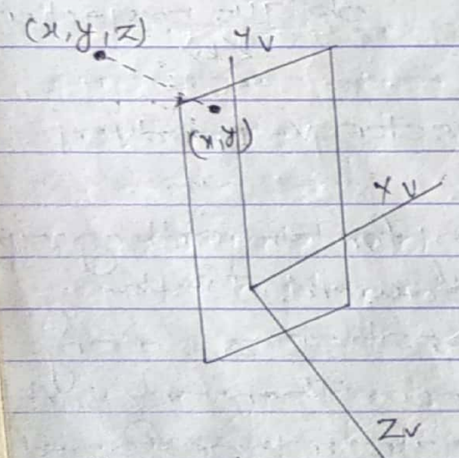
$$x_p = x$$
$$y_p = y$$

where the original z-coordinate value is preserved for the depth information needed in depth cueing and visible-surface determination procedure

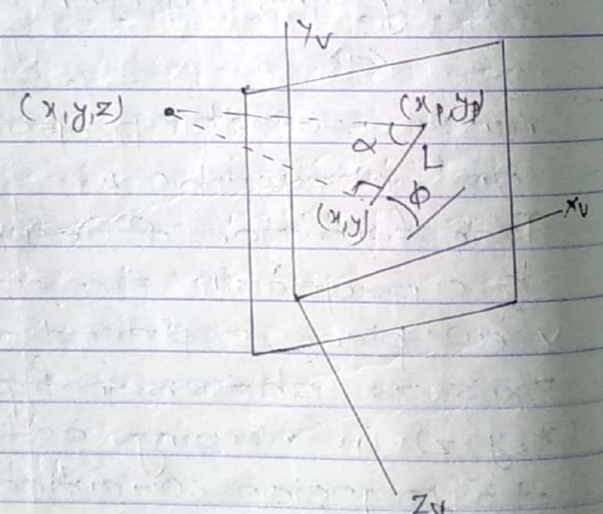
An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection ~~line~~ plane. ~~In some ap~~

In some application packages, an oblique projection vector is specified with two angles α, & ϕ.

Point (x, y, z) is projected to position (x_p, y_p) on the view plane. Orthographic projection co-ordinates on the plane are (x, y). The oblique projection line from (x, y, z) to (x_p, y_p) makes an angle α with the line on the projection plane that joins (x_p, y_p) and (x, y). This line of length L, is at angle ϕ with the horizontal direction in the projection plane. We can express the projection coordinate in terms of x, y, L and ϕ as



orthographic projection of a point onto a viewing plane



Orthographic projection of a point onto a viewing plane.

$$x_p = x + L\cos\phi$$
$$y_p = y + L\sin\phi$$

Length L depends on the angle α and the z co-ordinate of the point to be projected

Thus

$$\tan\alpha = \frac{z}{L}$$

$$L = \frac{z}{\tan\alpha} = zL_1$$

where L_1 is the inverse of $\tan\alpha$, which is also the value of L when $z=1$. We can then write the oblique projection eqn as

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

The transformation matrix for producing any parallel projection onto x_v, y_v plane can be written as

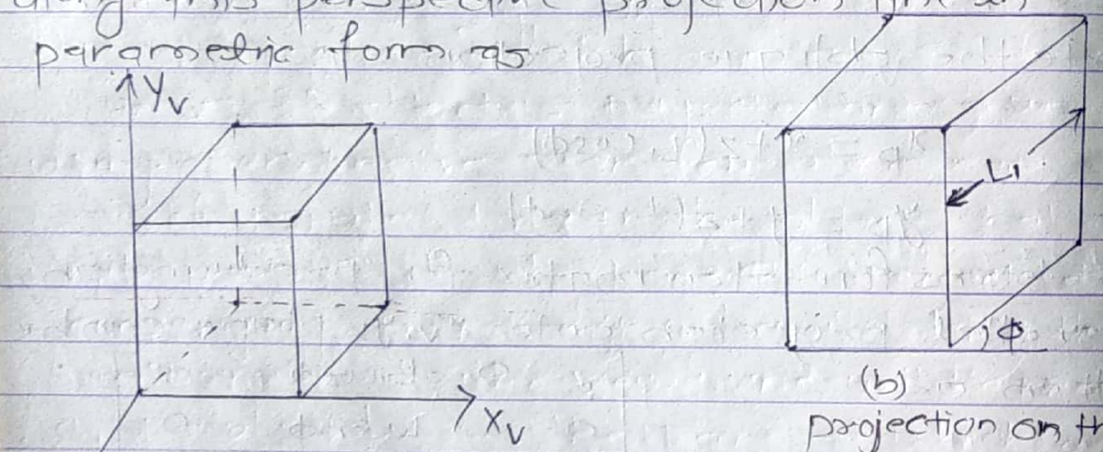
$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

An orthographic projection is obtained when $L_1 = 0$ (which occurs at a projection angle α of $90°$). Oblique projection are generated with nonzero values for L_1

Scanned by CamScanner

Perspective Projection:

To obtain a perspective projection of a 3D object, we transform points along projection lines that meets at the projection reference point. Suppose we set the projection reference point at position z_{prp} along the z_v axis and we place the ~~new~~ view plane at z_{vp} as shown in fig. We can write equations describing coordinate positions along this perspective projection line in parametric form as



Original coordinate Description of object

(b) projection on the viewing plane

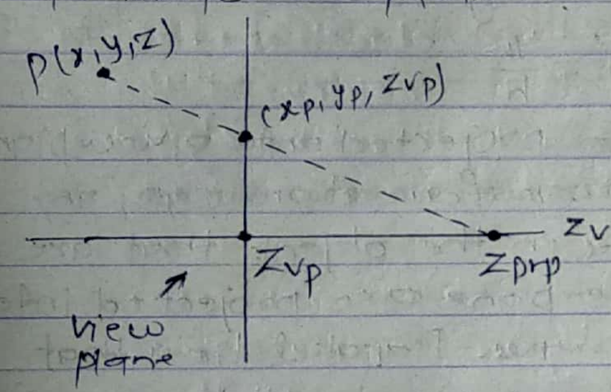
$$x' = x - xu$$
$$y' = y - yu$$
$$z' = z - (z - z_{prp})u$$

Parameter u takes values from 0 to 1, and coordinate position (x', y', z') represents any pt along the projection line. When $u = 0$, we are at position $P = (x, y, z)$. At the other end of the line, $u = 1$ and we have the projection reference point

coordinates $(0, 0, z_{prp})$. On the view plane, $z' = z_{vp}$ and we can solve the z' eqn for parameter u at this position along the projection line

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

Substituting this value of u into the eqn for x' and y', we obtain the perspective transform equations.



Perspective projection of a point P with co-ordinates (x, y, z) to position (x_p, y_p, z_{vp}) on the viewing plane

$$x_p = x\left(\frac{z_{prp} - z_v}{z - z_{prp}}\right) = x\left(\frac{d_p}{z - z_{prp}}\right) \quad \text{—(1)}$$

$$y_p = y\left(\frac{z_{prp} - z_v}{z - z_{prp}}\right) = y\left(\frac{d_p}{z - z_{prp}}\right) \quad \text{—(1)}$$

where $d = z_{prp} - z_{vp}$ is the distance of the view plane from the projection reference point.

Using a 3D homogeneous-coordinate representation, we can write the perspective-projection transformation 12-13 matrix form as

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

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

and projection coordinates on the view plane are calculated from the homogeneous coordinates as

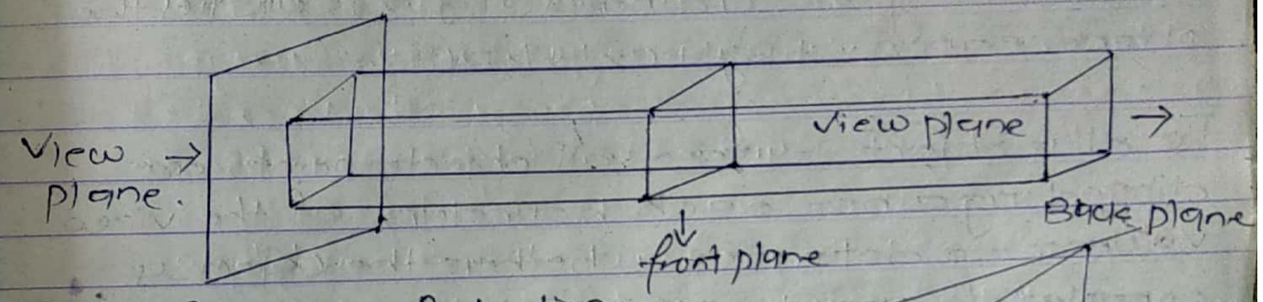
$$x_p = \frac{x_h}{h}, \quad y_p = \frac{y_h}{h}$$

When a 3D object is projected onto a view plane using perspective transformation eqn, any set of parallel lines in the object that are not parallel to the plane are projected into converging ~~planes~~ lines. Parallel lines that are parallel to the view plane will be projected as parallel lines. The point at which a set of projected parallel lines appears to converge is called vanishing point. Each such set of projected parallel lines will have a separate vanishing point; and many sets of parallel lines there are in the scene.

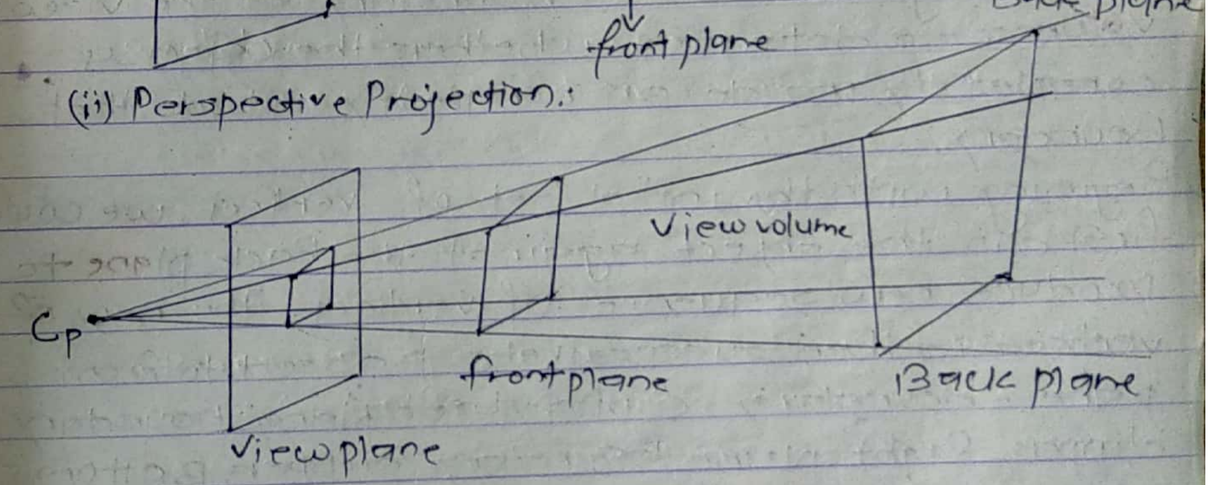
* Clipping In 3D:

In 3D space, objects within the view volume are visible while those outside are not displayed. Other crossing the boundary are cut & only the portion within the view volume is displayed.

(i) Parallel projection:



(ii) Perspective Projection:



Clipping Algorithm In 3D:

Eqn of the plane is given by

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

where A, B, C, D are the plane parameter. Any point $P_1(x_1, y_1, z_1)$ is outside a boundary plane if

$$Ax_1 + By_1 + Cz_1 + D > 0$$

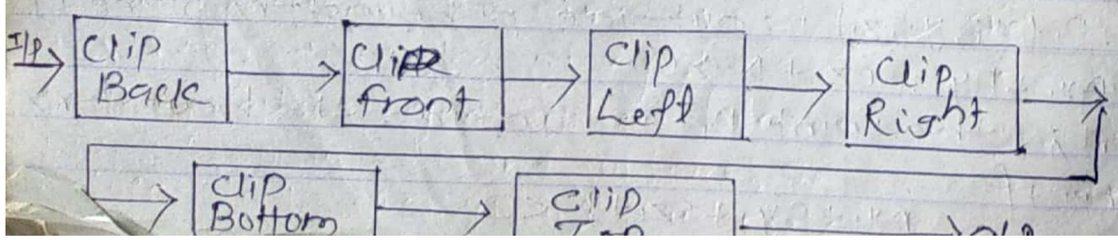
Similarly, the point P_1 is inside the boundary plane if

$$Ax_1 + By_1 + Cz_1 + D < 0$$

Scanned by CamScanner

Lines with bothendpoints outside a boundary plane are discarded and those with bothend-points inside all boundary planes are saved.
The intersection of a line with a boundary is found using the line equation along with the plane equation.
Intersecting co-ordinates (x', y', z') satisfies the plane eqn $Ax' + By' + Cz' + D = 0$.

To clip object surface, all objects points are clipped against each boundary of the view volume to determine whether the object is completely inside or completely outside that boundary.
Begining with the initial set of vertices, we could first clip the object against the Back plane to produce new sequence of vertices. The new of vertices is then successively passed to front plane - boundary clipper, Left plane boundary clipper, Right plane boundary clipper, Bottom plane boundary clipper & Top plane boundary clipper. At each step new sequence of output vertices is generated and passed to the next new plane boundary clipper as shown in fig.



The front and Back clipping planes are parallel to view plane.
Let z_v = front - z and z_1 = Back - z be the view plane co-ordinates for front and Back planes resp where front - z and Back - z are constants giving the positions of these planes on the z-axis.
Any point $P_1(x_1, y_1, z_1)$ is completely inside the front plane and Back plane if

$$\text{front} - z \le z_1 \le \text{Back} - z$$

* ~~Visible~~ VISIBLE surface Detection Method:

→ The method that are used for identifying those of a scene that are visible from a chosen viewing position are referred to as visible surface detection method.
→ Sometimes these methods are also ~~also~~ known as hidden surface elimination method.

Classification of Visible surface Detection Algorithm

- Visible surface detection algorithm are broadly classified according to whether they deal with the object definitions directly or with their projected image. These two approaches are called (a) Object space method
(b) Image space method.

Scanned by CamScanner

- An object space method compares objects and parts of objects to each ~~of~~ other within a scene definition to determine which surfaces are visible.
- An image space method visibility is decided point by point at each pixel position on the projection plane.
- Most visible surface algorithm use image-space method although object space method can be used effectly to locate visible surface in some cases.
- Line display algorithm generally use object space method to identify visible lines in wireframe display.

* Depth Buffer Method (Z-Buffer)

Depth Buffer method is an image space method, that compares surface depth at each pixel position on the projection plane and keeps the record of the depth surface that lies closest to the observer. The depth of the object is measured from view plane along z-axis of a viewing system.

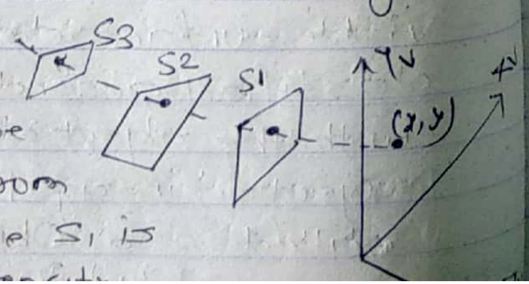


figure shows, three surfaces at varying distances along the orthographic projection line from pixel (x,y) in a plane surface S1 is closed to pixel (x,y) so its intensity

value at (x,y) is saved.

To implement this idea, two buffers namely z-buffer and refresh buffer are required to store depth value and intensity for each (x,y) position. Initially all positions in the depth buffer are set to maximum value and refresh buffer is initialized to the background intensity.

Each surface listed in the polygon table is then processed, one scan line at a time, calculating the depth i.e. z-value at each (x,y) position. The calculated depth is compared to the value stored in the depth buffer at that position. If the calculated depth is less than the value stored in the depth buffer, the new depth value is stored and the surface intensity at that position determined and placed in the same x-y location in the refresh buffer.

A depth Buffer Algorithm is as follows:

i. Initialize the z-buffer and refresh buffer so that for all buffer position (x,y)
   z-buffer(x,y) = z-back (max. value)
   refresh (x,y) = background intensity

ii. Scan convert the polygon in the scene during this process
   a) Calculate the depth z for each (x,y) position on the polygon
   b) If z-(x,y) < z-buffer(x,y)

   Then,

z-buffer(x,y) = z-(x,y)
refresh(x,y) = Intensity of polygon at (x,y) position

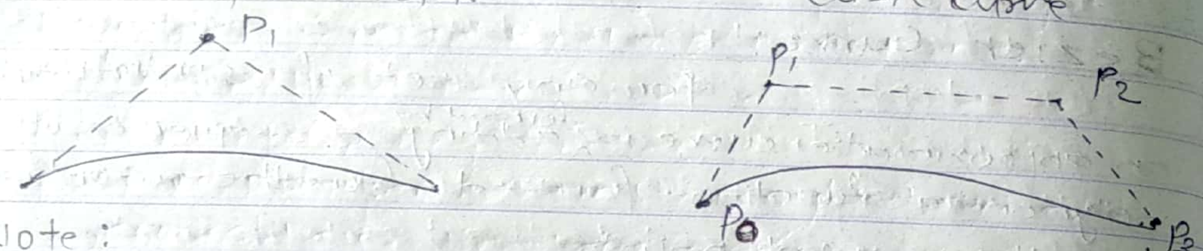
(ii) Repeat step (ii) for each polygon surface listed in the polygon table.

After all polygons have been processed the refresh buffer will contain the required solution.

Advantages
- It is easy to implement
- It can be implement in hardware to overcome the speed problem
- Since the algorithm processes objects one at a time the total number of polygons in a picture can be arbitrarily range.

Disadvantages
- It requires addition buffer and hence large memory.
- It is a time consuming processing as it requires comparision for each pixel instead of for entire polygon.

eg. Three points generate a parabola. four points generate a cubic curve



Note:
for a set of 4 control points, equation of Bezier curve are

$x(t) = x_3 t^3 + x_2 t^2 (1-t) + x_1 t (1-t)^2 + x_0 (1-t)^3$

$y(t) = y_3 t^3 + y_2 t^2 (1-t) + y_1 t (1-t)^2 + y_0 (1-t)^3$

As t goes from 0 to 1, the curve goes from the first control points to fourth control point.

is. $x(0) = x_0$

$y(0) = y_0$

& $x(1) = x_3$

$y(1) = y_3$

Application:

→ Bezier curves are used in painting and drawing packages, CAD systems.

→ They are easy to implement and reasonably powerful in curve design.

Drawbacks:

→ Bezier curve do not provide localized control ie. moving any control point changes the shape of whole curve.

→ At joints, complex geometric constructions are required to guarantee continuity. These drawbacks are overcomed by B-spline curve

* Bezier Surface:

Bezier surface can be designed by using two sets of Bezier curve by specifying an input set of control points. The parametric vector function for the Bezier surface is formed as the Cartesian product of Bezier blending function:

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k} B_{j,m}(v) B_{k,n}(u)$$

where $P_{j,k}$ specifying the location of $(m+1)$ by $(n+1)$ control points.

As entire surface is not always best described as a Bezier surface, surfaces are pieced together. Zero order continuity is achieved by making control co-incident (to coincide)

Properties Of Bezier Curves:

i. Curve always passes through the first and last control points.

ie. $P(0) = P_0$

& $P(1) = P_n$

ii. Multiple values: The parametric formulation of Bezier curve allows it to represent multiple valued shaps. If the first and last point coincide the curve is closed.

iii. A Bezier curve is independent of the co-ordinate system used to measure the location of control points.

iv. Bezier curve do not provide localized control ie. moving any control will change the shape

of the whole curve.

v. Bezier curve lies within the convex polygon boundary formed by the control points.

vi. $\sum_{k=0}^{n} B_{k,n}(u) = 1$

∴ Curve position is simply weighted sum of the control points positions.

vii. Bezier curve can be pieced together to describe a more complex curve.

* B-Spline Curves:

A general expression for the calculation of co-ordinate position of B-spline curve is

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

where P_k is input set of $(n+1)$ control points

- The range of parameter u is 0 to $(n+d)$
- The B-spline blending function, $B_{k,d}(u)$ are polynomials of degree $d-1$.

Bending functions for B-spline curve are defined by

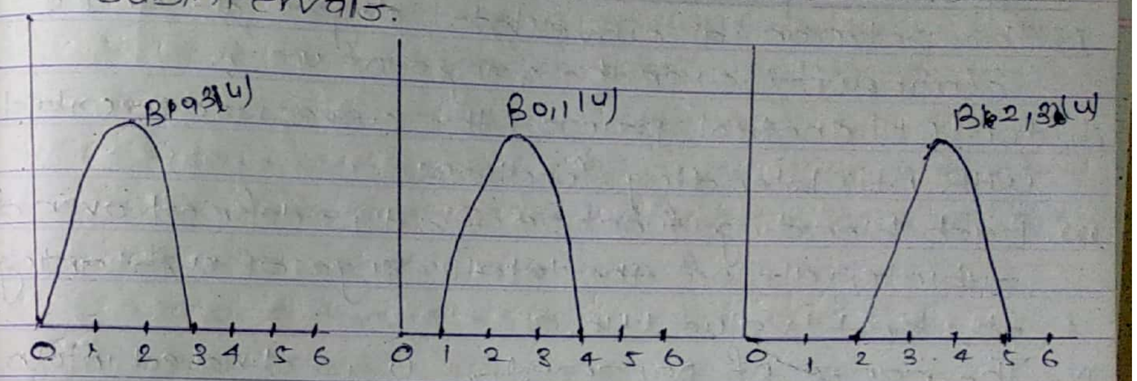
$$B_{k,1}(u) = \begin{cases} 1 & \text{if } u_k \le u \le u_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{k,d}(u) = \frac{u-u_k}{u_{k+d-1}-u_k} \cdot B_{k,d-1}(u) + \frac{u_{k+d}-u}{u_{k+d}-u_{k+1}} \cdot B_{k+1,d-1}(u)$$

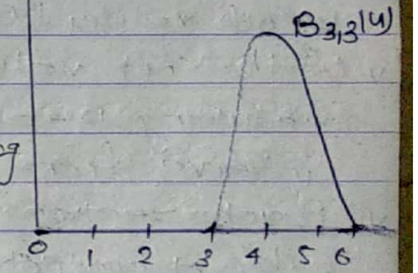
where each function is defined over d interval of the total range of u starting at u_k for

eg let $n=3$, $d=3$

Then range of u is from 0 to 6 i.e. $n+d = 3+3 = 6$ subintervals.

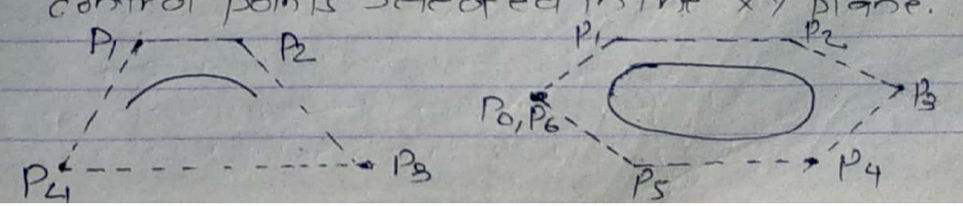


The 4th quadratic function demonstrate the local feature of B-spline. The first control point is multiplied by bending bending $B_{0,3}(u)$. Therefore, changing the position of the first control point only affects the shape of curve upto $u=3$.



similarly, the last control point influences the shape of the curve in the interval where $B_{3,3}$ is defined i.e. $u = 3$ to 6. The localized control of the curve can be increased by decreasing the value of d.

Ex: An eg. of the quadratic periodic B-spline curve is given in the following fig. for four control points selected in the xy plane.



Properties Of B-spline Curves:

i. The polynomial curve has degree d-1 and C^{d-2} continuity over the range of u

ii. for (n+1) control points, the curve is described with n+1 blending functions.

iii. Each blending function $B_{k,d}$ is defined over d subintervals of the total range of u, starting at knot value u_k.

iv. The range of parameter u is divided into n+d subintervals of ~~the total range~~ by the n+d+1 knot values ($u_0 \rightarrow u_{k+d}$)

v. With knot values labeled as $\{u_0, u_1, \ldots u_{n+d}\}$, the resulting B-spline curve is defined only in the interval from knot values u_{d-1} upto knot value u_{n+1}

vi. Each section of the spline curve is influenced by d control points.

vii. Any one control point can affect the shape of almost d curve sections.

If depth field is positive, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point. If the depth field is negative, this indicates multi-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data.

The A-buffer can be constructed using method similar to those in depth-buffer algorithm. Scan lines are processed to determine surface overlaps of pixels across the invidual scanlines. Surfaces are subdivided into a polygon mesh and clipped against the pixel boundaries. Using the opacity factors and percent of surface overlaps, we can calculate the intensity of each pixel as an average of the contributions from the overlapping surfaces.

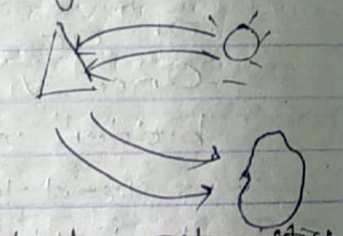
→ If a surface is texture or has a patterned printed on it, the reactance will vary with position on the surface.

→ Another surface property that play a role in shaded picture is transparency.

* Basic Illumination Models:

Ambient Light:

Surface i.e. not exposed ~~to~~ directly to a light source, still be visible if near by objects are illuminated. This light is called ambient light.



Characteristics of Ambient Light:

- It provides general level of brightness for all surfaces.
- It has no spatial or directional characteristics
- It has same intensity for each surface regardless of viewing direction.

If intensity of ambient light = I_a

Then, $I_{amb} = I_a$

If we take surface reflectivity then

$$I_{amb} = K_a . I_a$$

where K_a is co-efficient of ambient reflection.

Diffuse Reflection:

Diffuse reflection is constant over each surface in a scene independent of viewing direction. The intensity of diffuse

reflection due to ambient light is

$$I_{diff} = K_d . I_a$$

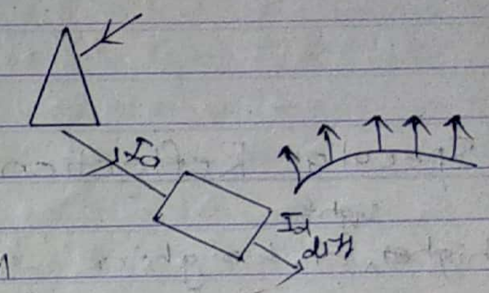
where

K_d = diffuse reflection co-efficient

for highly reflective surface, $K_d = 1$

" low " " $K_d = 0$

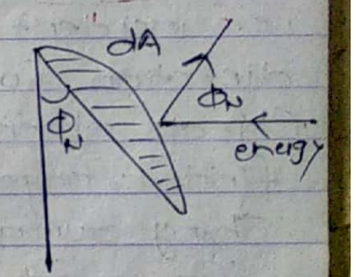
If surface is exposed to a point source we assume diffuse reflection from this surface ~~with~~ are scattered with equal intensity in all directions.



Such surfaces are also called ideal diffuse reflectors which follows Lambert's cosine law.

Lambert's Cosine Law:

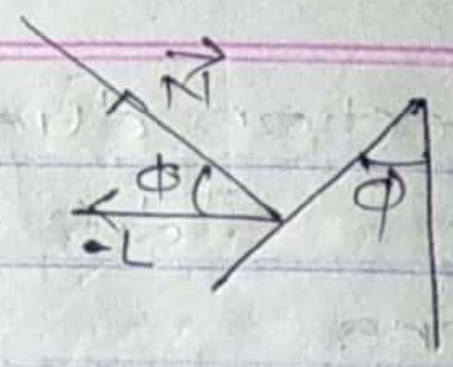
The radiant energy force i.e. light intensity from any small surface dA in any direction ϕ_N relative to surface normal is proportional to $\cos\phi_N$.



∴ Light intensity $\propto \cos\phi_N$.

Although there is equal light scattering in all direction from ~~any~~ an ideal diffuse reflection, the brightness of the surface depends on the orientation of the surface related to the light source. If surface is perpendicular to

source it appears brighter.



$I_{diff} = K_d \cdot I_1 \cos\phi$

$= K_d I_1 (\vec{L} \cdot \vec{N})$

$\therefore$ Total diffuse reflection = Diff (due to amb) + Diff. due to point source

$= K_a I_a + K_d I_L (\vec{L} \cdot \vec{N})$

Specular Reflection:

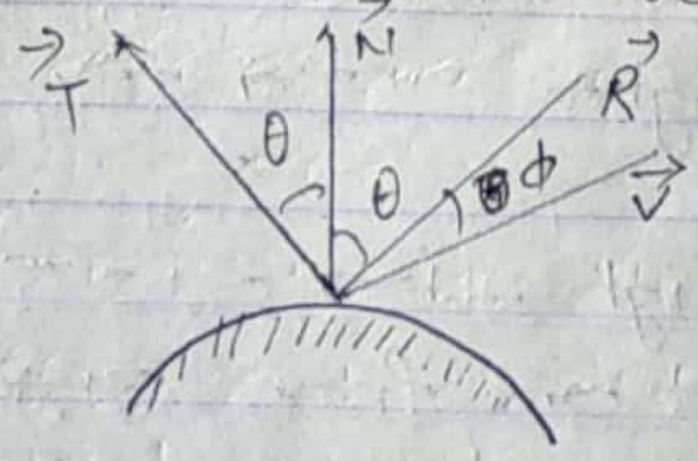
In shiny surfaces we see higher light or brighter spot from certain viewing direction called specular reflection. This is due to total or nearly total reflection of light.

for ideal reflector, (perfect mirror), $\phi = 0$

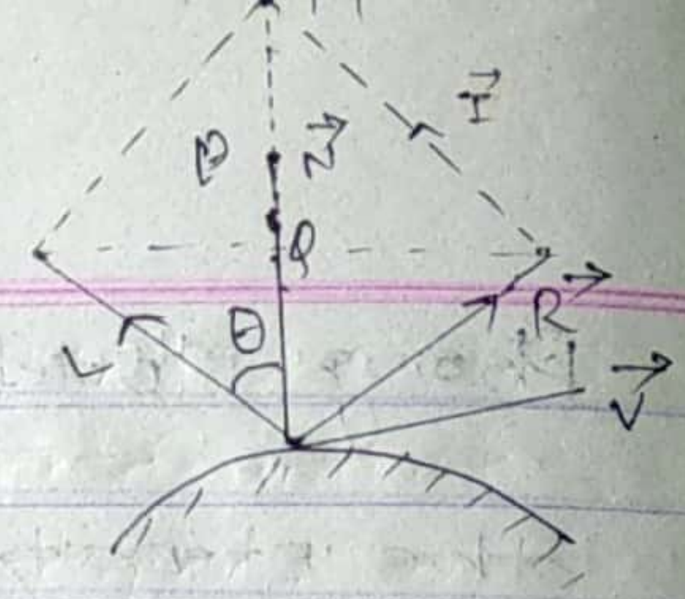
i.e. incident light is reflected only specular reflection direction. So $\vec{V}$ and $\vec{R}$ coincides.

Objects other than ideal reflection reflection for a finite range of viewing position around R.

Shiny surfaces have narrow specular reflection range and dull surfaces have a wider range.



The empirical formula for calculating the specular reflection range is given by phong model.



Expanding $\vec{R}$ in terms of $\vec{L}$ and $\vec{N}$

from fig aside

$\vec{R} + \vec{L} = \vec{OM}$

$= 2\vec{OQ}$

$= 2|\vec{ON}| \cdot \vec{N}$

$= 2\, O \cos\theta \cdot \vec{N}$

$\vec{R} + \vec{L} = 2(\vec{L} \cdot \vec{N}) \cdot \vec{N}$

$\vec{R} = 2(\vec{L} \cdot \vec{N}) \cdot \vec{N} - \vec{L}$

Total intensity is, $I = I_{amb} + I_{diff} + I_{spec}$

$= K_a I_a + K_d I_L (\vec{N} \cdot \vec{L}) + K_s I_L (\vec{V} \cdot \vec{R})^{n_s}$

where

$\vec{R} = 2(L \cdot \vec{N}) \cdot \vec{N} - \vec{L}$

* Polygon Rendering Methods:

1. Constant Intensity shading model.
2. Gouraud shading method (Intensity Interpolation)
3. Phone shading Method (Normal vector Interpolation)

⇒ Constant Intensity Shading Method:

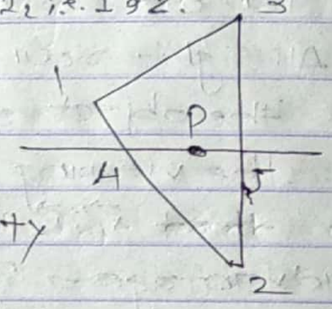
Any object with plane surface can be shaded with constant surface intensities. for a surface illuminated by a point source the shading model produces a constant surface intensity provided that the point source and view reference point are sufficiently far from the surface. With this method, the intensity is calculated at the interior point of each plane and the surface is shaded with the calculated intensity.

$I = K_a I_a + K_d I_L (\vec{N} \cdot \vec{R}) + K_s I_L (\vec{V} \cdot \vec{R})^{n_s}$, $R = 2(\vec{N} \cdot \vec{L}) \cdot \vec{N}$

each vertex to calculate intensity
- Linearly interpolate the vertex intensities over the surface of the polygon.

Intensity at 4 is obtained by I_1 & I_2 i.e. 1 & 2.

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



Similarly intensity at point 5 is interpolated from intensity values at point 2 and 3.

Once these bounding intensities are established for a scan line, an interior point is interpolated from the bounding intensities at point 4 and 5.

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

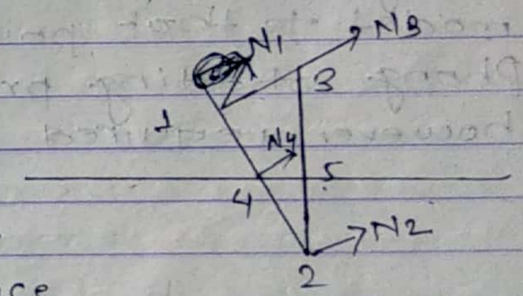
This process is repeated for each scanline passing through polygon.

Gouraud shading removes the intensity discontinuities associated with constant shading model. However, the linear ~~model~~ intensity interpolation can cause bright or dark intensity band, to appear on the surface.

Phone Shading :

Phong shading model also called normal vector interpolation. shading method interpolated normal vectors and then apply lighting model to each point.

Phong shading method
- Determine the average unit normal vector at each polygon vertex.
- Linearly interpolate the vertex normal over the surface of the polygon.
- Apply an illumination model along each scan line to calculate pixel intensities at each point.



At each vertex, the polygon average unit normal vector is obtained by averaging the surface normal of polygons sharing that vertex.

Thus for any vertex A

$$N_A = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

The normal vector N_4 for the scan line intersects point is obtained by interpolating edge endpoints normals.

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

Scanned by CamScanner

Similarly normal vector at point 5 is obtained by interpolating between edge point normals N_2 & N_3. All interior points along the scan line are interpolated from the bounding normals.

$$N_p = \frac{x_5 - x_p}{x_5 - x_4} N_4 + \frac{x_p - x_4}{x_5 - x_4} N_5$$

This process is repeated for each scan line passing the polygon.

3. Intensity values are calculated using an approximated normal vector at each point along the scan line by applying a lighting model to that point.

Phong shading produces an accurate results however required considerably more calculation.

CH# 07 GRAPHICAL LANGUAGES:

Graphics Softwares
There are two ~~types~~ general classifications for graphics:
1) General Programming Packages
2) Special Purpose Application packages.

(1) General Programming Packages:

- A general programming package provides extensive set of graphics function that can be used in higher level programming language such as C or FORTAN
- An example of a general graphics programming package is the GL (Graphics Language Library) system on silicon Graphics equipment.
- Basic function in a general package include those for generating picture components (straight line, polygon, circles and other figures) setting color and intensity values, selecting views and 3C applying transformations.

(2) Special Purpose Application Packages:

- Application graphics packages are designed for non-programmers, so that users can generate displays without worrying about how graphics operation work.
- The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms.

Scanned by CamScanner

Example : CAD

* Software Standards :

→ The primary goal of standardized graphics software is portability.

→ When packages are designed with standard graphic functions, software's can be moved easily from one hardware system to another.

→ Without standard programs designed for one hardware can't be transformed to another system without extensive rewriting of the programs. Thus we required a machine independent graphical language.

→ General Kernel system (GKS) was adopted as the first graphics software standard by the International Standard Organization (ISO) and by various national standards organizations, including the American National Standards Institute (ANSI)

→ Although GKS was originally designed as 2D graphics, a 3D GKS extension was subsequently developed.

→ The second software standard to be developed and approved by the standards organization was PHIGS (Programmer's Hierarchical Interactive Graphics standard) which is an extension of GKS.

→ Increased capabilities for object modeling, color specifications and picture manipulation

provided in PHIGS.

→ Standard graphics functions are defined as a set of specifications that is independent of any programming language. A language binding is then defined as a particular high level programming language. This language gives the syntax for accessing the various standard graphic functions from this language.

* Discussion Of available Languages and file formats :

file format :→ A file format is a standard method of encoding data for storage. There are many types of file formats, they are important because they tell programs what kind of data is contained in the file and how data is organized.

file formats are of two types :

<1> Property

<1> Proprietary

<2> Universal.

Proprietary And Universal :

The structure of proprietary is under the sole control of the software developer who invented the format.

This file can be opened or saved without the use of an import or export filter by the program that created them.

Universal file formats are based on openly published specification and are commonly used by many different program and different operating system.

For example: Adobe photoshop by default save images in its proprietary format. but it can also save its file in several universal format such as TIFF, GIF, JPEG, PICT and TGA. A word processing program can read and save files in specific formats such as DOC and TXT.

All the bitmap based to graphics programs can use any of the file formats listed in fig below:

Format — Description

1. BMP : A graphics format which is based in windows BMP is used for wallpaper.
2. PICT : (picture). It is generally used on MACS computers.
3. TIFF : (Tagged Imagefile format). A bitmap format defined in 1986 by Microsoft and Aldus and widely used in both Macs and PCs. This format is usually the best to use when exchanging bit map files that will be printed or edited further.
4. JPEG: (Joint photographic Export Group). This bit map format is common on the world wide

web and is often used for photos and high resolutions images. (24-bit or millions of colors) that will be viewed on the screen.
5. GIF : (Graphic Interchange Format). Like JPEG images GIF images are often found on world wide web pages. Unlike JPEG images, GIF images are reduced to 256 or fewer unique colors.
6. PNG : (Portable network graphics). This format was developed as an alternative to GIF & JPEG. PNG, like JPEG, can store the color images in a small amount of space, but PNG will file can store transparency info. the way GIF files do. The PNG format was initially designed for use in web pages.

Most vector base programs create and save file in a proprietary file format. These formats are either incompatible with other programs not totally supported by the other programs. In the later case, other programs may use import filters to use these file formats. Import filters never work perfectly because program using the import filter will not have the same exact features as the program that created the file. This lack of commonality has forced the software developer to create universal files formats, which enables user of one program to work with the files

created in other programs. Only a handful of commonfile formats such as date exchange formats (DXF) and initial graphics exchange specifications (IGES), exists for vector graphics. This universal format should enable you to create a vector file in one program, such as AUTOCAD and use it in another program such as corel DRAW or VISIO.

* Graphics Software :

1. Paint program : It works with bit map images
2. Photo manipulation program : It works with bit map images and are widely used to edit digitized photographs.
3. Computer Aided Design program : CAD ~~pro~~ software is used in technical design fields to create models of objects that will be built or manifactured. CAD software allows user to design objects in 3D and can produce 3-D frames and solid models.
4. 3-D Modeling Programs : It is used to create visual effects. 3D modeling program works by creating objects like surface, solid polygon, etc.

Imp → 5. Animation :

Computer are used to create animation for use in various fields, including games, and

These include hidden surface removal, alpha blending (transparency), antialiasing, texture mapping, pixel operations, viewing and modeling transforms, and atmospheric effects (for eg: smoke and haze).

Silicon Graphics, makers of advanced graphics workstations; initiated the development of OpenGL.

The earliest version of OpenGL were released with a companion library called GLU, the OpenGL Utility. It provides simple, useful features which were unlikely to be supported in contemporary hardware (like mipmap

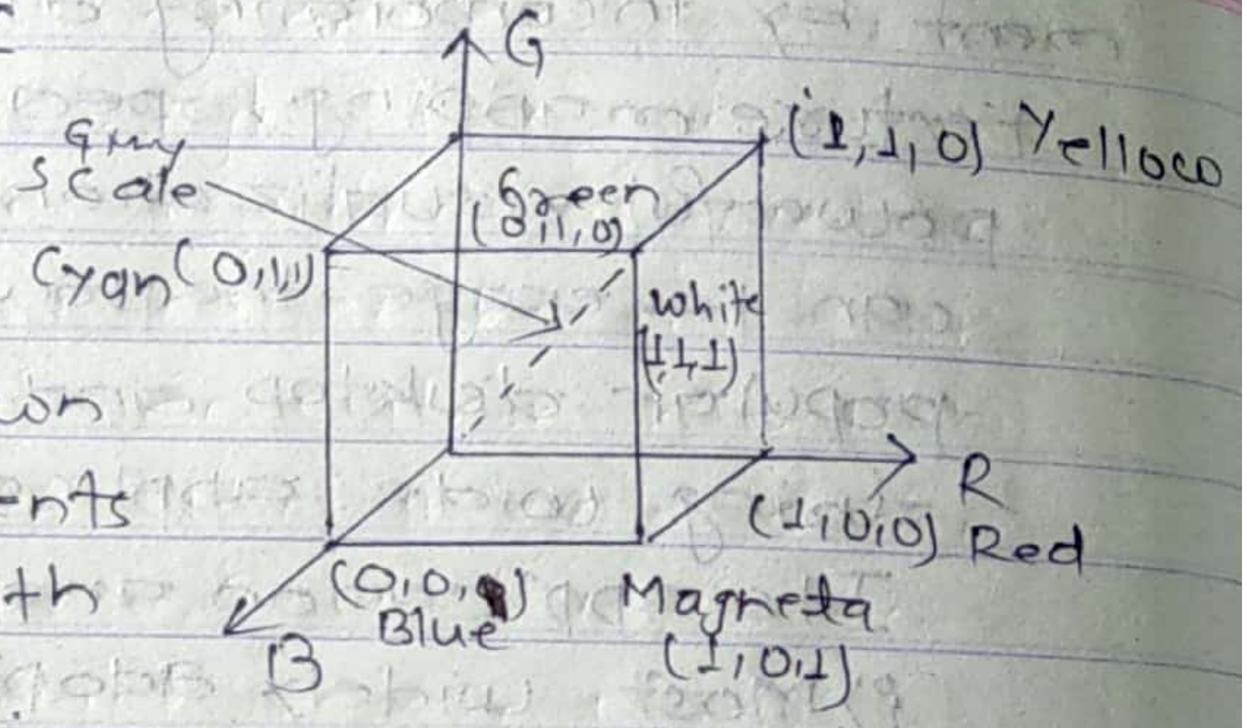
The OpenGL is an evolving API (API is defined as a number of functions which may be called by the client program, alongside a number of named integer constants). New version of the OpenGL specifications are regularly released by Khronos Group each of which extends the API to support various new features.

*Application:-

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application develop

* RGB Colour Model :-

This colour mode is represented with the unit cube defined on R, G and B axes as shown in fig. The origin represents black and the vertex with co-ordinate (1,1,1) is white.



Vertices of the cube on the axes represent the ~~binary~~ primary colours and the remaining vertices represent the complementary colors for each of the primary colours.

As with the XYZ colour system, the RGB colour scheme is an additive model. Intensities of the primary colours are added to produce other colours. Each colour point within the bounds of the cube can be represented as the triple (R, G, B), where values for R, G & B are assigned in the range from 0 to 1. Thus, a colour C_λ expressed in the RGB components as

$$C_\lambda = R'R + G'G + B'B.$$

* CMY Colour Model :

A unit cube representation for CMY model is illustrated in the adjoining fig.

In the CMY model, point (1,1,1) represents black, because all components of the incident light are subtracted.

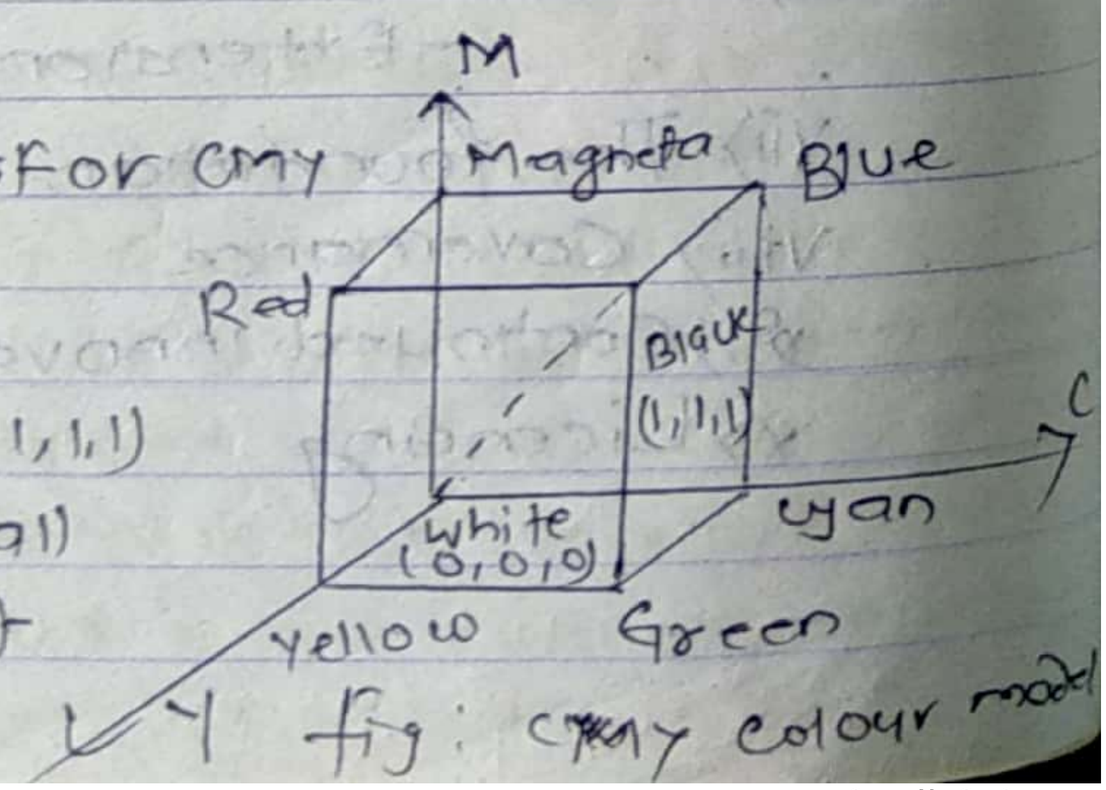


fig: CMY colour model

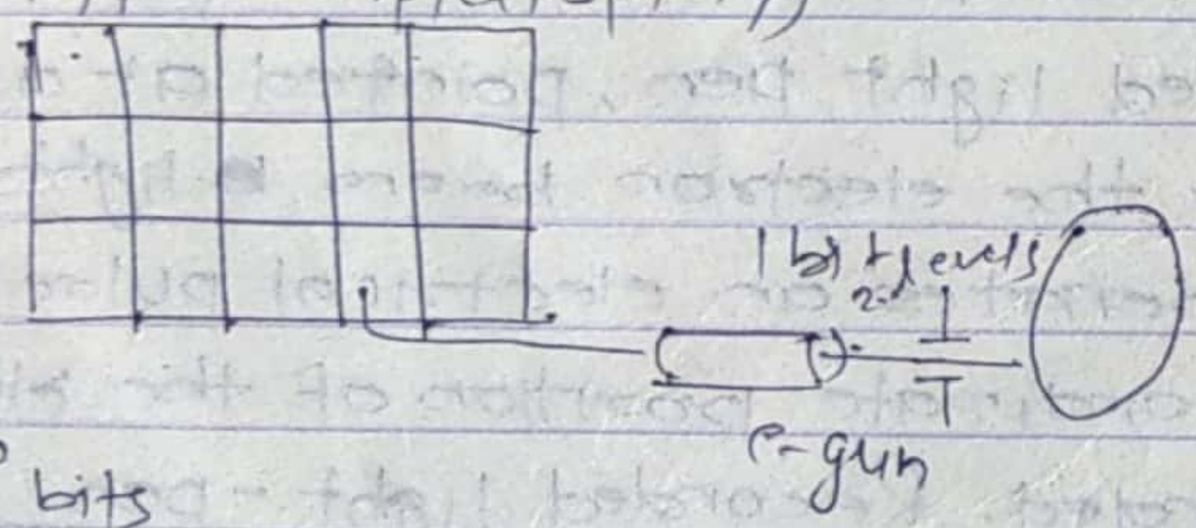
image is obscured by the hand and pen. And prolonged used of the light pen can cause arm fatigue. Also, light pens requires special implementations for some application because they can not detect position within black areas. To be able to select position in any screen area with a light pen, we must have some non-zero intensity assigned to each screen pixel.

In addition, light pens sometimes give false readings due to background lighting in a room.

* Frame Buffer :-

→ It is a set of arrays.

→ Total memory area is known as memory buffer monochromatic display (Bit - map display)

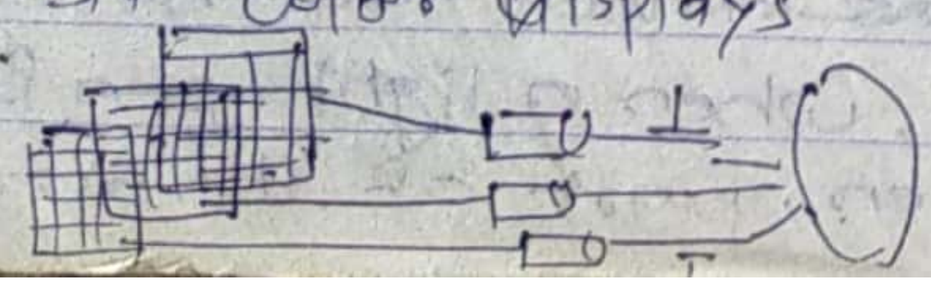


2^{20} bits

$\frac{1024 \times 1024}{1\text{-bit}}$ 1 pixel — 1 bit

frame buffer produces digital dp. To convert this digital to analog signal, there is one device (digital to analog converter device) used before the CRT

3-bit color displays



True color display

24 bit planes, 8-bit per color gun, $2^{24} = 16,777,216$

* Aspect ratio: ratio of vertical pt. to horizontal pt (necessary to produce equal length lines in both direction on screen)

Refresh rate = no. of times per sec the image is redrawn

Q. If we want to resize a 1024 x 768 image to 640 pixel width with the same aspect ratio. What would be the ht. of resized image.

→ $\frac{640}{1024} \times 768 = ht. = 480$

Q. Compute the size of 640 x 480 image at 240 pixel per

$\frac{640}{240}$ 2.6 $\frac{480}{240}$ 2

2.67 x 2 image //

* Q. An image has ht. of 2 inch & aspect ratio 1.5 What would be the width.

→ $\frac{w}{2} = 1.5 \Rightarrow w = 3$ inch. | Aspect ratio = $\frac{width}{height}$