# COMPUTER GRAPHICS

It is used in various fields:

- Science
- Engineering
- Medicine
- Government
- Art
- Business
- Industry
- Advertising
- Education
- Training

Major uses:

① DESIGN PROCESSES: (Engg. & Archi. System)

→ CAD methods: design buildings, spacecraft, design automobiles, computers, aircraft, etc.

→ OBJECTS IN WIREFRAME OUTLINE FORM:
Over all shape & internal features.
Realistic lighting models
Surface texture applied.

② PRESENTATION GRAPHICS:

→ used to produce illusions for reports or generate slides or transparancies for use with projectors.

eg. Bar charts
line graphs
pie charts, etc

③ COMPUTER ART : computer graphics methods are widely used in both fine arts and commercial art applications.

Artists use:

- Special purpose H/W
- Paint brush Prog. (eg. Lumene)
- special developed S/W
- CAD packages
- Animation packages, etc.

common graphics methods used in many commercials is morphing, where one object is transformed (METAMORPHOSED) into another.

eg.

This method has been used in TV commercials to turn water into tire, one person's face into another, etc

④ ENTERTAINMENT :

CG is used in making motion pictures, music videos and TV shows.

Sometimes graphics scenes are displayed themselves or combined with actors and live scenes.

eg. for a movie → Star Trek.

⑤ EDUCATION & TRAINING :

In this computer generated models of physical, financial and economic system is used.

Models of physical system, population trends etc. can help trainees to understand the operation of the system.

Eg. Specialized systems are designed systems such as simulators for practice sessions or training of ship captains, aircraft pilots etc.

Color coded diagram can be used to explain the operation of a NUCLEAR REACTOR.

Automobile driving simulators are used to investigate the behaviour of driver in critical situations.

⑥ VISUALIZATION:

Scientists, engineers, medical personnel, business analysts and others often need to analyse large amounts of information to study behaviour of certain processes

Supercomputers frequently produce data files containing thousands or even millions of data values.

↓ If data is converted to pictorial form, the trends & patterns are often immediately apparent.

**Scientific visualizations:** graphical representation for scientific engg. and medical data sets & procedures.

**Business visualization:** data sets related to commerce, industry and other non-scientific areas.

Additional techniques include - contour plots
- graphs & charts
- surface rendering, etc.

⑦ **IMAGE PROCESSING:**

It is used to modify or interpret existing pictures such as photographs & TV scans.
Two principale applications of image processing are:

- Improving Picture quality
- Machine perception of visual information as used in robotics.

Firstly, photo is digitized into image file. Then digital methods can be applied to rearrange picture parts to improve quality of shading, etc.

Medical applications use image processing for picture enhancements in tomography & in simulations of operations.

TOMOGRAPHY = X-ray photography for views of physiological system.

medicine uses CG and image processing to model & study physical functions, to design artificial limbs & to plan and practice surgery.
This is called COMPUTER AIDED SURGERY.

⑧ GRAPHICAL USER INTERFACE :

Major components of GUI is window manager used for displaying multiple window areas.
Icon - graphical symbol. Icons take less space as compared to textual description of icon.

Menus - lists of textual description & icons.

# Additional topics #

⑨ OB~~PR~~JECT & IMAGE SPACE :

In real life drawing of a triangle would begin with a decision in our mind regarding such geometric characteristics as type & size of the triangle, followed by pen moving on paper.

In CG, this is called as OBJECT DEFINATION. It defins triangle in abstract space. This space is continious & is called OBJECT SPACE.

Drawing of triangle on paper maps the imaginary objects on paper, which constitutes display image in another space called IMAGE SPACE.

## SCAN CONVERSION:

Area of computer graphics that is responsible for converting continious figure into its discrete approximation called SCAN CONVERSION.

## IMAGE REPRESENTATION:

Pixels

A digital image or image is composed of discrete pixels or picture elements.

Raster

These pixels are arranged in rows & columns to form a rectangular picture area, sometimes called RASTER.

Resolution

Total no. of pixels in an image is a function of its size and no. of pixels per unit length in horizontal & vertical direction.

Total no. of Pixels.

This no. of pixel per unit length is reffered to as the RESOLUTION of the image.

Size of Image

eg. 3×2 inch image at resolution of 300 pixels/inch whould have

$$\underbrace{3\times300}_{900} \times \underbrace{2\times300}_{600} = 540000 \text{ pixels.}$$

SEQUENCE IMAGE SIZE is given as total no. of pixels in the horizontal direction times the total no. of pixels in the vertical direction.

eg. 512 × 512, etc.

This does not specify size of image or its resolution.

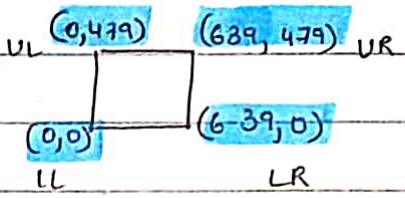
eg. A 640 × 480 image whould measure 1.6 inch × 1.2 inch at 400 pixels/inch.

ASPECT RATIO = $\frac{\text{image width}}{\text{image height}}$

Individual pixels in an array can be refferred to by their coordinates.

Pixel at left lower corner of an image is considered as origin (0,0) of a pixel coordinate system.

Q. A picture 640 x 480




DIRECT CODING:

Image representation is essentially representation of pixel codes.

In direct coding we allocate certain storage space for each pixel to code its color.

Q. 3 bits per pixel, ∴ 1 bit per each primary color.

Hence, each pixel can take one value out of total of 8 values.

But industry standard is 3 bytes or 24 bits per pixel i.e. 8 bit per primary color.

∴ 256 different industry levels.

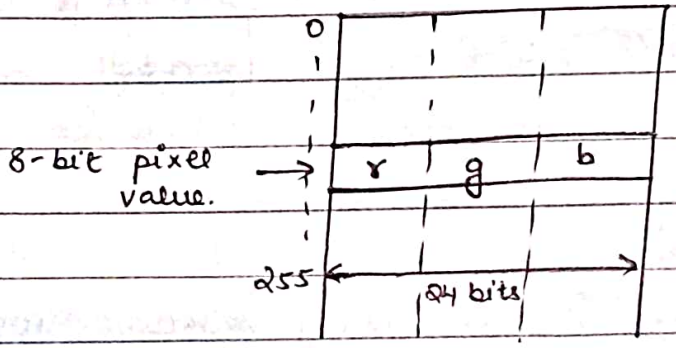
Thus, pixel can take color out of 256 x 256 x 256 possible values.

this 24 bit format is called true color representation.

## LOOK-UP TABLE :

In this strategy, pixel values have color in the form of addresses or indices into a table.
Color of particular pixel is determined by code value in table entry that the value of pixel references.

entry in table has addresses from 0 – 255




The color of a pixel whose value is i, where $0 < i < 255$ is determined by color value in the table entry whose address is i.

This 24 bit, 256 entry lookup table representation is often called as 8-bit format.
It reduces storage requirement of a 1000 x 1000 image.
It allows 256 simultaneous colors.

Image is defined not only by pixel value but by color value in corresponding look up table.
This forms COLOR MAP for the image.

## Display Monitor :

→ WORKING PRINCIPLE OF A MONOCHROMATIC DISPLAY MONITOR.

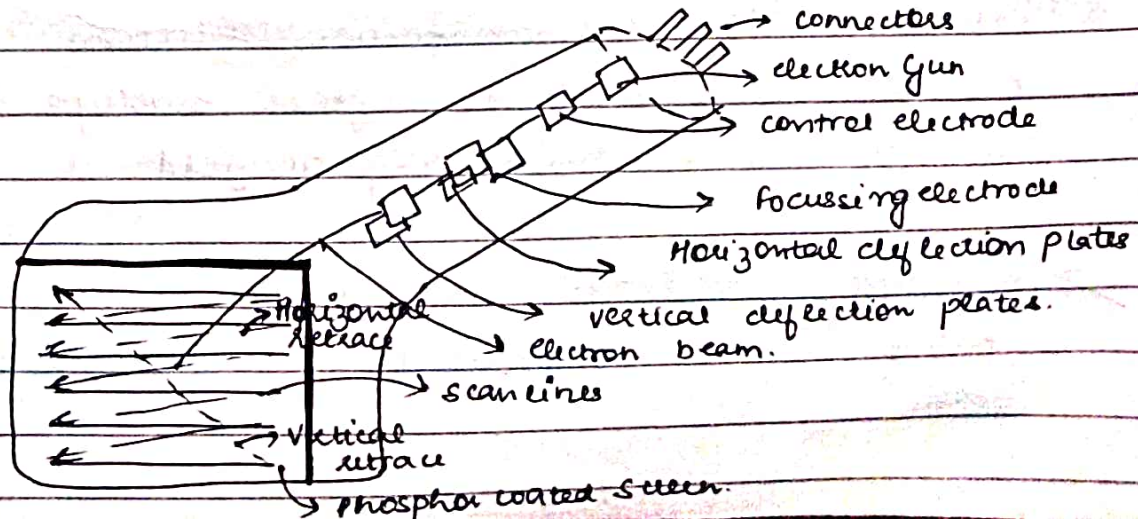
CATHODE RAY TUBE (CRT) is a vaccum glass tube with the display screen at one end and connected to control circuit at the other.
coated on the inside of display screen is a special material called PHOSPHOR, which emits for a period of time when hit by a beam of electrons.

Color of light and time period of may vary from one type of phosphor to another.

light given off by the phosphor during exposure to the electrons is known as FLOROSCENCE.

The continuing glow given off after then beam is removed is known as PHOSPHORESCENCE.
and the duration of phosphorescence is known as PHOSPHOIS PERSISTANCE.

ANATOMY OF MONOCHROMATIC CRT



Opposite to phosphor coated screen is electron gun that is heated to send out electrons.

Electrons are regulated by control electrode by setting voltage levels on the control electrode & forced by the focussing electrode into a narrow beam striking phosphor coating at small points.

Focussing is needed because otherwise electrons will repel each other and beam would spread as it approaches the screen.

Electron beam passes through horizontal & vertical plates and is bent by electric field b/w the plates.

Horizontal plates control the beam to scan from left to right and retrace from right to left.

Vertical plates control the beam to go from 1st scan line at the top to the last scan line at the bottom and retrace from bottom, back to top.

Some CRT's use magnetic deflection coils mounted on the outside of the glass envelope to bend the electron beam with magnetic field.

Intensity of light emitted by electron beam phosphor coating is a function of the intensity of electron beam.

Image displayed is stored in dedicated system memory area called FRAME BUFFER or REFRESH BUFFER.

Frequency at which contents on frame buffer is sent to display monitor is called REFRESHING RATE.

typically it is 60Hz.

lower than this leads to flicker

↓

To reduce flicker, some monitor use INTERLACING.

ie. half of scan lines are refreshed at a time.

even then odd.

## # RESOLUTION:

Max no. of pts that can be displayed without overlap on a CRT is called resolution.

More precise defination of resolution is no. of points per cm that can be plotted horizontally & vertically

Resolution of a CRT is dependent on the type of phosphor, intensity of to be displayed and the focussing and dyflection systems.

Typically resolution of high Quality systems is 1280 x 1024.

# RASTER SCAN DISPLAYS:

Objective

Electron beam is swept across the screen, one row at a time from top to bottom.

Pic 3

Picture definition is stored in a M/M area called REFRESH / FRAME BUFFER.

M/M area

This M/M area holds set of intensities for all the screen points. Stored intensity values are then retrieved from refresh buffer and 'PAINTED' on the screen, one row (scanline) at a time.
Each screen point is called PIXEL or PEL (picture element)

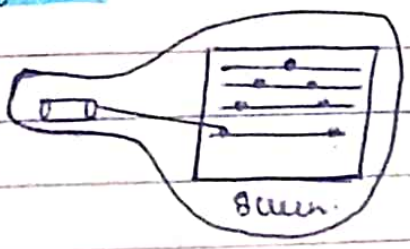
Typ & Disp

This capability of raster scan to store intensity info is well suited for the realistic display of scenes containing subtle shading & color patterns.
eg. Home TV's, printers etc.

Ref. rate

Refreshing or raster scan display is carried out at 60 to 80 frames per second.

At the end of each frame electron beam returns to the top of left corner of the screen to begin the next frame (vertical trace).



# RANDOM SCAN DISPLAY:

Concept.

CRT has electron beam directed only to the parts of the screen where a picture is to be drawn.

Random scan monitors draw a picture one line at a time and for this reason are also called VECTOR DISPLAYS.

The component lines of a picture can be drawn & refreshed by Random scan systems in any specified order.

eg. Pen plotter, hard copy device devices operates on random scan principle.

Ref rate.

Refresh rate depends upon no. of lines to be displayed. Picture definition is stored as a set of lines-drawing commands in M/M area called REFRESH DISPLAY FILE. or DISPLAY LIST or REFRESH BUFFER.

M/M area.

Random scan displays a picture through a set of commands in display file.
After all line commands, have been processed, system cycles back to 1st line command in the list.

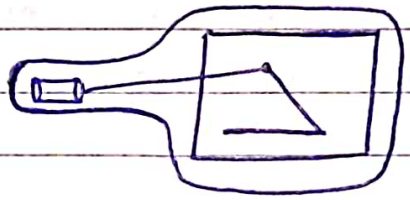
Drawing occurs 30 to 60 times/second. Faster refresh rates cored burn the phosphor.

Random scan systems are designed for line-drawing applications and can't display realistic shaded scenes.

Since, picture definition is stored as a set of line drawing instructions and not as set of intensities, vector displays generally have higher resolution than raster systems.

adv

Also, they produce smooth lines.



## LINE DRAWING ALGO :

The cartesian slope-intercept equation for a straight line is

$$y = mx + b \quad \text{—— (1)}$$

slope can be calculated as

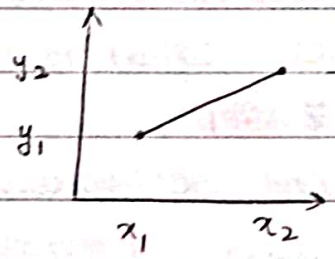
$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{—— (2)}$$

Also,

$$b = y_1 - mx_1 \quad \text{—— (3)}$$

for any given $x$ interval $\Delta x$, we can compute $y$ interval $\Delta y$ as

$$\Delta y = m \Delta x \quad \text{—— (4)}$$




Similarly we obtain $x$ interval $\Delta x$ corresponding to specified $\Delta y$ as

$$\Delta x = \frac{\Delta y}{m} \quad \text{—— (5)}$$

On raster systems, lines are plotted with pixels & step size in horizontal & vertical directions are constrained by pixel separation.

That is, we must sample a line at discrete positions and determine nearest pixel to the line at each sampled point position.

→ DDA algorithm (Digital Differential analyzer)

scan conv

It is a scan conversion line algo. based on calculating either $\Delta y$ or $\Delta x$ using equation ④ & ⑤.

---

sample

we sample the line at unit intervals in one coordinate and determine corresponding integer values, nearest the line path for the other coordinates.

L → R

+ve m ≤ 1

consider, a line with positive slope.

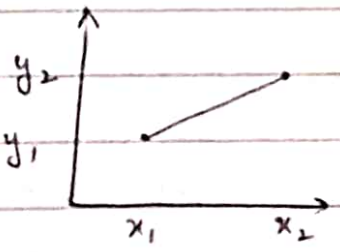
If slope is less than or equal to 1

i.e. $m \leq 1$,

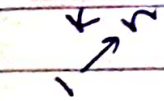
sample at unit interval $\Delta x = 1$

we sample at unit $x$ intervals ($\Delta x = 1$) & compute each successive $y$ values as

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

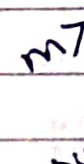


k takes integer values from 1, for 1st point and increases by 1 until final end point is reached.



since, (m) can be any real no. b/w 0 & 1, the calculated values must be rounded to nearest integer.

For lines, with the slope >1, we reverse roles of x and y .ie.



we sample at unit y intervals ($\Delta y = 1$) and calculate succeeding x values as

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{—②}$$

Equation ① & ② are based on assumption that lines are to be processed from left to right endpoints.

If processing is reversed, so

$$\Delta x = -1 \quad \text{and}$$

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

or,

when slope is >1, we have

$$\Delta y = -1 \quad \text{and}$$

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

DDA algo is faster method to calculate pixel positions that direct use of equation

$$y = mx + b$$

It eliminates multiplication in, $y = mx + b$, by making use of raster characteristics, so that appropriate increments are applied.

Accumulation of round off error in successive additions can, however, cause calculated pixels to drift away from true line path.

Calculations in DDA algo. is still time consuming.

→ Advantage of DDA:

It is faster method for calculating pixel positions.
It eliminates the multiplication in

$$y = mx + b.$$

→ Disadvantage of DDA:

① Floating calculations can cause calculated pixel positions to drift away from true line path for long line segments.

② Rounding off operations and floating pt arithmetic is still time consuming.

- accurate & efficient
- incremental integer calculation.

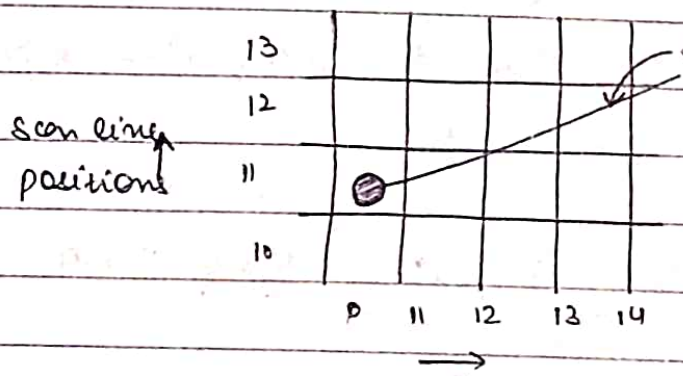
## BRESENHAM'S LINE ALGO

scan converts lines using only incremental integer calculations

Assumptions: scan conversion of lines with the slope less than one.

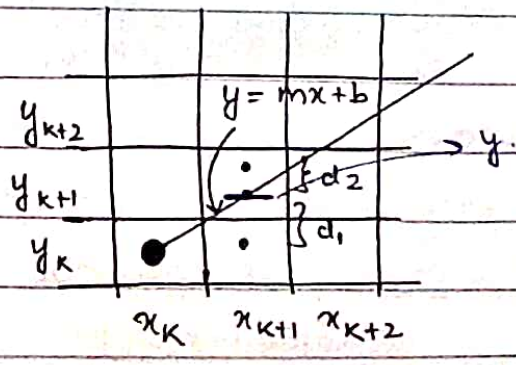
scan converts line with slope $m \leq 1$

Starting at left end point of a line ie. $(x_0, y_0)$ sample in unit x intervals.
Step to each successive columns & plot pixel value which is closest to line path.




Let $d_1$ & $d_2$ be vertical pixel positions from line path. (y) coordinate on mathematical line at pixel column $(x_{k+1})$ is

$$y = m(x_k + 1) + b \quad \text{——①}$$



Now,

$$d_1 = y - y_k$$
$$= m(x_k + 1) + b - y_k$$

Also,

$$d_2 = (y_k + 1) - y$$
$$= y_k + 1 - m(x_k + 1) - b$$

Now, $d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1$ ——— (2)

$$\Delta x (d_1 - d_2) = 2\Delta y (x_k + 1) - 2\Delta x \cdot y_k + \Delta x \cdot 2b - \Delta x$$

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

$\Delta y$ ; $\Delta x$ are vertical & Horizontal seperations of end point positions.

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

where,

$P_k$ is decision parameter.

$$P_k = 2\Delta y\, x_k - 2\Delta x\, y_k + \underbrace{2\Delta y + \Delta x(2b - 1)}_{c}$$

$$P_k = 2\Delta y\, x_k - 2\Delta x\, y_k + c \quad \text{——— (3)}$$

If, $P_k < 0$ i.e.

$$d_1 - d_2 < 0$$

or $d_1 < d_2$,

then LOWER PIXEL is plotted, otherwise

UPPER PIXEL

$K \rightarrow K+1$

Now,

At step $K+1$, decision parameter is

$$p_{K+1} = 2\Delta y \cdot x_{K+1} - 2\Delta x \cdot y_{K+1} + C$$

$$p_{K+1} - p_K = 2\Delta y (x_{K+1} - x_K) - 2\Delta x (y_{K+1} - y_K)$$

But, $x_{K+1} = x_K + 1$

$$\therefore p_{K+1} - p_K = 2\Delta y - 2\Delta x (y_{K+1} - y_K)$$

If, $p_K < 0$, then

$y_{K+1} = y_K$

$$p_{K+1} = p_K + 2\Delta y$$

If, $p_K \geq 0$, then

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

$$p_{K+1} = p_K + 2\Delta y - 2\Delta x$$

$$p_K = 2\Delta y \cdot x_K - 2\Delta x y_K + \Delta x (2b - 1) + 2\Delta y$$

$$p_0 = 2\Delta y \cdot 0 - 2\Delta x \cdot 0 + \Delta x (2 \times 0 - 1) + 2\Delta y$$

$$p_0 = 2\Delta y - \Delta x$$

↓

First parameter at starting pixel position.

$(x_0, y_0)$

eg.

$x_1, y_1$ $x_2, y_2$

Let end points of line be (20, 10) & (30, 18)

$\Delta x$ $\Delta x = 10\ (x_2 - x_1)$

$\Delta y$ $\Delta y = 8\ (y_2 - y_1)$

$P_0$

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

$2\Delta x$ $= 2 \times 8 - 10$

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

$P_k$ $x_{k+1}, y_{k+1}$

$2\Delta y = 16$ $\quad 2\Delta y - 2\Delta x = 16 - 20$

$= -4.$

initial point (20, 10)

| 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 |
| | | ⋮ |
| | | till 30, 18 |

+ / − ] y same

$P_0 < 0$

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

$= 6 - 4$

$= 2$

$2 + (-4)$

$= -2$

$P_1 > 0$

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

$= -2 + 16$

$= 14$

$14 - 4$

$= 10$

## PROPERTIES OF CIRCLE :

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

The distance relationship by Pythagorean theorem in cartesian coordinates is :

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

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

Problem with equation (2) is that it requires considerable computation at each step.

computation + spacing not uniform

Also,

Spacing b/w plotted pixels is not uniform.

One way to eliminate the unequal spacing is to calculate points using POLAR COORDINATES r & θ

The equation,

$$x = x_c + r \cos\theta \quad \text{——— (3)}$$

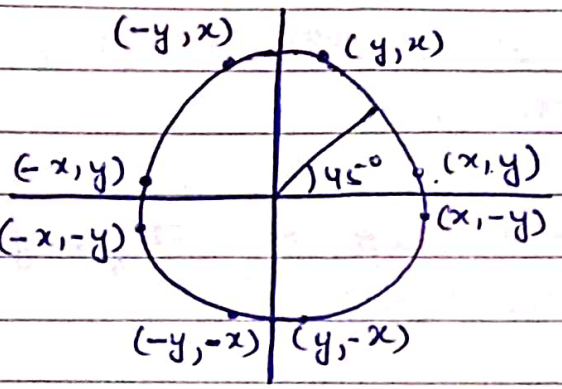
$$y = y_c + r \sin\theta \quad \text{——— (4)}$$

The step size chosen for θ depends on applications.

eg. for RASTER DISPLAY it is $\frac{1}{r}$.

It plots pixel positions that are 1 unit apart.

computation can be reduced by considering symmetry of circle.




if we calculate just one value of (x, y)
It will yield values in all 7 octants

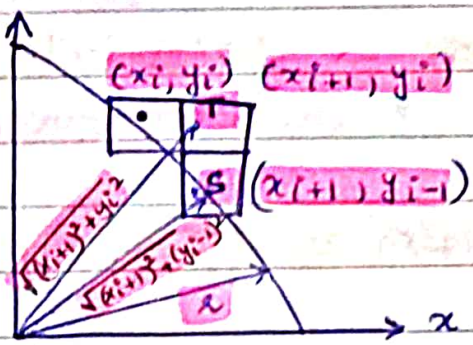
we can generate all pixel positions around a circle by calculating only points within the sector from

$x = 0$ to $x = y$.

Avoids ∫ calculations

## BRESENHAM'S CIRCLE ALGO :

If points are generated from 90° to 45°, moves will be made only in +x & -y direction.




Best approximation of the true circle will be described by those pixels in the raster that fall at least distance from the true circle.

dist from origin to T be = $\sqrt{(x_i+1)^2 + y_i^2}$

ie the distance from the origin to pixel T squared MINUS the distance to the true circle squared = D(T)

distance to true circle = r

$\therefore \sqrt{(x_i+1)^2+y_i^2} - r = D(T)$

and

for pixel S it be D(S)

coordinate of T are $(x_i+1, y_i)$

" " S are $(x_{i+1}, y_i-1)$

$\therefore$ $D(T) = (x_i+1)^2 + y_i^2 - r^2$

$D(S) = (x_i+1)^2 + (y_i-1)^2 - r^2$

Since, D(T) always +ve (T outside circle)

D(S) " -ve (S inside circle)

$\therefore$ Decision parameter $d_i$ is

$$d_i = D(T) + D(S)$$

$$\therefore \quad d_i = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2$$

When, $d_i < 0 \rightarrow |D(T)| < |D(S)|$ &
T is chosen.

$d_i \geq 0 \rightarrow |D(T)| \geq |D(S)|$ &
S is chosen.

$$d_{i+1} = 2(x_{i+1}+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$$

$$d_{i+1} - d_i = 2(x_{i+1}+1)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2(x_i+1)^2 - y_i^2 - (y_i-1)^2$$

Since,

$x_{i+1} = x_i + 1$, we have

$$d_{i+1} = d_i + 4x_i + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) + 6$$

# If T is chosen i.e. $d_i < 0$ then $y_{i+1} = y_i$

$$\therefore \quad d_{i+1} = d_i + 4x_i + 6$$

# If S is chosen i.e. $d_i \geq 0$ then $y_{i+1} = y_i - 1$

$$\therefore \quad d_{i+1} = d_i + 4(x_i - y_i) + 10$$

If starting pixel coordinates are $(0, r)$ then

$$d_i = 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2$$
$$= 3 - 2r$$

## MIDPOINT CIRCLE ALGORITHM :

sample at unit intervals

In this algorithm, we sample at unit intervals and determine the closest pixel position to the specified path at each step.

For a given radius (r) & screen centre position $(x_c, y_c)$, we can set up our algorithm to calculate pixel positions around a circle path centred at the coordinate origin (0,0)

Then each calculated position (x,y) is moved to its proper screen position by

adding $x_c$ to x

and, $y_c$ to y

From, $x = 0$

to $x = y$, it is one quadrant.

Slope of curve varies from 0 to −1.

∴, we take unit steps in positive x-direction in an octant

& use DECISION PARAMETER to determine & choose the point i.e. closest to the circle's true path.

calculate for 1 octant & other found by symmetry

Positions in other seven octants are obtained by symmetry.

We define a circle function :

$$f_{circle}(x,y) = x^2 + y^2 - r^2 \quad \text{——①}$$

Any point $(x, y)$ on boundary of circle with radius $= r$ satisfies the equation

$$f_{circle}(x,y) = 0$$

If the point is in the interior of circle, the circle function is -ve

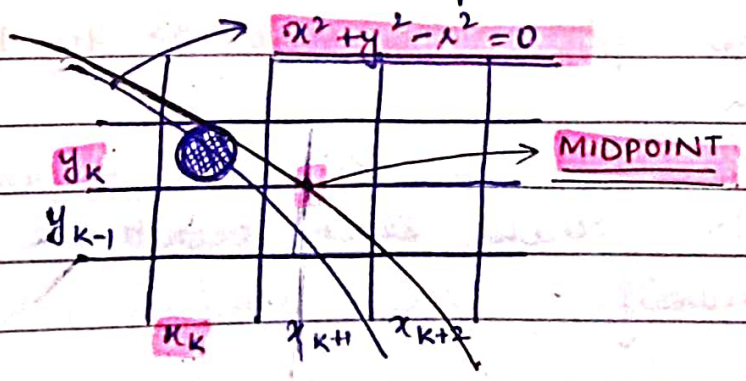
If the point is outside the circle, the circle function is +ve

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

——②

This is the DECISION PARAMETER.

Tests are performed for the mid positions b/w pixels near the circle path at each step.



The above figure shows the mid point b/w the two candidate pixels at sampling position $x_{k+1}$

Assume, $(x_k, y_k)$ has just been plotted

Next, we need to determine whether the pixel position $(x_{k+1}, y_k)$ or $(x_{k+1}, y_{k-1})$ is closer to the circle.

our decision parameter will be based on evaluating equation (1) at mid pt b/w these 2 pixels.

$$p_k = f_{circle}\left(x_{k+1}, y_k - \frac{1}{2}\right)$$

$$= (x_{k+1})^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad \text{---(3)}$$

Mid point is $\left(x_{k+1} + x_{k+1}, \frac{y_k + y_{k-1}}{2}\right)$

If, $p_k < 0$ → Mid point INSIDE circle pixel on $y_k$ is closer to circle boundary

else,

Mid point OUTSIDE circle & pixel on $y_{k-1}$ is chosen

Successive decision parameter's are obtained using incremental calculations.

for instance,

Next decision parameter would be

$$p_{k+1} = f_{circle}\left(x_{k+1}+1,\ y_{k+1}-\frac{1}{2}\right)$$

$$= \left[(x_k+1)+1\right]^2 + \left(y_{k+1}-\frac{1}{2}\right)^2 - r^2 \quad \text{——— (4)}$$

Subtracting $p_k$ from $p_{k+1}$
i.e. (4) − (3)

$$p_{k+1} - p_k = \left[(x_k+1)+1\right]^2 - \left[(x_k+1)^2\right] + \left[\left(y_{k+1}-\frac{1}{2}\right)^2 - \left(y_k-\frac{1}{2}\right)^2\right]$$

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

$$= (x_k+2+x_k+1)(x_k+1+1-x_k-1) + \left(y_{k+1}-\frac{1}{2}+y_k-\frac{1}{2}\right)\left(y_{k+1}-\frac{1}{2}-y_k+\frac{1}{2}\right)$$

$$p_{k+1} - p_k = 2x_k + 3 + (y_{k+1}+y_k-1)(y_{k+1}-y_k)$$

# If,

$p_k < 0$, then

$y_{k+1} = y_k$

$\therefore$ $p_{k+1} = p_k + 2x_k + 3$ ——— (5)

# If,

$p_k > 0$, then

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

$\therefore$ $p_{k+1} = p_k + 2x_k + 3 + (y_k-1+y_k-1)(y_k-1-y_k)$

$= p_k + 2x_k + 3 + (2y_k-2)(-1)$

$p_{k+1} = p_k + 2x_k - 2y_k + 5$ ——— (6)

Initial decision parameter ($p_0$) is obtained by evaluating the circle function at the start $(x_0, y_0) = (0, r)$

$$p_0 = f_{circle}\left(1, r - \frac{1}{2}\right)$$

$$= 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

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

If radius ($r$) is specified as integer, then.

$$p_0 = 1 - r$$

Q. $p_0 = \frac{5}{4} - r$ or $1 - r$

If, $p_k < 0$, then mid-point is $(x_{k+1}, y_k)$

and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

otherwise,

Next point is $(x_{k+1}, y_k - 1)$

and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

---

# let, $r = 10$

we have to determine pixel in 1st quadrant from $x = 0$ to $y$.

| 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 |

$k = 0$

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

$= \frac{5}{4} - 10$ $= \frac{5 - 40}{4}$ (1, 10)

$= \frac{-35}{4}$ $= -9$

$p_k < 0$ $p_k < 0$

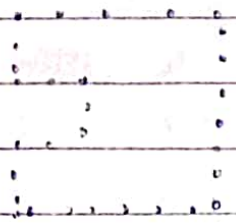
$p_{k+1} = p_k + 2x_{k+1} + 1$ $p_{k+1} = p_k + 2x_{k+1} + 1$

$= -9 + 2 + 1$

$= -6$

## POLYGON FILLING ALGO

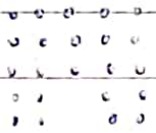
Region is described by pixels that outline it.



(a) Boundary-defined region

Or

Total pixels that make up a region.



(b) Interior-defined region.

Algorithms for filling up region (a) are called BOUNDARY FILL ALGO's.

Algo's for region (b) are called FLOOD-FILL ALGO's.

Regions can also be described by Geometry such as LINES & CURVES.

→ **BOUNDARY FILL ALGO:**

- Recursive Algo begins with starting pixel (SEED) inside the region.

- Algo checks whether this is a boundary pixel or has already been filled.

If (NO)
then, it fills the pixel and makes recursive call to itself using each & every neighbouring pixel.

If (yes)
Algo simply returns to caller.

**ADV:**
- works fine for arbitrarily shaped region.

**DISADV:**
- Takes TIME & MEMORY because of too many recursive calls.

→ FLOOD FILL ALGO

It begins with seed inside the region.

It checks if pixel has region's original same color

If (Yes)

It fills pixel with given a new color & uses each of the neighbour as a new seed in a recursive call.

If (NO)

It returns to caller.

Adv:

- useful if region has non-uniform colored boundary

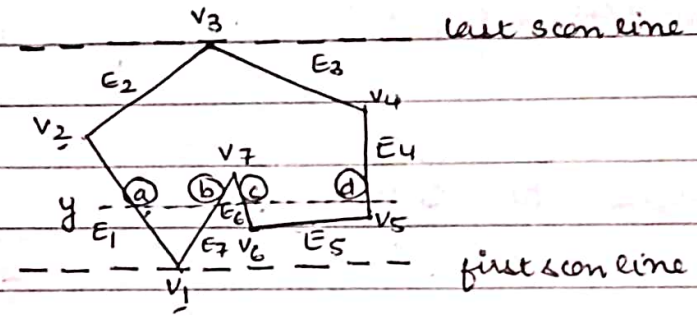
Disadv:

- Takes time & MM because of recursive calls.

## SCAN LINE ALGO:

- Boundary & Flood fill algos have pixel level defined regions

- Scan line algo has geometric defined regions.
  (defined by coordinates of vertices of polygon along with edges)




→ ## SCAN CONVERTING A POLYGONAL REGION:

Assumption: $V_i$ has already be scan-converted to integer coordinates $(x_i, y_i)$

→ • Each scan line finds all intersection pts b/w the current scan line & the edges

eg:
y intersects $E_1$, $E_7$, $E_6$ and $E_4$ at a, b, c, d resp.

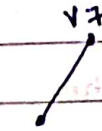
→ Intersection pts sorted by x coordinates & grouped into pairs.

eg. (a, b) & (c, d)

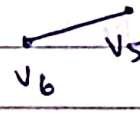
line is drawn from 1st to 2nd pt in pair.

→ • Horizontal edges are ignored
∴ pixels are filled automatically.

# Case of scan line intersecting a VERTEX.




• If vertex local MIN or MAX eg $V_1$ & $V_7$ ↓
NO SPECIAL TREATMENT.
2 edges that joins at the vertex will each yield intersection pt with scan line.




• For $V_5$ & $V_6$ → they are local MIN
the edge b/w them produces intersection pt.


• If intersects $V_4$ then we get 2 intersection pts.
but we need only 1 intersection pt.
$V \rightarrow y$.

| edge | $y_{min}$ | $y_{max}$ | x coordinate with $y = y_{min}$ | $\frac{1}{m}$ |
|---|---|---|---|---|
| $E_1$ | $y_1$ | $y_2 - 1$ | $x_1$ | $1/m_1$ |
| $E_7$ | $y_1$ | $y_7$ | $x_1$ | $1/m_7$ |
| $E_4$ | $y_5$ | $y_4 - 1$ | $x_5$ | $1/m_4$ |
| $E_6$ | $y_6$ | $y_7$ | $x_6$ | $1/m_6$ |
| $E_2$ | $y_2$ | $y_3$ | $x_2$ | $1/m_2$ |
| $E_3$ | $y_4$ | $y_3$ | $x_4$ | $1/m_3$. |

$E_5$ ✗ as horizontal edge not considered

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

$y_{min}$ & $y_{max}$ → y coordinates of edge's 2 end points.

- Edges become ACTIVE when y- coordinate of current scan line matches their $y_{min}$ value.

These are used in calculation of intersection pts. (Active edges)

- If edge intersects current scan line at $(x, y)$ then it intersects next scan line at $\left(x + \frac{1}{m}, y+1\right)$

Q. for $E_7$ → insects scan line $y$ at $b$ ∴ at $y+1$ its intersec$^n$ pt can be computed as $\Delta x = \frac{1}{m_7}$

$\Delta y = 1$

- Why do we have $y_2 - 1$ & $y_4 - 1$?

↓

To deactivate lower edge one line before the scan line intersects the vertex.

$$\begin{cases} y = mx + c \\ y_2 = mx_2 + c \\ y_1 = mx_1 + c \end{cases}$$

$$y_2 - y_1 = m(x_2 - x_1)$$

$$\frac{1}{m} = \frac{x_2 - x_1}{y_2 - y_1}$$

$$\text{or } \frac{1}{m} = x_2 - x_1$$

$$x_2 = \frac{1}{m} + x_1$$

# SCAN CONVERTING AN ELLIPSE :

ellipse has 4 way symmetry.

$(h,k)$ = ellipse centre

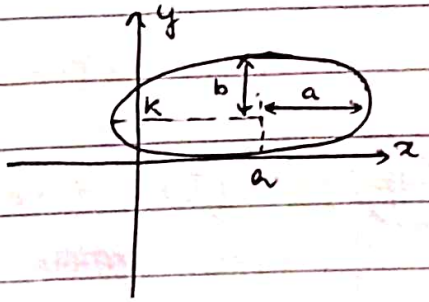
a = length of major axis

b = length of minor axis.

(I) $$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

Incrementing x from h to a

$$y = b\sqrt{1 - \frac{(x-h)^2}{a^2}} + k$$




(II) $$x = a\cos\theta + h \qquad \left(0 \leq \theta \leq \frac{\pi}{2}\right)$$

$$y = b\sin\theta + k$$

(III) MID POINT ELLIPSE ALGORITHM :

centre of ellipse is ORIGIN, then $\{(0,0)\}$

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

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

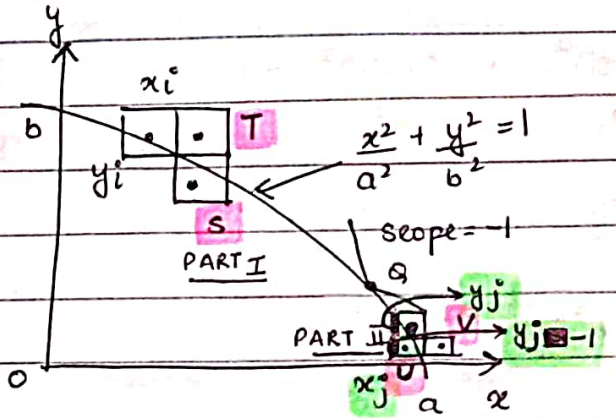
Now, $f(x,y) = b^2x^2 + a^2y^2 - a^2b^2$

- $< 0$ $(x,y)$ inside ellipse
- $= 0$ $(x,y)$ on ellipse
- $> 0$ $(x,y)$ outside ellipse.

Because of 4 way symmetry of ellipse, we need to consider the curve in 1st Quadrant.

Let,

Major & minor axes be ||l to the coordinate axis.




(#) PART (I):

Suppose curve is $f(x,y) = 0$,

then slope is $\frac{dy}{dx} = \frac{-f(x)}{f(y)}$

$f(x)$ & $f(y)$ are partial derivatives of $f(x,y)$ wrt $x$ & $y$.

$$f(x) = 2b^2x$$

$$f(y) = 2a^2y$$

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

ie. slope of curve changes MONOTONICALLY from 1 side of Q to another.

mid point of line b/w T & S is used to define decision parameter.

$x = x_i + 1$
$y = y_i - 1/2$

$$d_i = b^2(x_i+1)^2 + a^2\left(y_i - \frac{1}{2}\right)^2 - a^2b^2$$

if → $d_i < 0$, mid pt inside the curve, Choose (T).

→ $d_i \geq 0$, " " outside " " " (S)

Decision parameter for next step

$$d_{i+1} = f(x_{i+1}, y_{i+1} - \tfrac{1}{2})$$

$$= b^2 (x_{i+1} + 1)^2 + a^2 (y_{i+1} - \tfrac{1}{2})^2 - a^2 b^2$$

but,

$$x_{i+1} = x_i + 1$$

$$\therefore d_{i+1} = b^2 (x_{i+1} + 1)^2 + a^2 (y_{i+1} - \tfrac{1}{2})^2 - a^2 b^2$$

$$d_{i+1} - d_i = b^2 [(x_{i+1}+1)^2 - (x_{i+1})^2] + a^2 [(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$

$a^2 - b^2$

$$d_{i+1} = d_i + 2b^2 x_{i+1} + b^2 + a^2 [(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$

If, (T) chosen, then, $y_{i+1} = y_i$

(S) chosen, then $y_{i+1} = y_i - 1$

$$d_{i+1} = \begin{cases} d_i + 2b^2 x_{i+1} + b^2 & \text{if } d_i < 0 \\ d_i + 2b^2 x_{i+1} + b^2 - 2a^2 y_{i+1} & \text{if } d_i \geq 0 \end{cases}$$

Or

if, $d_i < 0$ ($y_{i+1} = y_i$)

↳ $d_{i+1} = d_i + b^2 (2x_i + 3)$

and if, $d_i \geq 0$ ($y_{i+1} = y_i - 1$)

↳ $d_{i+1} = d_i + b^2 (3 + 2x_i) + 2a^2 - 2a^2 y_i$

(0, b)

$$\Rightarrow d_0 = b^2 + a^2 (b - \tfrac{1}{2})^2 - a^2 b^2 \quad (x = 0, y = b)$$

$$d_0 = b^2 - a^2 b + \frac{a^2}{4}$$

(#) PART (II) :

mid point of horizontal line connecting U, V is

$$\left(x_j + \frac{1}{2},\ y_j - 1\right)$$

$$q_j = f\left(x_j + \frac{1}{2},\ y_j - 1\right)$$

$$= b^2\left(x_j + \frac{1}{2}\right)^2 + a^2(y_j - 1)^2 - a^2 b^2$$

$$q_{j+1} = f\left(x_{j+1} + \frac{1}{2},\ y_{j+1} - 1\right)$$

$$= b^2\left(x_{j+1} + \frac{1}{2}\right)^2 + a^2(y_{j+1} - 1)^2 - a^2 b^2$$

⟹ but, $y_{j+1} = y_j - 1$

$$q_{j+1} = b^2\left(x_{j+1} + \frac{1}{2}\right)^2 + a^2(y_j - 1 - 1)^2 - a^2 b^2$$

If, $\rightarrow$ $q_j < 0$

mid point inside the curve, choose (V)

then

$$x_{j+1} = x_j + 1$$

$$q_{j+1} - q_j = b^2\left[\left(x_{j+1} + \frac{1}{2}\right)^2 - \left(x_j + \frac{1}{2}\right)^2\right] + a^2\left[(y_j - 2)^2 - (y_j - 1)^2\right]$$

$(-1-1)$

$$= b^2\left[\left(x_j + \frac{1}{2} + 1\right)^2 - \left(x_j + \frac{1}{2}\right)^2\right] + a^2\left[(y_j - 1)^2 + 1 - 2(y_j - 1)(y_j - 1)\right]$$

$(a+b)^2$

$$= b^2\left[1 + 2x_j + 1\right] + a^2\left[1 - 2y_j + 2\right]$$

$$\therefore \boxed{q_{j+1} = q_j + b^2[2x_j + 2] + a^2[3 - 2y_j]}$$

If, $\longrightarrow$ $q_j \geq 0$, then

$x_{j+1} = x_j$

$$q_{j+1} - q_j = b^2 \left[ \left(x_{j+1} + \tfrac{1}{2}\right)^2 - \left(x_j + \tfrac{1}{2}\right)^2 \right] + a^2 \left[ (y_j - 2)^2 - (y_j - 1)^2 \right]$$

($x_{j+1}$ and $x_j$ cancel; $-2$ and $-1$ are marked)

$$= a^2 \left[ (y_j - 1)^2 + 1 - 2(y_j - 1) - (y_j - 1)^2 \right]$$

$$= a^2 \left[ 1 - 2y_j + 2 \right]$$

$$\therefore \quad \boxed{q_{j+1} = q_j + 3a^2 - 2a^2 y_j}$$

Initial decision parameter for part II is calculated using.

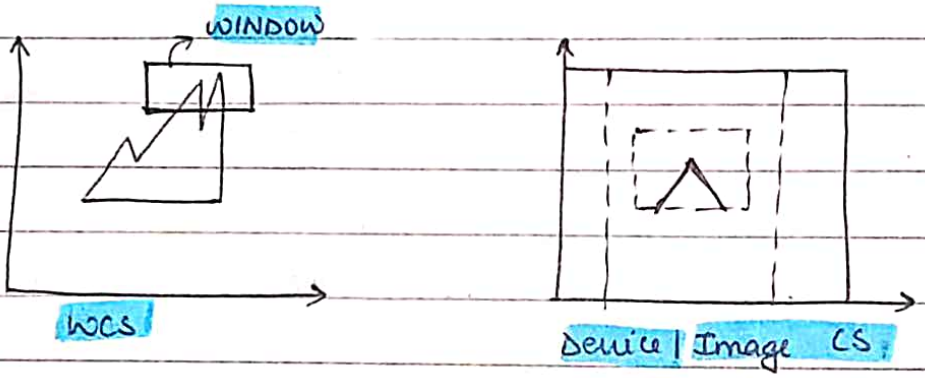
$q_j$ & coordinates $(x_k, y_k)$ of last pixel of part I of curve

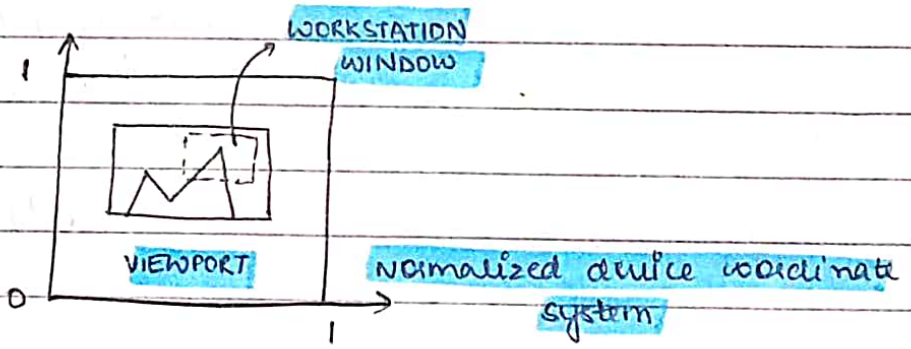
i.e $$q_0 = b^2 \left( x_k + \frac{1}{2} \right)^2 + a^2 (y_k - 1)^2 - a^2 b^2$$

# WORLD COORDINATE SYSTEM:

Objects are placed in a scene by modelling transformations to a master coordinate system. called WCS.

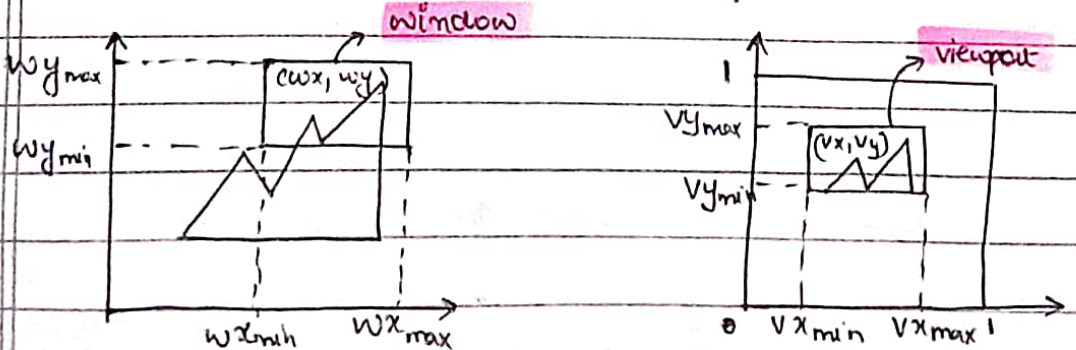
**WINDOW:** edges lie to axis of WCS is used to select the portion of scene for which image is to be generated.




**VIEWPORT:** Area on display device to which window is mapped.

## Window to viewport mapping:



normalized device coordinate

WC $(W_x, W_y) \rightarrow (V_x, V_y)$

**Objective:** Is to convert WC $(W_x, W_y)$ of an arbitrary point to its corresponding normalized device coordinates $(V_x, V_y)$

We have,

$$\frac{W_x - W_{x_{min}}}{W_{x_{max}} - W_{x_{min}}} = \frac{V_x - V_{x_{min}}}{V_{x_{max}} - V_{x_{min}}}$$

Also,

$$\frac{W_y - W_{y_{min}}}{W_{y_{max}} - W_{y_{min}}} = \frac{V_y - V_{y_{min}}}{V_{y_{max}} - V_{y_{min}}}$$

$$V_x = \frac{V_{x_{max}} - V_{x_{min}}}{W_{x_{max}} - W_{x_{min}}} (W_x - W_{x_{min}}) + V_{x_{min}}$$

$$V_y = \frac{V_{y_{max}} - V_{y_{min}}}{W_{y_{max}} - W_{y_{min}}} (W_y - W_{y_{min}}) + V_{y_{min}}$$

Since, all 8 coordinates of windows & viewports are constant

$$\therefore \begin{pmatrix} V_x \\ V_y \\ 1 \end{pmatrix} = N \cdot \begin{pmatrix} W_x \\ W_y \\ 1 \end{pmatrix}$$

where,

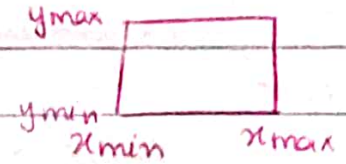
$$N = \begin{pmatrix} 1 & 0 & V_{x_{min}} \\ 0 & 1 & V_{y_{min}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{V_{x_{max}} - V_{x_{min}}}{W_{x_{max}} - W_{x_{min}}} & 0 & 0 \\ 0 & \frac{V_{y_{max}} - V_{y_{min}}}{W_{y_{max}} - W_{y_{min}}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -W_{x_{min}} \\ 0 & 1 & -W_{y_{min}} \\ 0 & 0 & 1 \end{pmatrix}$$

## POINT CLIPPING

$x_{min} \leq x \leq x_{max}$

&

$y_{min} \leq y \leq y_{max}$




where, $x_{min}, x_{max}, y_{min}, y_{max}$ → WINDOW

(x, y) point is INSIDE the window of INEQUALITIES are TRUE

---

## LINE CLIPPING

→ COHEN - SUTHERLAND ALGO (line clipping algo)

It is divided into 2 phases.

① Identify lines which intersect clipping window & so need to be clipped.

② Perform clipping.

- All lines fall into following categories.

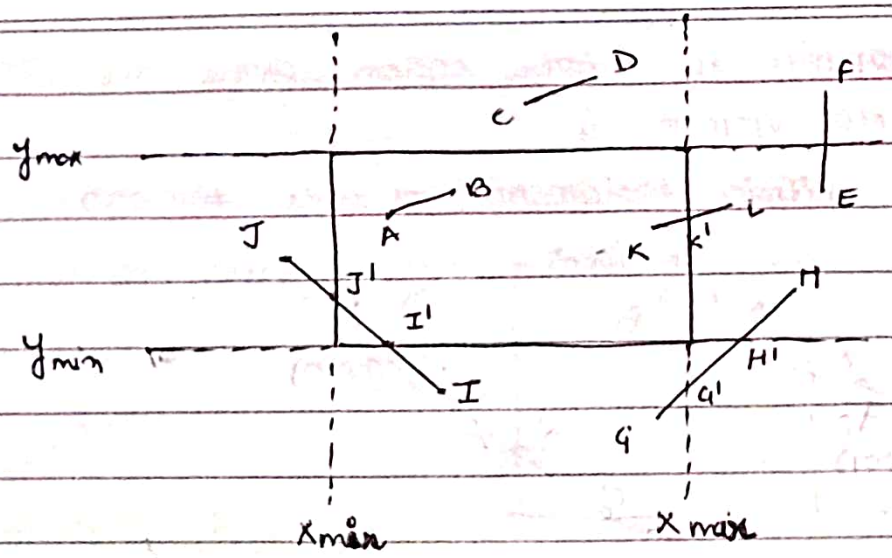
(i) VISIBLE: both end pts of line lie in the window.

(ii) NOT-VISIBLE: line definitely lie outside the window. it satisfies one of the following condition:

$x_1, x_2 > x_{max}$ $\quad y_1, y_2 > y_{max}$

$x_1, x_2 < x_{min}$ $\quad y_1, y_2 < y_{min}$.

(iii) CLIPPING CANDIDATE: line is neither category 1 nor 2.




**Procedure :**

① Assign **4 bit region code** to each end pt of line. code is determined according to

| | | | |
|---|---|---|---|
| $y_{max}$ | 1001 | 1000 | 1010 |
| | 0001 | 0000 | 0010 |
| $y_{min}$ | 0101 | 0100 | 0110 |
| | $x_{min}$ | $x_{max}$ | |

Starting from **LEFT**, each bit of code is set to **True (1)**

Or, **False (0)**, According to

Bit 1 = end pt is **ABOVE** the window = sign $(y - y_{max})$

Bit 2 = " " **BELOW** " = sign $(y_{min} - y)$

Bit 3 = " " **RIGHT OF** " = sign $(x - x_{max})$

Bit 4 = " " **LEFT OF** " = sign $(x_{min} - x)$

Convention followed: sign (a) = 1 if a is +ve

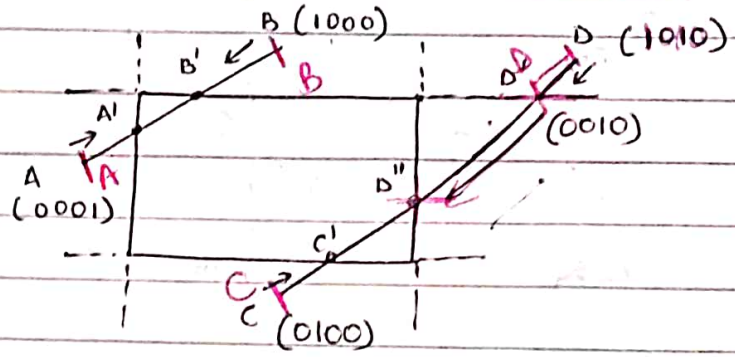
Otherwise, 0

pt with code **0000** is **INSIDE** the window.

(2) Line is VISIBLE if both region codes are 0000

Line is NOT VISIBLE if

bitwise logical AND of code ≠ 0000




Codes

1 → $y = y_{max}$

$y = y_{min}$

$x = x_{max}$

$x = x_{min}$.

0100

0100

1000

The 1 in the code is changed to 0. ie.

If bit 1 is 1, intersect with line $y = y_{max}$

" 2 " , " $y = y_{min}$

" 3 " , " $x = x_{max}$

" 4 " , " $x = x_{min}$.

If C is chosen then → 0100

$\therefore y = y_{min}$ is selected for intersec$^n$.

If D is chosen then → 1010

$\therefore y = y_{max}$ or $x = x_{max}$ is used.

coordinates of intersection are:

$x_i = x_{min}$ or $x_{max}$ (If Boundary line is vertical)

$y_i = y_1 + m(x_i - x_1)$

Or.

$x_i = x_1 + (y_i - y_1)/m$ (If Boundary line is Horizontal.)

$y_i = y_{min}$ or $y_{max}$

where $m = \dfrac{(y_2 - y_1)}{(x_2 - x_1)}$

we replace end point $(x_1, y_1)$ with intersection pt. $(x_i, y_i)$

↓

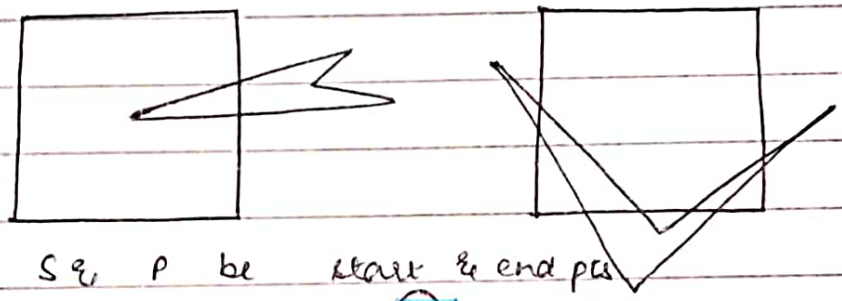
eliminating portion of original line outside the window. & we get new end points.

area clipping.

## POLYGON CLIPPING :

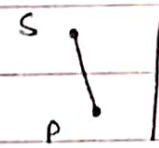
(Sutherland Hodgeman)

Sutherland Hodgeman



Let S & P be start & end pts

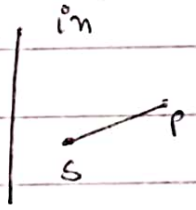
(1) out in




S & P outside

∴ NO OUTPUT

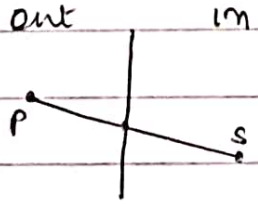
(2) out in




S & P inside

OUTPUT = P

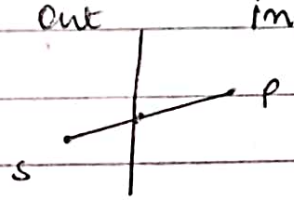
(3) out in




S → inside
P → outside

OUTPUT is intersection

(4) Out in




S → outside
P → inside

OUTPUT is intersection & P.

## ANTI-ALIASING :

**Scan conversion:** Systematic approach to mapping objects in continious space to discrete approximation.

Forms of distortion resulting from above is collectively called ALIASING. effects of scan conversion.

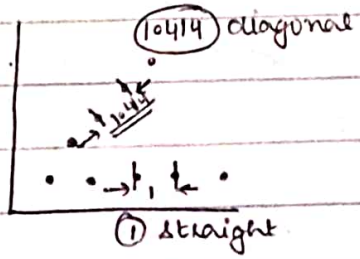
Eg: Scan converting a line give us jagged or staircase appearance.

## UNEQUAL BRIGHTNESS :

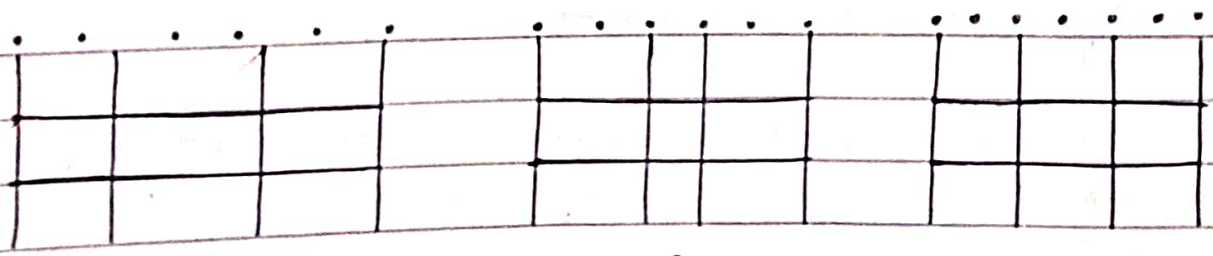
Slanted line appears dimmer than horizontal or vertical line.

Although, all are presented at the same intensity level.

Reason




## PICKET FENCE PROBLEM :



object doesn't fit pixel grid.

normal scan conversion (global aliasing)

equal spacing (local aliasing)

## Anti-Aliasing:

- Increase image resolution

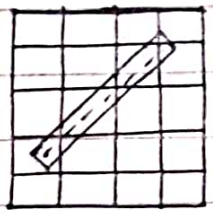
But, going from W x H to 2W x 2H means usage of more system resources.
& Results are not always good.

Two types of techniques:
(a) Prefiltering
(b) Postfiltering

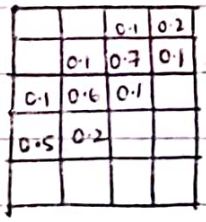
(a) **PREFILTERING:**

works on TRUE SIGNAL in CONTINIOUS SPACE to derive proper values for individual pixels.

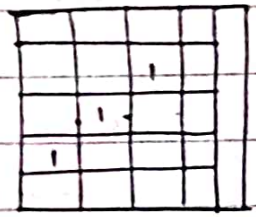
# • Area sampling:



Mathematical line (dotted) represented by rectangular region 1 pixel wide.




Black background & white line




ordinary scale

object

% of overlap → rectangle & intersecting pixel calculated analytically.

Background = gray (0.5, 0.5, 0.5)
Line = green (0, 1, 0)

Each pixel has value $[0.5(1-f) + 0.5(1-f) + f \times 1 + 0.5(1-f)]$

(b) **POST FILTERING :**

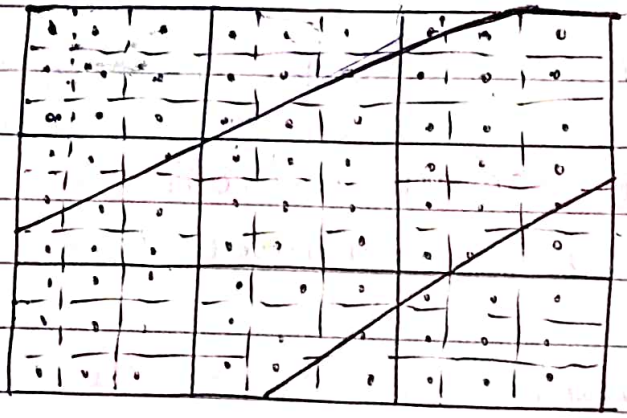
# • super sampling :

objects contributing / contribution to pixel's overall intensity

↓ proportional to

No. of sub pixels inside the area covered by object.

$$0.5(1-f) + 0 \times f$$



| | | |
|---|---|---|
| 0 | 2/9 | 7/9 |
| 6/9 | 1 | 8/9 |
| 1 | 8/9 | 1/9 |

each pixel is divided into (3×3) sub pixels.

object → WHITE

Background → BLACK

Take for instance,

object → Red (1, 0, 0)

Background → light yellow (.5, .5, 0)

Pixel on upper right corner will be assigned a value

$$= \left(1 \times \frac{7}{9} + 0.5 \times \frac{2}{9},\ 0.5 \times \frac{2}{9},\ 0\right)$$

$$= \left(\frac{8}{9}, \frac{1}{9}, 0\right)$$

- **Low Pass filtering:**

Pixel is assigned a new value = weighted average of original value & original value of its neighbours.

It is in form of

$(2n+1) \times (2n+1)$ grid

$n \geq 1$

Weighted average = sum of products of each weight in filter TIMES corresponding pixel's original value.

eg FILTER:

| | | |
|---|---|---|
| 0 | 1/8 | 0 |
| 1/8 | 1/2 | 1/8 |
| 0 | 1/8 | 0 |

- **Pixel Phasing:**

It is H/W based Anti-Aliasing technique.

graphics system is capable of shifting individual pixels from their normal position in pixel grid by fraction $\left(\frac{1}{4} \text{ & } \frac{1}{2}\right)$

It is helpful in smoothing out staircase effect without reducing sharpness of edges.

# COLOR MODELS :

fundamental

Color is a fundamental attribute of our viewing experience.

perception

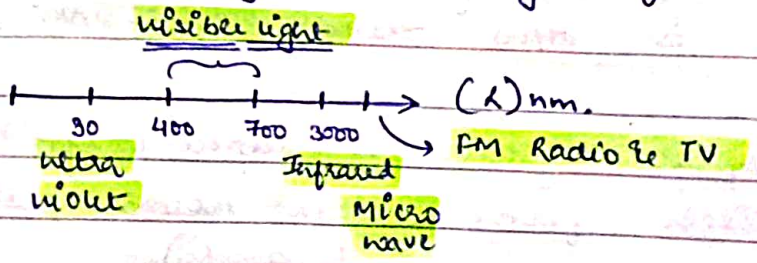
Perception of color arises from light energy entering our visual system and triggering a chain of not yet fully understood neurological and physiological responses.

This complex process is relevant to computer graphics because a realistic image is one that seems indistinguishable from a true record of light energy coming from the real scene.

It has become a constant challenge to find effective and efficient computational models for constructing such images.

## → LIGHT AND COLOR :

Light (visible light) is electromagnetic energy in the 400 to 700 nm ($10^{-9}$ m) wavelength ($\lambda$) range of the spectrum




## → BASIC CHARACTERESTICS OF LIGHT :

Light source such as sun or light bulb emits all frequencies within the visible range to produce white light.

When white light is incident upon some object, some is reflected, some is absorbed. The reflected

Light is what is perceived as color of object.

low frequencies in reflected light = Red

This implies perceived light has dominant frequency or wavelength at red end of the spectrum.

→ dominant frequency is also called HUE.

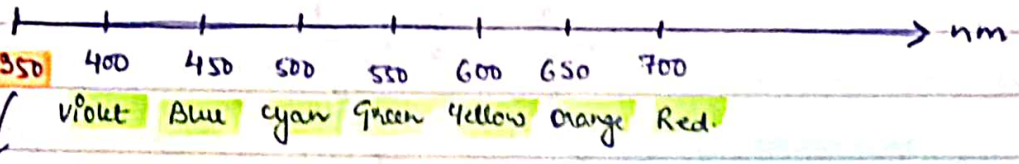
BRIGHTNESS: It is the perceived intensity of light. It is related to luminance. Luminance measure the total energy in the light.

Higher the luminance → Brighter the light to the Observer.

SATURATION / PURITY: It describes how "Pure" the color of light appears.

PALE colors are called "LESS PURE"

CHROMATICITY: is describable Purity & dominant wavelength collectively.




It is also color but retina can't recognize it.

COLOR MODELS

Hardware Oriented
- RGB
- CMY
- CMYK
- YIQ

Intuitive
- HSB
- HLS

## RGB: (Additive color model)

To represent any color using 3 1° colors (RGB) i.e. combining these 3 colors we produce infinite no. of colors within visible spectrum.

→ **conditions** for choosing 3 1° colors:

① No 2 1° colors should produce 3rd 1° color.

② 1° colors should be widely spread.

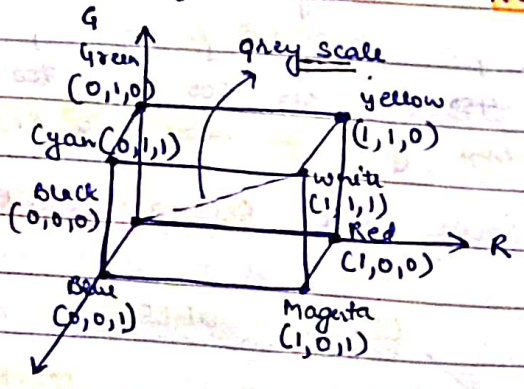
③ combining 3 1° colors should produce white color.

---

**Visual Pigments** in retina have a **peak sensitivity** at wavelengths of about **630 nm (Red)**, **530 nm (green)** and **450 nm (blue)**.

By comparing intensities in a light source, we perceive the color of the light.

This is the **theory of vision** that is basis for **displaying color output** on a video monitor using 3 1° colors R, G, B referred to as **RGB color model**.




Cyan, Magenta, yellow } complementary colors.

Each color pt. within bounds of cube can be represented as (R, G, B) where R, G, B range from 0 to 1.

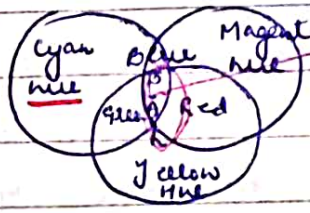
- RB → magenta
- GR → yellow
- GB → cyan

If RGB values are same then we are on grey scale.
It is widely used model in CRT.

CMY :

(1) Primary colors are Cyan, Magenta & yellow.

(2) Useful for describing color O/P to hard copy devices.




→ all three produce black.

(3) We cant produce all the other colors by adding combination of three on a monitor, because whichever (2) we add, we will always have mixture of all 3 1° & hence a color close to white.

We will always end up with tint of white when luminance are higher or tint of grey when luminance are low.

(4) There is no way we can create PURE BLACK

(5) CMY can be viewed as primaries when we consider paints & dyes, which are quite different from light emitting sources.
They never produce their own light, instead they reflect light that falls on them or filter light that passes through them
Absorbing more or less light in the process.

Q: Cyan dye painted on a paper, acts only as pass light from paper surface consisting of only green & blue

wavelengths. Red wavelengths is missing i.e. absorbed by cyan dye.

(6) This color system is SUBTRACTIVE.

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

(7) If we paint all 3 subtractive 1°s with water colors onto a white paper, we don't get black, but greyish brown. Although yellow subtracts blue wavelength but not completely.

CMYK:

(1) CMY model produce every color except black. ∴ we add black (K) as a fourth color, because CMY should theoretically produce B but it doesn't due to imperfections in ink.

(2) Printing on paper requires transparent inks of CMYK. Perfect ink would subtract all the red, blue or green wavelengths from white light, but nothing more. Then black would not be needed.

(3) eg. blue intensity yan + full intensity magenta
= full intensity blue (Theoretically)
= blue with little greyish (Actual)
(because some red & green wavelengths are present)

(4) This imperfection of getting "blue with little grey" can be countered by adding a little black.

→ TRANSFORMATION FROM RGB TO CMYK :

(1) Needed when hard copy of CRT display is required.

R + G = Yellow (on screen)
= Brown (on paper)

R + G + B = White (on screen)
= very dark brown (on paper)

(2) It is not possible or desirable to avoid overlapping diff. colors on paper. It is best to work on paper with 1° that each extinguish only 1 of the R's.

(3) These 1°'s are cyan, Magenta & yellow (CMY). This color is produced on paper by subtracting unnecessary R's from original white.

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

(4) 1 is the highest intensity in both color models. This formula is theoretical as it assumes pure intensities of CMY produce black, which is not so.

(5) → For real hard copy O/p we use CMYK.

$$C = \max(RGB) - R$$
$$M = \max(RGB) - G$$
$$Y = \max(RGB) - B$$
$$K = 1 - \max(RGB)$$

→ max(RGB) is the maximum intensity of 1° of the RGB combination.

Eg. To transform a dark greenish yellow on the screen to proper CMYK values.

(6) Dark greenish yellow is (RGB) = (0.4, 0.5, 0)

max (RGB) = 0.5

∴ C = .5 − .4 = .1

M = .5 − .5 = 0

Y = .5 − 0 = .5

K = 1.0 − .5 = .5

We obtain mainly a mixture of yellow & black dots on the paper, resulting in dark yellow. Bit of cyan will give dark yellow a greenish tint.

In printing industry, transformation from RGB to CMYK involves more complicated conversion formulas for several reasons.
One reason being imperfections in inks. Also, transparencies of inks is not upto the mark.
So, to produce a printed picture like the CRT display is largely a matter of personal experience & intuition of those running printing press.

YIQ:

(1) RGB monitor requires separate signals for R, G & B component of an image.
Television uses a single composite signal.

(2) It is used because:
① – better transmission efficiency.
② – Higher / better downloading compatibility in signal for B/W TV.

(3) Y → luminance (brightness)

I, Q → chromaticity info (Hue & Purity)

(4) B/W TVs use only Y- signal.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} .299 & .587 & .144 \\ .596 & -.275 & -.321 \\ .212 & -.528 & .311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RGB converted to YIQ using NTSC encoder, then superimposes I & Q information on Y signal for B/W TV's.

(5) We compress filters to export compression implies compression of luminance & chromaticity. Human eye can perceive change in luminance than in chromaticity

∴ compress chromaticity channel only.

lossy compression techniques are used for chromatic values.

(6) NTSC video signals can be converted to RGB signal using NTSC decoder.

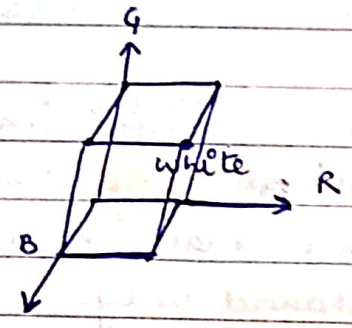
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -.272 & -.647 \\ 1.000 & -1.108 & 1.705 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

## INTUTIVE MODELS / SOFTWARE ORIENTED MODELS :

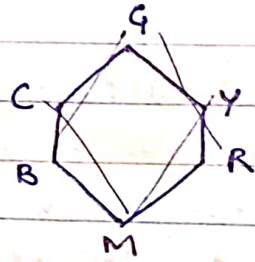
### HSV :

These models are available in any graphics s/w.
To give a color specification ./ user selects a spectral color & amounts of white & black to be added to obtain diff shades, tints & tones.

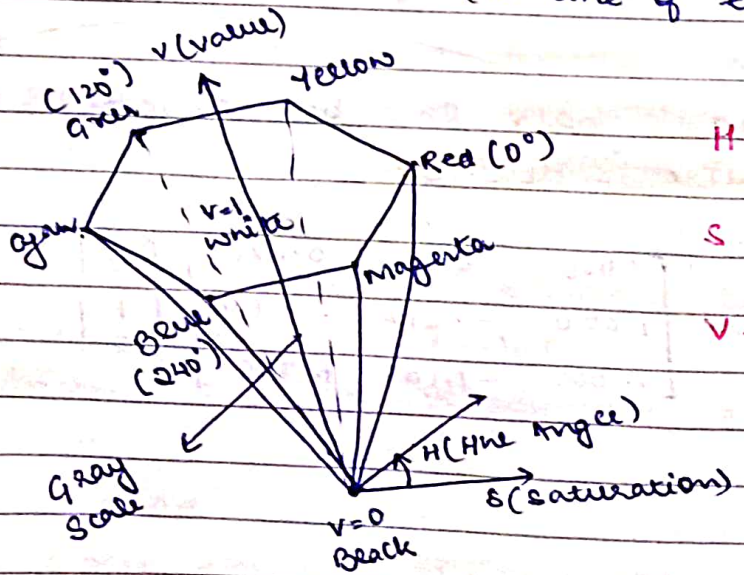
color parameters in their model are { Hue (H), Saturation (S) } & { Value (V) }




RGB color cube




View of cube along the diagonal (from white to black.) grey scale.
(outline of this view = Hexagon)




H = 0° / 360°

S = 0 gray scale
1 Pure color

V = 0 black
1 white

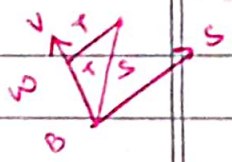
V=1, S=1 Pure Hue
V=1, S=0 white

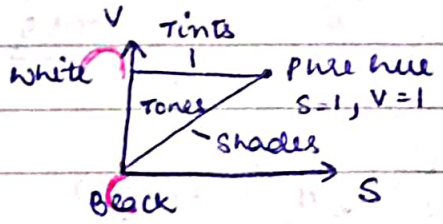
| H | S | V |
|---|---|---|
| 0° – 360° | 0 – 1 | 0 – 1 |
| | S=1 (Pure color) | V=1 (white) |
| | S=0 (gray scale) | |

If, V=1 & S=1 we have pure Hues

If, V=1 & S=0 we have white

Starting with selection for a pure hue, which is the angle (H) & set V=S=1, we describe color we want in terms of adding either white or black to pure hue.




+B ↓V S✓

Adding Black, decrease V & S is constant.

Adding white, V= constant, S is decreased.

+W V✓ S↓

Adding B & W, we decrease both V & S +B&W ↓V&S

Interface for HSV model is a color pallet. (?)

HLS:

H= Hue L = Lightness S= Saturation.

(H)

H = 0° Blue

H = 60° Magenta

H = 120° Red

H = 180° Cyan

0 60 120 180

BMRC.

Pure Hue at L = 0.5 plane

S → 0 to 1 ( Relative purity of color)

Pure Hues → S=1 , L= 0.5

S=0 → gray scale

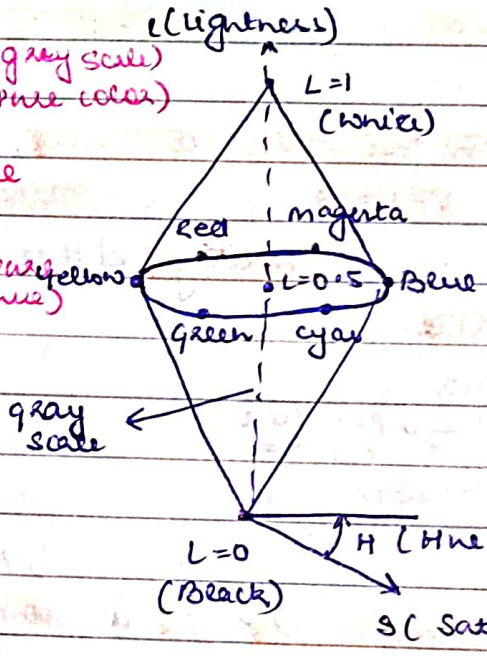
As S decrease, hues are less pure.

S = 0 (gray scale)
1 (Pure color)

H = angle

L = 1
0.5 (Pure hue)
0

Pure Hue.




HLS DOUBLE CONE

Desired shade, tint or tone is obtained by adjusting L & S.

Colors lighter by increasing L
" darker " decreasing L

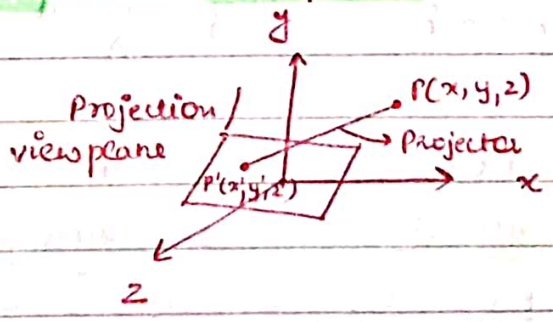
If S is decreased, colors move towards gray.

# PROJECTIONS

For centuries artists, engineers, designers, drafters & architects have tried to come to terms with the difficulties & constraints imposed by the problem of representing a 3-D object or scene in a 2-D medium (The problem of projection)

→ It can be defined as a mapping of points P(x,y,z) onto its image P'(x', y', z') in the projection plane or view plane which constitutes the display surface.

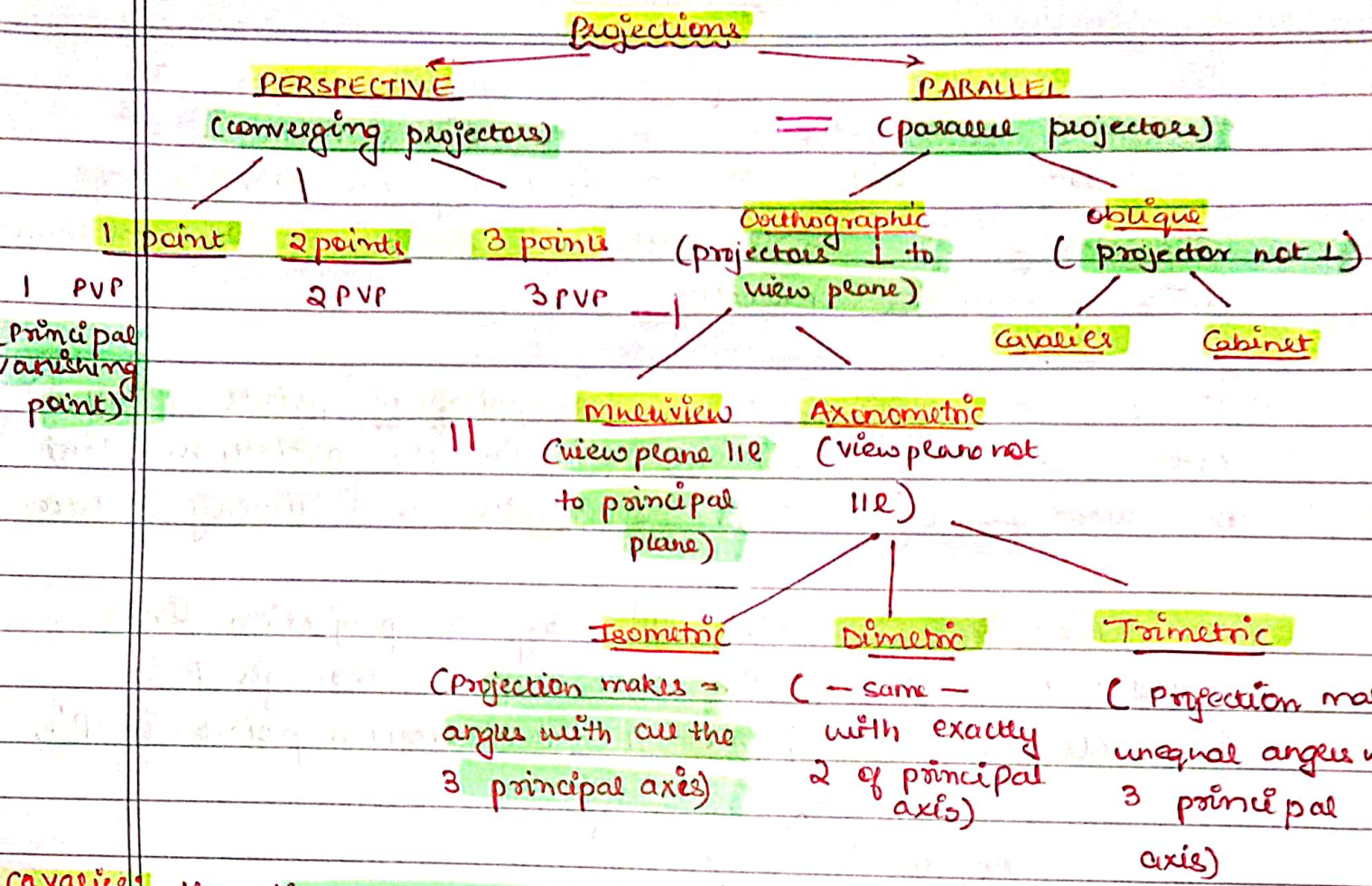
The mapping is determined by a projection line called the PROJECTOR. that passes through P & intersects the view plane. The intersection point is P'.




The result of projecting an object is dependent on the spatial relationship among the projectors & the spatial relationship b/w the projectors & the view plane.

Two basic method of projection are perspective & parallel

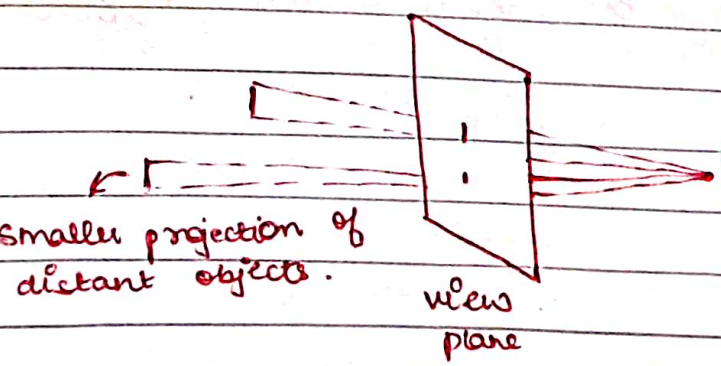
# TAXONOMY OF PROJECTIONS




**Cavalier:** the direction of projection is chosen so that there is no fore shortening of lines ⊥ to the xy plane.

**Cabinet:** the direction of projection is chosen so that lines ⊥ to the xy planes are fore shortened by half (1/2) their length

\# Perspective projection of equal sized obj. at diff dist from the view plane.



In perspective projection using a 3-D object, a set of parallel lines in object that are not ||el to the plane are projected into converging lines.
The converging point is called VANISHING POINT.

A scene can have any no. of vanishing pts depending upon set of ||el lines in scene.

The vanishing pt for any set of lines that are parallel to one of the principal axes of the obj is called PRINCIPAL VANISHING POINT.
We control the no. of PVP (1, 2 & 3) with the orientation of the projection plane.

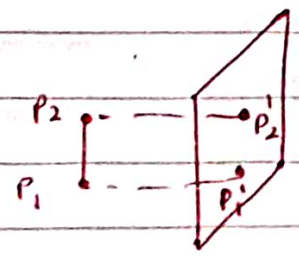
Anomalies:

① Farther the obj from center of projection, the smaller it appears.

② Due to vanishing pts, railroad tracks appear to meet at a point on the horizon.

③ Obj behind the centre of projection are projected upside down & backward onto the view plane.

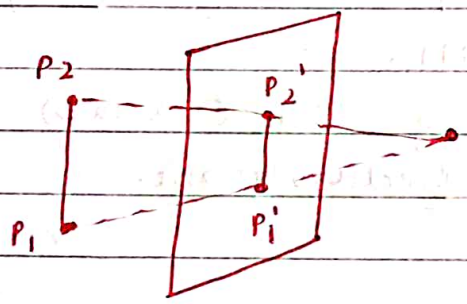
# In Parallel projection, coordinate positions are transformed to the view plane along parallel lines as shown below




Obj position is transformed to view plane along ||el lines

# For a perspective projection obj. positions are transformed to the view plane along lines that converge to a point called the PROJECTION REFERENCE POINT or CENTRE OF PROJECTION.
It is the viewer's eye.

The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.




# Parallel projection preserves relative proportions of obj & this is the method used in drafting to produce scale drawings of 3-D objects. // Accurate views of the various sides of an obj. are obtained with ||e projection, // but realistic views are obtained by perspective projection.

# 2-D GEOMETRIC TRANSFORMATIONS

The ability to simulate the manipulation of obj in space is called TRANSFORMATIONS.

GEOMETRIC  COORDINATE

basic geometric transformations are :
- Translation
- Rotation
- Scaling

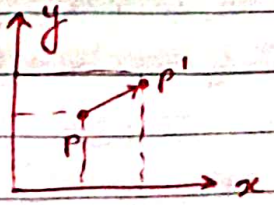
others are :
- Reflection
- Shear

## TRANSLATION :

Translation is applied to objects by repositioning it along a straight line path from 1 coordinate location to another.

Translating a 2D point involves adding translation distances $(t_x)$ & $(t_y)$ to original co-ordinate position $(x, y)$ to move point to new position $(x', y')$

$$x' = x + t_x \quad , \quad y' = y + t_y$$




$(t_x, t_y)$ is known as TRANSLATION VECTOR

Or SHIFT VECTOR.

$$P = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad P' = \begin{bmatrix} x_1' \\ y_1' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$P' = P + T$$

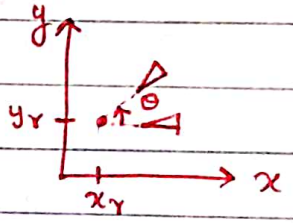
## ROTATION:

2D rotation is applied to objects by repositioning it along a circular path in xy plane.

Rotation angle (θ), position of rotation point (pivot point) ie $(x_r, y_r)$ about which the obj is to be rotated.

counterclockwise means θ → +ve

clockwise means θ → −ve




→ Transformation equations for rotation of a pt P when pivot pt is at origin.

$\frac{x'}{r} = \cos(\phi + \theta)$ ie $x' = r\cos(\phi + \theta)$

$x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta$ —①

$\frac{y'}{r} = \sin(\phi + \theta)$ ie

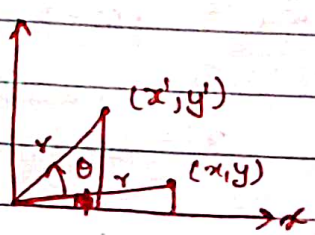
$y' = r\cos\phi\sin\theta + r\sin\phi\cos\theta$ —②

PBP / HHB (S C T)

Also,

$x = r\cos\phi$ —③

$y = r\sin\phi$ —④




Putting ③ & ④ in ① & ②

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

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

$P' = R \cdot P$

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

→ for rotation about $(x_r, y_r)$

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

**SCALING :**

They alter the size of an obj. / This is achieved by multiplying the coordinate values $(x, y)$ of each vertex of polygon by scaling factor $(S_x, S_y)$ to get transformed coordinates $(x', y')$

$x' = x \cdot S_x$ (Scale in x direc$^n$) $\qquad y' = y \cdot S_y$ (Scale in y direc$^n$)

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

$$P' = S \cdot P$$

if $S_x = S_y$ — UNIFORM SCALING

$S_x \neq S_y$ — DIFFERENTIAL SCALING.

They can take any numeric +ve values.

values $< 1$ reduce size of obj

$> 1$ produce enlargement of obj

$= 1$ size unchanged.

location of scaled obj. can be controlled by choosing a fixed point, that is to remain unchanged. A polygon is scaled relative to the fixed point.

$$x' = x_f + (x - x_f)S_x \rightarrow x' = x \cdot S_x + x_f(1 - S_x)$$
$$y' = y_f + (y - y_f)S_y \rightarrow y' = y \cdot S_y + y_f(1 - S_y)$$

## Matrix representation & homogeneous coordinates:

each of basic transformations can be expressed as in general

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

$M_1$ = 2×2 array Matrix containing multiplicative factors

$M_2$ = 2 element column matrix containing translational terms.

In translation, $M_1$ = Identity matrix

In rotation or scaling $M_2$ = translational terms associated with pivot pt (↑) or scaling fixed pt (↑)

For sequence of transformations, we must calculate transformed coordinates (one) step at a time.

↓

Better approach is to combine the transformations so that final coordinate positions are obtained from initial coordinates, thus eliminating calculation of intermediate coordinate values.

For combined transformation we change eq (A) we expand 2×2 matrix to 3×3 matrices. allowing to represent all transformation eqⁿ as MATRIX MULTIPLICATION

We represent each cartesian coordinate position $(x, y)$ with the homogeneous coordinate triple $(x_h, y_h, h)$

where $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$

General homogeneous coordinate representation is

$$(h \cdot x, h \cdot y, h)$$

for 2-D transformation, we choose $h$ to be any non zero value. For convenience we set $h = 1$.

Each 2D position is represented in homogeneous coordinates as $(x, y, 1)$

↓

expressing positions in homogeneous coordinates allows to represent all geometric transformations eq$^n$ as MATRIX MULTIPLICATIONS.

For translation:

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

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

Inverse of translation matrix $T$ is obtained by replacing translation parameters $t_x$ & $t_y$ by $-t_x$ & $-t_y$

For rotation: (About origin)

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

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

Inverse of $R(\theta)$ is obtained by replacing $\theta$ with $-\theta$

For scaling:

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

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

Inverse of $S$ is obtained by replacing $S_x, S_y$ by $\frac{1}{S_x}$ & $\frac{1}{S_y}$

## Composite Transformations:

It is matrix product of the individual transformations.

→ Product of Transformation matrices is called concatenation or composition of matrices.

→ **Translations:**

Two successive translation vectors $(tx_1, ty_1)$ and $(tx_2, ty_2)$ are applied to P, then.

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

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

$$T(tx_2, ty_2) \cdot T(tx_1, ty_1) = T(tx_1 + tx_2, ty_1 + ty_2)$$

→ **Rotations:**

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

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

→ **Scaling:**

$$\begin{bmatrix} sx_2 & 0 & 0 \\ 0 & sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx_1 & 0 & 0 \\ 0 & sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} sx_1 \cdot sx_2 & 0 & 0 \\ 0 & sy_1 \cdot sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or $S(s_{x_2}, s_{y_2}) \cdot S(s_{x_1}, s_{y_1}) = S(s_{x_1} \cdot s_{x_2}, s_{y_1} \cdot s_{y_2})$

ie scaling operations are MULTIPLICATIVE

# General Pivot point Rotation :

① Translate the obj. so that pivot pt position is moved to coordinate origin.

② Rotate the obj about the coordinate origin.

③ Translate the obj so that pivot pt is returned to its original position.

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

$$T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r) = R(x_r, y_r, \theta)$$

where

$$T(-x_r, -y_r) = T^{-1}(x_r, y_r)$$

# General fixed pt Scaling :

Let fixed pt be $(x_f, y_f)$

① Translate obj. so that fixed pt coincides with origin.

② Scale obj wrt coordinate origin.

③ use inverse Translation of step 1 to return obj to its original position.

$$\begin{bmatrix} 1 & 0 & x_b \\ 0 & 1 & y_b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_b \\ 0 & 1 & -y_b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_b(1-s_x) \\ 0 & s_y & y_b(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(x_b, y_b) \cdot S(s_x, s_y) \cdot T(-x_b, -y_b) = S(x_b, y_b, s_x, s_y)$$

→ **REFLECTION :**

It produces mirror image of an obj.
Reflection is generated wrt axis of reflection by rotating object by 180° about reflection axis.

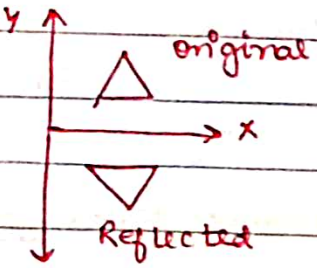
Reflection about line $y=0$ i.e. x-axis is by

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

This keeps x values but flips y values by of coordinate position.

Reflection about y-axis flips x coordinate keeping y coordinate as same.

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




→ **SHEAR :**

It distorts shape of an object.

$y' = y$ $\quad x' = x + Sh_x \cdot y$ $\quad$ (shear along x-axis or wrt x-axis)

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

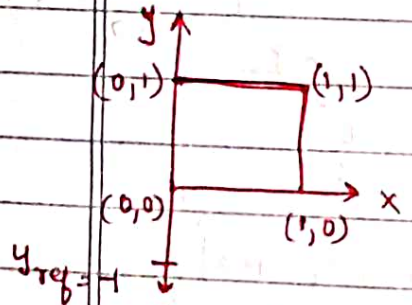
$x' = x$ $\quad y' = y + Shy \cdot x$ (Shear wrt y-axis)

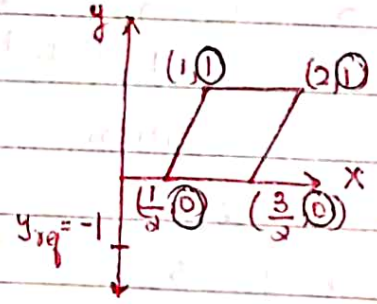
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Now,


$y' = y$ $\quad x' = x + Sh_x(y - y_{ref})$ {x-direction shear relative to ref. line}


$x' = x$ $\quad y' = y + Shy(x - x_{ref})$ {y-direction shear relating to ref line $x = x_{ref}$.




A unit sq. is transformed to a shifted parallelogram with

$$Sh_x = \frac{1}{2} \quad \& \quad y_{ref} = -1$$

✓ $y' = y = 0$.

① $(\frac{1}{2},0)$ $x' = 0 + \frac{1}{2}(0 - (-1))$

$= 0 + \frac{1}{2}(1)$ $\quad \therefore (\frac{1}{2},0)$

$= \frac{1}{2}$.

$y' = y = 1$

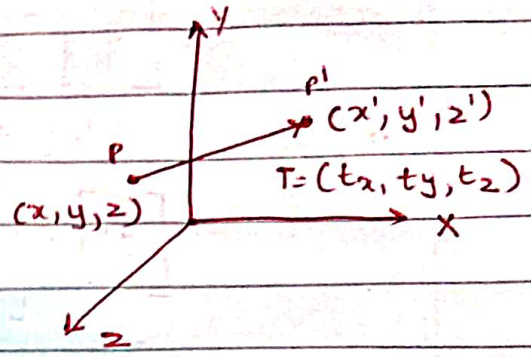
$x' = 1 + \frac{1}{2}(1 - (-1))$

$= 1 + \frac{1}{2} \times 2 = 2$.

cal for all coordinates.

## TRANSLATION:

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



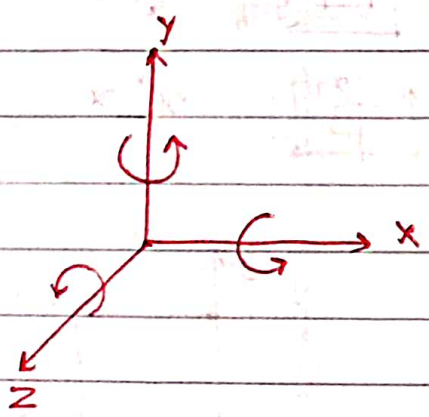

$$P' = T \cdot P$$

$$x' = x + t_x \quad , \quad y' = y + t_y \quad , \quad z' = z + t_z$$

## ROTATION:

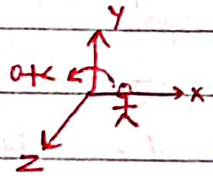
We must designate an axis of rotation.

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




counter clock wise = +ve rotation.

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}\right\} \text{ z-axis rotation}$$

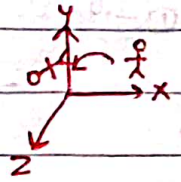



$$P' = R_z(\theta) \cdot P$$

(2.) Replace $x \to y$, $y \to z$, $z \to x$ for respective axis rotation.

for x-axis rotations

$$\left.\begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned}\right\}$$




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

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

# y-axis rotation

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

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

$y' = y$

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


ie. $P' = R_y(\theta) \cdot P$

# General 3-D rotation:

obj is rotated about axis not ||e to one of the coordinate axis.

STEPS:

① Translate obj so that rotation axis passes through coordinate origin.

② Rotate the obj so that the axis of rotation coincides with one of the coordinate axis.

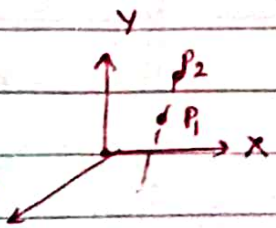
③ Perform the specified rotation about that coordinate axis.

④ Apply inverse rotations to bring the rotation axis back to its original orientation.

⑤ Apply invese translation to bring rotation axis back to its original position.

Let the rotation axis be defined by 2 pts & direction of rotation in counterclockwise.

When looking along the axis from $P_2$ to $P_1$.




Axis vector is:

$V = P_2 - P_1$

$= (x_2 - x_1, y_2 - y_1, z_2 - z_1)$

Unit vector (u) is defined along the rotation axis as

$$u = \frac{V}{|V|} = (a, b, c)$$

where components a, b, c of unit vectors u are

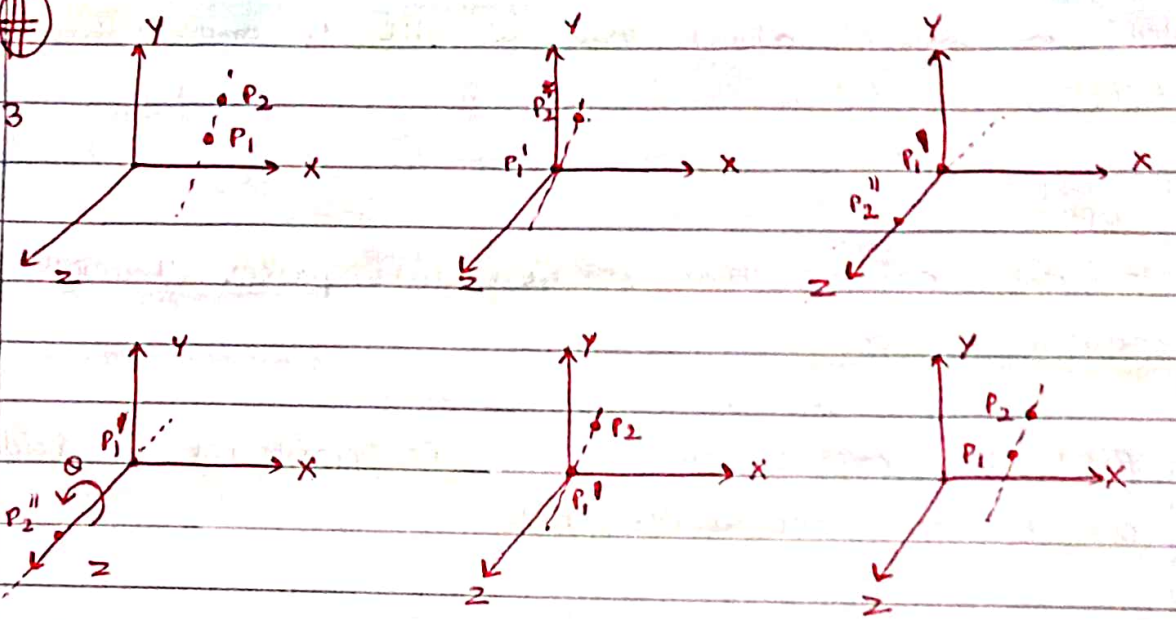
$$a = \frac{x_2 - x_1}{|V|}, \quad b = \frac{y_2 - y_1}{|V|}, \quad c = \frac{z_2 - z_1}{|V|}$$

(2) Fig 285 (#)

$P_2$ or $P_2'$

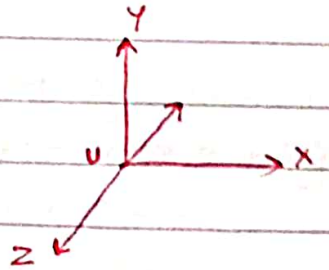
- Fig 5, 4, 3

$P_1''$ or $P_1'$




→ 1st step. $T_2 = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

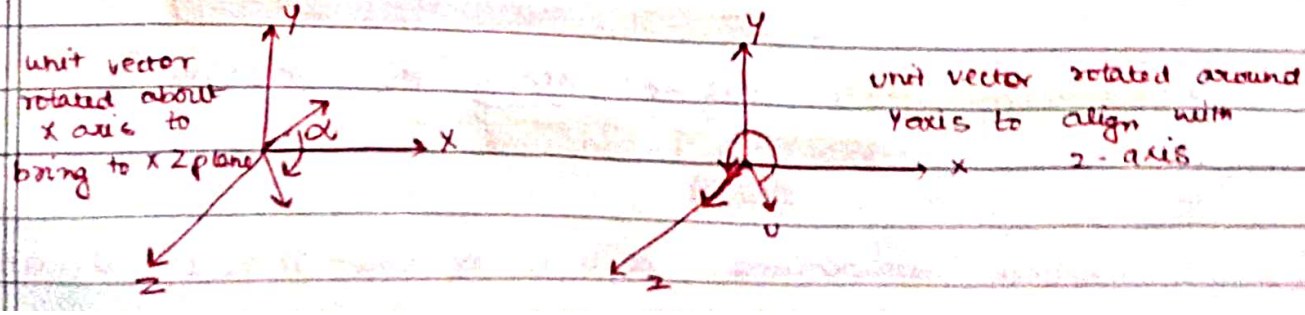
Translation of rotation axis to origin.




→ 2nd step. Put rotation axis on the Z-axis

# 1st rotate about X-axis to transform vector U into XZ plane.

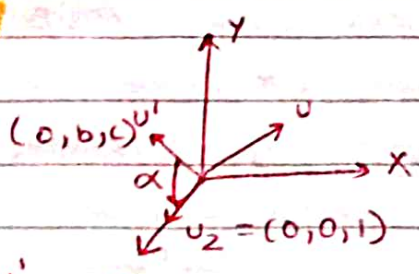
# Then swing U around to Z-axis using Y-axis rotation.



Rotation angle is angle b/w the projection of v in the yz plane and the +ve z axis

Let (v') be projection of v in yz plane, then

$v' = (0, b, c)$




$$\cos\alpha = \frac{v' \cdot v_z}{|v'||v_z|} = \frac{c}{d}$$

where (d) is magnitude of v' $\rightarrow$ $d = \sqrt{b^2 + c^2}$

Also,

$$v' \times v_z = v_x |v'||v_z| \sin\alpha \quad \text{——} (1)$$

$$v' \times v_z = v_x \cdot b \quad \text{——} (2)$$

Now,

$|v_z| = 1$ &

$|v'| = d$

considering, RHS of eq (1) & (2) we get

$$v_x |v'||v_z| \sin\alpha = v_x \cdot b$$

i.e.

$$d \sin\alpha = b$$

$$\sin\alpha = \frac{b}{d}$$

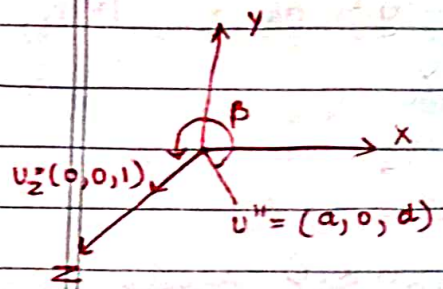
Matrix for rotation of v about x axis is

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

by this matrix vector v rotates itself into xz plane.

→ **Step 3.**

Transformation matrix for transforming unit vector in x z plane around y axis onto +ve z-axis.




$$\cos\beta = \frac{U'' \cdot U_z}{|U''|\,|U_z|} = d$$

Since,

$$|U_z| = |U''| = 1$$

$$U'' \times U_z = U_y\,|U''|\,|U_z|\sin\beta$$

$$U'' \times U_z = U_y \cdot (-a)$$

$$\sin\beta = -a$$

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

→ **Step 4.**

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

$$R(\theta) = T^{-1} \cdot R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ now, $R \cdot \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$, $R \cdot \begin{bmatrix} r_{21} \\ \vdots \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$,

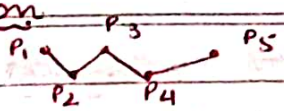
$$R \cdot \begin{bmatrix} r_{31} \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

# NOTE :

$$\begin{cases} V_1 \cdot V_2 = V_{1x}V_{2x} + V_{1y}V_{2y} + V_{1z}V_{2z} \\ V_1 \times V_2 = \left(V_{1y}V_{2z} - V_{1z}V_{2y},\ V_{1z}V_{2x} - V_{1x}V_{2z},\ V_{1x}V_{2y} - V_{1y}V_{2x}\right) \end{cases}$$

# 3-D object Representation

**POLYLINE:** chain of connected line segments




**POLYGON:** closed polyline.

**WIREFRAME MODELS:** consists of edges, vertices & polygons.
Edges may be curved or straight line segments.

**POLYGONAL MESH:** collection of polygon.

**MODELLING OF OBJECTS:** Numerical description of the obj. in terms of their geometry (size, shape) & interaction with light.

wireframe represen" of cylinder with back (hidden lines removed.

**CURVED SURFACES:** allows higher level of modelling for highly realistic models.

① Model obj. by using small curved surface patches placed next to each other.

② Another way is use surface that define solid objects such as q. cylinders, spheres.
Model is then constructed with these solid objects used as building blocks.
This is known as SOLID MODELING.

→ **WAYS TO CONSTRUCT A MODEL:**

- Additive modelling
- Subtractive Modelling

**POLYGONAL NET:** Is polygonal mesh where polygons are made of straight line edges.

Representation of polygon net Model.

① **EXPLICIT VERTEX LIST:**

$V = \{ P_0, P_1, \ldots P_N \}$

$P_i = (x_i, y_i, z_i)$ are vertices stored in order in which they would be encountered by travelling around the model.

$P_k = (x_i, y_i, z_i)$

$V = \{ P_0 \ldots P_n \}$

Disadv: Shared vertices & edges repeated several times.

② **POLYGON LISTING:** vertex stored exactly once in vertex list.

$V = ( P_0 \ldots P_N )$

each polygon is defined by pointing into this vertex list. / Shared edges drawn several times in displaying the model

③ **EXPLICIT EDGE LISTING:** A vertex list is kept, storing each vertex exactly once. / Edge is also stored exactly once.

Edge points to 2 vertices.

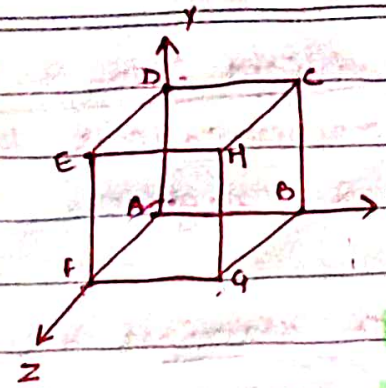
Polygon is represented as list of pointers into the edge list.

$P_1 = \{ \overline{AB}, \overline{AD}, \overline{BC}, \overline{CD} \}$

Polygons sharing a specific edge can be identified by extending edge's representation to include pointers to those polygons.

eg: $\overline{AB} \rightarrow P_1, P_4$ $\overline{AD} \rightarrow P_1, P_3$

Above 3 categories belong to representation of a Polygon net model.




Explicit vertex list =

V = { ABCDAFED CHEF GHGB }

Polygon listing =

V = { ABCDEFGH }

$P_1$ = [ABCD] $P_2$ = [CDEH]

$P_3$ = [ADEF] $P_4$ = [ABGF]

## 3-D OBJECTS REPRESENTATIONS

Graphics scenes can have diff. kinds of obj & material surfaces such as trees, flowers, clouds, papes etc.

There is (no) single method to describe obj. that includ all the characteristics of these diff. materials.

polygon & quadric surfaces.

Polygon & quadratic surfaces provide precise descriptions for simple euclidean obj. such as polyhedrons & ellipsoids // spline surfaces & constructive solid geometry techniques are useful for designing aircraft wings etc.

spline surfaces & construction solid geometry tech.

procedural methods
- Particle system
- fractal construction

Procedural methods, such as fractal constructions & particle systems, to model terrain features, clouds etc.

octree representation Encoding

Octree encodings are used to represent internal features of objects, such as those obtained from medical CT images.

visualization techniques
Vol. rendering

Volume renderings & other visualization techniques are applied to 3-D discrete data sets to obtain visual representation of the data.

Representation for solid obj divided into 2 broad categories, although not all representations fall neatly into one or other of these 2 categories.

① Boundary representation: It describes a 3-D obj as a set of surface that seperate the obj interior from the environment.
Eg. ✓ polygon facets
✓ spline patches.

② Space partitioning representations: They are used to describe interior properties, by partitioning the spatial region containing an obj into a set of small non-overlapping contiguous solids (usually cubes)


common space partitioning discription for a 3-D obj is an octree representation.

Most commonly used boundary representation for a 3-D obj is set of surface polygons that enclose the obj interior.

Eqⁿ for obj with curved boundaries can be expressed in either a parametric or non-parametric form.
The various obj that are often useful in graphics applications include quadric surfaces.
✓ super quadrics
✓ polynomial & exponential functions
✓ spline surfaces.

These input obj descriptions are often tessellated to produce polygon - mesh approximations for the surfaces.

Sphere (quadric surface) with radius (r), centered on coordinate origin is defined as the set of pts (x, y, z) that satisfy eq$^n$

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

↓

Super quadrics are generalization of quadric representa$^n$ they provide more flexibility in adjusting obj shape

## BLOBBY OBJECTS :

fluidity

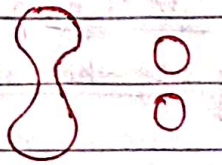
obj such as molecular structure, liquids & water droplets, human muscle shape etc exhibit a degree of fluidity.

change surface characteristics in certain motion

These obj. change their surface characterestics in certain motion (or) when in proximity to other objects & they have curved surfaces that cannot be easily represented with std. shapes.

No sp's by std shapes.

As a class they are refered to as Blobby obj.



Two molecules move away from each other and finally seperate into 2 spheres.

## NON-PARAMETRIC & PARAMETRIC CURVE

3-D curve line can be represented in non-parametric form as

$$y = f(x)$$
$$z = g(x)$$

$x$ is independant variable varying from starting point of curve to ending point.

Parametric Curve: Path of a 3-D curve can be described with a single parameter (u). ie. we can express each of 3 cartesian coordinates in terms of parameter (u) and any point on the curve can be represented as

$$P(u) = (x(u), y(u), z(u))$$

$u$ is varied over range of 0 to 1.

## SPLINE REPRESENTATIONS:

It is a flexible strip used to produce a smooth curve through a disignated set of points. / several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.

In CG spline curve now refers to any composite curve formed with polynomial sections, satisfying specified continuity conditions at the boundary of the pieces.

## Interpolation & Approximation Splines:

Interpolation Spline: When curve passes through each control point ie → I-Spline

Approximation spline: when curve is made without necessarily passing through (any) control point.


→ Interpolation curves are commonly used to digitize drawings or specify animation paths.

→ Approximation curves are primarily used as design tools to structure obj. surfaces.

→ Parametric continuity conditions:

① Zero order Parametric continuity: curves meet.

② First order Parametric " : 1st parametric derivatives (tangent lines) of 2 curve sections are same at intersection

③ Second order " " : both 1st & 2nd parametric derivatives of the 2 curve sections are same at intersection

→ Geometric continuity conditions:

① Zero order geometric continuity: — same as above PC —

② First order " " : parametric 1st derivatives are proportional at the intersection of 2 successive sections

③ Second order " " : both 1st & 2nd parametric derivatives of the 2 curve sections are proportional at boundary

# SPLINE SPECIFICATIONS:

3 equivalent methods for specifying a particular spline representation, given degree of polynomial & control points.

① we can state boundary conditions that are imposed on the spline.

② we can state the matrix that characterizes the spline.

③ we can state the set of blending functions (a basic functions) that determine how specified constraints on the curve are combined to calculate positions along the curve path.

Suppose, we have following parametric cubic polynomial for the (x) coordinate along the path of a spline curve section:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \qquad 0 \leq u \leq 1$$

Boundary conditions for this curve can be set for the end points $x(0)$ & $x(1)$

& for parametric 1st derivatives at the end points $x'(0)$ & $x'(1)$

These (4) boundary conditions are sufficient to determine the values of 4 coefficients

→ $a_x$, $b_x$, $c_x$ & $d_x$

From the boundary conditions we can obtain the matrix that characterizes this spline curve by 1st rewriting the above equn. as the matrix product:

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

i.e. $x(u) = U \cdot C$

## BEZIER SPLINE CURVE:

This spline approximation method was developed by French engineer Pierre Bezier for use in the design of Renault automobile bodies.
Bezier splines have a no. of properties that make them highly useful & convenient for curve & surface design.

They are easy to implement. For these reasons Bezier splines are widely available in various graphics packages.

In general Bezier curve sections can be fitted to any no. of control points, although some graphics packages limit the no. of control points to 4.

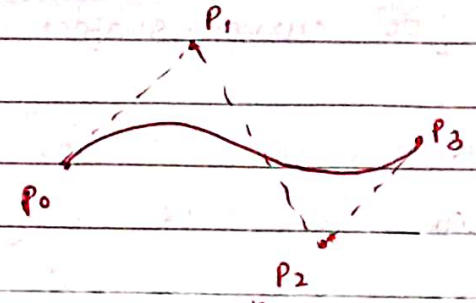
Degree of bezier polynomial is determined by no of control pts to be approximated & their relative position

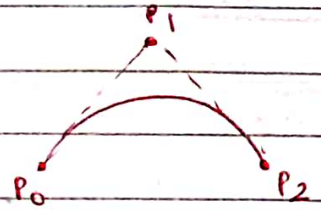
For general bezier curves with no restrictions on the no. of control pts, the blending function specification is the most convenient representation.

As with interpolation splines, a Bezier curve can be specified with boundary conditions, with a characterizing matrix, or with blending functions.

specified with boundary conditions
matrix
blending function.

Suppose we are given n+1 control point positions $P_k = (x_k, y_k, z_k)$ with (k) varying from 0 → n. These coordinate pts can be blended to produce the following position vectors (P(u)), which describes path of an approximating bezier polynomial function b/w $P_0$ & $P_n$.

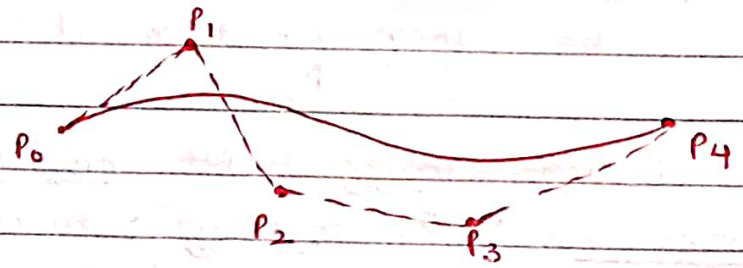



$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u) \qquad 0 \le u \le 1 \qquad \text{——— (1)}$$

The eq$^n$ (1) represents a set of 3 parametric equations for the individual curve coordinates at (2), (3), (4)

BEZIER CURVE




n+1 control pts generates polynomial of degree (n).

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

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

↓

Bernstein polynomials

Take 4 control pts & degree = 3

$$C(n,k) = \frac{n!}{k!(n-k)!} \qquad n=3$$

At $k=0$

$$P_0 \times 1 \cdot U^0 (1-U)^3$$

At $k=1$

$$P_1 \times \frac{3!}{1!\,2!} \cdot U^1 (1-U)^2$$

At $k=2$

$$P_2 \times \frac{3!}{2!\,1!} \cdot U^2 (1-U)$$

At $k=3$

$$P_3 \times \frac{3!}{3!\,0!} \cdot U^3 (1-U)^0$$

$$\rightarrow \quad x(U) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(U) \qquad \text{——— (2)}$$

$$\rightarrow \quad y(U) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(U) \qquad \text{——— (3)}$$

$$\rightarrow \quad z(U) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(U) \qquad \text{——— (4)}$$

$$P(U) = P_0(1 - U^3 + 3U^2 - 3U) + 3P_1(U + U^3 - 2U^2) + 3P_2(U^2 - U^3) + P_3U^3$$

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

$$= \begin{bmatrix} U^3 & U^2 & U & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Disadvantage:

① Global control.

② Degree of curve is function of no. of control points.

Properties:

① Curve passes through 1st & last control points.

② Tangent to curve at end points is along the line joining that end point to the adjacent control pt.

③ Bezier curve lies within the convex hull of the control points.

B-Spline curves

They are generated by approximating a set of control pts.
They have 2 adv. over bezier:

① The degree of b-spline polynomial can be set independently of the no. of control pts.

② B-spline allow local control over the shape of a spline.

Equations:

General expression for the calculation of coordinate position along a B-Spline curve using a blending function

$$P(U) = \sum_{K=0}^{n} P_K B_{K,d}(U) \qquad U_{min} \leq U \leq U_{max} \quad \text{—①}$$
$$2 \leq d \leq n+1$$

$P_k$ ≠ n+1 control points.

Range of parameter (U) depends on how to choose the other B-spline parameters.

$B_{k,d}$ (B-spline blending functions) are polynomials of degree (d-1), where (d) → degree of parameter.

If d=1 then curve is just a point plot of the control pt.

→ Local control for B-splines is achieved by defining the blending functions over sub-intervals of the total range of U.

Blending functions of B-spline curves are defined by "COX-DEBOOK" recursion formula:

→ $$B_{k,1}(U) = \begin{cases} 1 & \text{if } U_k \leq U \leq U_{k+1} \\ 0 & \text{otherwise} \end{cases} \quad \text{——— (2)}$$

→ $$B_{k,d}(U) = \frac{U - U_k}{U_{k+d-1} - U_k} B_{k,d-1}(U) + \frac{U_{k+d} - U}{U_{k+d} - U_{k+1}} B_{k+1,d-1}(U) \quad \text{———}$$

where each blending function is defined over d subintervals of the total range of U.

Each subinterval end point ($U_j$) is refered as a (KNOT). E entire set of selected subinterval is called a (KNOT VECTOR)

Any values for subinterval endpoints can be chosen, subject to condition $U_j \leq U_{j+1}$

values of $u_{min}$ & $u_{max}$ depend on no. of control pts we select,

The value of degree parameter (d) & How we set up subintervals (knot vectors)

Assumption: Any term evaluated as $\frac{0}{0}$ are treated as 0

B-splines allows us to vary no. of control pts. used to design a curve without changing the degree of the polynomial.

We can increase the no. of values in knot vector to aid in curve design.

When we do this, however, we must add control pts since the size of knot vector depends on parameter (n).

Properties:

① For n+1 control pts, curve is described with n+1 blending functions.

② Each blending function $B_{k,d}$ is defined over (d) subintervals of the total range of u, starting at knot value $u_k$.

③ Range of parameter (u) is divided into n+d subintervals by the n+d+1 values specified in the knot vector.

④ Polynomial curve has degree (d-1) & $C^{d-2}$ continuity over the range of u.

⑤ With knot values labeled as $\{u_0, u_1, \cdots u_{n+d}\}$ the resulting B-splines curve is defined only in the interval from knot value $u_{d-1}$ upto knot value $u_{n+1}$

⑥ Each section of the spline curve (b/w 2 successive knot values) is influenced by (d) control pts.

⑦ Any one control pt can affect the shape of at most (d) curve sections.

NOTES

(Ref Pg 357)

Uniform B-spline curves: → The resulting curve is UNIFORM B-SPLINE.

when spacing b/w knot values is constant. eg:

$\{-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$ —①

$\{0, 1, 2, 3, 4, 5, 6, 7\}$ —②

To illustrate blending function for uniform integer knot vector, we select parameter values.

$n = d = 3$.

The knot vector must then contain

$n+d+1 = 7$ knot values: {i.e eq ②}

& range of parameter u is from 0 to 6 with $n+d = 6$ subintervals.

Each 4 blending functions span $d = 3$ subintervals of the total range of u.

First blending function is // using eqn ③.

$$B_{0,3}(u)\begin{cases}\frac{1}{2}u^2 & \text{for } 0\le u<1\\ \frac{1}{2}u(2-u)+\frac{1}{2}(u-1)(3-u) & 1\le u<2\\ \frac{1}{2}(3-u)^2 & 2\le u<3\end{cases}$$

$$B_{1,3}(u)=\begin{cases}\frac{1}{2}(u-1)^2 & \text{for } 1\le u<2\\ \frac{1}{2}(u-1)(3-u)+\frac{1}{2}(u-2)(4-u) & 2\le u<3\\ \frac{1}{2}(4-u)^2 & 3\le u<4\end{cases}$$

$$B_{2,3}(u)=\begin{cases}\frac{1}{2}(u-2)^2 & \text{for } 2\le u<3\\ \frac{1}{2}(u-2)(4-u)+\frac{1}{2}(u-3)(5-u) & 3\le u<4\\ \frac{1}{2}(5-u)^2 & 4\le u<5\end{cases}$$

$$B_{3,3}(u)=\begin{cases}\frac{1}{2}(u-3)^2 & \text{for } 3\le u<4\\ \frac{1}{2}(u-3)(5-u)+\frac{1}{2}(u-4)(6-u) & 4\le u<5\\ \frac{1}{2}(6-u)^2 & 5\le u<6.\end{cases}$$

$$B_{0,3}(u)=\frac{u-u_0}{u_2-u_0}B_{0,2}(u)+\frac{u_3-u}{u_3-u_1}B_{1,2}(u)$$

$$=\frac{u-0}{2-0}B_{0,2}(u)+\frac{3-u}{3-1}B_{1,2}(u)$$

$$=\frac{u}{2}B_{0,2}(u)+\frac{3-u}{2}B_{1,2}(u)$$

$$B_{0,2}(u)=\frac{u-u_0}{4-u_0}B_{0,1}(u)+\frac{u_2-u}{u_2-u_1}B_{1,1}(u)$$

$$=uB_{0,1}(u)+(2-u)B_{1,1}(u)$$

$$B_{1,2}(u) = \frac{u-u_1}{u_2-u_1} B_{1,1} + \frac{u_3-u}{u_3-u_2} B_{2,1}$$

$$= (u-1) B_{1,1} + (3-u) B_{2,1}$$

$$B_{0,3}(u) = \frac{1}{2} u^2 B_{0,1}(u) + \frac{1}{2} u (2-u) B_{1,1}(u) + \frac{1}{2} (3-u)(u-1) B_{1,1}(u)$$
$$+ \frac{1}{2} (3-u)^2 B_{2,1}(u)$$

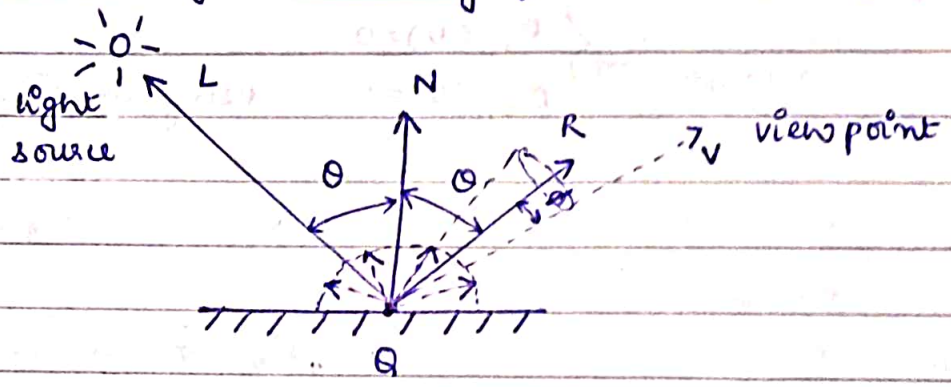
for $0 \le u < 1$ $\begin{cases} B_{1,1}(u) = 0 \\ B_{2,1}(u) = 0 \\ B_{0,1}(u) = 1 \end{cases}$ using eq. (2).

# ILLUMINATION MODELS & SURFACE RENDERING METHODS

## PHONG MODEL / LOCAL ILLUMINATION MODEL

Local illumination model is named so because its main focus is on direct impact of the light coming from the light source.

GLOBAL ILLUMINATION MODEL → attempts to include secondary effects as light going through transparent / translucent material & light bouncing from 1 obj surface to another.




consider a point light source, which is an idealized light with all its energy coming from a single point in space (approximation to a bulb).

our eye is at viewpoint looking at point Q on the surface of an object. what should be the color of light reflected into our eye from Q (in direction of vector V)

There are 2 case of light reflection

→ Diffuse Reflection :→

light from light source (in direction of -L) gets reflected equally in all directions.

→ **Specular reflection** :→

captures the characteristics of a shiny or mirror like surface. If surface is a perfect mirror, light would be reflected in exactly one direction (direction of vector R)

If not a perfect mirror (actual condition), then reflection is along a small cone-shaped space centered around R, with reflection being the strongest along direction of R ie ($\phi$)=0


→ In **PHONG MODEL** surfaces are thought to produce a combination of ambient light reflection & light source dependant diffuse / specular reflection.

Total reflected energy intensity I is

$$I = I_a k_a + I_p (k_d \cos\theta + k_s \cos^k \phi)$$

$I_a$ = intensity of ambient light
$I_p$ = " " point "

$0 \leq k_a, k_d, k_s \leq 1.0$ are reflection coeff.

↓

they represent surface ability to reflect ambient light to produce diffused reflection & to produce specular reflection.

LAMBERTS SHADING: The diffuse reflections from the surface are scattered with - equal intensity in - all directions - independent of viewing direction.

↓

Such surfaces are called LAMBERTIAN REFLECTORS

If L, N, R & V are UNIT VECTORS, then

$$L \cdot N = \cos\theta \quad \&$$

$$R \cdot V = \cos\phi$$

$\therefore$

$$I = I_a k_a + I_p (k_d L \cdot N + k_s (R \cdot V)^k)$$

If more than one light source, then

$$I = I_a k_a + \sum_{i=1}^{n} I_{p_i} (k_d L_i \cdot N + K_s (R_i \cdot V)^k)$$

## SHADING METHODS / TECHNIQUES (POLYGON RENDERING METHODS)

Phong formula at every point of surface is very expensive.

$\therefore$ we use COLOR INTERPOLATION & SURFACE INTERPOLA^ to shade other surface points.

→ CONSTANT SHADING: (flat shading)

color values of selected surface pts are used to shade entire surface.

Disadv:

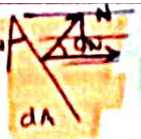
① false contours appear b/w adjacent polygons.

② specular reflection of light source tends to get lost.

→ GOURAUD SHADING:

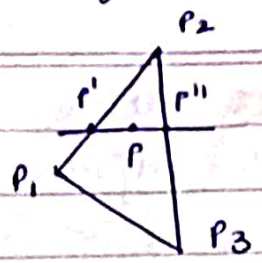
Evaluate illumination formula at vertices of a polygon mesh. Then interpolate resultant color values to get gouraud shading within each polygon.

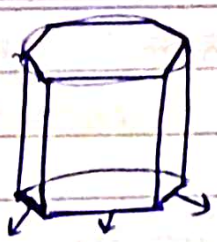
Lambert's cosine law:




It states that radiant energy from surface dA in direction $\phi_N$ is proportional to $\cos \phi_n$.

∴ for Lambertian reflectors intensity of light is same over all viewing directions.



linear interpolation applied to D to find color of point P scan line intersects at P' & P''

Interpolate color of $P_1$ & $P_2$ to get color of P'
" " $P_2$ & $P_3$ " P''
" " P' & P'' " P

$$I' = I_1 \frac{y_2 - y'}{y_2 - y_1} + I_2 \frac{y' - y_1}{y_2 - y_1}$$

$$I'' = I_2 \frac{y'' - y_3}{y_2 - y_3} + I_3 \frac{y_2 - y''}{y_2 - y_3}$$

$$I = I' \frac{x'' - x}{x'' - x'} + I'' \frac{x - x'}{x'' - x'}$$

For RGB formulas apply to each color component.

Eg:

Pt $P_1$ has RGB color (1, 0.5, 0) on line 5
& $P_2$ has (0.2, 0.5, 0.6) on line 15
color at line 8?

$\Delta R = (0.2 - 1) / (15 - 5) = -0.08$
$\Delta G = (0.5 - 0.5) / (15 - 5) = 0$
$\Delta B = (0.6 - 0) / (15 - 5) = 0.06$

0.5

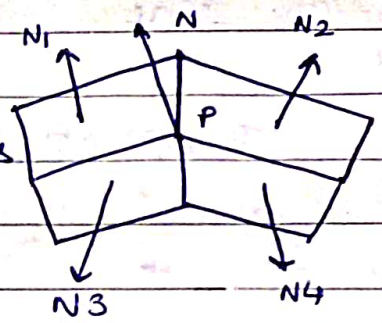
$1 + (3 \times (-0.08)),\ 0.5 + 3 \times 0,\ 0 + 3 \times 0.06$

$(.76, 0.5, 0.18)$

2 ways to determine NORMAL VECTORS at vertices of polygon to get smooth transition b/w adjacent polygons.

① use curved surfaces for true normal.

② Average normal vectors of adjacent polygons.




→ Average of normal vectors of polygons that meet at P.

$$N = N_1 + N_2 + N_3 + N_4$$

→ N is normalized by dividing by |N|

Common normal vectors at the vertices of polygon mesh result in color values shared by adjacent polygons, eliminating abrupt changes in shading across polygon edges.

→ Gouraud shading is not very effective in specular reflection.

→ PHONG SHADING: (normal vector interpolation shading)

Instead of color values, interpolation of normal vector is done.

Normal vector at P in Δ is found as →

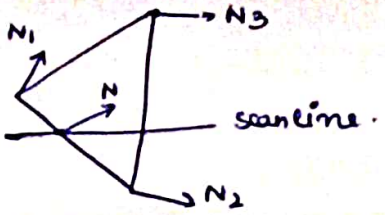
First interpolate $N_1$ & $N_2$ → to get $N'$

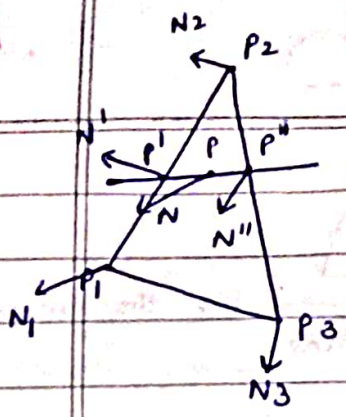
then " $N_2$ & $N_3$ → " $N''$

the " $N'$ & $N''$ → " N

This is time consuming as every point is evaluated using interpolated normal vectors.

#




$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

## RAY TRACING METHODS:

It is the generalization of the basic ray casting procedure. In Ray tracing pixel ray is pounced around in the scene as shown below, to collect various intensity contributions.

collect intensity contributions

This provide a simple & powerful rendering technique for obtaining global reflection & transmission effects.

simple powerful global R-T

Basic ray tracing algo also detects visible surfaces, shadow areas, transparency effects, generates perspective projectional views & accomodates illumination effects from multiple light sources.

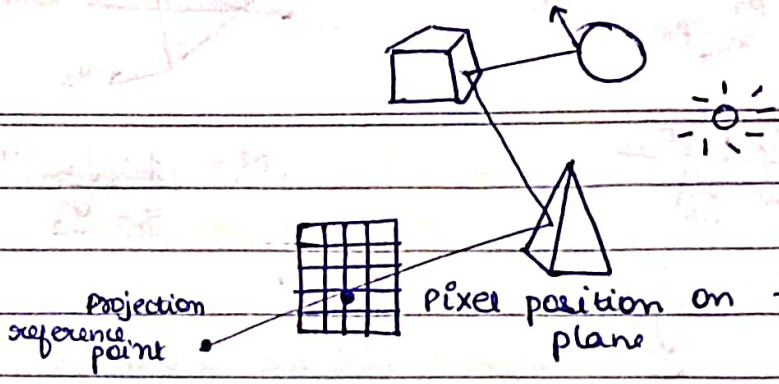
detects

Many extensions to the basic algo have been developed to produce photo-realistic displays.

Ray traced pictures of scenes can be highly realistic, particularly when the scene contains shiny objects, but ray-tracing algo involve considerable computation time.

realistic

computation time

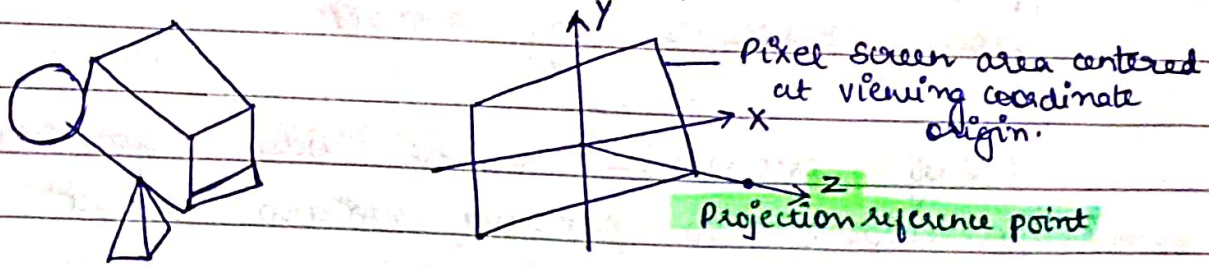



## BASIC RAY TRACING ALGORITHM :

The coordinate system for a ray tracing algo is typically set up as shown below, with the projection reference point on the z-axis & the pixel position on the xy plane.

We then describe the geometry of a scene in this coordinate system & generate the pixel rays.

For a perspective - projection view of the scene, each ray originates at the projection reference pt (center of projection), passes through a pixel center & continues into the scene to form various branches of the ray after reflection & transmission paths.




Contributions to the pixel intensities are then accumulated at the intersected surface.
This rendering approach is passed on principles of geometric optics.

light rays from the surfaces in a scene are emitted in all directions. & some pass through the pixel positio on the projection plane.

Since there are an infinite no. of rays, we determine the intensity contributions for a particular pixel by tracing a light path backward from the pixel position into the scene.

In basic ray tracing algo one reverse light ray is generated for each pixel, which is approximately equivalent to viewing the scene through a pinhole camera.

As each pixel ray is generated, the list of surfaces in the scene is processed to determine whether there are any ray-surface intersections. If ray intersects a surface, we calculate distance from the pixel to the surface intersection point.

After all the surfaces have been tested for a ray intersection. The smallest calculated intersection distan identified the visible surface for that pixel.

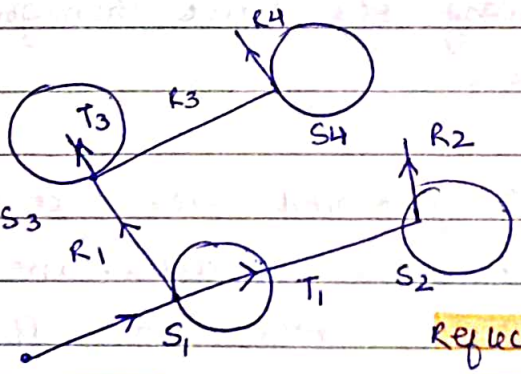
We then reflect the ray off the visible surface along a specular-reflection path (angle of reflection = angle of incidence).

For a transparent surface, we also send a ray through the surface in the refraction direction.

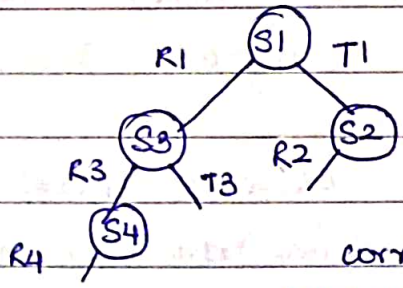
Reflection & surface refraction rays are called SECONDARY RAYS.

Repeat the ray-processing procedure for the secondary rays. Surfaces are tested for intersections & the nearest intersected surface, if any, along a secondary ray path is used to recursively produce the next generation of reflection & refraction paths.

Each successively intersected surface is added to binary ray tracing tree as shown below.




Projection reference point




Reflection & Refraction paths through scene

corresponding Binary-ray tracing tree.

Left branches = reflection paths
Right branches = transmission paths

Max depth of ray-tracing tree can be set as a user option or can be determined by amt. of M/M space.

We terminate a path in the binary tree for a pixel if any one of the following conditions is satisfied.

- Ray intersects no surfaces.

- Ray intersects a light source that is not a reflecting surface.
- the tree has been generated to its max allowable depth.

## RADIOSITY LIGHTING MODEL :

We can more precisely model lightening effects by considering the physical laws governing the radiant-energy transfers within an illuminated scene

This method for computing pixel color values is generally referred to as the RADIOSITY MODEL

For monochromatic light radiation energy of each photon is given by

$$E_{photon,f} = hf$$

$h = 6.6262 \times 10^{-34}$ Joules.sec , $f$ = frequency

$$\text{Total energy} = E_f = \sum_{\text{all photons}} hf$$

Actual light radiation contain range of frequencies.
∴ Energy is sum of all photon energies

$$E = \sum_{f} \sum_{\text{all photons}} hf$$

Amount of radiant energy transmitted / unit time → RADIANT FLUX (Φ)

where $\Phi = \frac{dE}{dt}$ Joules / sec.

Radiant flux per unit area leaving a surface called RADIOSITY $B = \frac{d\Phi}{dA}$ $W/m^2$