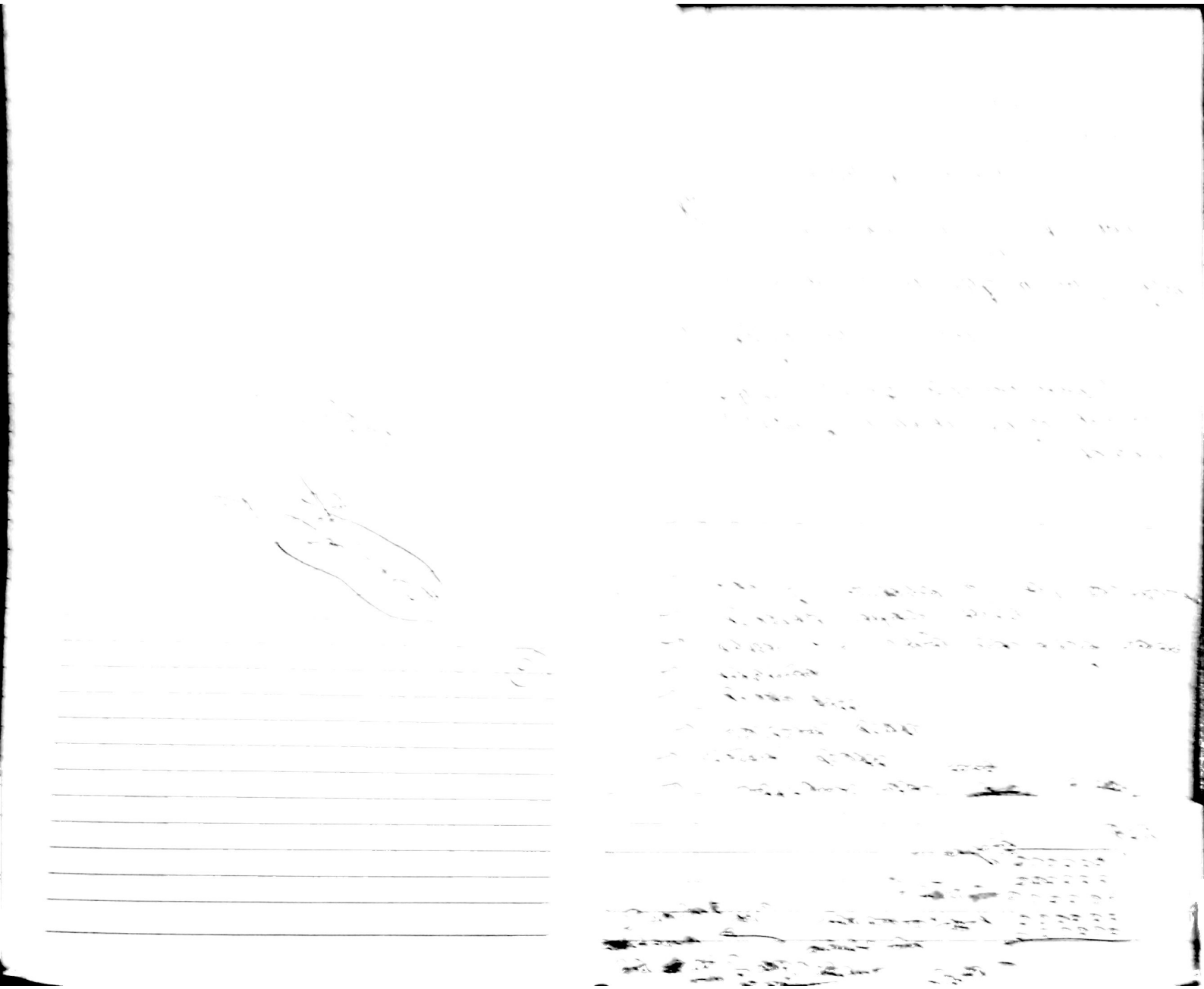


Date :
Page No.

Raster device - Raster is fixed.
Random " - Depends on complexity of figure.

CG Lab

Date :
Page No.

Digital Data Analysis
:DDA Algorithm

It help a computer to compute.

1) Pixel - which pixels to plot
2) Computations -

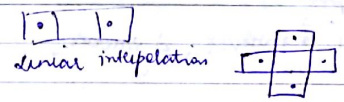
fill data b/w pts are ↓ to function

1) The DDA line drawing algorithm interpolates values in interval

$[(x_{start}, y_{start}), (x_{end}, y_{end})]$

$m = \frac{\Delta y}{\Delta x}$, $\Delta y = y_{end} - y_{start}$

& $\Delta x = x_{end} - y_{end}$



linear interpolation

bicubic smoothing interpolation

→ A line is sampled at x unit intervals in one coordinate and corresponding integer values nearest to the line path are determined for other coordinates.

* If slope < 1,
we sample at unit x intervals ($\Delta x = 1$) & compute successive y values as

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

i.e. plot (x, round(y))
$x = x + 1$ $y = y_0 + m.x$ $x = 1$
$y = y_0 + m$ till $x_0 < x$,

(1)

In DOS BOX –
Mount d C:\TC.
D:
cd BIN
TC.exe

In options → Directories
D: INCLUDE
D: LIB

Date :
Page No.

For slope > 1 PC-11
reverse the role of x & y i.e. we sample at dy = 1 & calculate consecutive x values as

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

#include <graphics.h>
int gdriver = DETECT; // for auto detection
int gmode = DETECT;

int graph;
initgraph(&gdriver, &gmode, " ");

To initialise graphics mode & drivers

putpixel (x, y, RED)

Colours that pixel red.

Exp-1

Aim: WAP to draw a line using DDA algorithm

Date :
Page No.

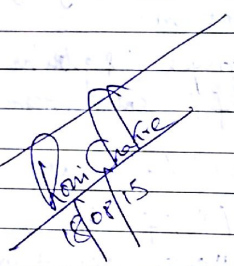
CG Lab

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

int main()
{
    int gdriver = DETECT;
    int gmode = DETECT;
    float x1, x2, y1, y2, x, y;
    float m;
    initgraph (&gdriver, &gmode, " ");
    printf (" Enter start & end pt in x axis\n");
    scanf ("%f", &x1);
    scanf (" %f", &x2);
    printf ("Enter start & end pt. in y axis\n");
    scanf ("%f", &y1);
    scanf ("%f", &y2);
    x = x1;
    y = y1;
    m = (y2-y1)/(x2-x1);
    if (m<1)
    {
        while (x != x2);
        {
            y = y+m;
            putpixel ((int)x, (int)y, RED);
            x++;
        }
    }
```

P.T.O

Date :
Page No.

```
else
{
    while(y != y2)
    {
        x = x + (1/m);
        putpixel (int)x, (int)y, WHITE);
        y++;
    }
}
getch ( );
return 0;
}
```



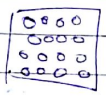
Date :
Page No.

CG Class

bitmap

If 1 pixel – 1 bit

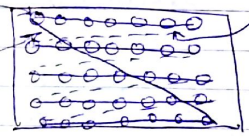
If > 1 bit → pixmap



Raster System

Fluorescence

Vertical retrace :



fluorescence travelling from last to first, horizontal retrace

If vertical/hori retrace large, flickering. To avoid, interlacing, as explained before

Refreshing Rate → Time taken to come to start point.

Random System - Not travelling whole of screen, rather only part it has to display, its time depends on complexity of figure.

Problem → depends on screen too.

Raster does not.

P.T.O

Date :
Page No.

Random Scan Display.

When operated as a Random Scan Disp. a CRT has an electron beam directed to only to the parts of screen where a picture is to be drawn. Random scan monitors draw a picture one at time & for this reason are also refered to as Vector displays.

Refresh rate on a Rand. Scan system depends on the no. of lines to be displayed. In a Rand. Scan Sys, picture definition is stored as set of line drawing commands in an area of memory referred to as Refresh Display file.

To display a specified picture, the system cycle through the set of commands in the display file drawing each component line in intrun. After all line drawing commands have been processed, the system cycles back to the first line command

Date :
Page No.

Raster Vs Random System.

Random Scan Sys are designed for line drawing applications & cannot display realistic shaded scenes. Also vector display produce smooth line design because CRT beam directly follow the line path.

The raster sys in contrast produces jagged lines that are plotted as discrete line set.

Uses of comp Graphics

1) User Interface - Point & Click facilities to allow users to select new items, icons, objects on the screen.

2) Interactive Plotting in business, science & Technology -
   Eg → Histogram
   → Inventory & Projection Chart
   → Bar Diagram.
   → 2D & 3D Graphs.

3) Office Automation & Electronic Publishing

4) Computer aided Drafting & Design.
   To design components of mech, elec. &

Date :
Page No.

electronic devices, eg VLSI chips, building & automobiles.

5) Simulation & Animation for scientific visualisation & entertainment eg. Animated movies.

6) Art & Commerce - Stock market, museums, hotels.

7) Process Control - Eg. flight controller at airport, military commanders view field data eg. weapons launched, casualities etc., air control.

8) Cartography - Weather, popu. density maps etc.

Date :
Page No.

Unit - 2

Basic Raster Graphics Algorithm

Drawing 2D Primitives

- Point
- Line
- Circle
- Ellipse

Algos for Line Drawing

1) Symmetric DDA (Digital differential Analyzers eg.) Algo
2) Simple DDA Algo
3) Bresenham's Algo (or Brick pattern algo)

Properties of a Good Line Algorithm

1) Line should be straight
2) Uniform density
3) Hardware Implementable
4) Termination Property (i.e. terminate at required pt.)

Scan Conversion

The process of representing continuous graphic object (for eg pts., line, circle, Ellipse) as collection of discrete

In Bresenham's line, line comes zig-zag most of the time, pixel is called ~~closest~~ Scan conversion.

→ Symmetric DDA

1) Let (x_1, y_1) & (x_2, y_2) be the 2 given end pts.. ~~in the algo.~~

2) Calculate, $\Delta x = x_2 - x_1$,
$\Delta y = y_2 - y_1$

3) $2^{n-1} \leq \max(|\Delta x|, |\Delta y|) < 2^n$
Find n from here

4) $\epsilon = 2^{-n}$

5) $x_{inc} = \epsilon \in \Delta x$.
$y_{inc} = \epsilon \Delta y$.

6) $x = x + x_{inc}$
$y = y + y_{inc}$.

(8,5)
(1,1)

$x' = x + 0.5$
$y' = y + 0.5$ rounded off.

Eg.

	x'	y'	$x = \text{trunc}(x')$	$y = \text{trunc}(y')$
(1, 1) →	1.5	1.5	1	1

Then $x + x_{inc}$.
$\Delta x = 7$, $\Delta y = 4$.
$\therefore 2^{3-1} \leq \max(7, 4) \leq 2^3$
$\therefore n = 3$. $\therefore \epsilon = \frac{1}{8}$.

~~Method~~ Method to round off. → $x = x + 0.5$
& then truncate x
i.e. round (x) = truncate $(x + 0.5)$

$\therefore x_{inc} = \frac{7}{8}$, $y_{inc} = \frac{4}{8}$.

x'	y'	x	y
1	1	1	1
$1\frac{7}{8}$	$1\frac{4}{8}$	2	2
$2\frac{6}{8}$	2	3	2
$3\frac{5}{8}$	$2\frac{1}{2}$	4	3
$4\frac{1}{2}$	3	5	3
$5\frac{3}{8}$	$3\frac{1}{2}$	5	4
$6\frac{2}{8}$	4	6	4
$7\frac{1}{8}$	$4\frac{1}{2}$	7	5
8	5	8	5 (same as end pt.)

Follow truncation logical
With truncating formula, Add as once only
& then a x_{inc}, y_{inc} & truncate

∴ Termination pt. satisfied

Here, drawback is that one may get repeated points., eg. for ~~for~~ Eg. (5,3) may come twice for some ratio.

Better Algo.

Simple DDA

1) (x_1, y_1) & (x_2, y_2) are 2 end pts.

$\Delta x = (x_2 - x_1)$
$\Delta y = y_2 - y_1$

Page No.

2) $l = \max(|\Delta x|, |\Delta y|)$

3) $x_{inc} = \Delta x / l$
$y_{inc} = \Delta y / l$ Here one of them becomes 1.

∴ No. of updated pt = 5.

4) $x = x + x_{inc}$.
$y = y + y_{inc}$.

(5,5)
(1,1)

Implementing

$\Delta x = 7$; $\Delta y = 4$.

$l = 7$.

∴ $\Delta x = 1$
$\Delta y = 4/7$.

x'	y'	x^s	y^s
1.5	1.5	1	1
2.5	2.07	2	2
3.5	2.64	3	2
4.5	3.21	4	3
5.5	3.78	5	3
6.5	4.35	6	4
7.5	4.92	7	4
8.5	5.49	8	5

Date :
Page No.

V. Imp

Bresenham's Algorithm for line

an assumption for this method

1) Compute the initial values $0 < m < 1$, $0 < \theta < 45°$

$\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$. (x_1, y_1) & (x_2, y_2) are end pts of line segment.

Increment $inc1 = 2 * \Delta y$,
$inc2 = 2 * (\Delta y - \Delta x)$

Decision value → $d = inc1 - \Delta x = 2\Delta y - \Delta x$.

2) Set (x, y) equal to the lower left-hand end pt & x_{end} is equal to the largest value of x.

3) Plot a point at the current (x, y) coordinates.

4) Test to see whether the entire line has been drawn. If $x \geq x_{end}$, stop. (x_{end} pt. of a coordinate ~~-ax~~.

5) Compute the location of next pixel
If $d \leq 0$, then $d = d + inc1$
If $d > 0$, then $d = d + inc2$ & $y = y + 1$.

6) Increment x : $x = x + 1$.

7) Draw a point at the current (x, y) coordinates.

8) Go to step 4.

$\Delta x = 7, \quad \Delta y = 4$

$Inc\ 1 = 8$

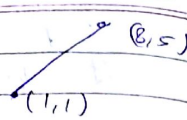
$Inc\ 2 = -6$

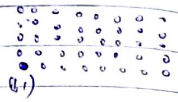
$d = 1$

Date :
Page No.

Eg.	d	x	y
	1	1	1
d+Inc 2 = 1-6 = -5	2	2	2
-5+8	3	3	2
	-3	4	3
	5	5	3
	-1	6	4
	7	7	4
	2	8	5

$x_{end} = 8$

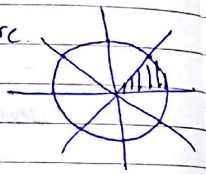




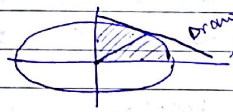
In paper, show calculation

Circle drawing Algo

Circle,
8 way symmetrical figure



Ellipse is 4 way symmetrical figure.

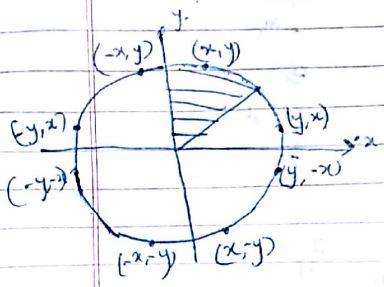


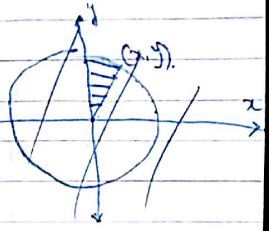
⇒ Question on these algos always come in mid & end sem.

Value - Pixel colour.

Date :
Page No.

Scan Converting Circles





For a circle centred at origin, the eight symmetrical pts can be displayed with procedure circle Points

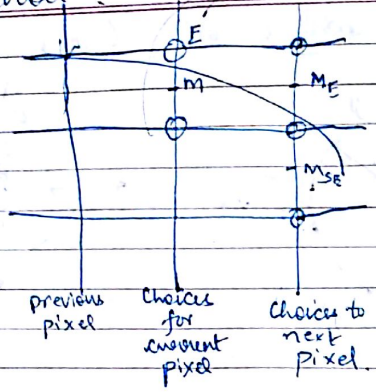
show location — value at that pt.

```
void Circle Points (int x, int y, int value)
{
    write pixel (x, y, value)
    write Pixel (y, x, value)
    write Pixel (y, -x, value)
    .
    .
    .
    8 pts.
}
```

We don't want to call circle pts. at $y = x$ as it will call same pt. 2 times ∴ Non-uniform P..

Derivation of mid pts method

Date :
Page No.



Let $F(x,y) = x^2 + y^2 - R^2$

If $F(x,y) = 0$ on circumference
If $F(x,y) < 0$ inside
$F(x,y) > 0$ outside.

If mid pt blw pixel East(E) & South East(SE) is outside the circle then pixel SE is closer to the circle.

On other hand, if mid pt is blw E & SE is inside, then E is closer to circle.

$d_{old} = F(x_p + 1, y_p - 1/2)$

$= (x_p + 1)^2 + (y_p - 1/2)^2 - R^2 - (1)$

If $d_{old} < 0$

Date :
Page No.

If $d_{old} < 0$ & the next mid pt is incremented & E is chosen over x,

$d_{new} = F(x_p + 2, y_p - 1/2)$

$= (x_p + 2)^2 + (y_p - 1/2)^2 - R^2 - (2)$

From (1) & (2)

1 → $x_p^2 + 1 + 2x_p + y_p^2 + 1/4 - y_p - R^2 = d_{old}$

2 → $x_p^2 + 4 + 4x_p + y_p^2 + 1/4 - y_p - R^2 = d_{new}$

Subtract,

$d_{new} - d_{old} = 3 + 2x_p$

$\Rightarrow \boxed{d_{new} = d_{old} + 2x_p + 3.}$

If $d_{old} \geq 0$.

$F(x_p + 2, y_p - 3/2)$

$= (x_p + 2)^2 + (y_p - 3/2)^2 - R^2$.

If $d_{old} \geq 0$. & SE is chosen & next mid pt is chosen as an increment in x & 1 decrement in y.

$F(x_p + 2, y_p - 3/2) = (x_p + 2)^2 + (y_p - 3/2)^2 - R^2$

Date :
Page No.

$\therefore x_p^2 + 4 + 4x_p + y_p^2 + y_p - 3y_p - R^2 + d_{old}$

$\therefore d_{new} = x_p^2 + 6x_p + 2 - 2y_p + d_{old}$

$\therefore \boxed{d_{new} = d_{old} + (2x_p - 2y_p + 5)}$ → SE

Initial condition.

(0, R)

$(1, R-\frac{1}{2})$

$F(1, R-\frac{1}{2}) = 1 + (R-\frac{1}{2})^2 - R^2$

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

$\boxed{d_{start} = \frac{5}{4} - R}$

Now decide d_{old}'s value.

MID PT CIRCLE ALGO.

Function ⇒

```
void midpoint circle (int radius, int value)
// centre (0,0)
{
  int x=0, y= radius;
  double d = 5.0/4.0 - radius;
  circle Point (x, y, value);
  while (y>x) {
  if (d<0)   // select E.
    d+= 2.0 *x + 3.0;
```

in exam, radius will be given, find 4-5 pts by using algo & mark these corresponding 7 other pts.

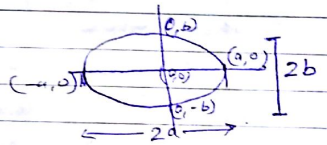
Date :
Page No.

```
  else
  { d+= 2.0 * (x-y) + 5.0.
    y--;
  }
  x++;
  Circle Point (x,y, value);
  }
}
```

3/10/24

Scan converting Ellipse

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



Mid pt ellipse Algo.

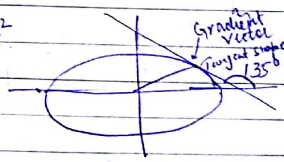
Da, Silvia's Algo (Also known as)

4 way symmetric figure

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

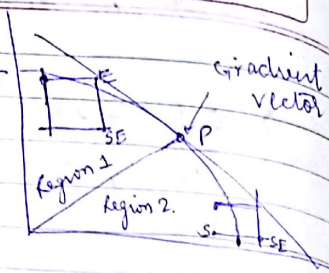
$2a$ – Major axis

$2b$ – Minor axis



Page No.

The vector ⊥ to tangent to curve at pt. p is called gradient vector.

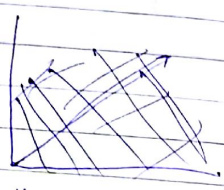


$$\text{grad } F(x,y) = \frac{\partial F}{\partial x} i + \frac{\partial F}{\partial y} j$$

$$= 2b^2 x\, i + 2a^2 y\, j \quad -(1)$$

is j component

The boundary b/w 2 regions is pt. at which slope of the curve is -1.



If that circle pt occurs when a gradient vector has a slope 1 i.e when i & j components of the gradient are of equal magnitudes. The j com

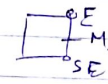
The j component of the gradient is larger than i component in region 1 & vice versa in region 2. Thus if at the next mid pt. $a^2(y_p - 1/2) < b^2(x_p + 1)$ (From (1)).

We switch from region 1 to region

Date :
Page No.

2.

If the current pixel is located at (x_p, y_p), then decision variable for region 1 is $F(x,y)$ located at $(x_p+1, y_p - 1/2)$, the mid pt b/w East & South East



For a move to E, the next mid pt. is 1 increment in x. Then for d_1,

$d_1 < 0$

$$d_{old} = F(x_p+1, y_p - 1/2)$$

$$= b^2(x_p+1)^2 + a^2(y_p - 1/2)^2 - a^2b^2 \quad (2)$$

$$d_{new} = F(x_p+2, y_p - 1/2)$$

$$= b^2(x_p+2)^2 + a^2(y_p - 1/2)^2 - a^2b^2 \quad (3)$$

From (2) & (3) we get

$$d_{new} = d_{old} + \underbrace{b^2(2x_p+3)}_{\Delta E} \quad -(4)$$

For a move to SE, the next mid point is 1 increment over in x & 1 increment down in y.

Then for $d_2 \geq 0$

$$d_{old} = b^2(x_p+1)^2 + a^2(y_p - 1/2)^2 - a^2b^2.$$

Q1) Beginning
Q2) Scan converting line, circle, ellipse
Q3) Transformation Mid Sem
Q4) clipping Till Mid Clipping
Q5) Mixed

Date :
Page No.

$$d_{new} = F(x_p + 2, y_p - \tfrac{3}{2})$$

$$= b^2 (x_p + 2)^2 + a^2 (y_p - \tfrac{3}{2})^2 - a^2 b^2 \quad \ldots (3)$$

$\therefore$ By (2) & (4).

$$d_{new} = d_{old} + \underbrace{b^2(2x_p + 3) + a^2(-2y_p + 2)}_{\Delta SE} \quad (5)$$

For region 2, the current pixel is at (x_p, y_p), the decision variable d_2 is $F(x_p + \tfrac{1}{2}, y_p - 1)$ (the mid pt. b/w S & SE)

For a move to SE, the next mid pt. is 1 increment down in y & one 1 increment over in x, then for

For $d_2 < 0$

$$d_{old} = F(x_p + \tfrac{1}{2}, y_p - 1)$$

$$= b^2 (x_p + \tfrac{1}{2})^2 + a^2 (y_p - 1) - a^2 b^2 \quad (6)$$

$$d_{new} = F(x_p + \tfrac{3}{2}, y_p - 2)$$

$$= b^2 (x_p + \tfrac{3}{2})^2 + a^2 (y_p - 2)^2 - a^2 b^2 \quad (7)$$

From (6) & (7)

$$\Rightarrow \boxed{d_{new} = d_{old} + \underbrace{b^2(2x_p + 2) + a^2(-2y_p + 3)}_{\Delta SE}} \quad (8)$$

For a move to S, the next mid point is 1 increment down in y. then

Graphic I/O devices.

Date :
Page No.

For $d_2 \geq 0$

$$d_{new} = F(x_p + \tfrac{1}{2}, y_p - 2)$$

$$= b^2 (x_p + \tfrac{1}{2})^2 + a^2 (y_p - 2)^2 - a^2 b^2$$

$$\therefore \boxed{d_{new} = d_{old} + a^2(-2y_p + 3)} \quad (9)$$

ΔS

Initial Conditions

Assuming integer values of a & b, ellipse start at $(0, b)$ & first mid pt. to be calculated is $(1, b - \tfrac{1}{2})$, then

Function values at $(1, b - \tfrac{1}{2})$:

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

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

$$\Rightarrow \boxed{d_1 = b^2 - a^2 b + 0.25 a^2} \quad // d \text{ start}$$

At every iteration in region 1, we must not only test decision variable at d_1 & update the delta function, but also see whether we should switch regions by evaluating the gradient at the mid pt b/w E & SE.

When mid pt crosses over into region 2, we change our choice of 2 pixels to compare from E & SE to SE & S.

At the same time, we have to initialise decision variable d_2 for region 2 for mid pt b/w SE & S, i.e. if the last pixel chosen

If in paper, it comes to describe, then desire & show.

Date :
Page No.

in region 1 is located at (x_p, y_p) then decision variable d_2 is initialised at $(x_p + \frac{1}{2}, y_p - 1)$. Thus initial value of d_2 is $F(x_p + \frac{1}{2}, y_p - 1)$

$$d_2 = b^2 (x_p + \tfrac{1}{2})^2 + a^2 (y_p - 1)^2 - a^2 b^2$$

We stop drawing pixel in region 2 when y value of pixel is equal to 0.

→ Implementing the algorithm in lab (for computer)

```
void midpointEllipse(int a, int b, int value)
// Assume centre of ellipse is at origin
{
    double d2;
    int x = 0;
    int y = b;
    double d1 = b^2 - a^2 b + (0.25) a^2 ;
    EllipsePoints (x, y, value); // 4 way symmetrical
                                 // write pixel

/* Test gradient if still in region 1 */

while ( a^2 (y - 0.5) > b^2 (x + 1) )   // region 1
{
    if (d1 < 0)                  // select E
    d1 += b^2 (2x + 3);
```

Date :
Page No.

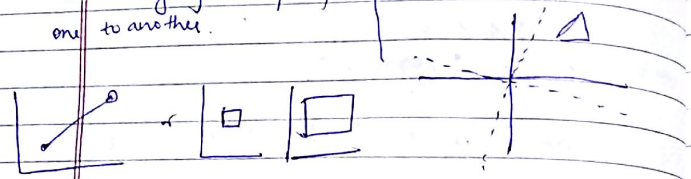
```
    else
    {
        d1 += b^2 (2x + 3) + a^2 (-2y + 2);
        y--;
    }
    x++;
    EllipsePoints (x, y, value);
}   /* Region 1 */

d2 = b^2 (x + 0.5)^2 + a^2 (y - 1)^2 - a^2 b^2 ;

                                  Region 2
while (y > 0)                 // ~~Select SE~~
{
    if (d2 < 0)               // select SE
    {
        d2 += b^2 (2x + 2) + a^2 (-2y + 3);
        x++;
    }
    else
    { d2 += a^2 (-2y + 3);    // Select S
    }
    y--;
    EllipsePoints (x, y, value);
}   /* Region 2 */
}
```

Date :
Page No.

2D Transformations

Geometric
Moving figure from 1 place to other place or changing shape from one to another.

Coordinate
To keep the figure stationary & move the coordinate system



Basic Transformations :-

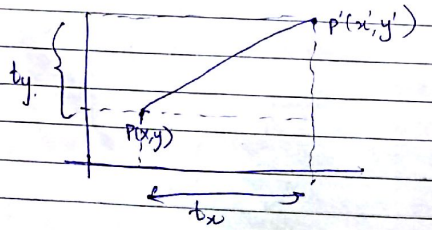
1) Translation
2) Rotation
3) Scaling

Other transformations

4) Mirror reflection
5) Reverse transformation

] can be obtained from above 3.

Translation



$x' = x + tx$

$y' = y + ty$

Matrix form

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

Remember this matrix

$P' = T_{tx,ty} \cdot P$

↳ Translation Matrix

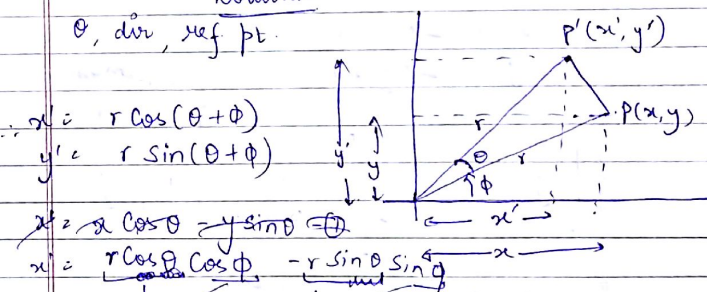
Matrix is easy to implement in computer based implementation. We represent coordinate pair (x,y) of a point by tuple $(x,y,1)$

Work in progress ..... Sorry !!

This is simply homogeneous representation of the pt P.

Rotation

θ, dir, ref pt.



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

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

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

$x' = x\cos\theta - y\sin\theta$ — (1)

$y' = x\sin\theta + y\cos\theta$ — (2).

Marketing office:

Date :
Page No.

Remember

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

If instead of column vector, we use row vector, sign of sin changes

$P' = (R_\theta) P$ → Rotation Matrix

Scaling -

S_x – Scaling x
S_y – Scaling y

$x' = S_x \cdot x \quad -①$
$y' = S_y \cdot y \quad -②$

Eg. $(S_x, S_y) = (2, 1/2)$, $P(2,2)$

$x' = 2 \times 2 = 4$
$y' = 2 \times \frac{1}{2} = 1$

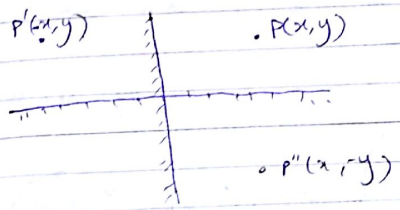
Remember

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

$P' = (S_{S_x, S_y}) \cdot P$

Date :
Page No.

Mirror Reflection



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

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

Inverse Transformation

$T^{-1}_{t_x, t_y} = T_{-t_x, -t_y}$

$R_\theta^{-1} = R_{-\theta}$

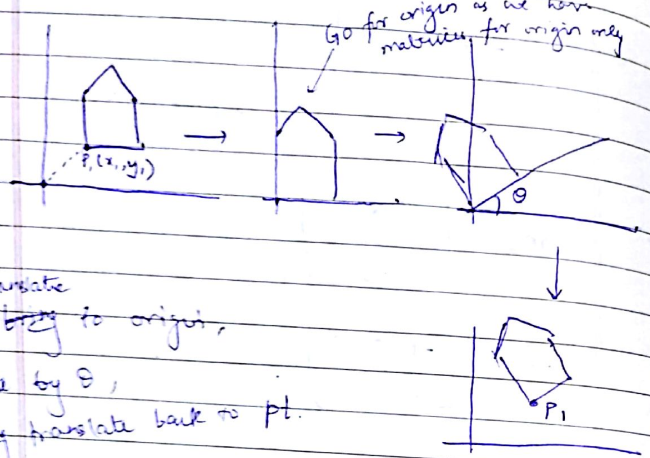
$S^{-1}_{S_x, S_y} = S_{\frac{1}{S_x}, \frac{1}{S_y}}$

$m_x^{-1} = m_x$

$m_y^{-1} = m_y$.

Composition of 2D Transformation

Eg - Rotation of an object ~~around~~ about 1 arbitrary pt:



We ~~trans~~ translate to origin,

rotate by θ,

~~trans~~ translate back to pt.

Req. Transformation = $T(x_1, y_1) + R_\theta \to T(-x_1, -y_1)$.

i.e. $T(x_1,y_1)\, R_\theta\, T(-x_1,-y_1)$

(translate back) (rot) (go origin; left to right)

Req. Trans = $\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

~~$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin & & \end{bmatrix}$~~

$t_x = x_1$ as we are shifting this much.

$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_1 \\ \sin\theta & \cos\theta & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x \\ 1 & -y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$= \begin{bmatrix} \cos\theta(x-x_1) - \sin\theta(y-y_1) + x_1 \\ \sin\theta(x-x_1) + \cos\theta(y-y_1) + y_1 \\ 1 \end{bmatrix}$

$= \begin{bmatrix} \cos\theta & -\sin\theta & -x_1\cos\theta + y_1\sin\theta + x_1 \\ \sin\theta & \cos\theta & -x_1\sin\theta - y_1\cos\theta + y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Q) To scale an object about an arbitrary pt. pt P_1.

Ans. Req. trans. = $T(x_1,y_1)\, S_{s_x,s_y}\, T(-x_1,-y_1)$.

$\therefore \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$= \begin{bmatrix} S_x & 0 & x_1 \\ 0 & S_y & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$

$= \begin{bmatrix} S_x & 0 & -x_1 S_x + x_1 \\ 0 & S_y & -y_1 S_y + y_1 \\ 0 & 0 & 1 \end{bmatrix}$

P.T.O

Date :
Page No.

Q3) To scale, rotate & position the trans with E_1 as the centre for rotation & scaling.

Ans. Req. Trans = $T(x_2, y_2) \cdot S_{s_x, s_y} \cdot R_\theta \cdot T(-x_1, -y_1)$

$$= \begin{bmatrix} 1 & 0 & x_2 \\ 0 & 1 & y_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_2 \\ \sin\theta & \cos\theta & y_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & -x_1 s_x \\ 0 & s_y & -y_1 s_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x\cos\theta & -s_y\sin\theta & -x_1 s_x\cos\theta + y_1 s_y \sin\theta + x_2 \\ s_x\sin\theta & s_y\cos\theta & -x_1 s_x \sin\theta - y_1 s_y \cos\theta + y_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Asked in exam

Q4) Perform a 60° rotation of a triangle A(0,0), B(1,1), C(5,2)

a) About origin
b) About (-1, -1)

A4)

a) Req. trans. = R_θ.

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pts. $\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$

$\therefore$ Req pt. $\begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$

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

$A \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \therefore A = (0,0)$

$B = \left(\frac{1-\sqrt{3}}{2}, \frac{1+\sqrt{3}}{2}\right)$

$C = \left(\frac{5-2\sqrt{3}}{2}, \frac{5\sqrt{3}+2}{2}\right)$

b) Req. Trans = $T(-1,-1) \cdot R_{60^\circ} \cdot T(1,1)$ (shifting (-1,-1) to (0,0))

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

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

$$= \begin{bmatrix} \cos\theta & -\sin\theta & \cos\theta - \sin\theta - 1 \\ \sin\theta & \cos\theta & \sin\theta + \cos\theta - 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta - \sin\theta - 1 & 2\cos\theta - 2\sin\theta - 1 & 6\cos\theta - 3\sin\theta - 1 \\ \sin\theta + \cos\theta - 1 & 2\sin\theta + 2\cos\theta - 1 & 6\sin\theta + 3\cos\theta - 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Page No.

$$= \begin{bmatrix} \frac{-\sqrt{3}-1}{2} & -\sqrt{3} & 2-\frac{3\sqrt{3}}{2} \\ \frac{\sqrt{3}+1}{2} & \sqrt{3} & 3\sqrt{3}+\frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore x = \left(\frac{-\sqrt{3}-1}{2}, \frac{\sqrt{3}+1}{2}\right)$

$y = (-\sqrt{3}, \sqrt{3})$

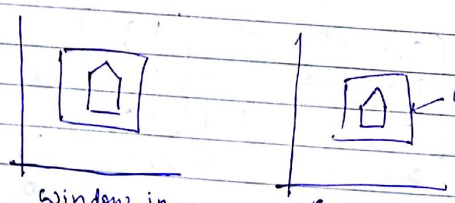
$z = \left(2-\frac{3\sqrt{3}}{2}, 3\sqrt{3}+\frac{1}{2}\right)$.

Type 2 / 22

Window to Viewport Transformation

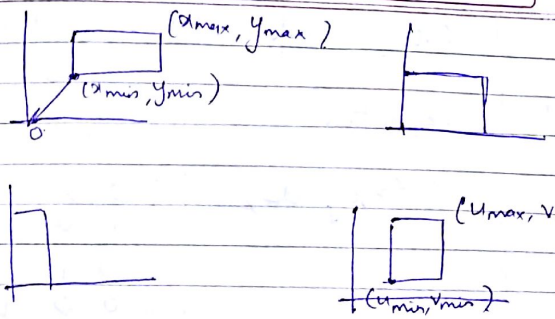
Window: Fixed area in world coordinate system

Viewport: Fixed area on a screen.



Given a window & viewport, What is the transformation matrix that maps the world coordinate system to screen coordinate system.

Date :
Page No.



$M_{WV} =$

$S_x = \frac{U_{max} - U_{min}}{x_{max} - x_{min}}$

$S_y = \frac{V_{max} - V_{min}}{y_{max} - y_{min}}$

$M_{WV} = T(U_{min}, V_{min})\ S_{Sx,Sy}\ T(-x_{min}, -y_{min})$

on paper, if asked where S_x S_y.

P.T.O

Page No.

Matrix Representation of 3-D Transformation

Translation:

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling:

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation:

Rotation Around z → $R_z(\theta)$ = Around z →

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

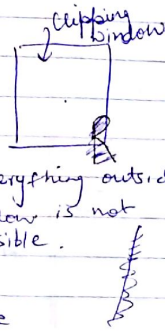
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Date :
Page No.

CLIPPING

* Point Clipping
* Line "
* Area " (Polygons)
* Curve "
* Text "

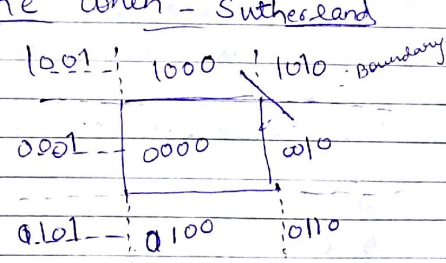


→ Point – Either outside or inside

→ Line –
1) Both ends inside
2) One end outside
3) Outside outside

The Cohen – Sutherland

Step-1



Generating code.
each bit → left most one 1.
second last → right most one
3 → bottom.

(x_1, y_1) (x_2, y_2)

Page No.

Step 2 P1 & P2 are two end points of the line segment, C1 & C2 are corresponding codes

I) If C1 = 0000, C2 = 0000
→ Line is completely visible.

II) If C1 bitwise AND C2 ≠ 0
→ Line completely invisible.

III) If C1 AND C2 = 0000
→ Line is partially visible.
Candidate for clipping

Q) Let R be a rectangular window whose lower left hand corner is at L(−3, 1) & upper right hand corner is at R(2, 6). Codes

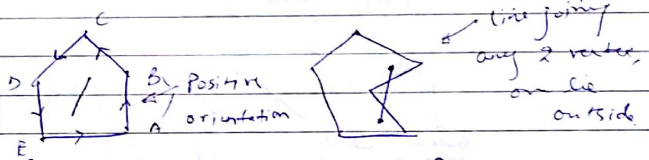
i) Find end pt. for pts. given below.

A (−4, 2)
B (−1, 7)
C (−1, 5)
D (3, 8)
E (−2, 3)
F (1, 2)
G (1, −2)
H (3, 5)
I (−4, 7)
J (−2, 10)

ii) Find Clipping Categories for line segments
AB, CD, EF, GH, IJ
Candidate for clipping ✓ ✗ clip ✓ NV

iii) Clip the line segment.

Polygon Clipping



Convex Polygon.

Non-convex Polygon.

→ A polygon is called convex if line joining any 2 interior pts lies complete inside the polygon.

→ Positive Orientation. - We travel in polygon anti-Clockwise.
-ve Clockwise.

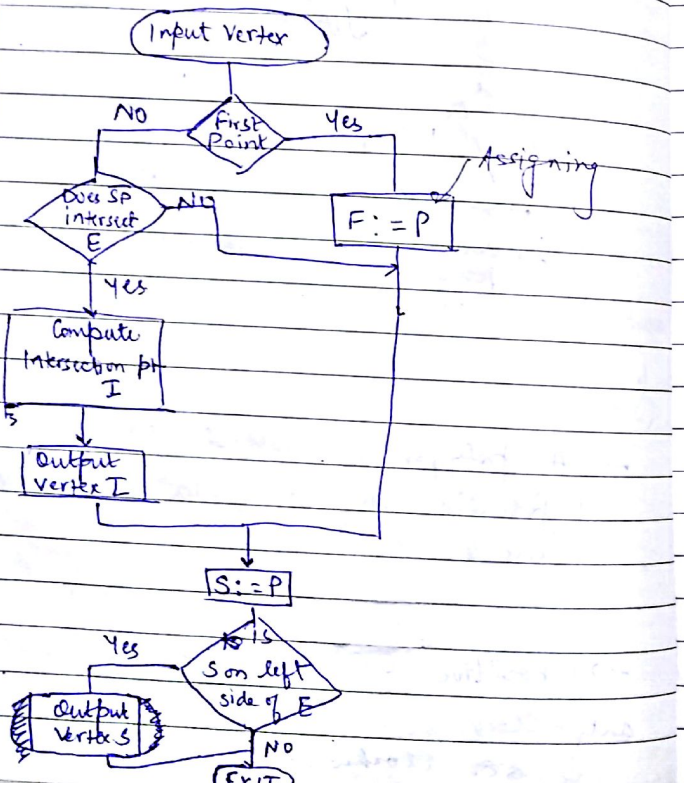
Date :
Page No.

Draw a flowchart illustrating the logic of

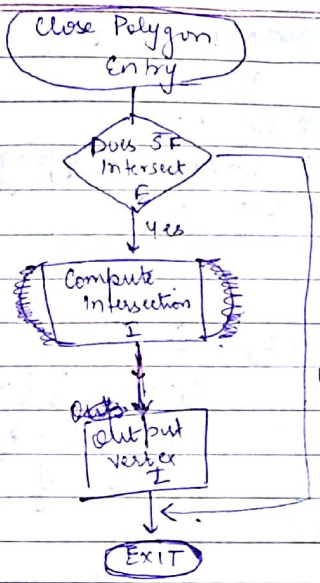
Sutherland - Hodgeman Polygon Clipping Algo

S: Previous I/P vertex
F: First Vertex
P: Input Vertex.

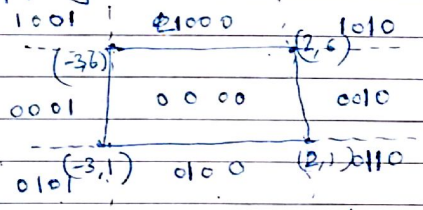
Consider a polygon with n vertices $P_1 P_2 \ldots P_n$ has edges $P_1P_2, \ldots \overline{P_{n-1}P_n}$ & $\overline{P_nP_1}$ closing the polygon.



Date :
Page No.



Ans for Clipping



A (-4,2) → 0001
B (-1,7) → 1000
C (-1,5) → 0000
D (3,8) → 1010
E (-2,3) → 0000
F (1,2) → 0000
G (1,-2) → 0100
H (3,3) → 0010
I (4,7) → 1001
J (-2,10) → 1000

Page No.

AB = Candidate for clipping
CD = "
EF = Inside frame
GH = Candidate for clipping.
IJ = Outside frame

AB → (-4, 2) – (-1, 7)

$m = \frac{5}{3}$ $\quad (y-2) = \frac{5}{3}(x+4)$

$x = -3$ $\quad \Rightarrow y = 2 + \frac{5}{3}(1)$

$= \frac{11}{3}$

$\left(-3, \frac{11}{3}\right)$

$y = 6$

$(6-2) = \frac{5}{3}(x+4)$

$\Rightarrow \frac{4}{5} \times 3 = x+4$

$\therefore x = \frac{12-20}{5} = \frac{-8}{5}$.

$\therefore \left(-\frac{8}{5}, 6\right)$

Area filling: Scan + boundary fill also in exam
Date :
Page No.
So, study.
2. Polygon filling too.

Liang – Barsky Line Clipping Algo

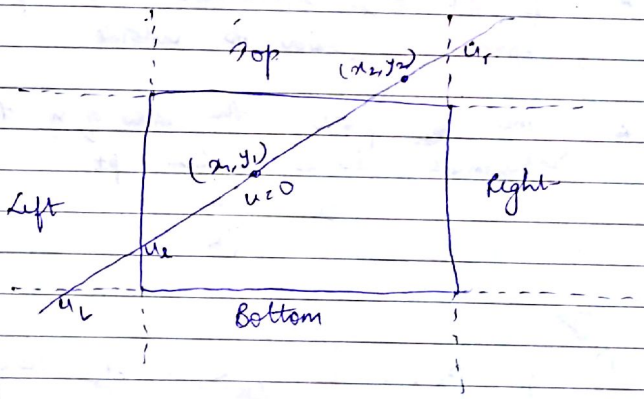
(x_1, y_1) & (x_2, y_2)

$x = x_1 + \Delta x \cdot u \quad \rightarrow \quad 0 \le u \le 1$

$y = y_1 + \Delta y \cdot u \quad \rightarrow \quad -\infty \le u < \infty$

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

$$\begin{cases} x_{min} \le x_1 + \Delta x \cdot u \le x_{max} \\ y_{min} \le y_1 + \Delta y \cdot u \le y_{max} \end{cases}$$



$p_k \cdot u \le q_k, \quad k = 1, 2, 3, 4$

where (window)

$p_1 = -\Delta x \quad q_1 = x_1 - x_{min}$ (left)

$p_2 = \Delta x \quad q_2 = x_{max} - x_1$ (right)

$p_3 = -\Delta y \quad q_3 = y_1 - y_{min}$ (bottom)

$p_4 = \Delta y \quad q_4 = y_{max} - y_1$ (top)

Date :
Page No.

We can observe the following effects:

* If $p_k = 0$, the line is parallel to the corresponding boundary.
* If $q_k < 0$, the line is completely outside the boundary & can be eliminated
* If $q_k \geq 0$, line is inside the boundary & need further consideration
* If $p_k < 0$, the extended line proceeds from the outside to the inside of corresponding boundary line
* If $p_k > 0$, extended line proceeds from inside to outside
* When $p \neq 0$ the value of u that corresponds to intersection pt.

$$u = \frac{q_k}{p_k}$$

1) If $p_k = 0$ & $q_k < 0$ for any k
Eliminate the line & stop.
Otherwise proceed to next step.

2) for all k such that $p_k < 0$, calculate

$$SO_k = so_k - dho_k = SO_k$$
or $SO_k = kha_k - p_k$

$$r_k = \frac{q_k}{p_k}$$

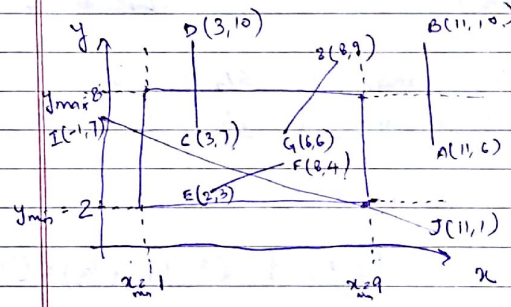
Let u_1 be the max. of set containing 0 & the calculated r values

3) For all k such that $p_k > 0$, calculate r_k

Let u_2 be min of set containing 1 & the calculated r valu.

4) If $u_1 > u_2$
Eliminate the line since it is completely outside the clipping window
Otherwise use u_1 & u_2 to calculate end pt. of clipped line

Q) Use the Liang - Barsky Algo to clip the line



Page No.

For line AB, (11,6), (11,10)

$p_1 = 0$ $q_1 = 10$

$p_2 = 0$ $q_2 = -2$

$p_3 = -4$ $q_3 = 4$

$p_4 = 4$ $q_4 = 2$

since, p_0 $p_1 = 0$, $q_2 < 0$.
∴ lie outside.

For CD, (3,7), (3,10)

$p_1 = 0$ $q_1 = 2$

$p_2 = 0$ $q_2 = 6$

$p_3 = -3$ $q_3 = 5$, $r_3 = -5/3$

$p_4 = 3$ $q_4 = 1$, $r_4 = 1/3$

$u_1 = \max(0, -5/3) = 0.$

$u_2 = \min(1, 1/3) = 1/3$

$u_1 < u_2$

On pt, (3,7) & (3, 7 + 3(1/3) = (3,8)

$x = x_1 + \Delta x \cdot u$

$y = y_1 + \Delta y \cdot u$

Date :
Page No.

For EF, (2,3), (8,4)

$p_1 = -6$ $q_1 = 1$

$p_2 = 6$ $q_2 = 7$

$p_3 = -1$ $q_3 = 1$, $r_3 = -1$

$p_4 = 1$ $q_4 = 5$ $r_4 = 5$

Page No.

Weiler - Atherton Algo

Let the clipping window be initially called the clip polygon & the polygon to be called the subject polygon.

We start with an arbitrary vertex of the subject polygon & trace around its border in the clockwise direction until an intersection with the clip polygon is encountered.

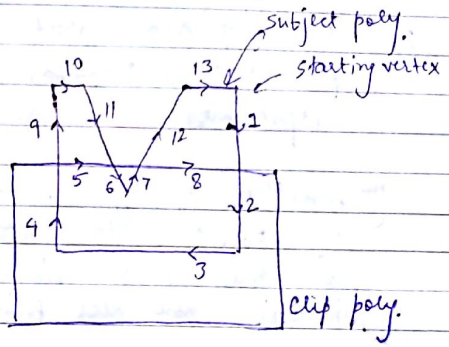
1) If the edge enter the clip polygon, we find intersecting pt & continue to trace the subject polygon.

2) If the edge leaves the polygon, record the intersection pts. & make a right turn to follow the clip polygon in the same manner (i.e. treat the clip poly. as subject polygon & subject poly. as clip poly. & proceed as before.)

Wherever our path of traversal forms a sub-polygon we output the sub poly. as part of overall result.

We then continue to trace the rest of the original subject poly. from a recorded intersec. pt. that marks the beginning of not yet traced edge or a portion of the edge.

The algo terminates when the entire border of original subject polygon has been traced exactly once.



~~The need for to represent curve~~

Date :
Page No.

Representing Curves & Surfaces

The need to represent curves & surfaces arises in 2 cases :

1) Modelling existing objects (a car, a face, a tree, a mountain)
2) Modelling "from scratch" where no pre-existing physical object is being represented.

By modelling of object, we mean description of object to the computer so as to produce a visual display that simulates the real thing.

One way to this is to use a set of primitive are geometric forms that are simple enough to be easily implemented on a computer but flexible enough to represent or model a variety of objects. We can build a model of an object by assembling these primitives either through the use of a Prepared Display File.

Geometric forms that can be used as primitives include Points, Lines, Line segments, polylines, polygon & polyhedra.

Date :
Page No.

More complex geometric forms, curved surfaces, curved segment, curved surface patches, quadratic surfaces (sphere, parabola).

Polyhedra - Polyhedra is a closed polygonal mesh that is one which encloses a definite volume in which each polygon is planar. The polygon are called the faces of polyhedra.

Polygon mesh is a collection of vertices, edges & faces that define the shape of polyhedron object in 3D computer graphics & solid modelling.

Eg. cube, aeroplane

How polygons are represented :-

1) Explicit vertex list
2) polygon Listing (or pointer to vertex list)
3) Explicit edge Listing (or Pointers to edge list)

P.T.O

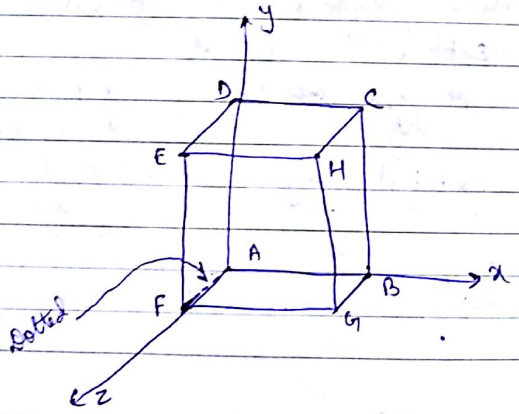
Page No.

1) Explicit Vertex list.

$V = \{P_0, P_1, P_2 \dots P_n\}$

Point → $P_i(x_i, y_i, z_i)$ or the vertices of polygonal mesh is stored in the order in which they will be encountered by traveling along the model. Although this form of representation is useful for simple polygons, it is inefficient for a complex polygonal net in which shared vertices are repeated several times in addition when displaying the order by drawing the edge shared eg. edges are drawn several time.

For. Drawing a cube using explicit Vertex list

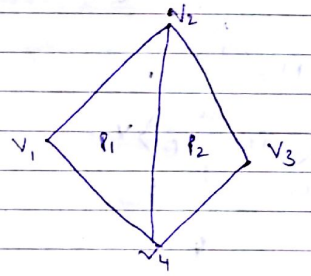


Date :
Page No.

V = { A B C D A F E D C H F G H C B }
repeated

Polygon Listing

In this form of Representation, each vertex is stored exactly once in a vertex list ($P = \{P_1, P_2, P_3 \dots\}$). Each vertex is defined by pointing into this vertex list again shared edges are drawn several time in displaying the model.



$V = \{V_1, V_2, V_3, V_4\}$
$= \{(x_1, y_1, z_1), (x_2, y_2, z_2), \dots\}$

$P_1 = (1, 2, 4)$
$P_2 = (4, 2, 3)$

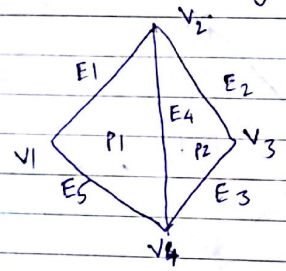
Date :
Page No.

Explicit Edge Listing

In this form of representation, we keep a vertex list in which each vertex is stored exactly once & an edge list in which each edge is stored exactly once.

Each edge in the edge list points to the two vertices in the vertex list which defines that edge. A polygon is now represented as a list of pointers into the edge list.

Additional info such as those polygons sharing a given edge can also be stored in the edge list.



$V = \{V1, V2, V3, V4\} = \{(x_1, y_1, z_1), (x_2, y_2, z_2)\}$

$E_1 = \{V_1, V_2, P1, \lambda\}$ $\lambda = \phi$

$E_2 = (V_2, V_3, P2)$ } for other polygon

$E_3 = (V_3, V_4, P2, \lambda)$

$E_4 = (V_4, V_2, P1, P2)$

-8/10/15 on the (one mark & class) weither ... start

Date :
Page No.

$E_5 = (V_4, V_1, P1, \lambda)$ ← λ represent null

$P1 = (E1, E4, E5)$

$P2 = (E2, E3, E4)$

In some situations, some edges are shared by 3 or more polygons. In such cases, the edge description can be extended to include an arbitrary no of polygons.

Curved Surfaces

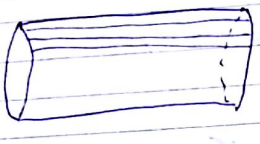
There are several approaches to model curved surface. We model an object by using small curved surface patches placed next to each other.

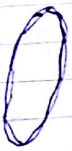
Another approach is to use surfaces that defines solid objects such as polyhedra, sphere, cylinder & cone. a model can then be constructed with these solid objects used as building blocks. This process is called solid modelling.

Polygon meshes can be used to represent object with curved surfaces.

Date :
Page No.

however the representation is only an approximate





Cross - section

Equation of a plane - $ax+by+cz+d=0$.

→ Parametric Cubic Curves

Polylines & polygons are first degree piece-wise linear approximations to curves & surfaces respectively. Unless the curve & surface being approximated are also piece wise linear, large no. of end pt. coordinates must be created & stored to achieve reasonable accuracy. The general ____ uses functions of more than ~~two~~ one degree. However the function is still, only approximate the desired shape but use less storage. The higher degree approximations can be based on one of the following 3 methods :

1) Explicit function of x :-

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

2) Implicit function

$$f(x,y,z) = 0$$

eg. $x^2+y^2=r^2$ models a circle but how do we model half a circle.

These two mathematical forms represent many difficulties to represent desired shape.

3) Parametric representation to curve

$$x = x(t) \quad y = y(t) \quad z = z(t)$$

→ segment

Here, a curve is approximated by piecewise polynomial curve instead of piecewise linear curve.

Each segment Q of overall curve is given by 3 functions of x, y, z which are cubic polynomials in parameter t :

Page No.

Cubic polynomials are most often used because lower polynomials give too little flexibility in controlling the shape of curve.

The cubic polynomial that define a curve segment $Q(t) = \{x(t)\ y(t)\ z(t)\}$ are of the form

$$x(t) = ax t^3 + bx t^2 + cx t + dx$$
$$y(t) = ay t^3 + by t^2 + cy t + dy$$
$$z(t) = az t^3 + bz t^2 + cz t + dz$$

$0 \le t \le 1$

To deal with finite segments of the curve without loss of generality we stick the parameter T from 0 to 1 interval with

$$T = \{t^3\ t^2\ t\ 1\}$$

$$C = \begin{bmatrix} ax & ay & az \\ bx & by & bz \\ cx & cy & cz \\ dx & dy & dz \end{bmatrix}$$

$$Q(t) = T \cdot C$$

The derivative of $Q(t)$ is a parametric tangent vector of the curve.

Date :
Page No.

We have,

$$\frac{dQ(t)}{dt} = Q'(t)$$

$$\Rightarrow \left[\frac{d}{dt}x(t) \quad \frac{d}{dt}y(t) \quad \frac{d}{dt}z(t)\right]$$

$$= \frac{(dT)}{dt} C = [3t^2\ 2t\ 1\ 0] \cdot C$$

$$= 3ax t^2 + 2bx + cx$$

$$=$$

Constraints on end pts, tangent vector & continuity b/w curve segment, each cubic polynomial of equation one has 4 constraints, so 4 constraints will be needed, allowing us to formulate 4 equation & then solving for 4 unknowns.

1) Hermite Curves

Defined by 2 end pts. & 2 end pt. tangent vectors.

2) Bezier Curves

Defined by 2 end pts. & 2 other pts. (not on curve) but control end. pt. tangent vector.

Date :
Page No.

3) Several kinds of splines :-
Each defined by 4 control pts.

→ To see how the coefficients of eq^n can depend on 4 constraints we know that a parametric cubic is represented as

$$Q(t) = T \cdot C$$

Let $C = m \cdot G$

where m is a 4×4 basis matrix
G is 4-element column vector of geometry constraints called geometric vector.

The geometric constraints are conditions such as end pts. or tangent vectors that define the curve.

Let G(x) be the column vector of just x components of geometry vector

Similarly, G_y & G_z.

m or G or both m & G differ for each type of curve.

Date :
Page No.

The elements of m & G are constant so the product

$Q = T \cdot C = T \cdot m \cdot G$ is just three cubic polynomials.

$$Q(t) = \{ x(t) \quad y(t) \quad z(t) \} =$$

$$[t^3 \; t^2 \; t \; 1] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$x(t) = T \cdot m \cdot G(x)$ $\quad G_x$ for x, G_y for y.

$$\begin{bmatrix} g_1^x \\ g_2^x \\ g_3^x \\ g_4^x \end{bmatrix}$$

$x(t) = (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_1^x$

$+ (\quad) g_2^x + (\quad) g_3^x + (\quad) g_4^x$ (3)

Eq. (3) emphasizes that the curve is weighted sum of elements of geometric matrix. The weights are each cubic polynomial of t called the blending functions. The blending func. b is given by

$B = T \cdot m$

Page No.

Hermite Curves

The Hermite form of the cubic poly. curve segment is determined by constraints on the end pts P_1 & P_4 & tangent vector at the end pt. R_1 & R_4.

To find the Hermite Basis Matrix (M_H) which relates the Hermite Geometry (G_H) to the polynomial coefficients, we write 4 eqns, 1 for each of the constraints on the 4 unknown poly. coefficients & then solve for the unknowns.

$$G_{Hx} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x \quad -(4)$$

$$x(t) = ax t^3 + bx t^2 + cx t + dx$$

$$T \cdot C_x = T \cdot M_H \cdot G_{Hx}$$

$$= [t^3 \; t^2 \; t \; 1] \, M_H \cdot G_{Hx} \quad -(5)$$

Substituting $x(0) = [0 \; 0 \; 0 \; 1] \cdot M_H \cdot G_{Hx}$ -(6)

$$x(1) = [1 \; 1 \; 1 \; 1] \, M_H \cdot G_{Hx} \quad -(7)$$

$$x'(t) = [3t^2 \; 2t \; 1 \; 0] \, M_H \cdot G_{Hx}$$

Date :
Page No.

$$x'(0) = [0 \; 0 \; 1 \; 0] \, M_H \, G_{Hx} \quad -(8)$$

$$x'(1) = [3 \; 2 \; 1 \; 0] \cdot M_H \cdot G_{Hx} \quad -(9)$$

Writing (6), (7), (8) & (9) in matrix form.

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = G_{Hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{Hx} \quad -(10)$$

$$\Rightarrow M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1}$$

$$\downarrow$$

$$A^{-1} = \frac{1}{|A|} \, \text{adj}(A) = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad -(11)$$

adj: $[\text{cofactors}]^T$

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

M_H which is unique, can now be used in

$$x(t) = T \cdot M_H \cdot G_{Hx}$$

To find $x(t)$ based on geometry vector G_{Hx}.

$$y(t) = T \cdot M_H \cdot G_{Hy}$$

Page No.

$Z(t) = T \cdot m_H \cdot G_{H_2}$

That means $Q(t) = \underbrace{T \cdot m_H}_{B_{H}} \cdot G_H$ — (12)

B_H = Hermite Blending func.

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$

$Q(t) =$

$T \cdot m_H = B_H$ i.e. Blending Function.

$$\therefore Q(t) = (2t^3 - 3t^2 + 1) P_1 + (-2t^3 + 3t^2) P_4 + (t^3 - 2t^2 + t) R_1 + (t^3 - t^2) R_4 \quad (13)$$

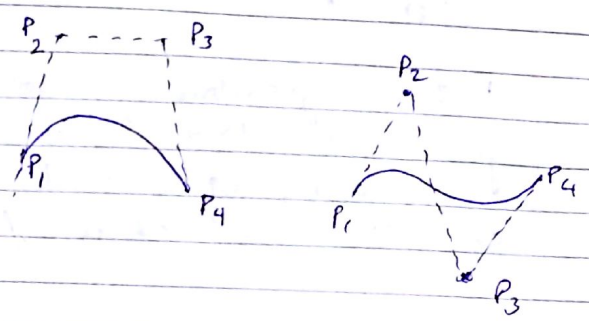
$$T = [t^3 \; t^2 \; t \; 1]$$

$$\therefore T m_H = [t^3 \; t^2 \; t \; 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2t^3 - 3t^2 + 1 & -2t^3 + 3t^2 & t^3 - 2t^2 + t & t^3 - t^2 \end{bmatrix}$$

Date :
Page No.

Bezier Curves



The Bezier form of the cubic polynomial curve segment indirectly specify the end pt. tangent vector by specifying 2 intermediate pts. that are not on the curve.

The starting & ending tangent vectors are determined by vectors P_1P_2 & P_3P_4 & are related to R_1 & R_4

$$\left.\begin{aligned} R_1 &= Q'(0) = 3(P_2 - P_1) \\ R_4 &= Q'(1) = 3(P_4 - P_3) \end{aligned}\right\} \quad (14)$$

$$G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \quad (14A)$$

$G_H = M_{HB} \cdot G_B$

The matrix M_{HB} that defines the

Page No.

relation $G_H = M_{HB} \cdot G_B$ b/w the Hermite Geometry vector G_H & Bezier geometric vector G_B is just the 4×4 matrix in the following eq^n which rewrite eq. (14) in matrix form.

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$= M_{HB} \cdot G_B \quad -(15)$$

From equation (12) & (15) we have

$$Q(t) = T \cdot M_H \cdot G_H = T \cdot M_H \cdot M_{HB} \cdot G_B$$

Define $M_B = M_H \cdot M_{HB}$ then

$$Q(t) = T \cdot M_B \cdot G_B \quad -(16)$$

Now,

$$M_B = M_H \cdot M_{HB}$$

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

Date :
Page No.

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad -(17)$$

From (16) $\quad Q(t) = T \cdot M_B$

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$= \begin{bmatrix} -t^3+3t^2-3t+1 & 3t^3-6t^2+3t & -3t^3+3t^2 & t^3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$= (-t^3+3t^2-3t+1)P_1 + (3t^3-6t^2+3t)P_2 + (-3t^3+3t^2)P_3 + (t^3)P_4$$

$$= (1-t)^3 P_1 + 3t(t^2-2t+1)P_2 + 3t^2(-t+1)P_3 + t^3 P_4$$

$$= (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$

$$\therefore \boxed{Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4} \quad -(18)$$

Page No.

The 4 polynomials

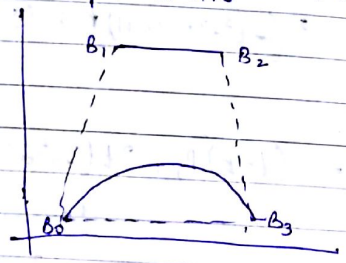
$B_B : T \cdot m_B$

which are weights in eq (18) are called Bernstein Polynomials

Q] Given $B_0[1,1]$, $B_1[2,3]$, $B_2[4,3]$ & $B_3[3,1]$.

The vertices of a Bezier polynomial determine the [0,1]

Determine the 7 points [b/w 0 & 1] i.e. find 5 pts. in b/w including 0 & 1



$P(t) = Q_3(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$

$P_1 = B_1$
$P_2 = B_2$
$P_3 = B_3$
$P_4 = B_4$.

Date :
Page No.

find t.

$P(0)$
$P(0.15)$
$P(0.35)$
$P(0.5)$
$P(0.65)$
$P(0.85)$
$P(1)$

Put t point at one pt, find corresponding x & y.

$P(0) = (1)P_1 + \dots$
$\therefore x = 1, \quad y = 1.$

$P(0.15) = (0.85)^3 P_1 + 3 \times 0.15 (0.85)^2 P_2 + 3(0.15)^2 (0.85) P_3 + (0.15)^3 P_4$
$= (0.6141) P_1 + (0.325) P_2 + (0.0574) P_3 + (0.0034) P_4$

$\therefore x_{0.15} = 0.6141 + 0.65 + 0.2296 + 0.0102 = 1.5$
$y_{0.15} = 0.6141 + 0.975 + 0.2296 + 0.0034 = 1.82.$

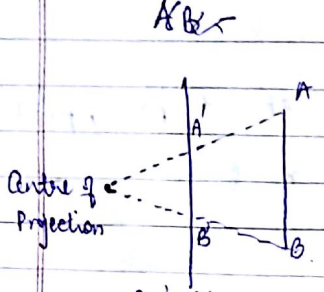
& Similarly, other points.

Page No.

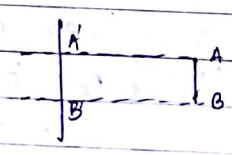
Projections

Perspective — Parallel

One point
Two points
Three points



AB → object
A'B' → Projection obtained

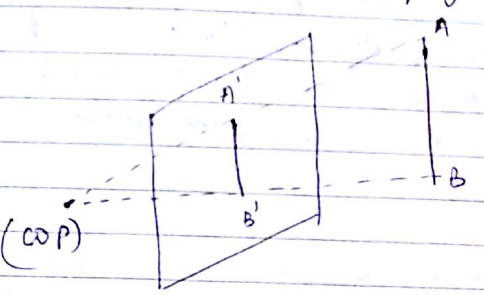


AB = A'B'

A perspective projection whose centre of projection is a pt. at ∞ becomes a ∥ projection.

The Class of projection we deal with here is known as planar geometric projection. because the projection is onto a plane rather than some curved surface & uses straight rather than curved projection.

Date :
Page No.



The projection of a 3D object is defined by straight projection rays called projectors emitted from a projection centre passing through each point of object & intersecting with the projection plane to form the projection.

In general, projection transforms pt. in a coordinate system in dimension n into a coordinate system of dimension < n.

	Perspective	Parallel
1)	Dist b/w centre of projection & proj. plane is finite	1) Dist. b/w centre of proj. & proj. plane is ∞
2)	When defining it, we explicitly specify its centre of projection	2) when defining it, we give its direction of approach.
3)	Due to perspective fore shortening object tend to look realistic.	3) It is less realistic view because perspective foreshortening is lacking.

Date :
Page No.

Perspective	Parallel
4) It is not particularly useful for recording exact shape & measurement of object. 11 lines don't in general project as 11 lines.	11 lines don't ↗ general proj. as 11 lines. It can be used for exact measurement & 11 lines do remain 11.

Perspective
Foreshorting - It is an illusion that the object & length appear smaller as their distance from centre of projection decreases. i.e. size of the perspective proj. varies inversely with dist. of that object from centre of projection.

Vanishing Point - The perspective proj of any set of ~~points~~ lines 11 lines that are not 11 to proj. plane converge at a pt. called vanishing pt.

If the set of lines is 11 to one of the 3 principal axis, the vanishing pt. is called an axis vanishing pt. or principal vanishing pt.
∴ there are at most 3 such pts.

Perspective projections are characterised by their no. of principle projection pts & therefore their no. of ~~axis~~ axis the proj plane cuts. 1, 2 or 3 principal vanishing pt.

for projection of a cube

1 Point Perspective - The plane cuts only 1 of the principal axis. All the 8 vertices of the cube are projected on the plane by joining each vertex to the centre of projection, we mark the points where these projections cut the projection plane & join these to form the projection of the cube.

We observe that the edges 11 to the x axis in the original cube are mutually 11 in the proj. plane. Same applies to the edges that are 11 to the y axis. All the edge that are 11 to z axis converge at a pt called the z axis vanishing point.

2 Point Perspective - The plane cuts 2 of the principal axis. The proj. plane cuts the x axis & z axis (any 2 planes). All

Page No

The 8 vertices of the cube are projected on the plane by joining each vertex to the centre of projection.

We mark the pts where these projectors cut the projection plane & join them to form projection of a cube. We observe that the edges that were ll to x axis in the original cube converge at a pt called the x-axis vanishing pt, ll^{ly} all the edges that are ll to z-axis in the cube converge at a pt called the z-axis vanishing pt.

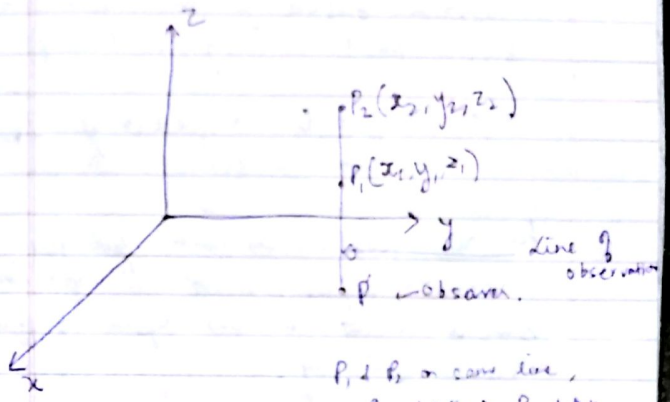
All the edges ll to y-axis are trivially ll in the proj plane.

Perspective projection

19/10/15

Date :
Page No

Hidden Surface Removal
or
Visible Surface determination



P_1 & P_2 on same line,
P_2 depth > P_1 depth.
$\therefore$ P_2 hidden

Given 2 pts. P_1 & P_2, does either pt obscure the other?

Ans Are P_1 & P_2 on same projection line?

If no, either pt. obscure the other.

If so, a depth comparison tells us which is in front of other.

Marketing Office : ... Delhi-110 002

Date :
Page No.

Z-buffer Method (or algo)
or
Depth buffer method

1) Initialise the depth buffer & refresh buffer so that for all buffer position (x,y) ← Pixel pos.
$depth(x,y) = 0$, $refresh(x,y) = I_{background}$ ← intensity

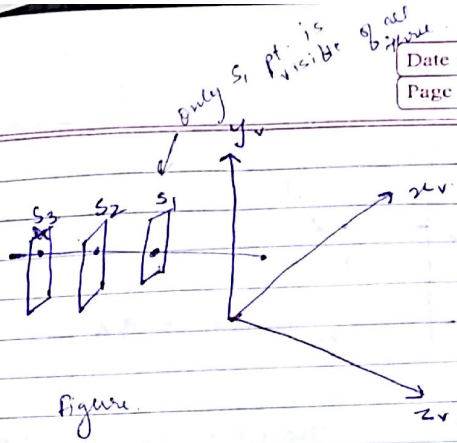
2) for each position on each polygon surface compare depth values to previously stored values in the depth buffer to determine visibility.

3) Calculate the depth z for each (x,y) position on the polygon.

4) If $z > depth(x,y)$, then set
$depth(x,y) = z$ & $refresh(x,y) = I_{surf}(x,y)$.

where $I_{background}$ is the value for the background intensity & $I_{surf}(x,y)$ is the projected intensity value for the surface at pixel position (x,y).

Date :
Page No.



Figure

At viewplane position (x,y) surface S_1 has the smaller depth from the viewplane.
S_1 is visible from (x,y).

After all surfaces have been processed depth buffer contains the depth values for visible surfaces & refresh buffer contains corresponding refresh values for those surfaces.

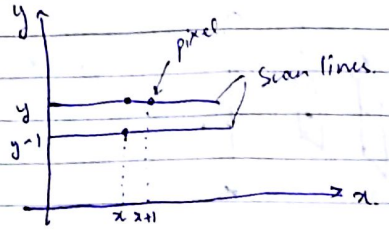
Depth Calculation

Depth values for a surface pos (x,y) are calculated from plane equation for each surface.

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

$$\Rightarrow z = \frac{-Ax - By - D}{C} \quad -(1)$$

Page No.



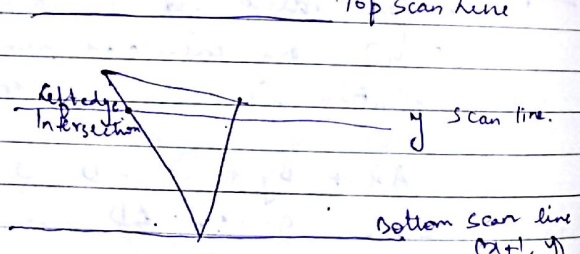
for any scan line, if the depth of pos (x, y) has been determined to be z, then the depth z' of the next position $(x+1, y)$ along the scan line is obtained from eqn. (1).

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

$$\Rightarrow z' = z - (A/C) \quad - (2)$$

A/C is a constant

So, succeeding the values across a scan line are obtained from the preceding values with a single addition.



Date :
Page No.

Left edge intersection

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

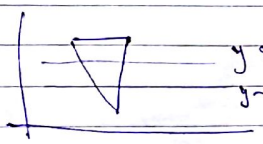
Depth values at each successive position across the scan line are then calculated by eqn (2).

We first determine the y coordinate extent of each polygon. Process the surfaces from the topmost scan line to the bottom scan line. Starting at top vertex, we can recursively calculate x position down a left edge of a polygon as

$$x' = x - \frac{1}{m}$$ → from $y = mx + c$ instead of y, $y-1$

$$y = mx + c$$
$$(y-1) = mx' + c.$$

$$\Rightarrow$$ gives $x' = x - \frac{1}{m}$.



Date :
Page No.

Depth value down the col edge are obtained recursively as.

$$z' = z + \frac{A/m + B}{C}$$ → from (1).

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

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

$A(x - \frac{1}{m}) + B(y-1) + Cz' + D = 0$

$Ax + By + Cz' + D - \frac{A}{m} - B = 0.$

$-Cz + Cz' - \frac{A}{m} - B = 0.$

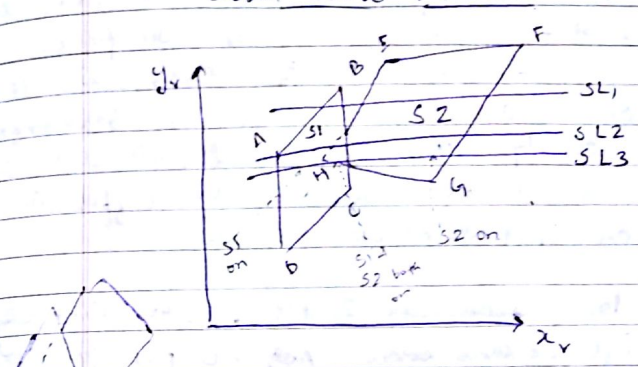
$z' = z + \frac{A/m + B}{C}$.

If we are processing down a vertical edge, slope is ∞ & the successive calculation reduces to.

$$z' = z + \frac{B}{C}.$$

Date :
Page No.

Scan line Method



Illustrate the scan line method for locating visible portion of surfaces for pixel positions along the line.

The active list for scan line 1 contains info. from the edges AB, BC, EH, FG.

for SL1 ↓ AB, BC, EH, FG.

For positions along these scan lines b/w the edges AB & BC, only the flag for surface S1 is ON.

∴ No depth calculations are necessary. & intensity information for surface S1 is entered from the polygon table into the refresh buffer.

For b/w edges EH & FG only the flag for surface S2 is ON.

No other positions along a scan line intersects the surfaces. So, the intensity values in the other areas are set to the background intensity. The background intensity can be loaded through the buffer in an initialisation routine.

For Scan line 2 & 3, the active edge list contains edges AD, EH, BC, FG. Along the scan line 2 from edge AD to edge EH only the flag for surface S1 is ON. But between edges EH & BC, flag for both surfaces are ON. In this interval, depth calculation must be made using the plane eqn for the two surfaces.

For this example, the depth of S1 is assumed to be less than that of S2. So, intensities of surface S1 are loaded into refresh buffer until boundary BC is encountered. Then the flag for S1 goes off & intensities for surface S2 are stored until edge FG is passed.

We can take advantage of coherence along the scan line as we pass from one scan line to the next.

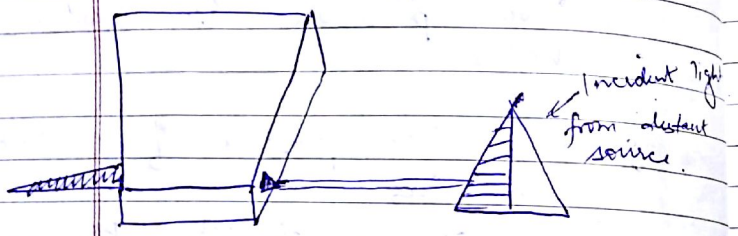
In this figure, SL3 has the same active list of edges as scan line 2. Since no changes have occurred in line intersection, it is unnecessary again to make depth calculation b/w edges EH & BC. The 2 surfaces must be in same orientation as determined in scan line 2. So, intensity for surface S1 can be entered without further calculation. Any no. of overlapping polygon surfaces can be processed with scan line method.

Coherence: The properties of one part of a scene are related in someway to other parts of the scene so that the relationship can be used to reduce the processing. Coherence method often involve incremental calculation applied along a single line or b/w successive scan line.

Date :
Page No.

Shading / Shadows

The surfaces that are visible from the light source are not in shadow. Those that are not visible from light source are in shadows. When there are multiple light sources, surface must be classified relative to each of them.



the generation of shading pattern for 2 objects in a table & a distant light source and shadow areas in this figure are surfaces that are not visible from the position of light source. Shad Shadow patterns generated by hidden surfaces method & are valid for any selected view positions as long as

Date :
Page No.

the light source positions are not changed.

2/11/15

Shading Models for Polygons

* Constant - Intensity shading
* Gouraud Shading
* Phong Shading

1) Constant - Intensity Shading

A fast and simple method for rendering an object with polygon surfaces is constant intensity shading. Also called flat shading.

In this method a single intensity is calculated for each polygon. All points over the surface of the polygon are then displayed with the same intensity value. Constant shading can be useful for quickly displaying general appearance of curved surface

2) Gouraud Shading -

Each polygon surface is rendering by

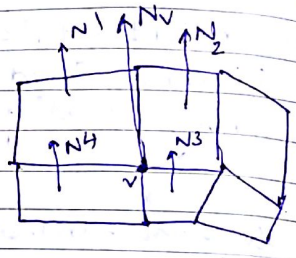
Date :
Page No.

Performing following calculations :-

a) Determine the avg. unit normal vector at each polygon vertex.

b) Apply Illumination model to each vertex to calculate vertex intensity.

c) Linearly interpolate the vertex intensity over the surface of the polygon.



n=4.

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

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

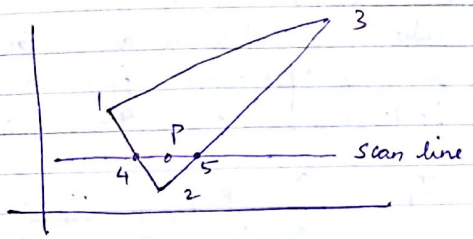
$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|}$$

Once we have the vertex normals

ASC/11

Date :
Page No.

we can determine the intensity at vertices from a lighting model.



For each scan line, the intensity at the intersection of scan line with polygon edge is linearly interpolated from intensity at the edge end pts.

For the example in figure, the polygon edge with end pt vertices at pos. 1 & 2 is intersected by scan line at pt 4. A fast method for obtaining the intensity at pt 4 is to interpolate intensity b/w I_1 & I_2 using only the vertical displacement of the scan line.

$$\therefore I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Similarly, intensity at the right intersection of this scan line (pt 5) is interpolated from intensity values at vertices

Date :
Page No.

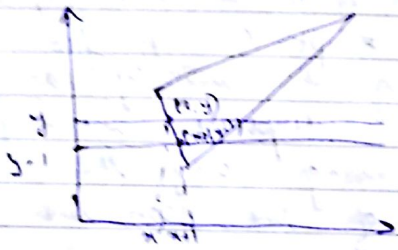
2 1 5 . Once

Once these boundary intensities are established for scan lines, the intensities for points such as p (i.e. P in fig.) in interpolated from the bounding intersections at 4 & 5 as:

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Incremental calculations are used to obtain successive edge intensity values between scan lines for obtaining successive intensity along a scan line.

As shown in the figure, if the intensity at edge pos.



$$I = \left(\frac{y - y_2}{y_1 - y_2}\right) I_1 + \left(\frac{y_1 - y}{y_1 - y_2}\right) I_2$$

then we can obtain the intensity along this edge for the next scan line y-1 as

$$I' = \left(\frac{(y-1) - y_2}{y_1 - y_2}\right) I_1 + \left(\frac{y_1 - (y-1)}{y_1 - y_2}\right) I_2$$

Date :
Page No.

$$I' = I - \left(\frac{1}{y_1 - y_2}\right) I_1$$

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

Similar calculations are used to obtain intensities at successive horizontal pixels along each scan line.

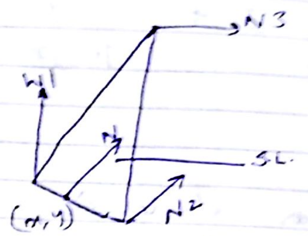
Phong Shading

A more accurate method for rendering a polygon surface is to interpolate normal vectors & then apply the illumination model to each surface pt.

A polygon surface is rendered by carrying out the following steps:

1) Determine the avg. unit vector at each polygon surface.
2) Linearly interpolate the vertex normals over the surface of the polygons
3) Apply an illumination model along each

scan line to calculate pixel projected pixel intensities for the surface pts



Interpolation of surface normals along a polygon H b/w 2 vertices is illustrated in figure.

The normal vector N for the scan line intersection pt. along the edge b/w vertices 1 + 2, can be obtained by vertically interpolating b/w edge & point normal

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

incremented methods are used to evaluate normals b/w scan lines & along each individual scan line at each pixel position along a scan line the illumination model is applied to determine the surface intensity calculations using an approximated normal at each pt along the scan line produces more accurate results than the direct interpolation ~~of interpolation of~~ ~~direct interpo~~ of intensities as in Gouraud shading. The tradeoff however is that Phong shading requires considerably more calculations.

Syllabus Over

7/8 ques, +1 each, 2 ques — A Everything in syllabus Extra will come.

Unit - 1 RGB and HSV model

Whole Unit — Unit - 2 → Filling algos also. Boundary, flood, character generation, Line attributes, fill styles & antialiasing

Unit - 3 → All trans. etc. Shearing + flip X.

Unit - 4 → Curve Clipping

Unit - 5 ~~the~~ Splines → Short notes mostly

P.T.O

CG → Extra topics

Only HSV RGB & Boundary & Flood fill.

Other Extra Backchodi

Date :

Page No.