Graphics : still images

Animation : movement of object artificially

Morphing : changing one object to another

Eg: Nijam pauk, Rajini face

8M Survey of computer graphics :- Graphic Application education, advertisement, medicine

① CAD : Computer Aided Design
draw images of building, textiles, aircraft etc

② Wireframe model : Interior design of an object can be seen.

Interactive Design
adjustments can be made
Can be used for circuit design.

Animations → CAD application
* Can be used in virtual reality
* VR :- walkthrough.

③ Create personalized symbols

④ Testing performance of a vehicle.

⑤ Surface rendering, lightening model.

TB.

CAD - uses, wireframe Display, Animation in VR, Animation in Wireframe Display, walkthrough in VR.

2M PRESENTATION GRAPHICS:

cricket score
35mm slides
financial, mathematical, statistical

data → to represent these data
eg bar graph, pie chart, line graph

COMPUTER ART:

Graphics can be used in

1. Fine Art
2. Commercial Art

Artists use paint brush prg.

to allows, Paint on video monitor

Animations used in ads TV commercial frame by frame

and saved as image file.

5 Diff btw frames → position change

6 Once all frames in animations are rendered they are stored in video buffer.

7 Film animation requires 24 frames each seconds.

8 Play Back → 30 frames per second

9 Morphing

ENTERTAINMENT :

movies, TV shows

making motion pictures

Graphic objects can be combined with live objects action. Eg: Morphing

Morphing def

EDUCATION & TRAINING

aircraft, pilot, Machine operators training

pen plotter, simulation training

VISUALIZATION

scientist, engineers, Medical person have large amount of data

Visualization / scientific

\ Business

effective visualization → based on characteristics of data

Data — scalar
— vector

color coding - to visualize a data set
Image processing can also be used to visualize data

2M diff btw comp graphics & image processing

6M IMAGE PROCESSING :-
2M
8M
→ apply technique to modify/interpret existing pictures.

applications of Image processing → improve picture quality
→ machine perception of visual information

How to do image processing
digitize the photo
digital method is applied
improve color

use : Medical purpose → Tomography (surgery)
x-ray photography
cross sectional views
CT, PET
comp X-ray tomography, Position Emission Tomography

Ultra sonic nuclear medicines
Model & study. physical
Artificial limbs (X)

GUI : graphical symbol – icon
Menus
Windows Manager – responsible for GUI.
To enable window, click it by using interactive pointing device

OVERVIEW OF GRAPHIC SYSTEM :-

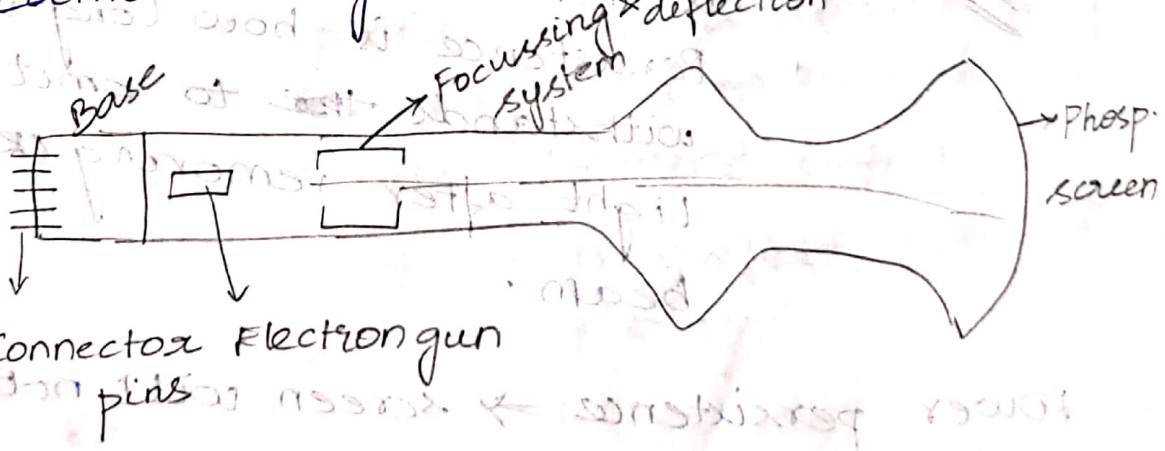
→ Video display devices 8M

Output Device

8M Video monitor → CRT

Refresh CRT → 2M

9) 2 Cathode Ray Tube :- Video monitor & TV → 8M diag & explanation



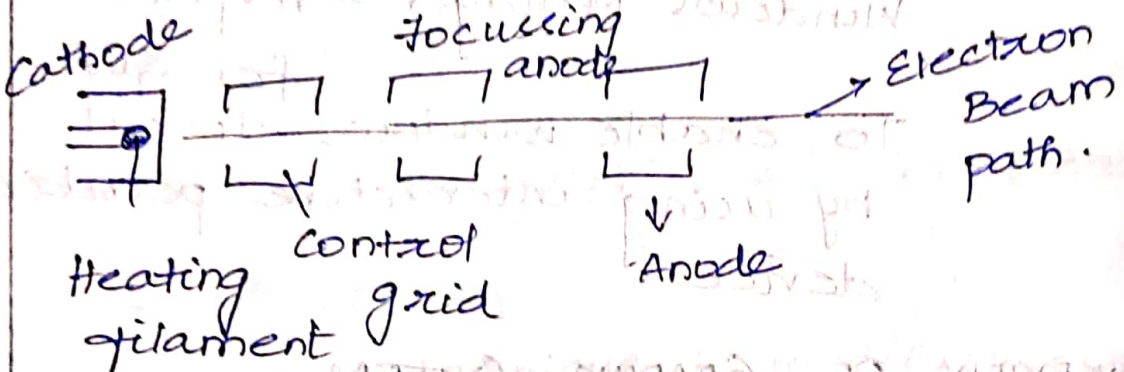
Refresh CRT → TB 2M

Components of Electron gun:

4M

1) Heating filament (Cathode)
2) control grid
3) Focussing anode
4) Anode
5) Electron Beam path.

Heated metal cathode
Control grid



(Focussing system) and para in TB.
forces electron beam to converge

If foc. sys. is not there e⁻ will repel.

Persistence: CRT beam is resp for beam phos. glowing. when CRT is removed it starts fading. Persistence is how long it withstands to emit light after removing CRT beam.

2M

Lower persistence → screen will not flicker.

Phos will low persistence → used for animation

" high " → complex static pic.

Resolution :- max no of pts that can be displayed in CRT without overlapping. 2M

pixel ⇒ picture element

Aspect Ratio : prop of video monitor. 2M
ratio of vertical pts & horizontal points.
eg: $\frac{3}{4}$ $\frac{\text{vertical pts}}{\text{horizontal pts}}$

8M Raster scan display :-
most common type of graphics display monitor (based on TV)

e⁻ beam one row at a time by one row from top to bottom.

Intensity will be turned on or off

picture defn is stored in memory area ⇒ refresh buffer / frame buffer.

each pt is called pixel / pel

realistic display

Eg : Home TV sets, printer

Pixel Intensity range for Raster scan system depends on the capability of Raster scan system.

In black & white system, each screen pt will be on/off. only one bit/pixel needed.

1 → e⁻ beam is on

0 → e⁻ beam is off

Frame Buffer is called bit map → one bit/pixel

Multiple bits/pixel → pixmap

Horizontal retrace
Vertical retrace
Interlaced refresh procedure } 2M

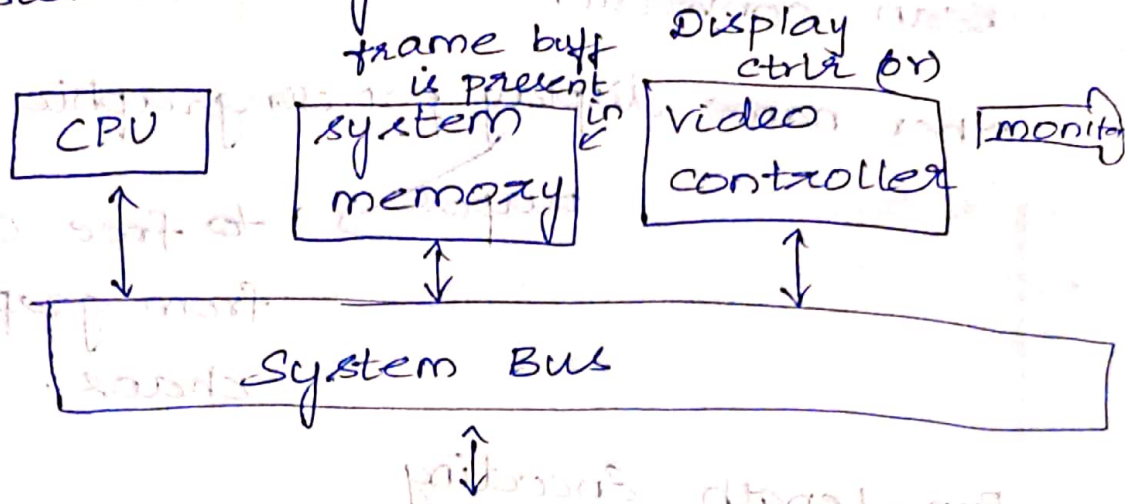
8M

Two passes 1st pass : top to bottom
2nd pass : vertical

used with slower refreshing rates

15/12

Raster Scan System:



Video controller → ctrls frame buffer

- 2 Register coz the co-ordinates of screen pixel has to be set.
  it increases when it goes to next scanline

cartesian co-ordinate scan lines - ranges from Ymax to 0 (top) (bottom)

Along each scan, screen fixed pos as labelled from 0 to Xmax.

character represent

Major task of display processor is digitizing the picture defngn in an appln prg ~~to~~ if a set of pixel intensity value for storage in frame buffer.

This digitization process is called scan conversion.

Other name display processor: graphic contr

purpose → to free CPU from graphic chaos.

Run-Length Encoding

Cell Encoding

To reduce memory req in Raster systems, methods have been derived for organizing frame buff as Linked list & encoding as intensity info.

One way to do is to store each scan line as a set of integer pairs.

One no of each pair indicates an intensity val & sec no indicates no of adj pixel on the scan line that are to have that intensity.

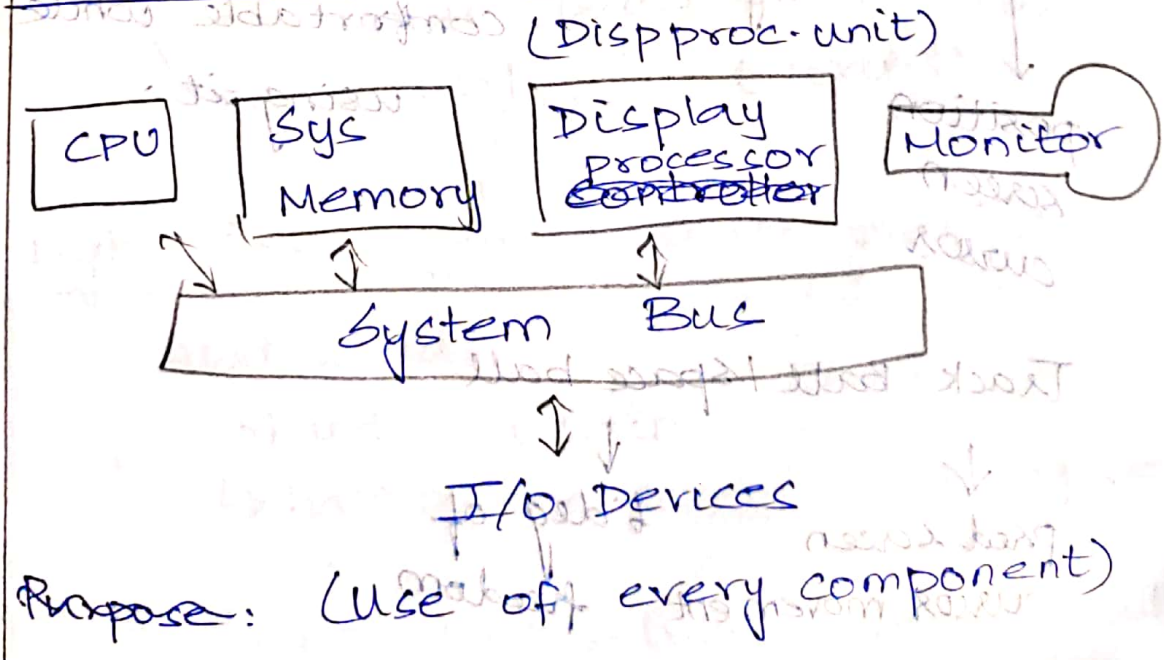
This technique is Run-Length Encodin Can result in a considerable saving

in storage space if a pic is to be constructed mostly with long runs of single color each.

2M When pixel intensity change linearly, approach used to encode the racter as a set of rectangular areas is called cell encoding.

Disadv of encoding Runs are that intensity changes are difficult to make & storage req actually increase, the length of runs decrease it is difficult for disp ctrlr to process the Raster when many short runs are involved.

4M Random Scan System / Graphics ctrlr

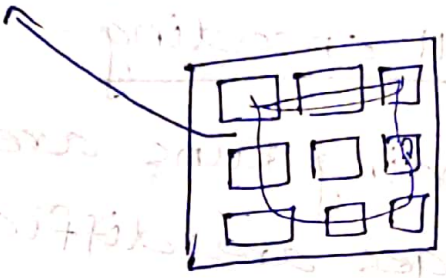


Purpose: (use of every component)

MEDIA WALL → large area display

eg: SunNews → images split into no of sections

MULLIONS → break btw pics



8M Input Devices :- → frq used operations

KeyBoard → fn keys, spl keys, alphanumeric

cursor ctrl key ↑ ↓ → ←

; keyboard; non-graphic data

Mouse ⇒ input to graphic.

ergonomic ⇒ how user should be comfortable while using it.

↓

position screen cursor

Track Ball / Space ball

↓ ↓

prod screen cursor movement 6 deg of freedom

Joy sticks : Games
Data glove
Transmitting antennas
Receiving antennas

Digitizer : drawing painting
main purpose : to select co-ordinate position accurately
⇓
eg : graphics tablet

Types - Electromagnetic Pulse
- Sonic / Acoustic

Image Scanner :

By passing optical scanning mechanism to the info to be stored.

Touch Panels : ATM
- optical (Led)
- electrical (Transplates)
- acoustical (sound waves)

Light Pens : pencil shape device

disadv & adv :
- ↳ hand will pain
- ↳ half of the screen will not be visible
- ↳ room temp & light → readings will change
- ↳ will not work in black light

Voice system : voice as input
predictionary of words

18/12/17

8M HARD COPY DEVICES / OUTPUT DEVICES :

Several formats : (Refer Book)

Printer :

Quality of pic depends on dot size
dots per inch

Impact method (impressions are there)
eg: Line printer, Dot matrix printer

Non impact method (no impressions)
eg: Xerox plotter
Laser, Inket space, electrostatic, electrothermal
methods

ELECTROSTATIC DEVICE
ELECTROTHERMAL DEVICE :- Dot matrix printer.

Pen plotter (8M)

s/w → set of prgs to do a task.

Graphics s/w

- General programming packages
  - GL - graphics lib
    - (straight line, polygon, circle)
    - (color, intensity, views, applying transformation)
- Spl application packages - designed for non-programmers

CO-ORDINATE REPRESENTATION:

~~i) cartesian~~

Modelling co-ordinates 2M.

or

Local co-ordinates

or

Master co-ordinates

Wall "

Device co-ordinates

Screen co-ordinates

Basic building blocks for picc are referred as o/p primitives They include ~~Screen co-ordinate~~ char strings

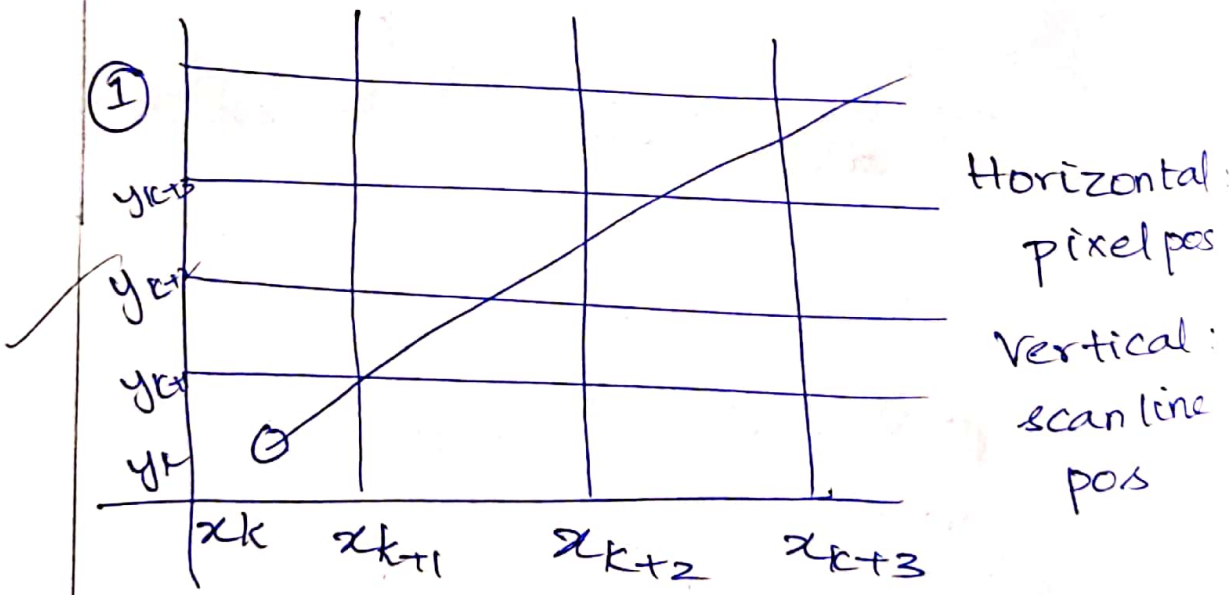
geometric entity like points, straight line, filled areas (polygon, circle etc)

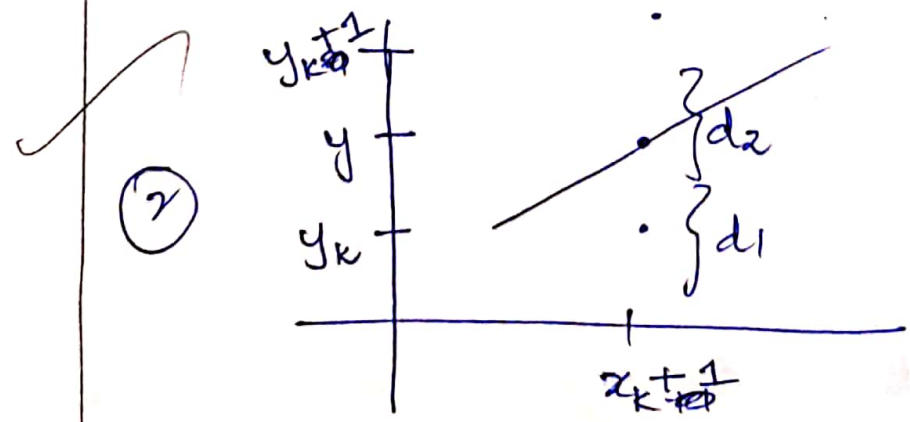
Attributes are the prop of o/p primitive i.e., attributes desc how a particular primitive is to be

29/12/17

Bresenham's Algorithm:

An accurate and efficient Raster Line generating algorithm by Bresenham, scan converts lines using only incremental integer calculations that can be adapted to display circle & other curves. (adv)





We first consider, the scan conversion process for lines with positive slope less than 1. Pixel pos along a line path are then determined by sampling at unit x intervals. Starting from the left end pt (x_0, y_0) of a given line, we step into each successive column (x position) and plot the pixel whose scan line y value is closest to the line path.

Fig (1) demonstrate the kth step in this process. Assuming we have determined that the pixel (x_k, y_k) is to be displayed. We next need to decide which pixel to plot in column x_{k+1}. Our choices are pixel at positions (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}).

At sampling position $x_k + 1$, we label vertical pixel seperations from mathematical line path as d_1 and d_2 (fig 2)

The y coordinate on the mathematical line at pixel column position x_k+1 is calculated as

$$y = mx + b$$

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

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

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

$$d_1 = mx_k + m + b - y_k$$

$$\boxed{d_1 = mx_k - y_k + m + b}$$

$$d_2 = y_{k+1} - m(x_k+1) - b$$

$$d_2 = y_k + 1 - mx_k - m - b$$

$$\boxed{d_2 = y_k - mx_k - m - b + 1}$$

$$d_1 - d_2 \Rightarrow mx_k - y_k + m + b$$

$$- mx_k + y_k - m - b + 1$$

$$+ \quad - \quad + \quad + \quad -$$

$$2mx_k - 2y_k + 2m + 2b - 1$$

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

A decision parameter P_k for the kth step in the line algorithm can be obtained by rearranging eqn (2) so that it involves only integer calculations. Subs $m = \frac{\Delta y}{\Delta x}$ where Δy and Δx are vertical & horizontal seperations for end point positions

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

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

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

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

$$= 2\Delta y x_k - 2y\Delta x + 2\Delta y + 2b\Delta x - \Delta x$$

$$P_k = 2\Delta y x_k - 2\Delta x y_k + c \quad (3)$$

where

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

Sign of p_k is same as $(d_1 - d_2)$ since $\Delta x > 0$.

y_k is closer to the line path than the pixel at $y_k + 1 \Rightarrow d_1 < d_2$.

Then, the decision parameter p_k is negative, in that case we plot lower pixel otherwise we plot upper pixel.

At step $k+1$, decision parameter is evaluated from (3)

$$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$$

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

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

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

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x (y_{k+1} - y_k)$$

where the term $y_{k+1} - y_k$ is either 0 or 1, depending on the sign of parameter p_k.

First parameter p_0 is evaluated from eqn (3), starting pixel pos (x_0, y_0) and with $m = \frac{\Delta y}{\Delta x}$,

$$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x(2b-1)$$

$$= 0 - 0 + 2\Delta y + 2b\Delta x - \Delta x$$

Since 'b' is constant,

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

(Algorithm ⇒ Refer Book)

If $p_k < 0$; $p_{k+1} = p_k + 2\Delta y$

$p_k >= 0$; $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

PROBLEM :

Digitize the lines with end pts (20, 10) & (30, 18); line slope = 0.8

with $\Delta x = 10$; $\Delta y = 8$

$$p_0 = 2\Delta y - \Delta x = 16 - 10 = 6$$

$$\boxed{p_0 = 6}$$

Initial pts $(x_0, y_0) = (20, 10)$

Successive pixel pos. along line path from the decision parameter

k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	$-2 \Rightarrow 20$ $y=12$	(23, 12)
3	14	(24, 13)
⋮	⋮	⋮
9		(30, 18)

For $k = 1$; $p_k >= 0$; $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

$p_{k+1} = 6 + 2(8) - 2(10)$

$p_1 = 6 + 16 - 20$

$p_1 = 22 - 20 = 12$

$k = 2$; $p_k >= 0$

$p_{k+1} = 2 + 2(8) - 2(10)$

$= 2 + 16 - 20$

$= -2$

$k=3$;

$p_{k+1} = p_k + 2\Delta y$

$p_{k+1} = p_k + 2\Delta y = -2 + 2(8) = 14$

(1) CIRCLE ALGORITHM : (16M)

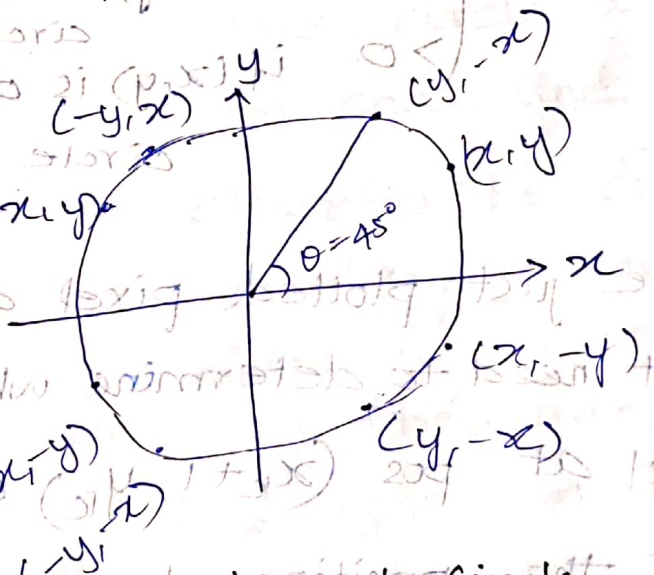
Cartesian co-ordinate form,

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

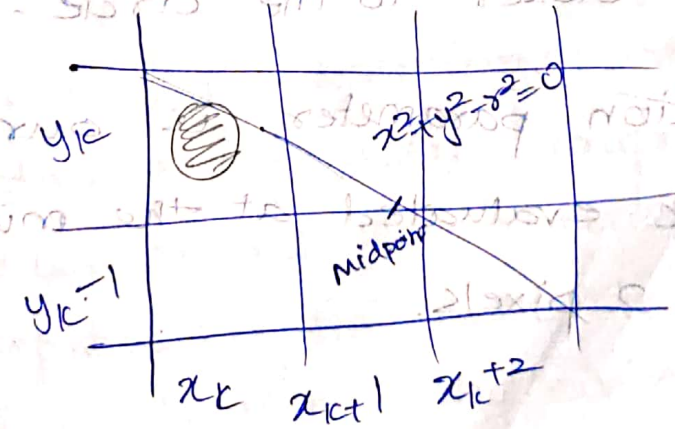
The polar form,

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

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



Midpoint Circle Algorithm



circle function $f_{circle}(x,y) = x^2 + y^2 - r^2$

Any pt (x,y) on the boundary of circle with radius r satisfies the eqn $f_{circle}(x,y) = 0$ if the pt is in the interior of the circle, f_{circle} is -ve & if the pt is outside the circle f_{circle} is +ve.

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

We have just plotted pixel at (x_k, y_k) we next need to determine whether the pixel at pos (x_k+1, y_k) or the one at the position at (x_k+1, y_k-1) is closer to the circle.

Our decision parameter is our circle func evaluated at the midpoint btw these 2 pixels.

$$\text{Midpt} = \left(\frac{x_k + 1 + x_k + 1}{2}, \frac{y_k + y_k - 1}{2} \right)$$

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

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

$$f_{circle}(x, y) \Rightarrow (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

If $p_k < 0$, this midpt is inside the circle and pixel on scan line y_k is closer to the circle boundary; otherwise the midpt is outside or on the circle boundary. We select the pixel on scan line $y_k - 1$

Successive decision parameters are obtained by using incremental recursive calculations. We obtain a ~~regressive~~ recursive expression for next decision parameter by evaluating the circle func at

sampling position

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

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

$$f_{circle}(x,y) \Rightarrow (x_{k+1}+1)^2 + (y_{k+1}-\frac{1}{2})^2 - r^2$$

$(a+b)^2$ $(a-b)^2$

to get book ans $\Rightarrow$ add & sub p_k on RHS

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

where y_{k+1} is either y_k or y_{k-1} depending on the sign of p_k.

$$2x_{k+1} = 2x_k + 2 \quad (x_k+1)$$

$$2y_{k+1} = 2y_k - 2 \quad (\because y_k - 1)$$

At start position $(0, r)$ these 2 terms have the values 0 and $2r$ respectively. Each successive value is obtained by adding 2 to the previous value of $2x$ & subtracting 2 from previous value of $2y$.

Initial decision param is obtained by evaluating circle func at start position $(x_0, y_0) = (0, r)$

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

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

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

$$= \frac{5}{4} - r + \frac{3}{4}$$

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

If 'r' is specified as an integer

$$p_0 = 1 - r$$

Algorithm ⇒ Refer Book.

$$p_{t+1}$$

5/1/18 Filled area primitives:

A std o/p primitive in gene graphics packages is a solid color or patterned polygon area. other kinds of primitives are available but polygons are easier since they have linear boundaries.

There are 2 basic approaches to area filling on Raster System

1) To determine the an area is to determi overlap intervals for scan lines that cros the area.

2) Start from a gvn interior p and paint outward from this pt until we encounter the specified boundary condition.

Scan line approach is used in general graphic packages to fill polygon, circle, ellipse and other simple curves. Fill methods starting ~~pt at~~ interior pts are useful ~~for~~ with from the

more complex boundaries and interactive painting system.

8M SCAN LINE POLYGON FILL ALGORITHM:

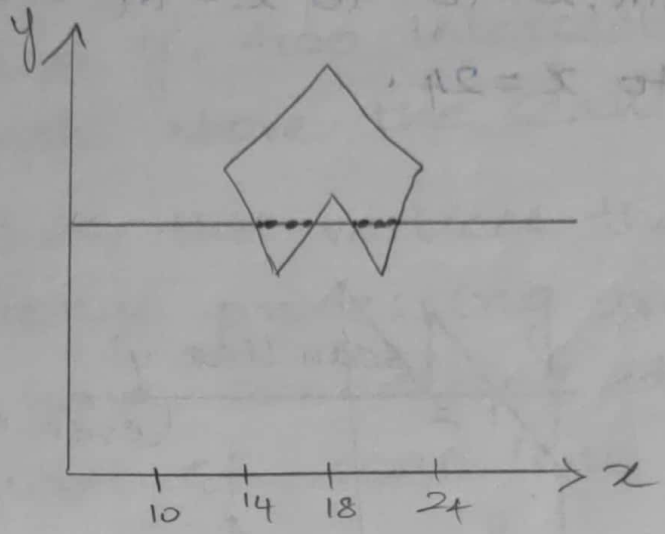


Fig (1)

Interior pixels along a scan line passing through a polygon area.

Fig (1) illustrates the scan line procedure for solid filling of polygon areas. For each scan line crossing a polygon, area fill algo locates the intersection pts of the scan line with the polygon edges.

These intersection pts are then sorted from left to right & the corresponding frame buffer positioned btw each intersection pair are ~~said~~ set to specify fill color.

Eg: Consider fig ①, there are 4 pixel intersection position with polygon boundaries. Interior pixels are from, $x = 10$ to $x = 14$ and from $x = 18$ to $x = 24$.

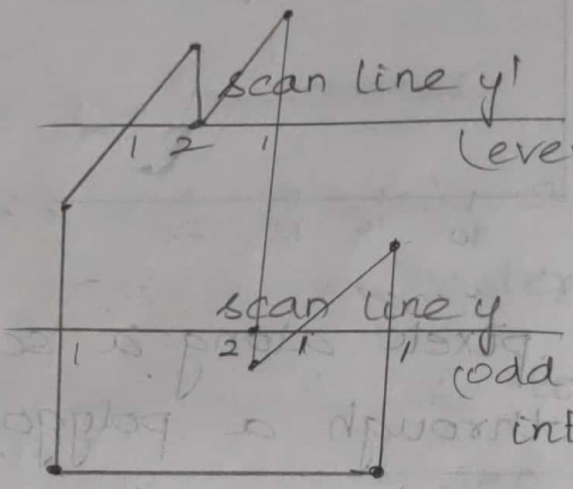


Fig ②

Fig ② shows 2 scan lines at pos y and y' that intersect edge n points. Scan line y intersect y polygon edges: Scan line y' intersect an even no of edges although it passes through a vertex.

Intersection pts along y' correctly identify the interior pixel, scan But with scan line y, we need to do some additional processing to deter

correct interior pts:

For scan line y, two intersecting edges sharing a vertex are on opp sides of the scan line. But for scan line y', two intersecting edges are both above the scan line.

Thus, the vertices that require additional processing are those that have connecting edges on the opp sides of scan line.

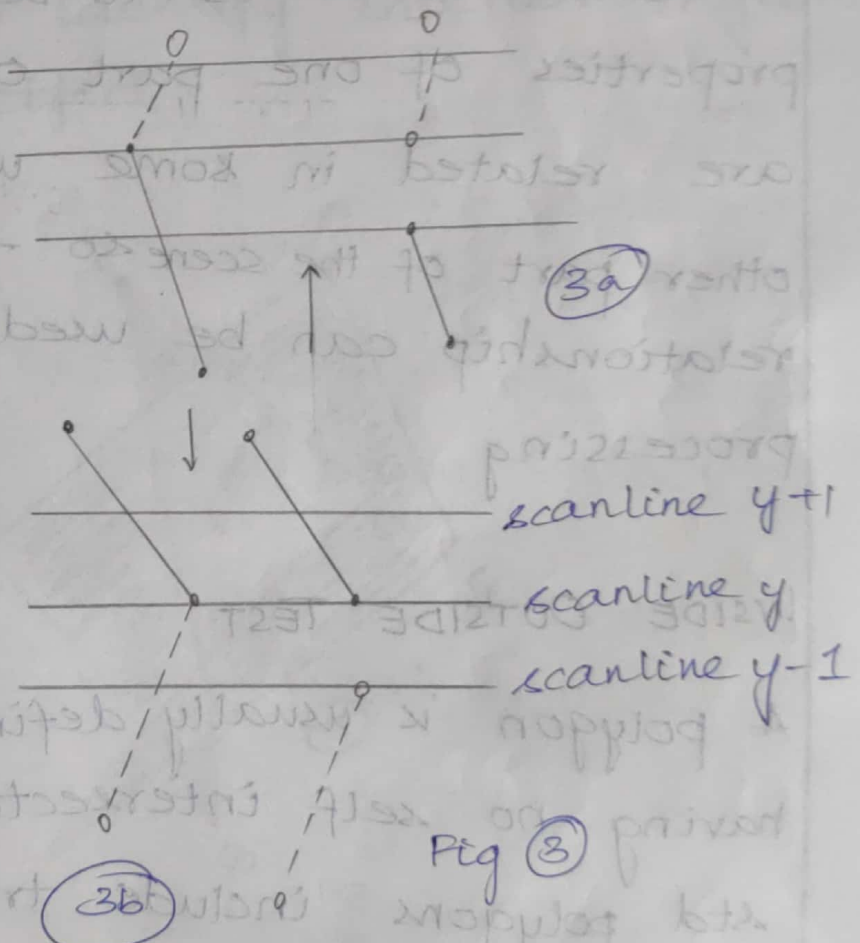


Fig (3)

Fig (3) illustrates shortening of edge when the end pts of y coordinates are increasing. The y value of upper end pt for the current edge is

decreased by 1 as in (3a). When the end pt y values are monotonically dec as in (3b), we decrease the y coordinate of the upper end of the edge fall the current edge.

2M COHERENCE :

The scene that is to be displaye properties of one part of a scene are related in some way to other part of the scene so that relationship can be used to redu processing

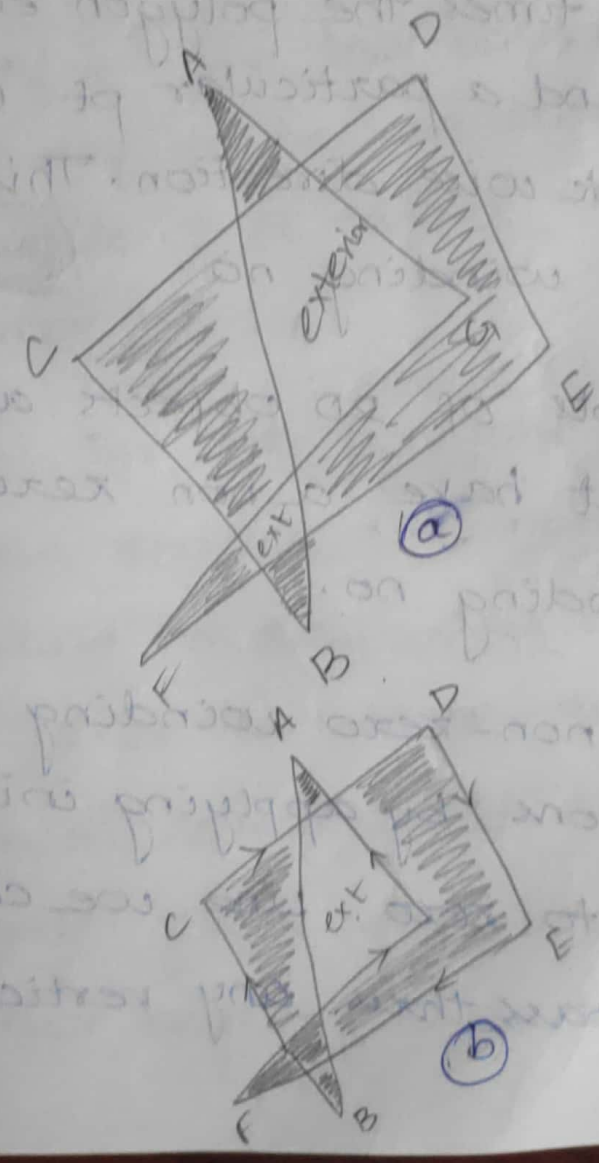
INSIDE OUTSIDE TEST

A polygon is usually defined as having no self intersection. Eg of std polygons include triangles, rectangles, octagons and decagons. The component edges of these objs are joined only at the vertices or otherwise the edges have no

common pts in the plane.

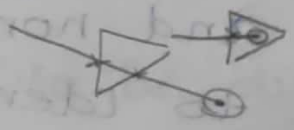
(Identifying interior regions of a 2d polygon is a straight forward process)

In most graphics applns, which is complicated graphics packages normally use odd/even rule. Also called odd parity rule / even-odd rule and non zero winding no rule to identify interior regions of an object:



Odd Even Rule:

Drawing a line from any pos p to a distant pt outside, the coordinate extend of the obj and counting the no of edge & crossing along the line. If the no of polygon edge crossing this line is odd, the p is an interior pt otherwise p is an exterior pt.



Non zero winding no rule counts the no of times the polygon edges wind around a particular pt in counterclockwise direction. This count is called winding no.

Interior pts of 2D objects are def to be that have a non zero value for a winding no.

We apply non-zero winding no rule to polygons by applying initial winding no to zero. Line we choose must not pass thru any vertices. As

we move along the line from pos P to distant pt, we count the no of edges that cross the line in each direction. We add 1 to the winding no everytime we intersect a polygon edge that crosses line from right to left & we sub 1 everytime we intersect an edge that crosses from left to right. Final value of winding no after all edges crossing have been counted determines the relative pos of P. If the winding is not 0, P is def to be an interior pt, otherwise P is taken to be an exterior pt.

NOTE : For std polygons and other simple shapes, the non-zero winding no rule and odd-even rule gives the same result. But for more complicated shapes, two methods yield diff interior and exterior region (Refer diag @ 1(b))

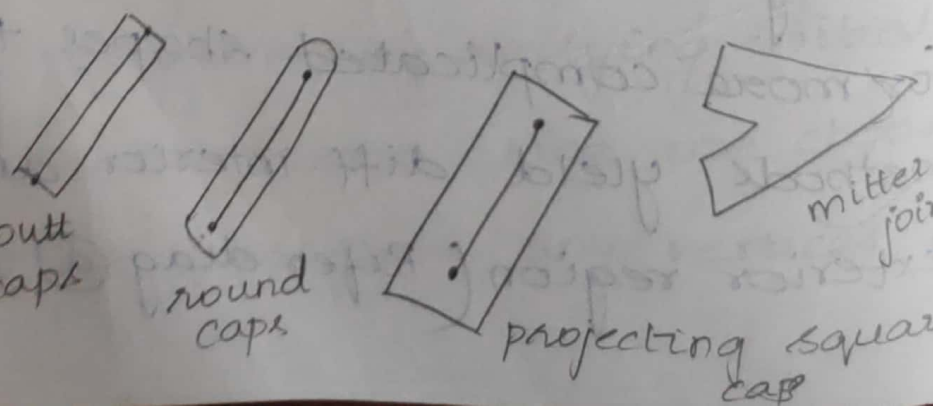
LINE ATTRIBUTES:

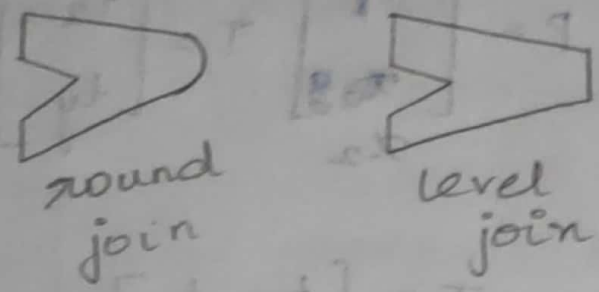
Basic attributes of straight line segment are its type, its width, its color. Line type — solid, dashed, dotted lines.

SetLineType (lt) where parameter lt is assigned a +ve integer value of 1, 2, 3 or 4 to generate lines that are solid, dashed, dotted or dashed-dotted

LineWidth, SetLineWidthScaleFactor (lw) lw is assigned a +ve no to indicate a relative width of the line to be displayed. Value of 1 specifies std width line. Values >1 produce lines thicker than std.

Thick Lines





Line Color – SetPolyLineColorIndex(lc)

2D GEOMETRIC TRANSFORMATIONS: (2)

The basic geometric transformations are translation, rotation and scaling. Other trans that are often applied to objs include reflections and shearing

Translation : A translation is applied to an obj by repositioning it along st line from one coordinate location to another. We translate a 2D pt by adding t_x (translation dist) and t_y to the orig coordinate pos (x, y) to move a pt to new pos (x', y').

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

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

$$P' = P + T$$

$P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$ $P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

$P = [x \quad y]$ $T = [t_x \quad t_y]$

6/11/18 2D – transformations :

$x' = x + t_x$; $y' = y + t_y$ — (1)

$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$ $T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

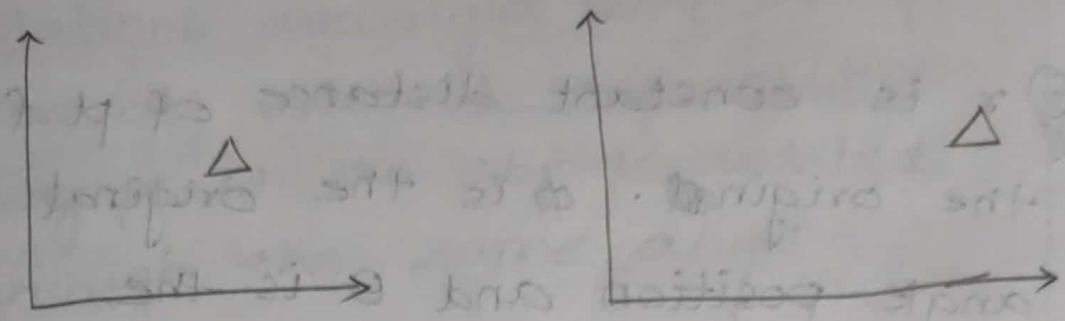
Translation (X) $\boxed{P' = P + T}$ — (3) ↳ (2)

def Translation is a rigid body transformation that moves obj without deformation. i.e., every pt on the obj is translated by the same amount. def and explain eqn (1) (2) (3) diag

A straight line segment is translated by applying transformation eqn (3) to each of the line end pts and redrawing line between new

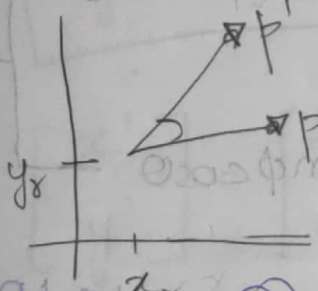
T+T '1

end pt position.



Rotation :

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



(1)

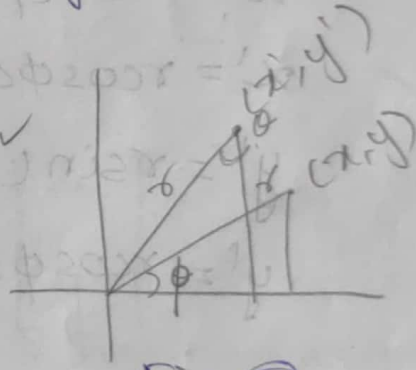


Fig (2)

To generate a rotation we specify rotation angle θ and the position (x_r, y_r) of the rotation pt (or pivot pt) about which obj is to be rotated. Positive values for the rotation angle define counter clockwise rotation abt the

pivot pt - Neg values rotate objs in clockwise direction.

Fig ② r is constant distance of pt from the origin. ϕ is the original angle position and θ is the rotation angle.

Using std trigo identities, the transformed co-ordinates in term of θ and ϕ.

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

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

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

$$y' = r\cos\phi\sin\theta + r\sin\phi\cos\theta \quad \text{— ①}$$

The org co-ordinates of the pts in polar co-ordinates are $x = r\cos\phi$;

$y = r\sin\phi$ — ②

Sub ② in ①

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

$\Rightarrow P' = RP$ (Rotation formula)

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

$$(P')^T = (R \cdot P)^T = R^T P^T$$

Rotations are rigid body transf that move obj without deformation.
Every pt on an obj is rotated through the same angle.

Scaling

Scaling transf alters the size of an objects. S_x and S_y. $s \Rightarrow$ scaling factor.
Scaling factors S_x and S_y

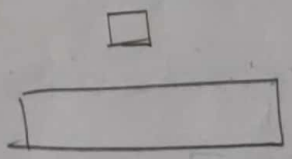
$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

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

$P' = S \cdot P$ (X) scaling

where S is a 2×2 scaling matrix.
Any +ve numeric values can be assigned to scaling factors S_x and S_y. Values less than 1 reduce the size of obj. Values > 1 produce an enlargement. Specifying value of 1

for both S_x and S_y leaves the size of obj unchanged. When S_x and S_y are assigned same val, it is called uniform scaling. Unequal S_x & S_y values is called differential scaling which is often used in design appln where pics are constructed from few basic shapes that can be adjusted by scaling & positioning transformation.



8/1/18 Matrix Representation and homogenous ~~transformation~~ coordinate:

Composite Transformation:

Forming prod of transf matrix is often ref to as concatenation or composition of matrices.

For col matrix representation of coordinate positions, we form composite transf by multiplying

matrices from right to left

Rotation about pivot point (Ref TB)

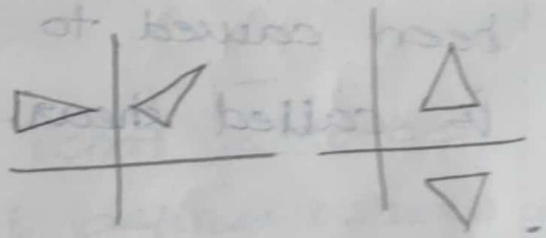
Scaling (Ref TB)

Other Transformation:

Reflection is a trans that produces a mirror image of an object. The mirror image for a 2D reflection is generated relative to axis of reflection by rotating the obj 180° about the refl axis.

Refl about the line $y=0$, x axis will have the transformation matrix,

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



x axis refl abt y axis x axis

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ y axis

Reflection abt any line $y=mx+b$ in xy plane can be accomplished with a combination of translate-rotate-reflect.

In general, we first translate line so that it passes through origin & we can

rotate line onto one of co ordinate axis and reflect abt that axis

Finally we restore line to its orig pos with inverse rotation and translation transformation.

Shear

A transf that distorts the shape of an object such that the transformed shape appears as if the obj were composed of internal layers that had been caused to slide over each other is called shear.

22/1/17

An x direction shear related to x axis is produced with the transformation matrix

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

which transforms co ordinate position as

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

$$y' = y$$

Two dimensional viewing :

The world co ordinate area selected for display is called window.

An area on a display device to which a window is mapped is called view port.

Window defines what is to be viewed. View port defines where is to be displayed. windows & view ports are rectangles in std position.

2M

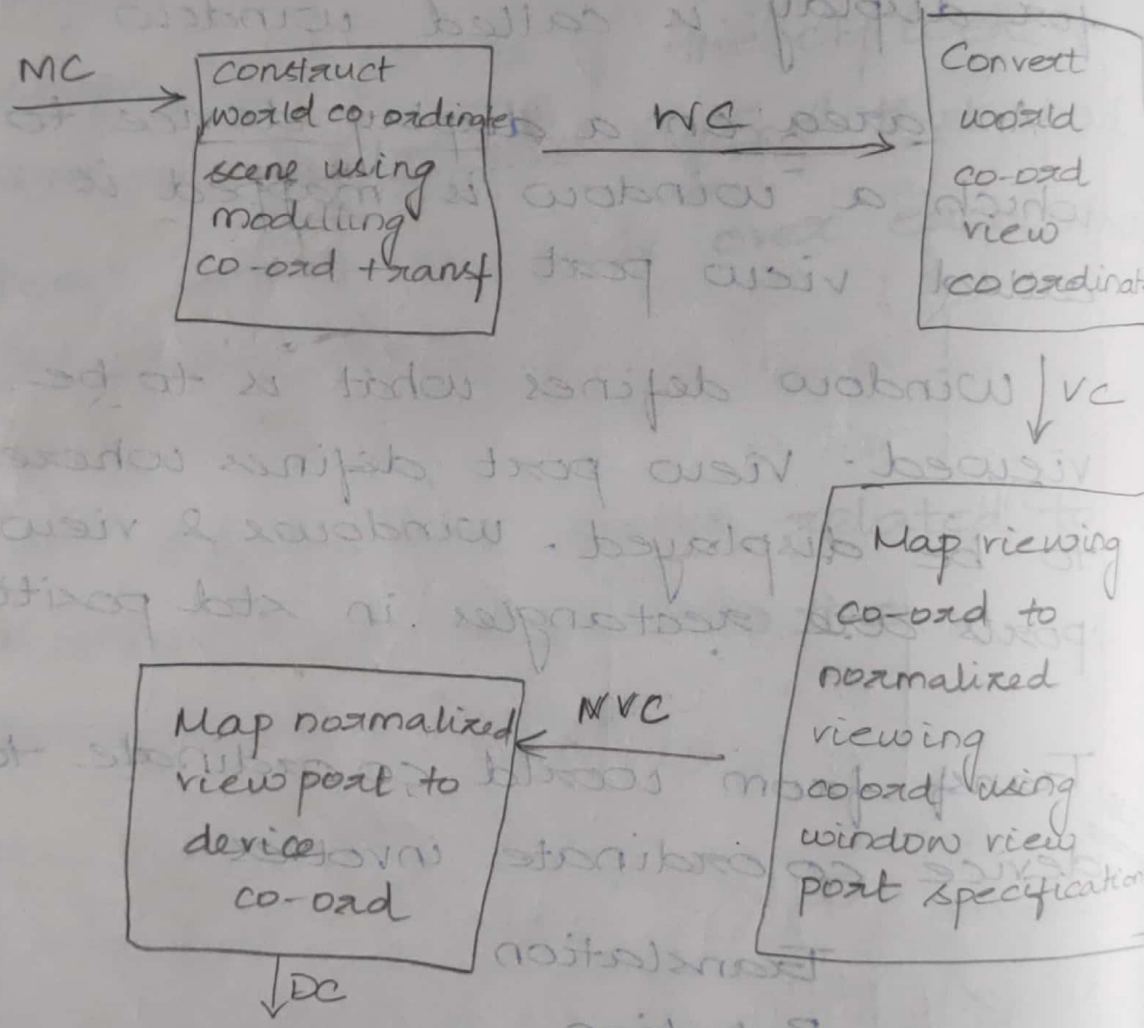
Transf from world co ordinate to device co ordinate involves

Translation
Rotation
scaling as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area.

Mapping of a part of a world co ordinate scene to a device coordinate is referred as viewing transformation or window to view port transformation or windowing transformation.

2M

2D Viewing Transformation pipeline:



view ports are typically defined within unit square (normalized co-ordinates). This provides means for separating viewing & other transf from specific o/p device req. so that the graphic package is largely device independent.

clipping procedures are of fundamental importance in comp graph. They are used not only in viewing transformation but also used in window-manager sys in painting & drawing packages to ~~elimate~~ eliminate parts of a picture inside or outside of a designed screen area & in many other application

2/2/18 CLIPPING OPERATIONS : Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as clipping algorithm or clipping.

Region against which an obj is clipped is called clip window.

Applications of clipping include extracting part of defined scene for viewing; identifying visible surfaces in 3D view; anti aliasing line segments or obj boundaries; creating obj using solid-modelling procedures; displaying a multi window env & drawing and painting oprtns that allows part of pic to be selected for copying, moving erasing and duplicating.

For viewing trans, we want display only those pic parts that are within window area assuming that clipping flags have not be set to noclip). Everything outside the window is ~~deleted~~ discarded.

Clipping alg can be applied in world coordinates so that contents of windows are mapped to device co-ordinates.

TYPES OF CLIPPING :-

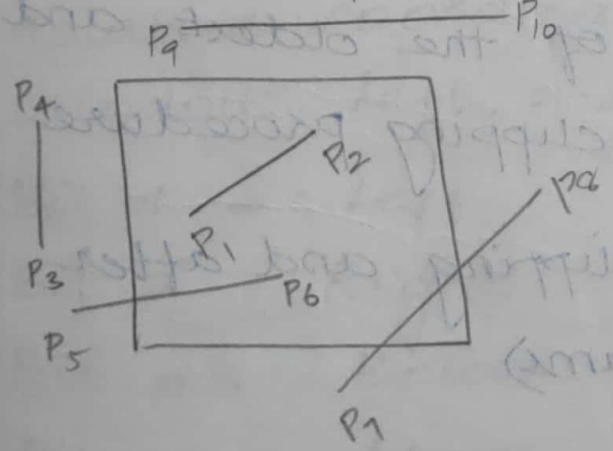
1. Point clipping
2. Line clipping (straight line segment)
3. Area clipping (polygon)
4. Curve clipping
5. Text Clipping

1. POINT CLIPPING : Assuming that the clip window is rect in std pos, we say $P = (x, y)$ for displaying if the following inequalities are satisfied

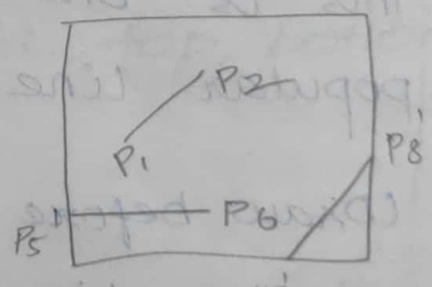
$$xw_{min} \leq x \leq xw_{max}$$
$$yw_{min} \leq y \leq yw_{max}$$

If any of these 4 inequality is not satisfied, point is clipped.

2. LINE CLIPPING :



Before clipping



After clipping

A line clipping proc involves several parts :

First, we can test gm line seg to test whether it lies completely inside clipping window. If not, we try to determine if it lies outside the window. finally if we cannot identify line completely inside or outside we must perform intersection calc with one or more clipping boundaries. we process lines through inside outside test by checking the line end pts.

16M

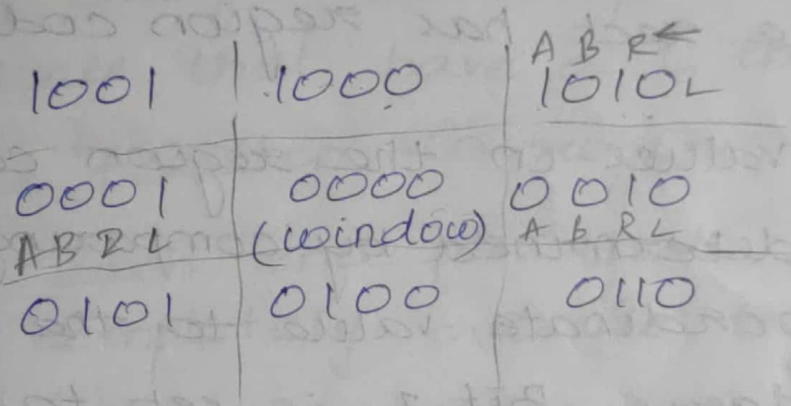
COHEN-SUTHERLAND LINE CLIPPING ALGORITHM : (3)

This is one of the oldest and popular line clipping procedure (Draw before clipping and after clipping diagrams)

Generally this method speeds up the processing the line segments by

performing initial tests by reducing the no of intersections that must be calculated.

Every line end pt in a pic is assigned a 4 digit binary code called region code that identifies location of pts relative to boundaries of the clipping rect.



Region code (4 digit binary code)

Each bit pos in the region code is used to indicate one of the 4 relative coordinate pos wrt clip window. left, right, top, bottom

Bit 1 : Left
Bit 2 : Right
Bit 3 : Below
Bit 4 : Above

Pt that is below and left of the rect is 0101.

A value of 1 in any bit pos indicates that the pt is in relative pos otherwise bit pos is set to zero. If a pt is within a clipping rect, the region code is 0000. The pt that is below & to the left of the rect has region code 0101

Bit values in the region code are determined by comparing end pt coordinate value to the clip boundary. Bit 1 is set to 1 if $x < x_{wmin}$ other 3 bit values can be determined using similar comparison.

Not necessary

Just for understanding

Bit 1 is sign bit of $x - x_{wmin}$

Bit 2 $x_{wmax} - x$

Bit 3 $y - y_{wmin}$

Bit 4 $y_{wmax} - y$

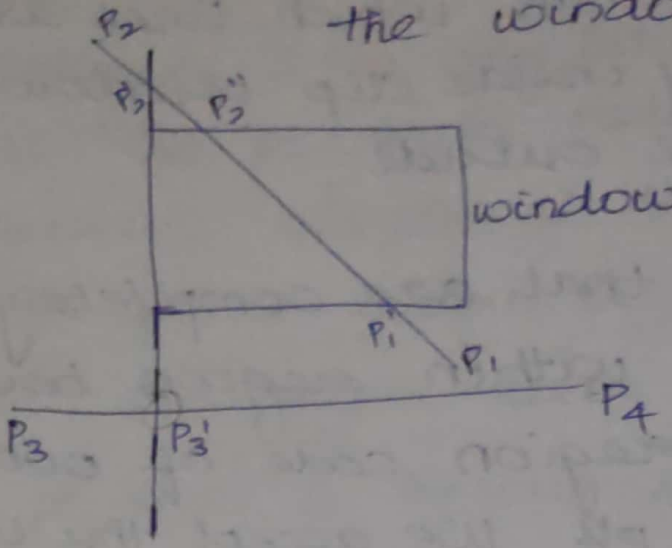
Once we have established region codes for all end pts we can determine which lines are completely inside clip window & which are outside

Any lines that are completely contained within region boundaries have a region code of 0000 for both end pts. We accept this line.
Any lines that have a 1 in the same bit position in region codes for each end pts that are completely outside the clipping rectangle. We reject these lines.
We would discard the line that has a region code of 1001 for one end pt and a code of 0101.
Method used is logical AND operation with both region codes to test line

If result is not 0000, line is completely outside clipping region.

Line that cannot be identified as completely inside / outside by a

clip window by these tests. are checked for intersection the window boundaries



We begin the clipping process for a line by comparing outside end pt to a clipping boundary to determine how much of line can be discarded. Then remaining part of the line is checked against other boundaries & we continue until either the line is totally discarded or a section is found inside the window.

Starting with bottom end pt of the line from P_1 to P_2 against the left, right & bottom boundaries in turn find that this pt is below the

clipping rectangle. We then find the intersection pt P_1' with the bottom boundary and discard the line section from P_1 to P_1'. The line has been reduced to the section from P_1' to P_2

Since P_2 is outside the clip window we check end pt against the bounda and find that it is to the left of window. Eq

Intersection pt P_2' is calculated but this pt is above the window. So the final intersection calculation is P_2'

Line from P_1' to P_2'' is saved.

This completes processing for this line so we save this part and go on to next ~~point~~ line. pt P_3 in the next line is to the left of clip rectangle so we determine intersection P_3' & eliminate line section from P_3 to P_3'. By checking region codes for line section P_3' to

P_4 we find that the remainder of line is below the clip window & can be discarded also.

(Refer algo from book)

LIANG – BARSKY LINE CLIPPING ALGORITHM: (4)

For fast line clippers have been developed that are based on analysis of parametric eqn of a line segment

$$x = x_1 + u\Delta x \qquad y = y_1 + u\Delta y \qquad 0 \le u \le 1$$

where $\Delta x = x_2 - x_1$ $\Delta y = y_2 - y_1$

Using these parametric eqns Cyrus and Beck developed an algo that is generally more efficient than Cohen Sutherland algorithm.

Later Liang and Barsky independently derived an even faster parametric line algo

$$xw_{min} \le x_1 + u\Delta x \le xw_{max}$$

$$yw_{min} \le y_1 + u\Delta y \le yw_{max}$$

Each of these 4 inequalities can be expressed as $up_k \leq q_k$ ~~where $p \neq 0$~~

$k = 1, 2, 3, 4$.

where parameters p and q are defined as $p_1 = -\Delta x$

$q_1 = x_1 - x_{wmin}$

$x_{wmin} \leq x_1 + u\Delta x$

$-x_1 - u\Delta x \leq -x_{wmin}$

$p_2 = \Delta x$

$q_2 = x_{wmax} - x_1$

$-u\Delta x \leq x_1 - x_{wmin}$

$p_3 = -\Delta y$

$q_3 = y_1 - y_{wmin}$

$p_4 = \Delta y$

$q_4 = y_{wmax} - y_1$

Any line parallel to one of the clipping boundaries has $p_k = 0$ for the value of k corresponding to that boundary ($k = 1, 2, 3, 4$) correspond to left, right, bottom and top boundaries respectively.

If for that value of k we also find $q_k < 0$ then the line is completely outside the boundary & can be eliminated from further

consideration.

If $q_k > 0$ line is inside the parallel clipping boundary.

5/2 When $p_k < 0$ the infinite extension of line proceeds from outside to inside of the infinite extension of this particular clipping boundary

If $p_k > 0$ line proceeds from inside to outside

for non-zero value of p_k, $u = q_k / p_k$

For each line, we can calculate values for parameters u_1 and u_2 that define that part of the line that lies within clipped rectangle.

Value of u_1 is determined by looking at the rect edges for which line proceeds from outside to inside p

For these edges we calculate q_k / p_k. Value of u_1 is taken as largest of set consisting of zero

and various values of q.

Value of u_2 is determined by examining the boundaries for which line proceeding from inside to outside $p>0$.

Value of q_k is calculated for each of these boundaries.

Value of u_2 is the min of set consisting of 1 & calculated q values.

Algo

If $u_1 > u_2$, line is completely outside clip window & it can be rejected. Otherwise end pts of clip lines are calculated from two values of parameters u.

line intersection parameters are initialized to values $u_1 = 0$ $u_2 = 1$

For each clipping boundary, approp values for p and q are calculated and used by func clipTest to determine the line can be rejected or whether the intersection parameter has to be adjusted.

when $p > 0$, param α is used to update u_2.

when $p < 0$, param α is used to update u_1.

If updating u_1 or u_2 results in $u_1 > u_2$ we reject the line.

We update the approp u param only if new value results in a shortening of line. When $p = 0$ and $q < 0$ we can discard the line since it is parallel to & outside of this boundary. If the line has not been rejected after all 4 val of p & q have been tested the end pts of clipped line are determined by the val of u_1 and u_2.

not needed

Liang Barsky algo is more efficient than Cohen Sutherland Line Algo since intersection calc are reduced. Each update of param u_1 and u_2 req only one division and window

only once when the final values of u_1 and u_2 have been computed.
On the other hand, CS Line Clipping Alg can repeatedly calc intersection along the line path even though the line may be completely outside the clip window & each intersection calc req both division & multiplication.

NOTE:
Both algo can be extended to 3D clipping.

(not there in syllabus)

(NLN) NICHOLL LINE CLIPPING: By creating more regions around clip window avoids multiple clipping of an individual clipping. Compared to Both Liang Barsky and Cohen Sutherland NLN has only fewer comparisons
It supports only 2D clipping whereas other two supports 3D clipping

~~3D CON~~ ——— X ———

Unit 3 3D concepts

Viewing trans in 3D are much more complicated cox we have many more param to select when specifying how a 3D scene is to be mapped to a display device.

3D Display Method:

Eg: Camera

Parallel projection:

Eg: Victory stand

One method of generating view of solid object is to project pts on the obj surface along parallel pts on to the disp plane. By selecting diff viewing position, we can project visible pts on the obj on to the disp plane to obtain diff 2D views of the obj.

In parallel projection, the parallel lines in world co-ord ~~scene~~ scene proj into parallel line on 2D disp plane

This technique is used in engineering & architectural drawings to represent obj with set of views that maintain relative proposition of the obj. The appearance of solid obj can then be reconstructed from the major views.

Perspective Projection: Another method for generating view of 3D scene is to project pts to the disp plane along converging path. This causes obj farther from viewing pos to be displayed smaller than obj of same size that are nearer to the viewing pos.

In a perspective ~~pos~~ projection, parallel lines in a scene that are not parallel to the disp plane are projected into converging lines. Scenes displayed using perspective projection appear more realistic. Since this is the way our eyes and a camera lens form images. Distant obj appear smaller than obj closer to the viewing pos.

Depth Cueing :- Depth info is important so that we can easily identify, for a particular viewing direction, which is front and which is back for a displayed obj.

A simple method for indicating depth with wire frame disp is to vary the intensity of obj according to dist from viewing pos. Lines closer to viewing pos are displayed with highest intensity. Lines farther away are displayed with dec intensities

Depth cueing is applied by choosing max and min intensity (or colors) values and a range of dist over which intensities are to vary.

Applications of Depth Cueing:

Modeling the effect of atmosphere on perceived intensity of objects. More distant obj appear dimmer to us than nearer obj due to light scattering by dust particles, haze and smoke. Some atmospheric effects can change the perceived color of an obj and we can model these effects with depth cueing.

Visible line and surface Identification:

we can also clarify depth relations in a wire frame disp by identifying visible lines in some way. The simplest method is to highlight visible line or disp them in a diff color.

Another technique commonly used for eng. drawing is to disp non-visible lines as dashed lines. Another approach is to simply remove non-visible lines. But removing hidden lines also removes info. abt the shape of back surfaces of an obj. These visible line methods also identify visible surfaces of objs.

When objs are to be disp with color or shaded surfaces we apply surface rendering procedures to the visible surfaces so that hidden surfs are obscured

Surface Rendering :- Added realism is attained in disp by setting the surf intensity of objs acc to lighting conditions in the scene and acc to assigned surf characteristics. Lighting specs include intensity and pos of light sources and general background

illumination req for a scene.

Surf prop of objs include degree of transparency and how rough or smooth the surf are to be. Procedures can then be applied to generate the crt illumindtions & shadow regions for the scene. Surf rendering methods are combined with perspective & visible surf identification to generate a degree of realism in a displayed scene.

d

Exploded and cutaway views :-

Many graphics packages allows objs to be defined as hierarchial struct so that internal details can be stored. Exploded and cutaway objs can then be used to show internal struct and relationship of obj parts. Eg: Mechanical Design.

An alternative exploding objs into component parts is cut away view. which removes part of visible surfaces to show internal surface structure.

any 4 in 8M

3Dimensional and stereoscopic views:

Another method for adding a sense of realism to a computer generated scenes is to disp objs using 3D or stereoscopic views. 3D views can be obtained by reflecting a rasterimage from a vibrating flexible mirror. The vibrations of mirror are synchronized with the disp of scene on the CRT. As the mirror vibrates, the focal length varies so that each pt in the scene is projected to a position correspondi. to its depth.

Stereoscopic devices present 2 views of a scene one for left eye & one for right eye. Two views are generated by selecting viewing pos that correspond to 2 eye position of a single viewer. These 2 views then can be displayed on alternate refresh cycles of a raster monitor and viewed through glasses that alternately darken first one lens then the other in synchronization with monitor refresh cycles.

3D object Representation

Graphics scene can contain many diff kinds of objs such as trees, flowers, rocks, water, bricks, wood paneling, rubber, paper, marble, steel, glass, plastic and cloth. There is no one method that we can use to describe objs that will include all characteristics of these diff materials scenes and to produce realistic disp, we need to use representations that accurately model obj characteristics.

Representation schemes for solid objs are often divided into 2 broad categories Although not all representation fall neatly into one or other of these 2 categories.

2M { Boundary Representation (B-Reps):
Space Partitioning Representation

B-Reps describes a 3D object as a set of surfaces that seperate obj interior from the environment.

Eg: Polyon facets and spline patche

Space Partitioning Representation are used to describe interior properties by partitioning the spacial region containing an obj into a set of small, non-overalapping, contiguous solids (usually cubes).

Eg: Octree representation.

Note: Polygon and quadratic surfaces provide precise description for simple euclidia objs such as polyhedrons & ellipsoids; spline surfaces and construction techniques are used for designing ~~and~~ gears & other eng. struct with curved surfaces.

Procedural Methods such as fractal construction and particle system allows us to give accurate representation for clumps (clouds) of grass and other natural objs.

Physically based modelling methods using system of interactive forces can be used to describe the non-rigid behaviour of a piece of cloth.

Octree encodings are used to

represent internal features of obj such as those obtained from medical CT images.

Isosurface displays, volume rendering, and other visualization techniques are applied to 3D discrete data sets to obtain visual representations of data.

I Polygon surfaces (8M) diag, table, eqns

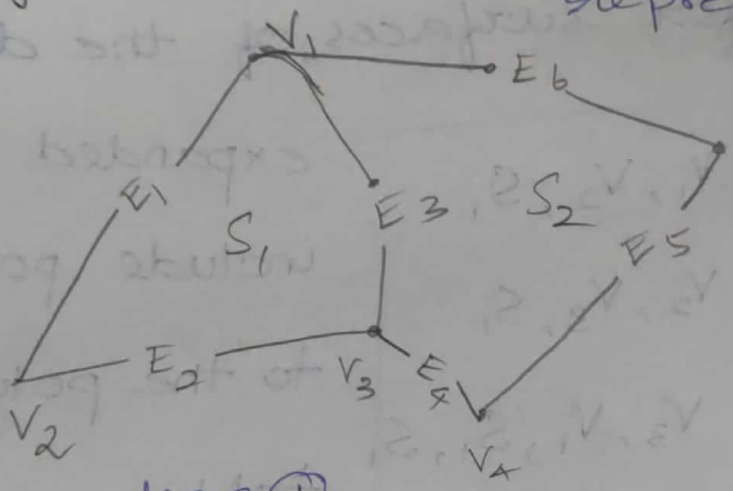
12/12

Polygon surface most commonly used boundary representation for a 3D graphics obj is a set of surface polygons that enclose the obj interior. Many graphic system store all obj descriptions as sets of surf polygons.

2M The reason for using polygon is polygon surf simplifies & speeds up surf rendering & disp obj since all surf all described with linear eqn. For this reason, polygon descriptions are often referred to as "standard graphics object".

NOTE : Many packages allow objs to be described with other schemes such as ~~pte~~ spline surf. that then converted to polygonal representations for processing.

II. Polygon table : (Geometric data table representation for 2 adjacent polygon surf with 6 edges and 5 vertices)



diag (1)

Table :

Vertex Table
$V_1 : x_1, y_1, z_1$
$V_2 : x_2, y_2, z_2$
$V_3 : x_3, y_3, z_3$
$V_4 : x_4, y_4, z_4$
$V_5 : x_5, y_5, z_5$

Edge Table
$E_1 : V_1, V_2$
$E_2 : V_2, V_3$
$E_3 : V_3, V_1$
$E_4 : V_3, V_4$
$E_5 : V_4, V_5$
$E_6 : V_5, V_1$

Polygon Surface Table
S_1 : E_1, E_2, E_3
S_2 : E_3, E_4, E_5, E_6

edge table for surfaces of the diag ① expanded to include pointers to the polygon table.

E_1	: V_1, V_2, S_1
E_2	: V_2, V_3, S_1
E_3	: V_3, V_1, S_1, S_2
E_4	: V_3, V_4, S_2
E_5	: V_4, V_5, S_2
E_6	: V_5, V_1, S_2

Polygon data tables can be organized into 2 groups

1. Geometric Tables
2. Attribute Tables

Geometric data table contain vertex co-ordinates & parameters to identify the spatial orientation of polygon

surfaces.

Attribute info for an obj include parameters specifying the deg of transparency of the obj and its surface reflextivity and texture characteristics.

A convinient organization for storing geometric data is to create 3 list i.e. a vertex table, an edge table and polygon table. Coordinate val for each vertex are stored in vertex table. Edge table contains ptrs back into vertex tables to identify the vertices for each polygon edge and polygon table contains ptrs back into edge table to identify edges for each polygon

Listing the geometric data in 3 tables as in the above diag provide a convinient reference to the individual components (vertices, edges, polygons) of each obj. Also the obj can be displayed efficiently by using the data from the edge table to draw the

component lines.

Alternate arrangement is to use just 2 tables a vertex table & a polygon table. But this scheme is less convinient and some edges could get drawn twice. Another possibility is to use only a polygon table. This duplicates coordinate info Since explicit coordinate values are listed for each vertex in each polygon, also edge info should have to be reconstructed from vertex listings in the polygon table

Since geometric data table may contain extensive listings of vertices and edges for complex objs. It is important that data be checked for consistency and completeness.

15/2 More info included in data tables, the easier it is to check for errors. ∴, error checking is easier when 3 data tables (vertex, edge and polygon)

are used since this scheme provides more info.

Some of the test that could be performed by a graphic packages are

1) Every vertex is listed as an end-point for atleast 2 edges

2) Every edge is a part of atleast one polygon.

3) Every polygon is closed

4) Each polygon has atleast one shared edge

5) If the edge table contains ptrs to polygons, every edge reference by a polygon ptr has a reciprocal ptr back to the polygon.

3D Geometric and modelling Transformation:

Translation: In a 3D homogenous co-ordinate representation, a point is translated from position $P = (x, y, z)$ to position $P' = (x', y', z')$

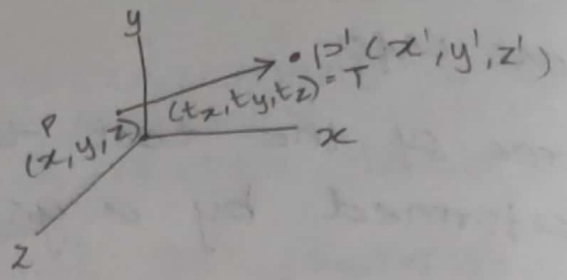
$$P' = T \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{---(1)}$$

Parameters t_x, t_y, t_z specifying translation dist for co-ord directions x, y, z are assigned any real values. Matrix

representation in ① is equivalent to 3 eqns

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



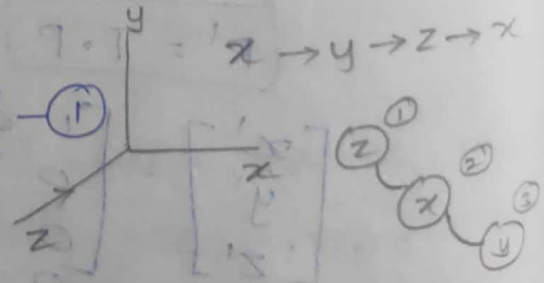
We obtain the inverse of translation matrix in eqn ① by negating the translation dist (t_x, t_y, t_z). This produces a translation in opp direction & the prod of translation matrix and its inverse produces the identity matrix

Rotation : To generate a rotation transf for an obj, we must designate an axis of rotation (abt which the obj is to be rotated) and amt of angular rotation.

Note : In 2D applications where all transf are carried out in xy plane, a 3D rotation can be specified around any line in the space. The easiest rotation axis to handle are those that are ||el to co ordinate axis

Co ordinate axis Rotation : 2D z-axis rotation equations are easily to extended to 3D

$x' = x\cos\theta - y\sin\theta$
$y' = x\sin\theta + y\cos\theta$
$z' = z$ } — ①



Parameter θ specifies the rotation angle

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

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

Transf eqns for rotation abt other 2 co ordinate axis can be obtained with a cyclic permutation of the co-ordinate parameters x, y and z in ① we use the replacements $x \to y \to z \to x$ → ②

Substituting permutation ② in ①, we get the eqn for x axis rotation

$$\left.\begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned}\right\} \to ③$$

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

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

By using cylic ~~parameters~~ permutation in ③, we get transf eqn for y-axis rotation,

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

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

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

(An inverse rotation matrix is formed by replacing the rotation angle 'θ' by '$-\theta$'. Neg values for rotation angles

generate rotation in clockwise dir. So the identity matrix is produced when any rotation matrix is multiplied by its inverse.

∴ only the sine func is affected by the change in sign of rotation angle. (No need) The inverse matrix can also be obtained by interchanging rows and columns.

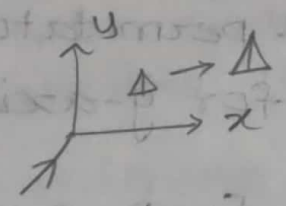
$$R^{-1} = R^T$$

Scaling: The matrix epression for scaling transf of a pos $P=(x,y,z)$ relative to the co-ord origin can be written as

$$P' = S \cdot P$$

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

where scaling parameters S_x, S_y and S_z are assigned the values.



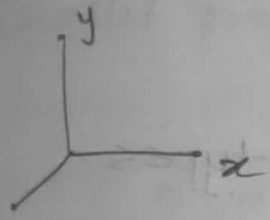
$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$
$$z' = z \cdot S_z$$

Scaling an obj with transf (1) changes the size of an obj & reposition the obj relative to the co-ord origin. Also if the transf parameters are not all equal, relative dimensions on the obj are changed. We preserve the original shape of an obj with uniform scaling $S_x = S_y = S_z$

Scaling wrt a selected fixed pos (x_f, y_f, z_f) can be represented with the foll sequence.

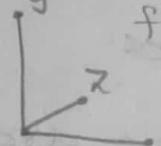
1. Translate the fixed pt to the origin
2. Scale the obj relative to the coord origin using eqn (1)
3. Translate the fixed pt back to original pos

Other Transformation :- Reflection



Refl relative to xy plane

conversion of coord specs from a RHed to a LHed sys can be carried out with the z refl trans using eqn (1)

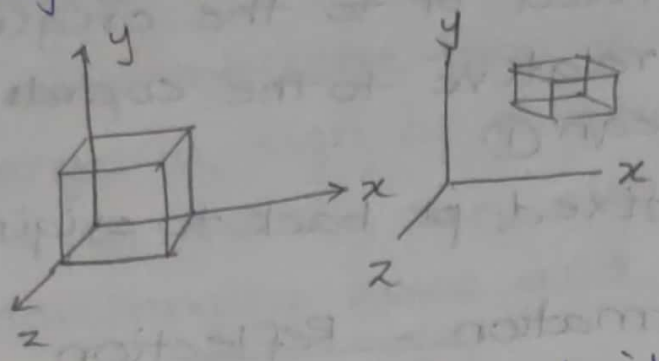


A 3D reflection can be performed relative to a selected refl axis or wrt a selected refl plane. Reflections related to a gvn axis are equivalent to 180° rotation abt the axis. When the refl plane is a coord plane (either xy, xz or yz). We can think of trans as a conversion b/w LHed and RHed system

An eg of a refl that converts co-ord specs from a RHed sys to a LHed sys (or vice versa) is shown in abv diag. This transf changes the sign of z-coord leaving x & y coord values unchanged. The matrix representation for this refl of pts relative to xy plane is

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{--- (1)}$$

in 3D viewing for obtaining projection transformation:



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

Parameters 'a' and 'b' can be assigned any real values.

8M Projection: There are two types

① Parallel projection ② Perspective proj

Once a world co-ordinate description of the objs in a scene are converted to viewing co-ord. We can project 3D objs onto the 2D view plane. There are 2 basic proj method

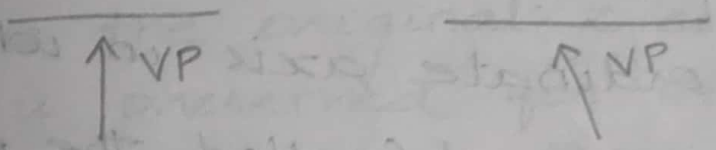
1st projection: co ord position are transformed to the view plane along parallel lines.

Perspective projection: Here, obj positions are transformed to the view plane along lines that converge to a pt called projection reference pt (or centre of projection). The projected view of an obj is determined by calculating the intersection of projection lines with the view plane.

19/2 SM Parallel projection 11th & persp 16M

We can specify a parallel projection with a projection vector that defines the direction for projection line

If The proj is perpendicular to view plane we have orthographic parallel proj otherwise we have oblique parallel projection.



orthographic projection

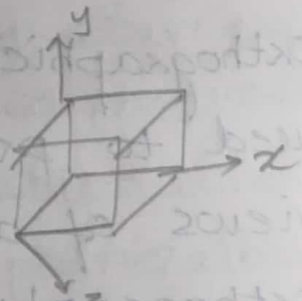
oblique projection

Orthographic projections are most often used to produce front, side and top views of an obj. Front, side and rear orthographic proj of an obj are called elevation. Top orthographic proj is called plan view

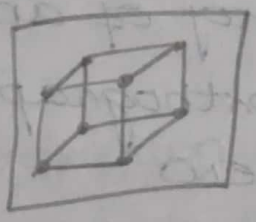
Engineering and architectural drawings commonly employ these orthographic proj bcz length and angles are accurately depicted and can be measured from the drawing.

We can also form orthographic projections that display more than one face of an obj. Such views are called axonometric orthographic projection. The most commonly used axonometric projection is isometric projection. We can generate an isometric projection by aligning the proj plane so that it intersects each coordinate axis in which the obj is defined (called the principle axis) at the same dist from the origin.

Isometric proj of cube

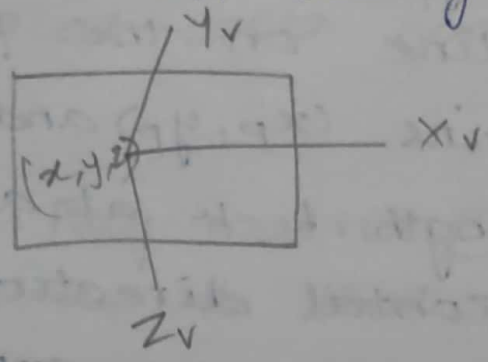


(Ref book)



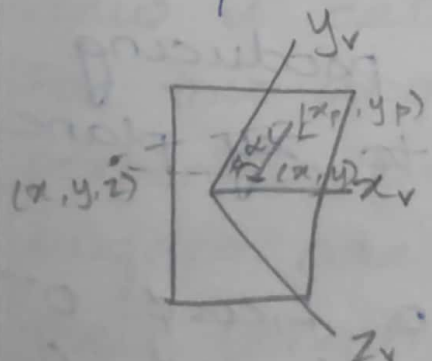
There are 8 pos one in each octant for obtaining an isometric view. All the 3 principle axis are equal in isometric proj whereas it is not equal in axonometric proj.

Trans eqn for orthographic parallel proj if the view plane is placed at position Z_{VP} along the Z_v axis



Then any pt (x, y, z) in viewing coordinates is transformed to projection coordinates, $x_p = x$; $y_p = y$ where the original z coordinate val is preserved for depth info needed in depth cueing and visible surface determination procedure.

An oblique proj is obtained by projecting pts along parallel lines that are not perpendicular to the proj plane. An oblique proj vector is specified with 2 angles, α and ϕ. Pts (x, y, z) is projected to a position (x_p, y_p)



oblique proj of coord pos (x, y, z) to position (x_p, y_p) on the viewplane

Orthographic projj coord on the plane are (x, y)

from (x, y, z) to (x_p, y_p) makes an angle ϕ with the line on the proj plane that joins (x_p, y_p) and (x, y). This line of length L is at angle ϕ with the horizontal direction in the proj plane. We can express proj coord interms of x, y, L and ϕ as $x_p = x + L\cos\phi$; $y_p = y + L\sin\phi$ ↳ (1)

Length L depends on angle α and z coord of pt to be projected,

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

where $L_1 = \frac{1}{\tan\alpha} = \tan^{-1}\alpha$ which is also called the value of L when $z = 1$.

(1) becomes,

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

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

The transf matrix for producing any parallel proj on to xy plane can be written as

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad -(2)$$

An orthographic proj is obtained when $L_1 = 0$ (which occurs at a proj angle of 90°). Oblique proj are generated with non-zero values for L_1.

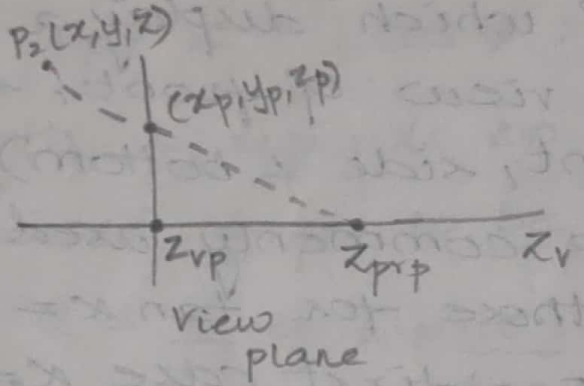
Note: Proj matrix (5) has a similar structure to that of z axis share matrix.

Common choices for angle ϕ are 30° and 45° which display a combination view of front, side & top (or front, side & bottom) of an object. Two commonly used values of α are those for $\tan\alpha = 1$ and $\tan\alpha = 2$. For first case $\alpha = 45°$ & the views obtained are cldcavalior projection. All lines perpendicular to proj plane and projected with no change in length.

When proj angle α is chosen, so that $\tan\alpha = 2$, the resultant view is called cabinet proj. For this angle (approx 63.4°), lines perpendicular to the viewing surface are projected at one-half their length. Cabinet proj appear more realistic than cavalior proj bcz of this reduction in length of

perpendicular — 8M

23/2 8M Perspective projections: To obtain a perspective projection of a 3D object, we transform points along proj lines that meet at the proj reference pt. Suppose we set the proj ref pt at position z_{prp} along the z_v & we place the view plane at z_{vp}



As shown in the abv diag, we can write eqns describing co ordinate pos along this perspective projection line in parametric form as,

$$\left.\begin{aligned} x' &= x - xu \\ y' &= y - yu \\ z' &= z - (z - z_{prp})u \end{aligned}\right\} \quad -(1)$$

Parameter 'u' takes values from 0 to 1 and co ordinate pos (x', y', z') represents any pt along the proj line. when $u = 0$, we are at pos $P = (x, y, z)$ when $u = 1$, we have proj ref pt coordinates $(0, 0, z_{prp})$

On the view plane $z' = z_{vp}$. We can solve z' eqn for parameter u.

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$ → (2)

Substituting this value of u, in eqns for x' and y', we obtain the perspective transf eqn,

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

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z - z_{prp}}\right) = y\left(\frac{d_p}{z - z_{prp}}\right)$$ → (3)

where $d_p = z_{prp} - z_{vp}$ is the dist of view plane from proj ref pt.

Using a 3D homogenous coordinate representation, we can write the perspective proj trans (3) in matrix form as,

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

In this representation, homogenous factor

$$h = \frac{z - z_{prp}}{d_p}, \quad x_p = \frac{x_h}{h}; \quad y_p = \frac{y_h}{h}.$$

If the view plane is taken to be uv plane, then $z_{vp} = 0$, then the proj coordinates are, Put $z_{vp} = 0$ in (3)

$$x_p = x z_{prp}/z - z_{prp}; \quad y_p = y z_{prp}/z - z_{prp}$$

$$x_p = \frac{l}{\frac{z}{z_{prp}} - 1} \qquad y_p = \text{(Ref Book)}$$

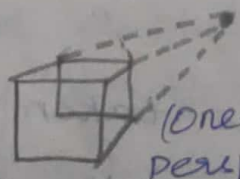
When a 3D obj is projected onto a view plane using persp eqn, any set of parallel lines in the obj that are not parallel to the plane are projected into converging lines. Parallel lines that are ||el to view plane will be projected as ||el lines.

Q.M Point at which a set of projected ||el lines appears to converge is called a vanishing point.

NOTE: In general a scene can any no of vanishing pts depending on how many set of ||el pts are there in scene.

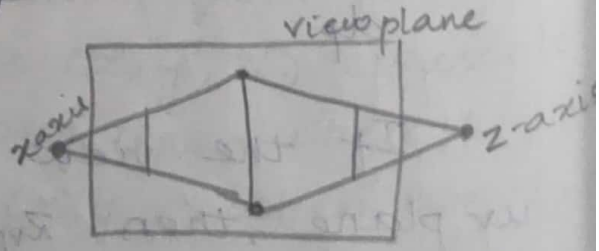
not needed for exam

Vanishing pt for any set of lines that are ||el to one of principle axis of an obj is referred to as principle vanishing.



(One-pt perspective projection)

(1)



(Two-pt perspective projection)

(2)

We control the no of principal vanishing pts (one, two or three) with the orientating of projection plane & perspective proj are acc classified as one-point, Two-point, Three-point proj. No of princ. vanishing pt in a proj is determined by the no of princ axis intersecting the view plane. In diag ① view plane is aligned ||el to xy object plane so that only z-axis is intersected. This orientation produces One-Point proj with z-axis vanishing pt.

In diag ②, the proj plane intersects both x and z axis but not y axis. The resulting Two-Point persp proj contains both x and z axis vanishing pts.

—8M

VISIBLE SURFACE REDUCTION METHOD:-

A major consideration in generat of graphic disp is identifying those parts of a scene that are visible from a chosen viewing position. There are many approaches we can take to solve this problem. Some methods req more memory. Some involve more processing time. Some apply only to spl types of objs. Deciding upon a method

for a particular application can depend on such factors as complexity of scene, types of objs to be displayed, available equipment, whether static or animated displays are to be generated. Various algo are referred to as Visible-Surface Reduction Methods. Sometimes these methods are ref to as Hidden-Surface Elimination method.

CLASSIFICATION OF VISIBLE SURFACE REDUCTION ALGORITHM:

2M

Visible surf red algo are broadly classified acc to whether they deal with obj defn directly or with projected imgs. Approaches are called Object-Space Method & Image-Space Method.

2M

An obj-space method compares objs & parts of objs to each other within the scene defn to determine which surf as a whole we should label as visible:

2M In img-space algo, visibility is decided pt by pt at each pixel position on the proj plane.

Most visible surf algo, use img-space method although obj-space method can be used effectively to locate visible surf. in some cases.

Diff btw: most visible surf ~~red~~ algo use sorting and coherence method to improve performance.

Sorting is used to facilitate depth comparison by ordering the individual surfaces acc to the dist from the view plane.

Coherence method are used to take adv of regularities in a scene.

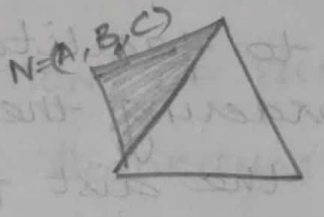
F 8M
BACK-~~S~~FACE REDUCTION: Fast and simple obj-space method for identifying the back faces of a polyhedron is based on inside-outside test. A pt (x, y, z) is inside a polygon surf with plain parameters A, B, C and D if ~~$ax+by+cz+d<0$~~

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

26/2

When an inside pt is along the line of side to the surf, the polygon must be a back face (we are inside that face and cannot see the front of it from our viewing position

We can simply test by considering the normal vector N,

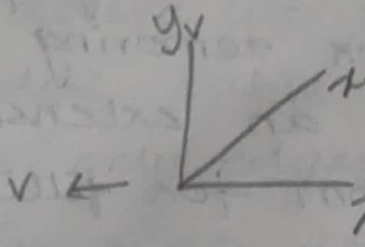


The vector N, to the polygon surf which has cartesian compone
N (A, B, C)
In general, V is a vector in the viewing direction from the eye (or camera) pos as shown in above diag. Then this polygon is a back face $V \cdot N > 0$

If obj description have been converte to proj coordinates and our viewing direction is ||el to Z_v axis

Then $V = (0, 0, V_z)$ and $V \cdot N = V_z C$

In a right handed viewing system with the viewing dir along the negative z_v axis as in below diag.



The polygon is a back face if $C < 0$

Also we cannot see any face whose normal as z component $C = 0$

Thus in general, we can label any polygon as a back face if it is normal vector as a z component val $C \leq 0$.

Left handed system - clockwise dir

Right handed system - counter clockwise dir

By examining parameter C for diff planes defining an obj we can immediately identify all back faces. If C contains only non-overlapping polyhedra, then again all hidden surf are identified with back face method.

Back face removal can be expected to remove abt half of polygon surf in a scene ~~from~~ to compute further the visibility test.

Depth Buffer Method - not there in syllabus

SCAN-LINE METHOD :- This img space method is used for removing hidden surfaces which is an extension of scan line algorithm for filling polygon interiors. Instead of filling just one surf, we now deal with multiple surfaces. As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible. Across each scan line, depth calculations are made for each overlapping surf to determine which is nearest to view plane. When visible surf has been determined, the intensity val for that pos is entered into refresh buffer. We assume that tables are set up for various surfaces that include edge table and polygon table. ~~A~~ edge table contains co ordinate end pts for each line in a scene, inverse slope of each line & pts into polygon table to identify surf bounded by each line.

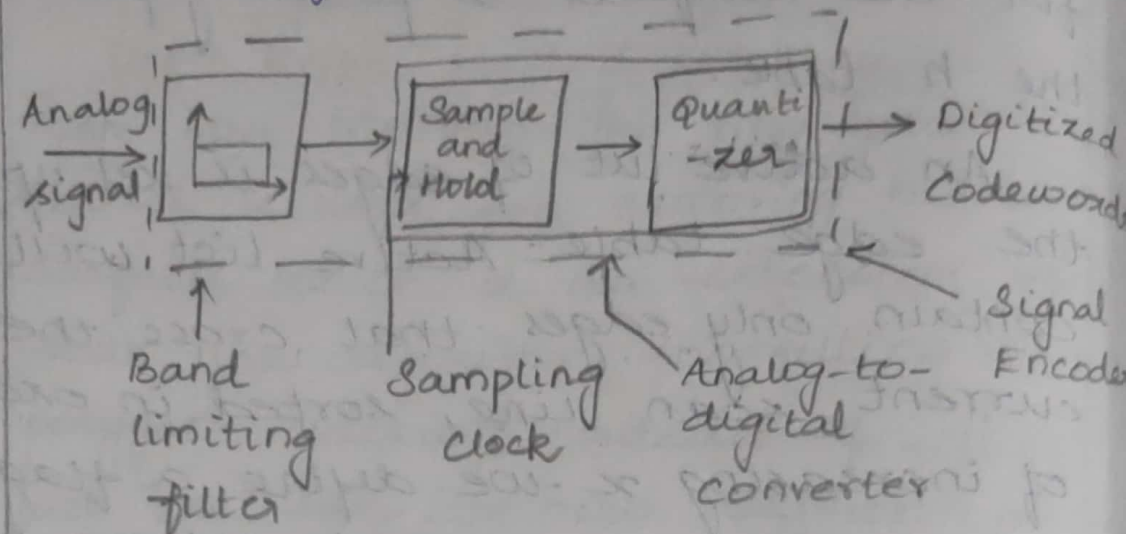
Polygon table contains co-eff of plane eqn for each surf, intensity info for the surfaces & possibly ptrs into the h table.

An active list of edges is set up in the edge table. Active list will contain only edges that cross the current scan line, sorted in order of increasing x. We define a flag for each surf that is set on or off to indicate whether a pos along a scan line is inside/outside a surface.

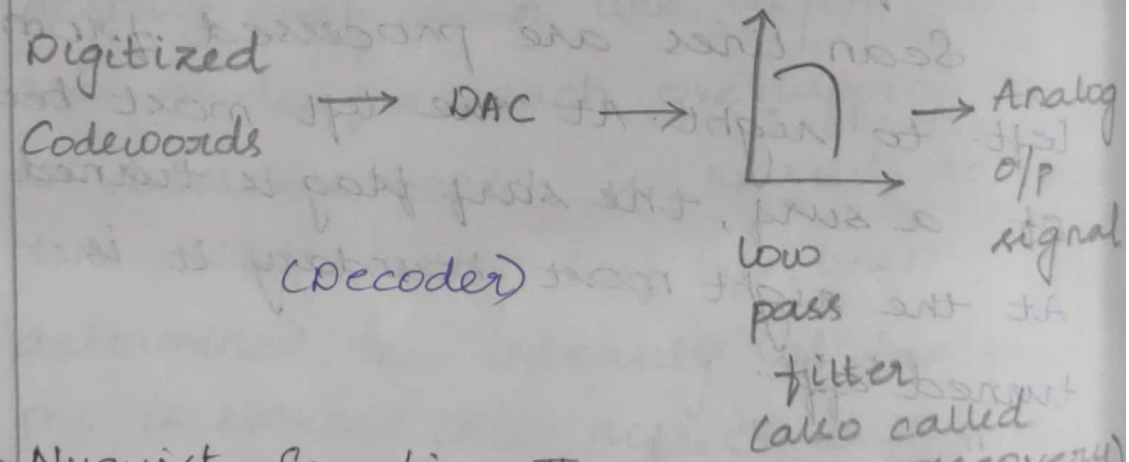
Scan lines are processed from left to right. At the left most bounda of a surf, the surf flag is turned on. At the right most boundary it is turned off.

2|3 UNIT 5

Encoded design, decoded design 8M. 4M 4M



diag: explain every block (Encoder)



(Decoder)

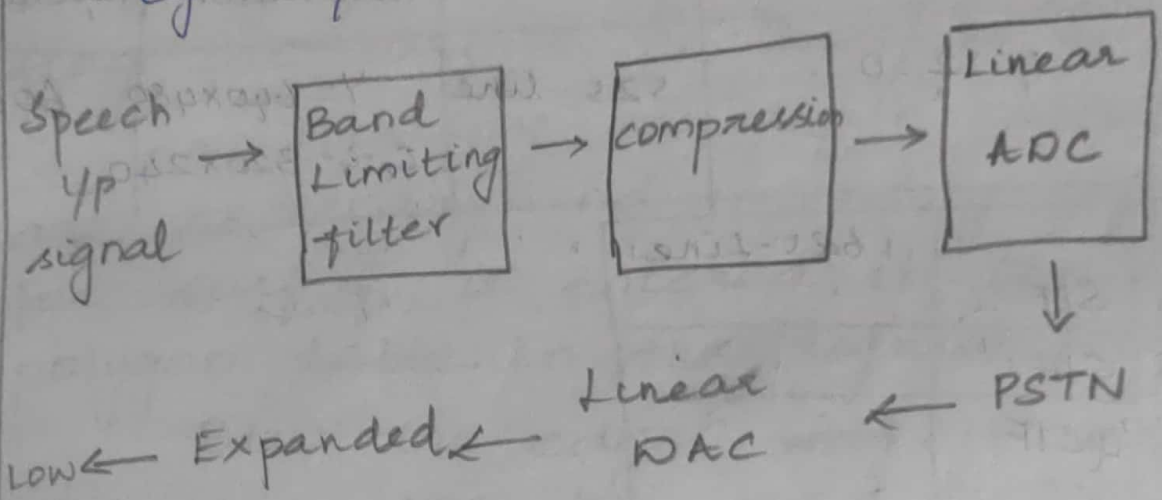
(2M) Nyquist Sampling Theorem

time varying analog signal

Text — Unformatted (ASCII) table not needed
— Formatted
— Hyper

- Digitized document (eg: FAX)
- Digitization Color Principle
- NTSC

- Digital cameras and scanners
  CCD → charge couple device
- Audio
- PCM speech (SM) diag ✗
  eg: Telephone



CD quality Audio
MIDI

original Digitization format 4:2:2

525 - Line System
4:2:0 format

Video formats (SM)

Refresh Rate Resolution
HDTV
Source Intermediate Format

CIF. video conferencing application

CIF / SIF ; QCIF ⇒ video telephony

PC Video Digitization format

Table (X) 2M | 8M

Digitization format	System	Spatial Resolution	Tempral Resolution
4:2:0	525-line	Y = 640×480 C_b-C_r = 320×240	60Hz

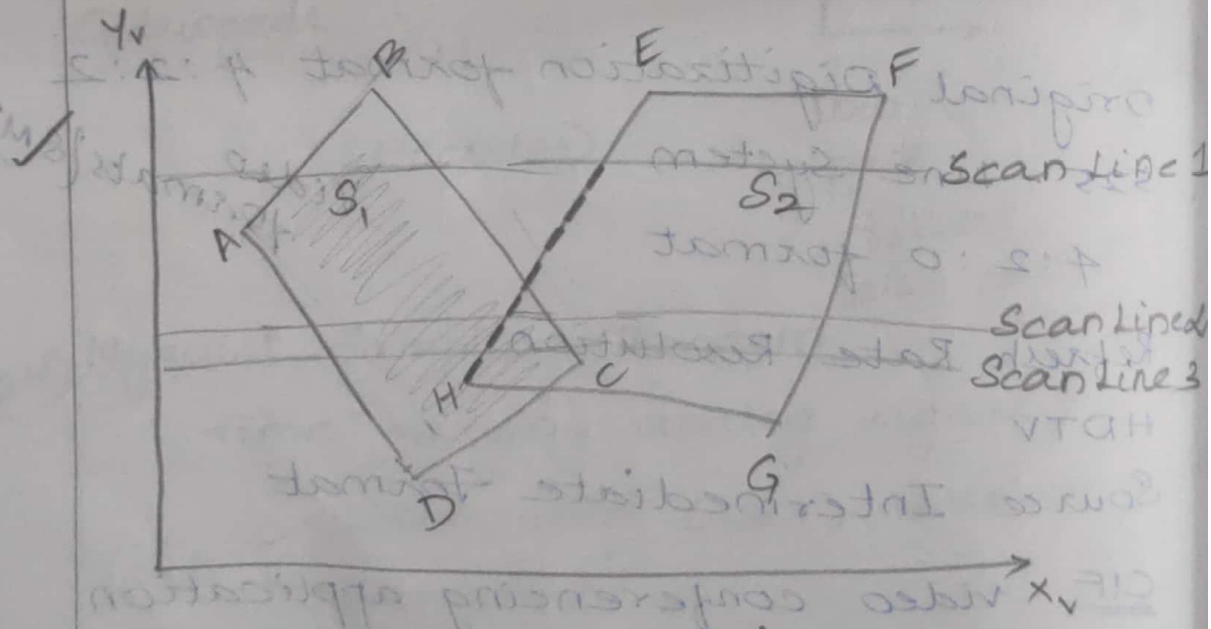
	625-Line
SIF	
CIF	
QCIF	

— X —

Unit-3

5/3/18

SCAN-LINE METHOD:



Edges, surface, & explanation

Dashed line indicate boundary of hidden surfaces

Active list for scan line 1 contains info from the edge table for the edges AB, BC, EH, FG. For positions along the scan line btw edges AB and BC only the flag for surface S_1 ~~Therefore~~ S_1 is on. Therefore no depth calculations are necessary and intensity info for surf S_1 is entered in the polygon table. in the refresh buffer.

Btw the edges ~~only~~ EH and FG only flag for the surf S_2. No other pos along the scan line 1 intersect surfaces, so ~~the~~ the intensity values in other areas are set to background intensity.

For scan lines 2 and 3 in the abr diag, active edge list contains edges AD, EH, BC & FG. Along scanline 2 from edge AD to edge EH only the flag for the surface S_1 is on. Btw the edges EH and BC flags for both surfaces are on. We can take advantage of coherence along the scan lines as we pass from one scan line to next.

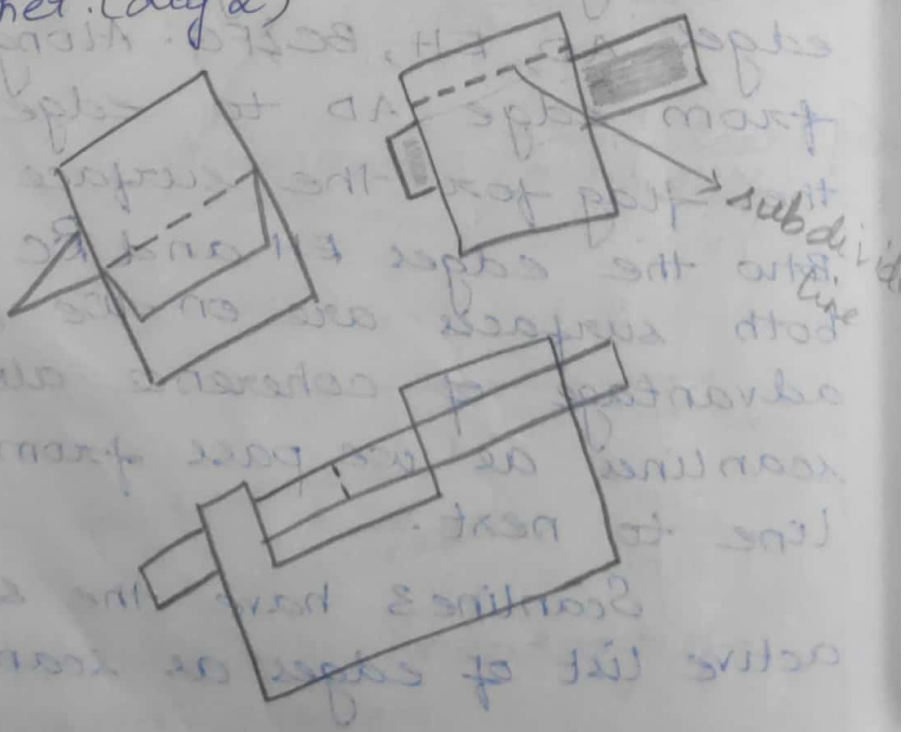
Scanline 3 have the same active list of edges as scan line 2.

Since no changes has occured in line intersection it is unnecessary again to make depth calculations btw edges BC and EH. Two surfaces must be in same orientation as determined on scanline a, so the intensity for surf 1 can be entered without further calc.

~~Dashed lines indicate the~~

Any no of overlapping polygon surfaces can be processed with this scan line method. Flags for the surfaces are said to indicate whether a position is inside or outside and depth calc are performed if surfaces are overlapped. When these coherence methods are used, we need to be careful to keep track which surface section is visible on each scan line.

This works only if surfaces dont cut through or otherwise cyclically overlap each other. (dig 2)

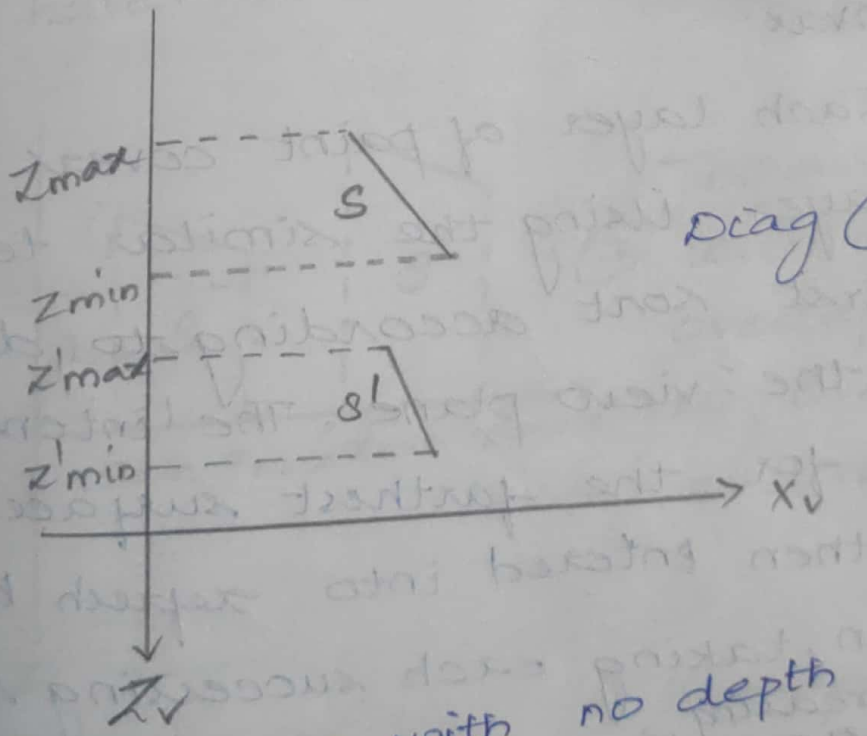


If any kind of cyclic overlap is present in a scene, we can divide the surfaces to eliminate overlaps. The dashed lines in this fig indicate where planes could be subdivided to form two distint surfaces so that cyclic overlaps are eliminated.

Depth Sorting Method :- (Painter's algo) Using both img space and obj space operations, depth sorting performs the foll basic funcs

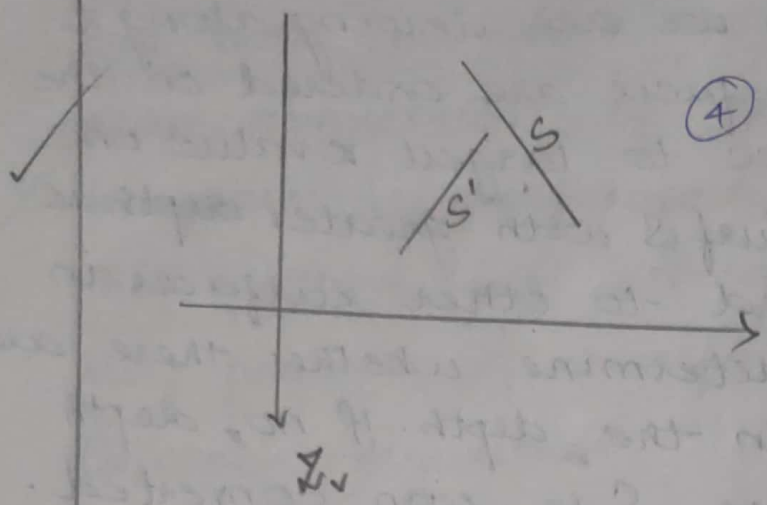
1) Surfaces are sorted in order of decreasing depth.

2) Surfaces are scanned converted in order, starting with surface of greatest depth

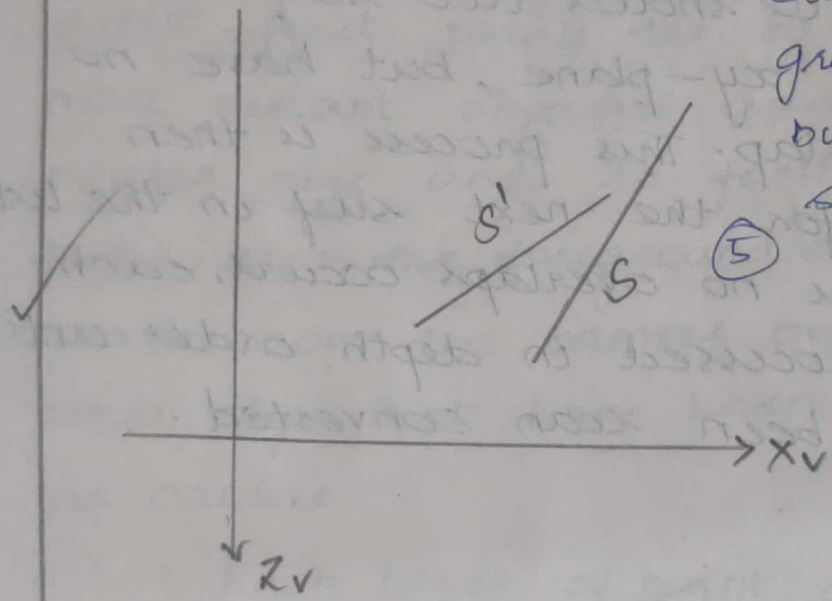


Diag (1)

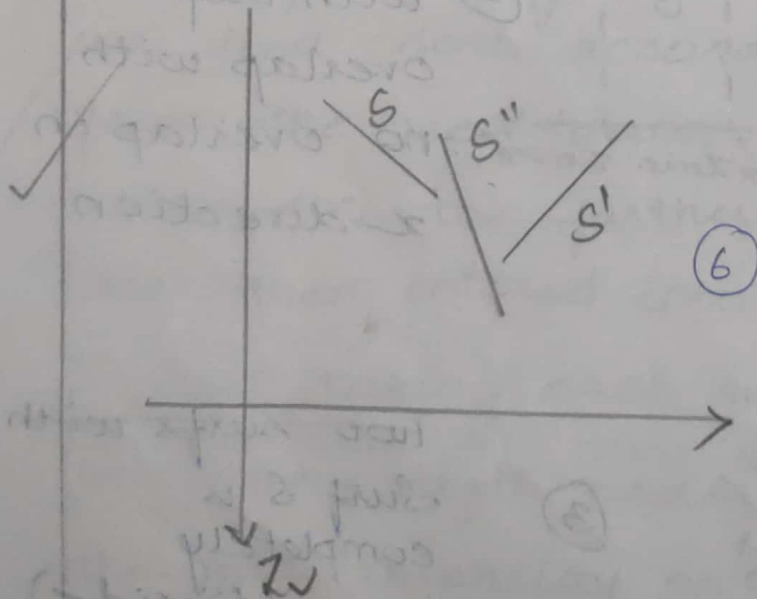
(Two surfaces with no depth overlap)



④ Overlapping surf S' is completely in front (outside) of the surf S but S is not completely beyond S'.



⑤ Surf S has greater depth but obscures surf S'.



⑥ Three surfaces entered into sorted surf list in the order S, S'', S' should be reordered as S', S'', S

If a depth overlap is detected in any pt in the list, we need to make additional comparison to determine whether any of the surf must be reordered.

Foll test for each surf that overlap with S must be considered. If any one of these test is true, no reordering is needed for that surf. Tests are listed in order of inc difficulties

1) Bounding rect in xy plane for 2 surfaces donot overlap

2) Surf S is completely ~~behind~~ beyond the overlapping surf relative to viewing pos

3) overlapping surf is completely in front of S, relative to viewing position

4) Proj of 2 surfaces on to the view plane donot overlap

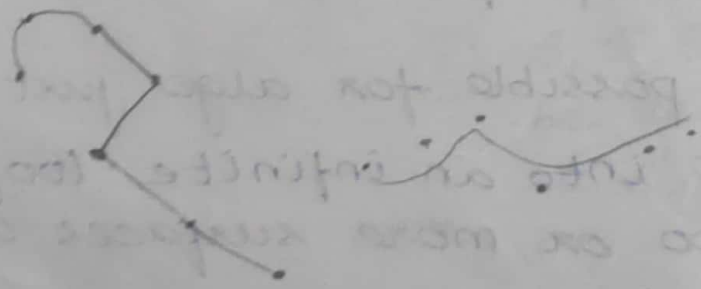
Test 1 is performed in two parts

7/3 It is possible for algo just outlined to get into an infinite loop if
2M ~~in~~ two or more surfaces alternatively obscured each other in such a situation, the algo will continually reshuffle the positions of the overlapping surf

surface that has been re ordered to a further depth position so that it cannot be moved again. If an attempt is made to switch surf is again timed we divide it into 2 parts to eliminate cyclic overlap. The original surf is then replaced by 2 new surfaces & we continue processing as before.

16M SPLINE REPRESENTATION:

2M A spline is a flexible strip used to produce a smooth curve through a designated set of pts. Several small blades are distributed along the leng of strip to hold it in the pos on the drafting table as the curve is draw



Mathematically, we can describe such a curve by piece wise polynomial func whose first and second derivatives

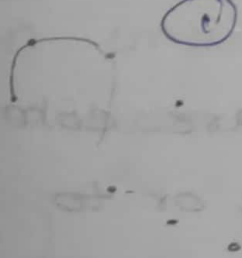
are continuous across the various curve sections.

Use:
Splines are used in graphic applns to design curve and surf shapes, to digitize drawings for computer storage and to specify animation path for the objs or camera in a scene. Typical CAD applns for spline include the design of automobile bodies, aircraft & space craft surfaces and ship hulls.

Spline curve refers to any composite term curve formed with polynomial section satisfying specifying continuity conditions at the boundary of pieces.

Spline surface can be described with 2 sets of orthogonal spline curves

Interpolation & Approximation Spline:



Set of 6 ctrl pts interpolated with piece wise continuous polynomial section

Set of 6 ctrl pts approximated with piece wise cont. polynomial section

A spline curve is specified by giving a set of coord position called ctrl pts which indicate the general shape of the curve. These ctrl pts are then fitted with piece wise continuous parametric polynomial func in one of the 2 ways

i) Interpolate

ii) Approximate

When polynomial sections are fitted so that the curve passes thru each ctrl pt as shown in diag (1). Resulting curve is said to interpola the set of ctrl pts.

Polynomials are fitted to general ctrl pt path without necessarily passing thru any ctrl pt. Resulting curve is said to approximate the set of ctrl pts.

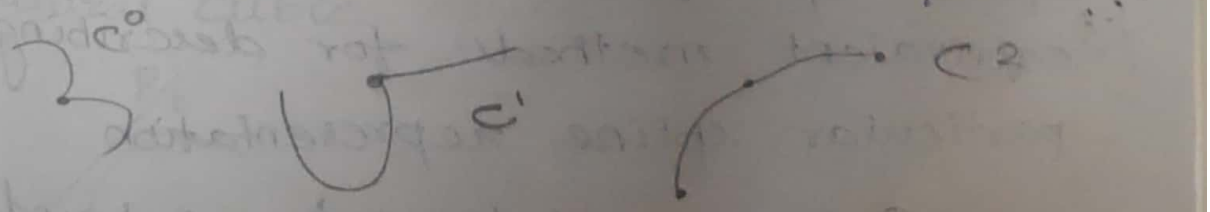
Interpolation curves are common used to digitize drawings or to specify animation ~~path~~ part.

Appr. curves are primarily used as design tool to structure obj surfa

A spline curve is defined, modified and manipulated with operations on the ctrl pts. By interactively collecting spacial pos for ctrl pts, a designer can set up a initial curve. After a polyn fit is displayed for a gvn set of ctrl pts, the designer can then reposition some or all of the ctrl pts to restructure the shape of the curve. In addition, curve can be translated, rotated or scaled with transf. applied to the ctrl pts. CAD packages can also insert extra ctrl pts to help a designer in adjusting the curve shapes.

Parametric Continuity Conditions :

To ensure a smooth transition from one section of piece wise parametric curve to the next, we can impose various continuity condition at the connection pt.



Zero-order parametric continuity described as C^0 continuity means

simply that the curve meets.

First order parametric continuity referred as C^1 continuity means the first order parametric continuity (tangent lines) of the co-ord func.

Second order parametric continuity or C^2 continuity means that both the first & seco parametric derivatives of the 2 curve sections are same at the intersection (6M

Geometric Continuity Condition: An alternate method for joining 2 successive curve sections is to specify conditions for geometric continuity.

Zero order geometric Continuity describ as G^0 is same as zero order parametric continuity

G^1 same as C^1 ; G^2 same as C^2

Spline specification: There are 3 equivalent methods for describing particular spline representation

(1) We can state set of bound conditions that are imposed on

the spline

(or) (2) we can state matrix that characterize the spline (or) (2) we can state the set of blending funct (basis func) that determine how to specify geometric constraints on the curve combined to calculate the pos along the curve path

Cubic Spline Interpolation Method

16M This class of splines is most often used to set a path for obj motions (or) to provide a representation for an existing object

$$P_k = (x_k, y_k, z_k) \quad k = 0, 1, 2 \ldots n$$

$$x(u) = a_x u^3 + b_x u^2 + c_x u$$

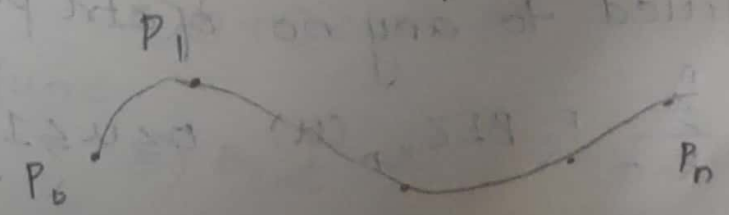
$$y(u) =$$

$$z(u) =$$

$$0 \leq u \leq 1$$

Types of cubic spline:

1) Natural Cubic spline:



2) Hermite Interpolation:

$P(0) = P_k$; $P(1) = P_{k+1}$; $P'(0) = DP_k$

$P'(1) = DP_{k+1}$

3) Coordinal Spline: piece wise cubics

$P(0) = P_k$; $P(1) = P_{k+1}$;

$P'(0) = \frac{1}{2}(1-t)(P_{k+1} - P_{k-1})$

$P'(1) =$

4) Kochanek-Bartels Splines

$P(0) =$

$P(1) =$

$P'(0) =$

$P'(1) =$

Bezier Curves and Surfaces:

(approximation method)

surf design, CAD systems ⇒ use

adv can be fitted to any no of ctrl pts

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

Bezier Blending func,

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

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

Types

1. Cubic Bezier Curve

$n=3$; $BEZ_{n,3}(u) = (1-u)^3$

Bezier surface :

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

B-Spline (Bezier spline)

advantages
1. deg of B-spline polyn is ——
2. local

B-Spline Curves

$$P(u) = \sum_{k=0}^{n} P_k\, B_{k,d} \quad \text{(ref book)}$$

Properties of B-spline curve

1) deg is $d-1$; continuity is C^{d-2}
2) $(n+1)$ ctrl pts will have $(n+1)$ blending func.
3) $B_{k,d}$ => knot value
4) $n+d$ $n+d+1$

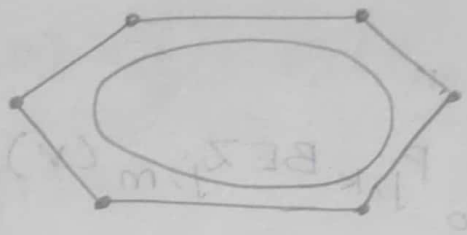
B-spline Types

① Uniform periodic B-splines

knot value, knot vector

② Uniform Quadratic B-spline

③ Cubic periodic B-spline → closed curve.



P(0) P(1) P'(0) P'(1)

④ Open, uniform B-splines

⑤ Non - Uniform B-spline

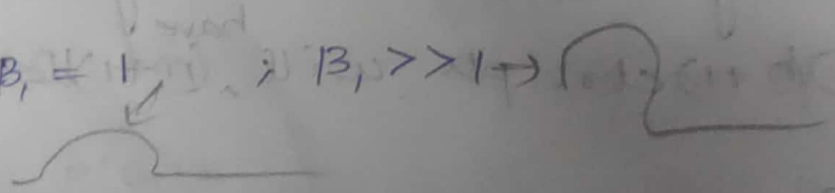
flexibility curve shape

B-spline surfaces

① Beta - splines (generalization of B-spline surf)

$$P_{j-1}(u_j) = P_j(u_j)$$

$\beta_1 = 1$; $\beta_1 >> 1$ →



Rational Splines : ratio of 2 spline

$$P(u) = \sum_{k=0}^{n} w_k P_k B_{k,d}(u)$$ } Book

adv model the curves, shapes

Displaying Spline curves and Surfaces

3 methods

1. Horner's Rule : successive factoring

$$x(u) = a_x u^3 + b_x u^2 + c_x u + 1.$$

2. Forward Difference calculation (Δ)

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

forward difference $x_0, \Delta x_0, \Delta^2 x_0$

3. Sub division method

before subdivision — after subdivision

book