1E

CSE 4201

computer graphics and animation.

computer Graphics

computer → graph (geometrical figure)

• point

— line

□ △ ○

computer

raw data process → pics (image)

কচ্ছে output দিচ্ছে

raw data —Process→ meaningful data

Lab; CSE 4202

OpenGL install করতে হবে।

But c দিয়ে code করতে হবে।

Historical point of view: Ivan sutherland.

(father of graphics)

scratch pad তৈরি করেছেন।

GUI, office automation, scanner, desktop publishing, photoshop

computer Graphics

Active

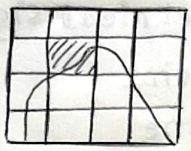
(user এর interaction এর আছে)

Passive

(interaction নেই)

Image representation: → Raster
→ vector

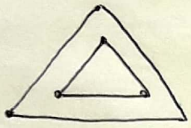
Pixel: smallest unit of picture
Raster graphics are composed of a combination of pixels.



Raster (space কম লাগে)

→ tiff, jpeg এ box box আকারে থাকে

→ Raster এ প্রত্যেকটা pixel কে box আকারে থাকে,



vector

Vector graphics are composed of paths and are based on mathematic

zoom করলেও same দেখায়,

Book: computer graphics (জায় ফ্রিরিজ)

Xiang & Ray

1: Introduction
2: Display Devices
3: Line drawing
4: Circle drawing
5:

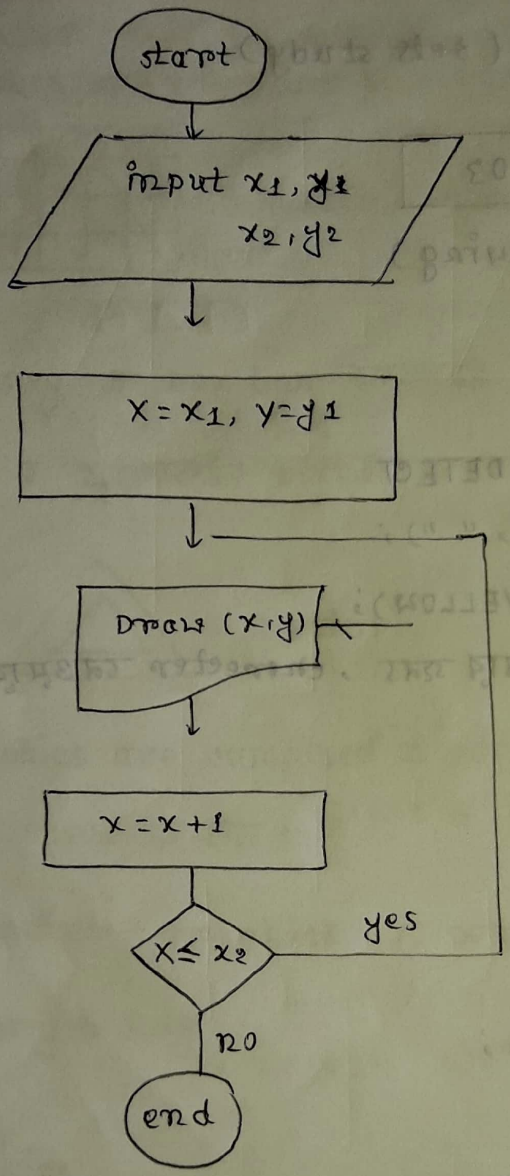
chapter 2 (self study)

chapter 03

(Line drawing)

```
int main()
{ int gm,gd;   gm=DETECT
  initgraph(&gm,&gd," ");
  putpixel(50,100,YELLOW);
  getch(); //wait করানোর জন্য , character নেওয়ার জন্য
  closegraph();
  return 0;
}
```

```
line(100,50,200,50);
int xm,ym;
xm=getmaxx()/2 (639 /2
ym=getmaxy()/2(479
putpixel(xm,ym)
line(0,ym,2*xm,ym);
line(xm,0,xm,2*ym);
putpixel(50+xm,ym-100)
```




```
x1=20, y1=50, x2=
x=x1, y=y1;
do {
    putpixel (x
    x++;
} while (x<=x2)
```

2E

$m = 1$ হলে

(2,2) (7,7)

$x = x_{start}$

$y = y_{start}$

$x++$

$y++$

Arbitary Lines Drawing Algorithm

Direct Line Algo:

$$y = mx + b$$

(2,3) (10,6)

$$m = \frac{6-3}{10-2} = \frac{3}{8}$$

$$b = y - mx$$

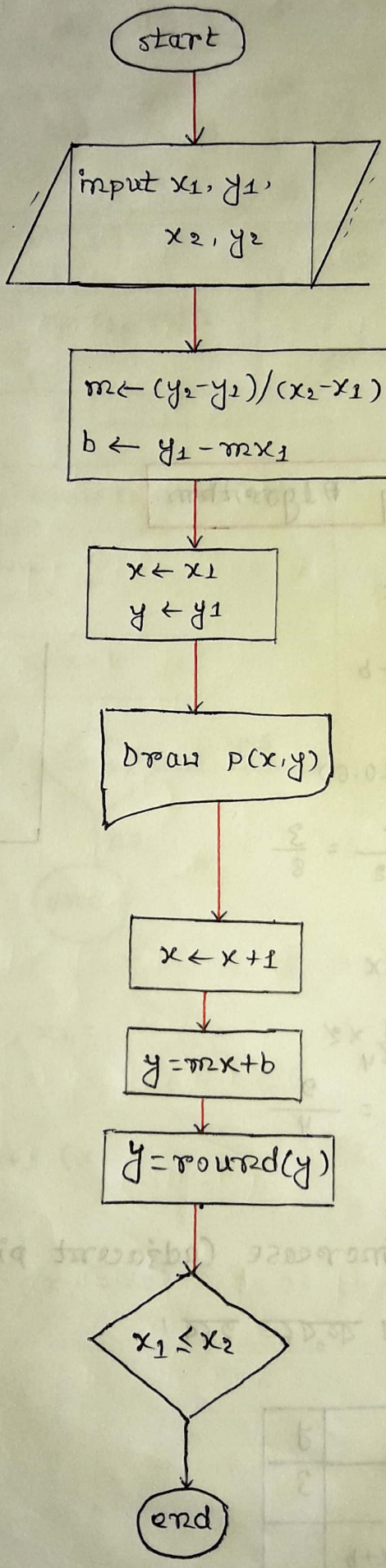
$$= 3 - \frac{3}{8} \times 2$$

$$= \frac{12-3}{4} = \frac{9}{4}$$

x এর value 1 করে increase (adjacent pixel নিতে হবে)

y calculate করে বের করতে হবে।

x		y
2		3
3	$y = mx+b$ $= \frac{3}{8} \times 3 + \frac{9}{4}$	
.		


start
input $x_1, y_1,$ x_2, y_2
$m \leftarrow (y_2 - y_1)/(x_2 - x_1)$
$b \leftarrow y_1 - mx_1$
$x \leftarrow x_1$
$y \leftarrow y_1$
Draw p(x,y)
$x \leftarrow x + 1$
$y = mx + b$
$y = \text{round}(y)$
$x_1 \leq x_2$
end

accuracy
fast
easy to implement

```
// axis
line (0, ym, 2*xm, ym)
line (xm, 0, xm, 2*ym)

x1 = 20; y1 = 20, x2 = 150, y2 = 80;

m = 1.0 * (y2-y1)/(x2-x1);

b = y1 - m * x1;

x = x1; y = y1;

do {
    putpixel (x+xm, ym-y, YELLOW)
    x++;
    y = int (m*x + b);
} while (x <= x2).

getch();
```

it involves floating point computation (multiplication and addition)

- multiplication এ time বেশি লাগে।
  round()
  ↳ time বেশি

vertical দিকে line টা হলে, x এর consequtive দুইটা মানের জন্য y এর difference অনেক বেশি

x
20 → 50 (diff 30)

y
20 → 150 (diff 130)
↳ y এর মাঝে 30টা value যাব।

abs(m) > 1 হলে y increament

$m < 1$ " x "

Digital Differential Analyzer [multiplication avoid করে]

(takes two consecutive point)

$P_2(x_2, y_2)$ → consecutive of (x_1, y_1)

P_1
(x_1, y_1) → any point in the straight line

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

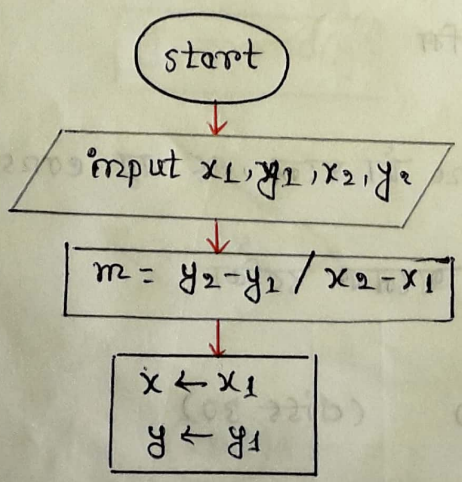
$$x_2 = x_1 + 1$$

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

$$y_2 = y_1 + m$$

So, $(x_1, y_1) \rightarrow (x_1+1, y_1+m) \rightarrow (x_1+1+1, y_1+m+m)$

multiplication problem ছিল
কিন্তু round হচ্ছে.



```
float z;
do{
   putpixel
   x++;    z=z+m;
   y=int(z)
      ↳ float
```

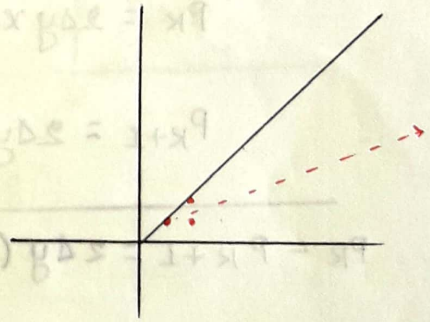
report : Algo name

flow chart

code

3A efficient কারণ addition, subtraction and multiplication by 2. shift operator use

Bresenham's Line Drawing Algorithm.



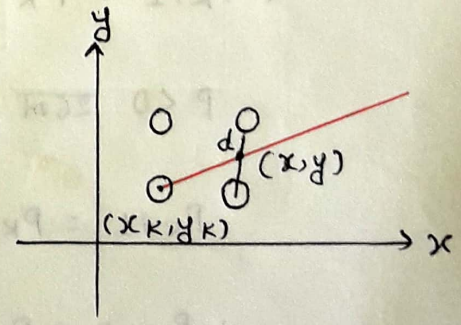
$m < 1$ হলে y অথবা y+1

```
x++;
y=y or y+1
```

current pixel (x_k, y_k)

next pixel (x_{k+1}, y_{k+1})

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



lower distance

$$d_L = y - y_k$$

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

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

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

$$du = (y_k+1) - y$$

$$= (y_k+1) - m(x_k+1) - b \qquad m = \frac{\Delta y}{\Delta x}$$

$dL - du = 2m(x_k+1) - 2y_k + 2b - 1$

division আকারে রাখা যাবে না।

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

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

P_k decision param

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

$$= 2\Delta y x_k - 2\Delta x y_k + \underbrace{2\Delta y + 2b\Delta x - \Delta x}_{c}$$

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

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

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

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

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

$P < 0$ হলে $y_{k+1} = y_k$

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x \times 0$$

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

$P > 0$ হলে $y_{k+1} = y_k + 1$

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

initial/starting p_0

$$p_k = 2\Delta y\, x_k - 2\Delta x\, y_k$$

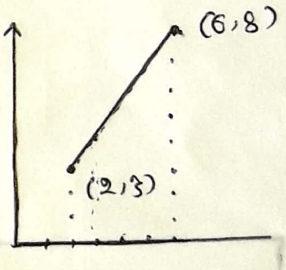
$$p_0 = 2\Delta y\, x - 2\Delta x\, y_k + \Delta x\,(2b - 1) + 2\Delta y$$

$$\Delta x\,(2[y_k - m x_k] - 1) + 2\Delta y$$

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

$$+ 2\Delta x\, y_k - 2\Delta y\, x_k - \Delta x + 2\Delta y$$

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



Example: (2, 3) (6,8)

$\Delta x = 4$

$\Delta y = 5$

$2y = 2\Delta x = 2 \times 4 = 8$

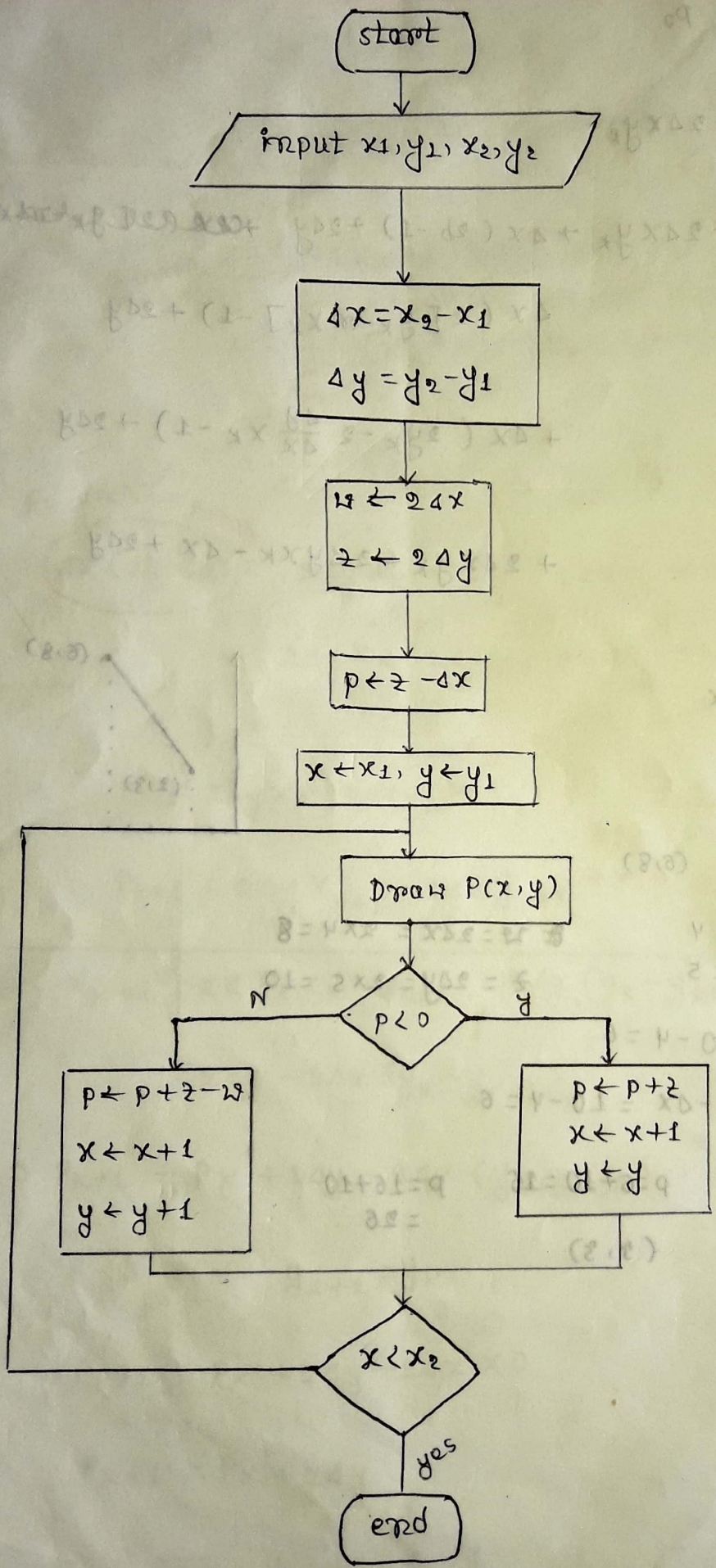
$z = 2\Delta y = 2 \times 5 = 10$

$p_0 = 10 - 4 = 6$

$= z - \Delta x = 10 - 4 = 6$

$p = 6$ $\quad p = 6 + 10 = 16$ $\quad p = 16 + 10 = 26$

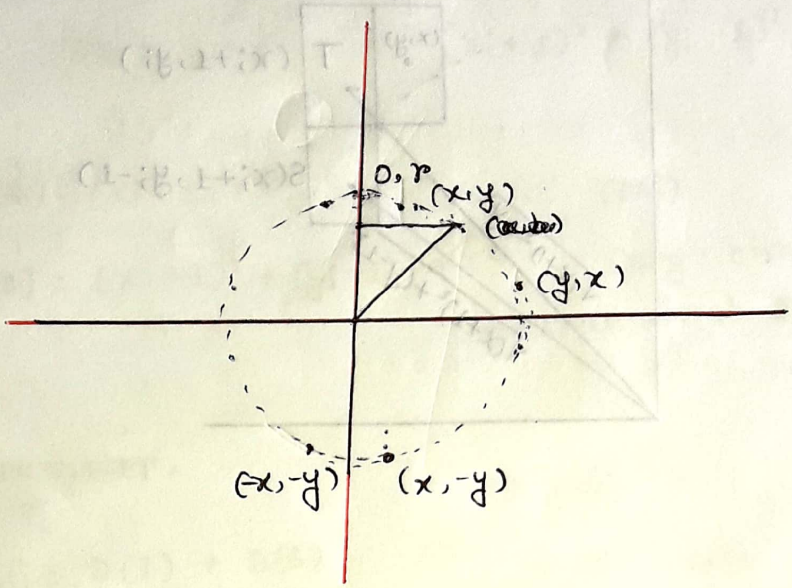
(2, 3) $\quad$ (3, 3)


start
input x_1, y_1, x_2, y_2
$\Delta x = x_2 - x_1$
$\Delta y = y_2 - y_1$
$n \leftarrow 2\Delta x$
$z \leftarrow 2\Delta y$
$p \leftarrow z - \Delta x$
$x \leftarrow x_1,\ y \leftarrow y_1$
Draw P(x, y)
$p < 0$
N
Y
$p \leftarrow p + z - n$
$x \leftarrow x + 1$
$y \leftarrow y + 1$
$p \leftarrow p + z$
$x \leftarrow x + 1$
$y \leftarrow y$
$x < x_2$
yes
end

3C

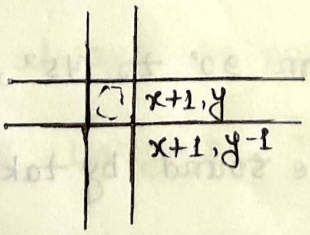
3.3 Scan-converting a circle

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



binding factor

circle is eight way symmetrical object.



x increase

y decision

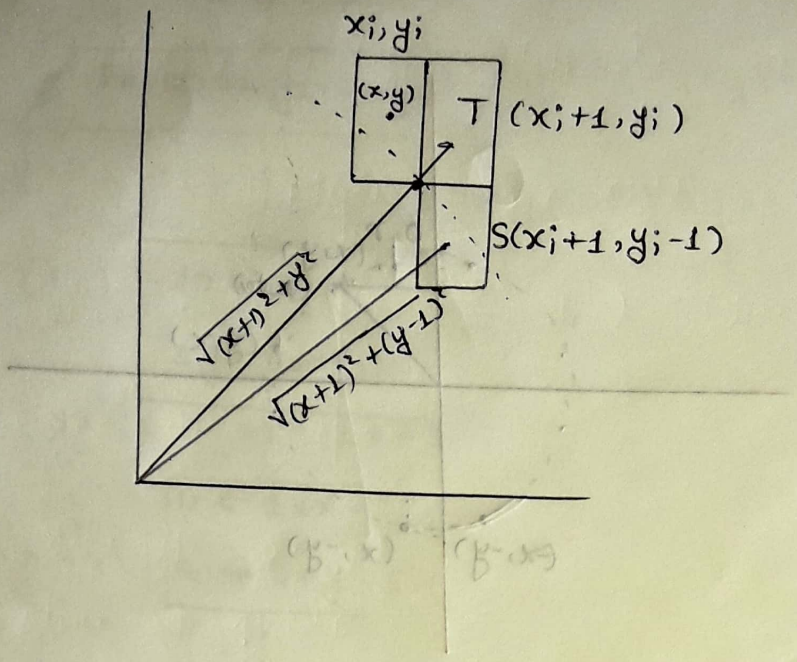
y or y-1

$a^2+b^2=r^2$

$\Rightarrow 2a^2=r^2$

$\Rightarrow a^2=\frac{r^2}{2}$

$\therefore a=\frac{r}{\sqrt{2}}$



3E

Bresenham's Circle Algorithm

Its points are generated from 90° to 45°, each new point closest to the true circle can be found by taking either two actions

i) move in the x direction, one unit.

or ii) move in the x " " " and move in the negative y direction one unit.

Therefore, a method of selecting between these two choices is all necessary to find the points closest to the true circle.

Distance of circle from origin r^2

Distance of T from origin $(x_i+1)^2 + y_i^2$

" of S " " $(x_i+1)^2 + (y_i-1)^2$

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

$D(S) = (x_i+1)^2 + (y_i-1)^2 - r^2$ (neg : circle এর ভিতরে)

↳ true circle এর dist

Decision parameter,

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

The function D provides a relative measurement of the distance from this center of a pixel to the true circle.

Since,

$D(r)$ is always be positive (T is outside the true circle)

and $D(S)$ will always be negatives (S is inside the true circle), is a decision variable d_i is defined.

Therefore, $d_i = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2$

When $d_i < 0$, $|D(T) < D(S)|$, T is chosen

$d_i > 0$, $|D(T) \geq D(S)|$, S is chosen

Decision variable d_{i+1} for next step

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

Finding $d_{i+1} - d_i$ and placing $x_{i+1} = x_i + 1$

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

If T is chosen ($d_i < 0$) then

$$y_{i+1} = y_i \text{ so,}$$

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

If S is chosen ($d_i > 0$) then

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

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

$$d_{i+1} = \begin{cases} d_i + 4x_i + 6 & \text{is } d_i < 0 \\ d_i + 4(x_i - y_i) + 10 & \text{id } d_i \geq 0 \end{cases}$$

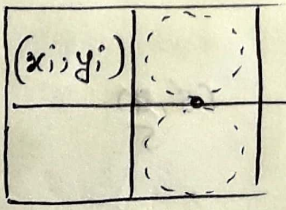
We set $(0, r)$ as the starting point

$$d_1 = 3 - \frac{2}{8}r$$

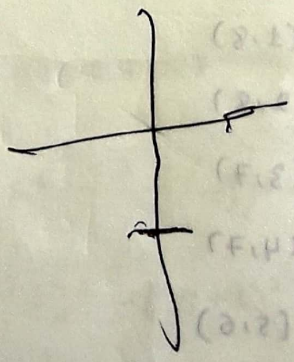
```
int x=0, y=r, d=3-2r

while (x <= y) {

    set pixel (x,y);

    if (d < 0)

        d = d + 4x + 6

    else

    { d = d + 4(x-y) + 10;

      y--;

    {

    x++;

    }
```

$$m = \frac{dy}{dx}$$

Midpoint circle Algorithm:

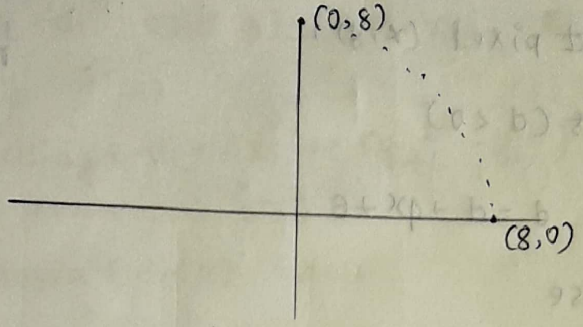


→ decision parameter $(x_{i+1}, y_i - \frac{1}{2})$



4c

Suppose the radius of a given circle is 8 unit. Please find out all the points in 1st quadrant.



$d = 3 - 2r =$

$d < 0$ $\quad d = d + 4x + 6;$

$d > 0$ $\quad d = d + 4(x-y) + 10;$

(0,8) (1,8) (2,8)

$d = (-13)$ $\quad d = -13 + 6 = -7$ $\quad d = -7 + 4 + 6 = 3$ $\quad d = 3 + 4(-5) + 10 = 3 - 20 + 10 = -7$

(0,8) $\quad$ (1,8) $\quad$ (2,8) $\quad$ (3,7)

$d = -7 + 4 \times 3 + 6 = -7 + 18 = 11$

(4,7)

$d = 11 + 4 \times (-3) + 10 = 11 - 12 + 10 = 9$

(5,6)

$d = 9$

(6,5)

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

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

$p = 1 - r$

$p < 0$ হলে $p = p + 2x + 3$

$p > 0$ হলে $p = p + 2(x-y) + 5$

Midpoint circle

$$f(x,y) = x^2 + y^2 - r^2 \begin{cases} < 0 & , (x,y) \text{ inside the circle} \\ = 0 & , (x,y) \text{ on the circle} \\ > 0 & , (x,y) \text{ on the out the circle.} \end{cases}$$

Where we have to select between T and S

consider a point halfway between pixel T and S that is

$(x_i + 1, y_i - \frac{1}{2})$ this is the mid point.

So decision parameter $p_i = f(x_i + 1, y_i - \frac{1}{2})$

$= (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$ ——————(i)

If p_i is negative the midpoint is inside the circle, so T is chosen

If p_i is positive, " outside " " , so S is "

similarly decision parameter

$p_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$ ——————(ii)

(ii) - (i) $\Rightarrow p_{i+1} - p_i = [(x_{i+1} + 1)^2 - (x_i + 1)^2] + [(y_{i+1} - \frac{1}{2})^2 + (y_i - \frac{1}{2})^2]$

$=$

Coordinate system

2D Geometric Transformation

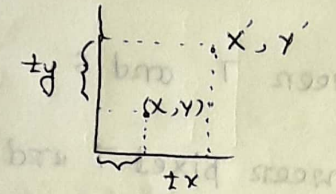
→ Translation

→ Rotation

→ Scaling

→ shearing

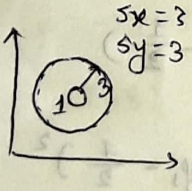
Translation:



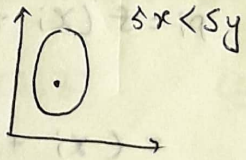
$X' = x + tx$

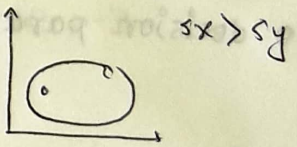
$Y' = Y + ty$

Scaling:



symmetric scaling ($sx = sy$)

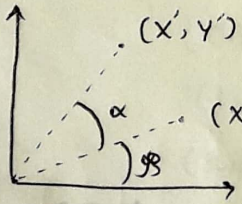




$X' = X sx$

$Y' = Y sy$

Rotation:



$X = R\cos\beta$

$Y = R\sin\beta$

$X' = R\cos(\alpha+\beta) = R(\cos\alpha\cos\beta - \sin\alpha\sin\beta)$

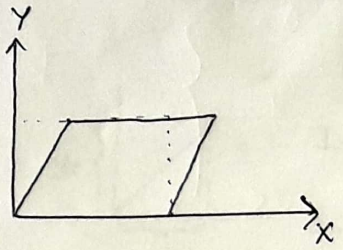
$Y' = R\sin(\alpha+\beta) = R(\sin\alpha\cos\beta + R\sin\beta\cos\alpha)$

$X' = R\cos\alpha\cos\beta - R\sin\alpha\sin\beta$

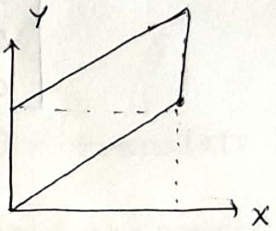
$= x\cos\alpha - y\sin\alpha$

$Y = X\sin\alpha + Y\cos\alpha$

Shearing



x এর মাধ্যমে shearing



shearing in Y

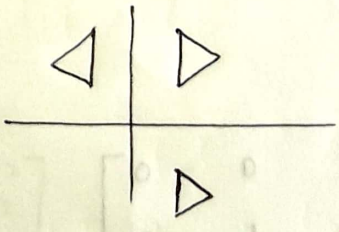
shearing in x

$X' = X + Y.a$ where $a \neq 0$

$Y' = Y + X.b$ where $b \neq 0$

it shear in both x and y then $a \neq 0$ and $b \neq 0$

Reflection



combine $X' = ax + by + c$

$Y' = dx + ey + f$

Matrix representation of transformations:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

CT: Bresenham, circle, 2D

Translation

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

Scale

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

Rotation

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

Shear

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

Identity Matrix

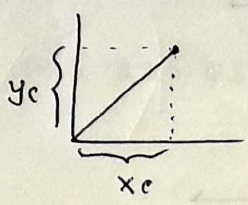
$$\begin{bmatrix} X' \\ Y' \\ H' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$X' = X$

$Y = Y$

$H = 1$

Rotating an object about its center



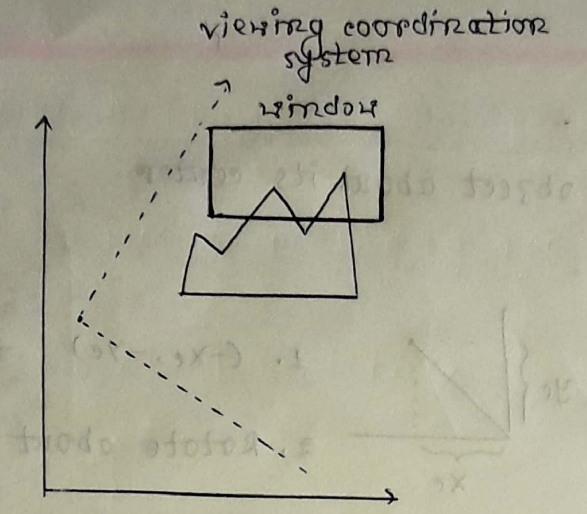
1. $(-x_c, -y_c)$ translate by
2. Rotate about the origin
3. Translate by (x_c, y_c)

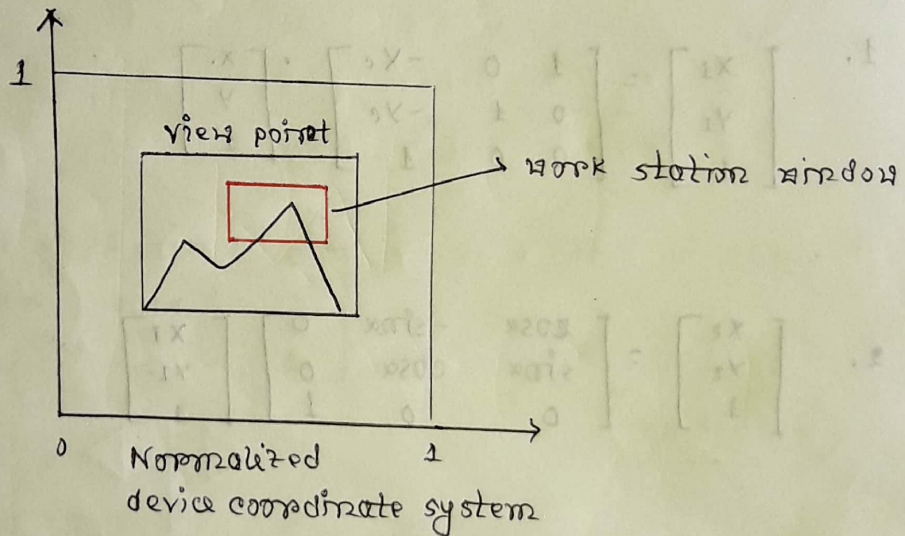
1. $$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2. $$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

3. $$\begin{bmatrix} x_3 \\ y_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$





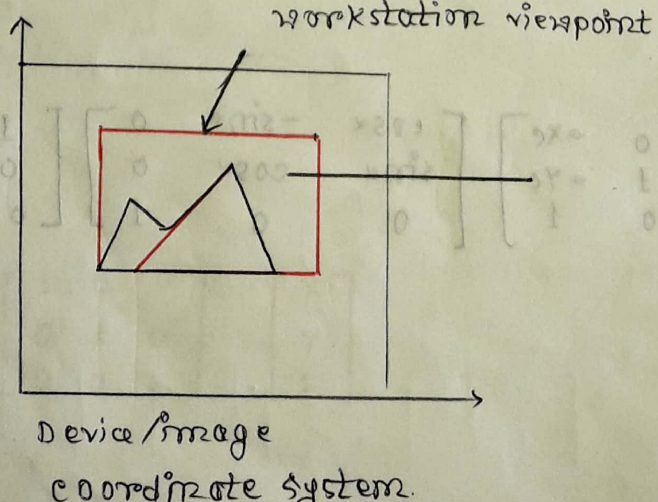


Fig: Viewing transformation

World coordinate system: Objects are placed into the scene by modeling transformation to a master coordinate system commonly refered to as WCS.

Viewing coordinate system: Rectangular window with its edges parallel to the axis of the WCS is used to select the portion of the scene for which image has to be generated. Sometimes an additional coordinate system called the viewing CS is introduced to simulate the effect of moving and/or tilting the camera

[Plateform/device independent coordinate system]

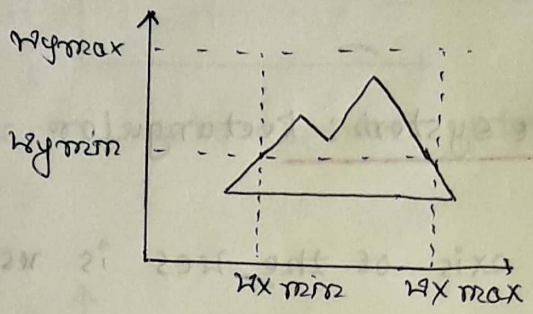
The process that converts object coordinates in WCS to normalize device coordinates is called "window to viewpoint" mapping or "normalization transformation."

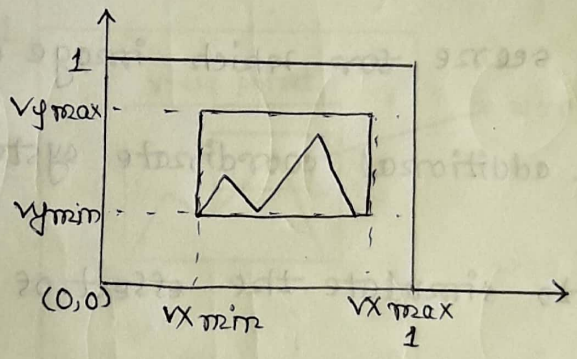
The process that maps normalized device coordinates to discrete device or image coordinate is called workstation transformation.

5.1 Window to viewport Mapping:

A window in WCS is specified by 4 (four) world coordinates:

Wx_{min}, Wx_{max}, Wy_{min}, Wy_{max}





The objectives of window to-viewport mapping is to convert the world coordinates (wx, wy) of an arbitary point to its corresponding normalized device coordinates (vx, vy)

$$N = \begin{bmatrix} 1 & 0 & Vx_{min} \\ 0 & 1 & Vy_{min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{Vx_{max} - Vx_{min}}{Wx_{max} - Wx_{min}} & 0 & 0 \\ 0 & \frac{Vy_{max} - Vy_{min}}{Wy_{max} - Wy_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -Wx_{min} \\ 0 & 1 & -Wy_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

point clipping

Line clipping → Coherent Sutherland

Polygone clipping → Sutherland hodgemen

In order to maintain same relative replacement of point in the viewport as in

$$\frac{Wx - Wx_{min}}{Wx_{max} - Wx_{min}} = \frac{Vx - Vx_{min}}{Vx_{max} - Vx_{min}}$$

$$\Rightarrow Vx = \frac{Vx_{max} - Vx_{min}}{Wx_{max} - Wx_{min}} (Wx - Wx_{min}) + Vx_{min}$$

$$\frac{Wy - Wy_{min}}{Wy_{max} - Wy_{min}} = \frac{Vy - Vy_{min}}{Vy_{max} - Vy_{min}}$$

$$\Rightarrow Vy = \frac{Vy_{max} - Vy_{min}}{Wy_{max} - Wy_{min}} (Wy - Wy_{min})$$

4.2 a) Find the matrix that represents rotation of an object by 30° about the origin.

b) What are the new coordinates of the point P(2, -4)

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

$$R_{30^\circ} = \begin{pmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ 1/2 & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ -4 \\ 1 \end{pmatrix}$$

$$= \frac{\sqrt{3}}{2} \times 2 - \frac{1}{2} \times (-4)$$

4.05 Perform a 45° rotation of triangle A(0,0), B(1,1), C(5,2)

a) about the origin and

b) about the P(-1,-1)

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

↳ origin এর নিচে কাজ করতে হবে।

$$\begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$ ← fixed

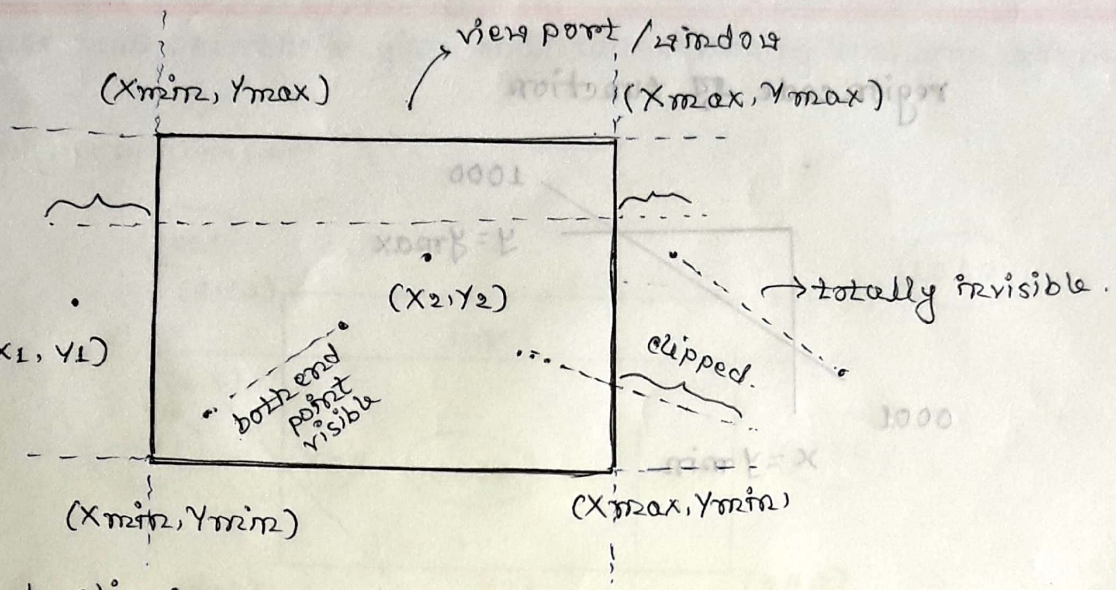
$$= \begin{pmatrix} 2 & 0 & -5\times 2+5 \\ 0 & 2 & -4+2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

Line drawing
circle
2D

$$R_{\theta,P} = \begin{pmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta & -h\cos\theta + k\sin\theta + h \\ \sin\theta & \cos\theta & -h\sin\theta - k\cos\theta + k \\ 0 & 0 & 1 \end{pmatrix}$$

4.5 ⓐ $[A'B'C'] = R_{45°}\cdot[ABC] = \begin{bmatrix} \cos45° & -\sin45° & 0 \\ \sin45° & \cos45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{3}{\sqrt{2}} \\ 0 & \sqrt{2} & \frac{7}{\sqrt{2}} \\ 1 & 1 & 1 \end{bmatrix}$



5.2 Point clipping:

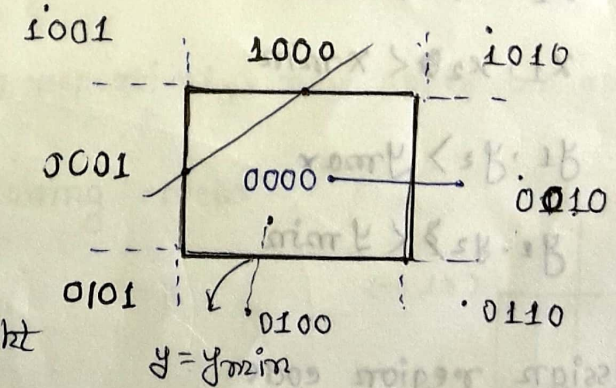
$x_{max} > x_2 > x_{min}$

$y_{min} < y_2 < y_{max}$

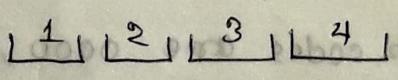
makes the point → visible

5.3 Line clipping: Cohen ~~Sodarl~~ Sutherland

endpoint
Assign ^ region code



start from left to right



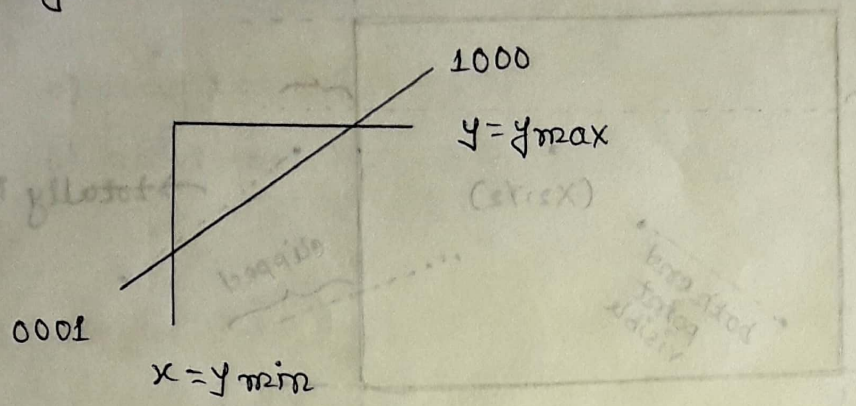
Bit 1 ≡ Endpoint is above the window = sign $(y - y_{max})$

Bit 2 ≡ " is below " " = sign $(y_{min} - y)$

Bit 3 ≡ " " right = sign $(x - x_{max})$

Bit 4 ≡ " " left = sign $(x_{min} - x)$

region code এর function



clipping categories:

1. visible (দুইটা endpoint 0000
2. Not visible (nonzero bo
3. clipping candidate (1 আর 2 এ না পড়লে)

$x_1, x_2 > x_{max}$

$x_1, x_2 < x_{min}$

$y_1, y_2 > y_{max}$

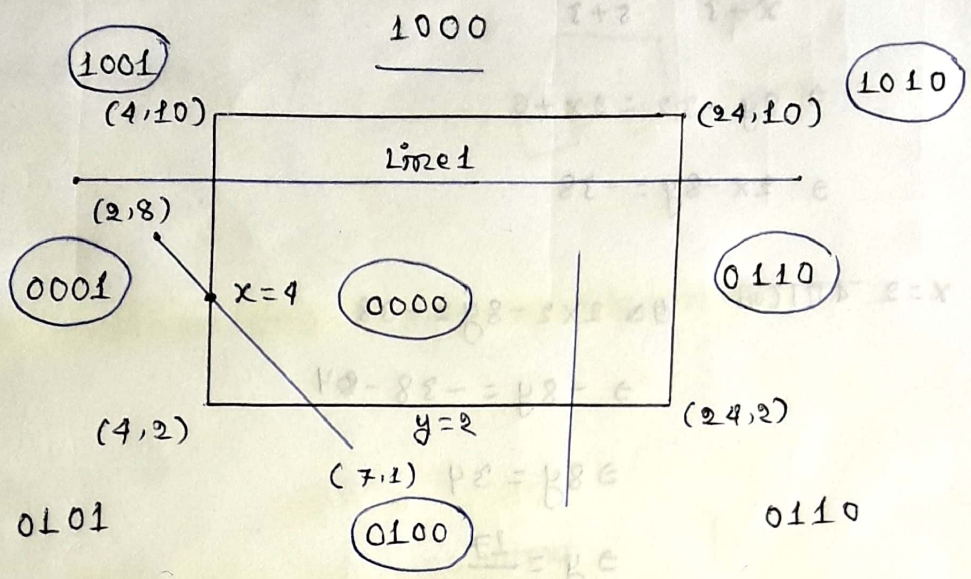
$y_1, y_2 < y_{min}$

1. Assign region code

2. The line is visible if both codes are 0000, and not visible if the bitwise logical AND of the codes is not 0000, and a candidate for clipping the bitwise, logical AND of the region.

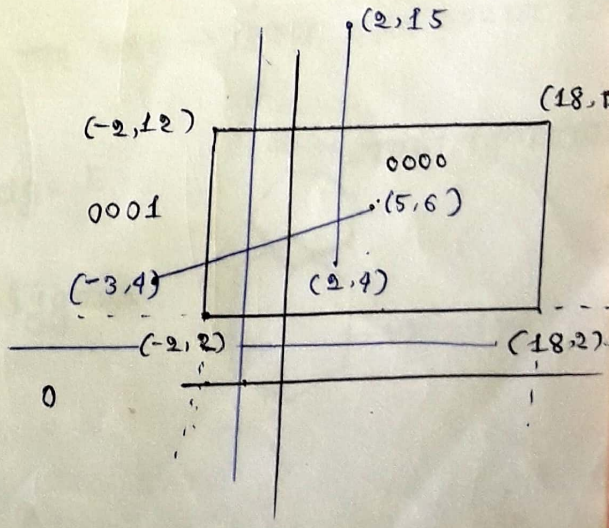
6A

Please find out the region code for following lines and clipped portion for given viewport.



endpoint of line 1 : 0001
0110

The lower left corner of a clipping window is (-2,2) with width and height are 20 and 10 repectively, now apply an appropiate algorithm to clip the following circles



(i) $y = 1$

(ii) $(-3,4)$ to $(5,6)$

(iii) $x = -1$

(iv) $(2,4)$ to $(2,15)$

straightline that goes through $(-3,4)$ to $(5,6)$

$$\frac{y-4}{x+3} = \frac{6-4}{5+3}$$

$$\Rightarrow 8y - 32 = 2x + 6$$

$$\Rightarrow 2x - 8y = -38$$

$x = 2$ বসালে,

$$\Rightarrow 2\times 2 - 8y = -38$$

$$\Rightarrow -8y = -38 - 4$$

$$\Rightarrow 8y = 34$$

$$\Rightarrow y = \frac{17}{4}$$

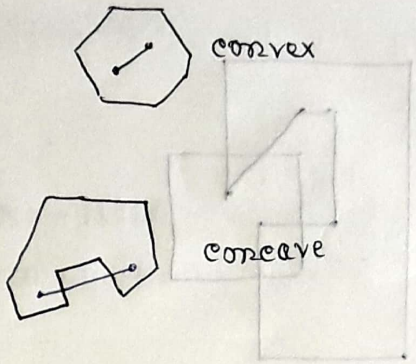
intersection point $(2, \frac{17}{4})$

convex: if the line joining any two interior points of polygon lies completely inside the polygon.

5.4 Polygone Clipping:

→ convex Polygone

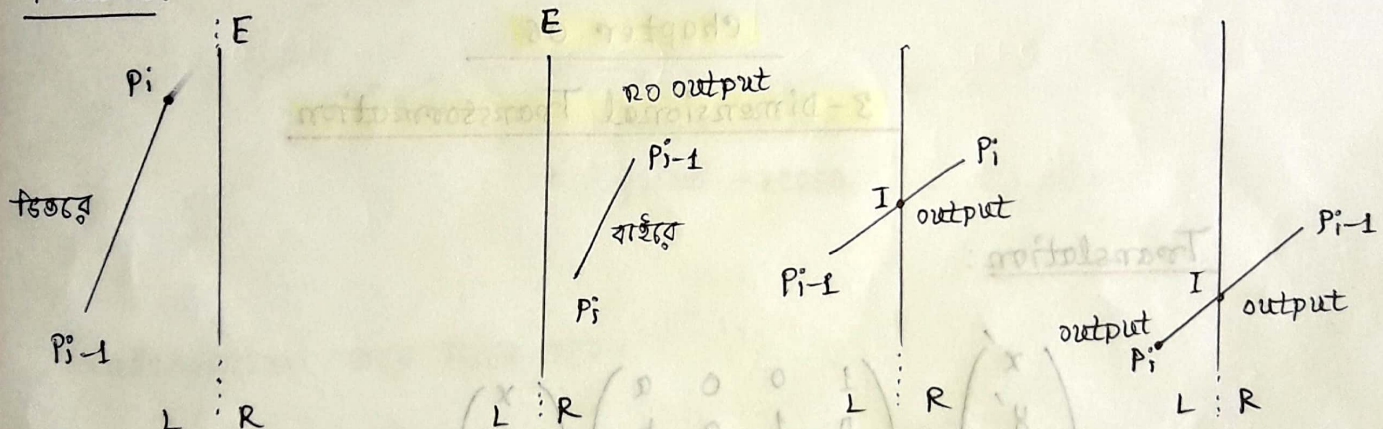
→ concave "



7B

"The sudtherland-Hodgman Algorithm"

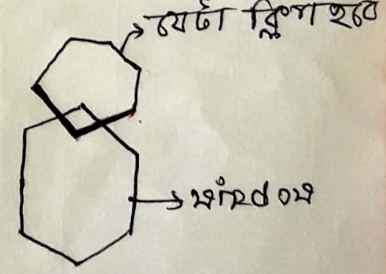
4 cases:



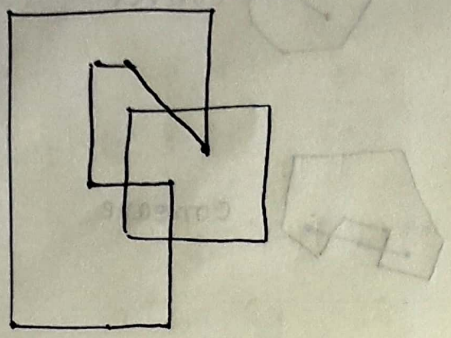
Let $P_i \ldots P_N$ be the vertex list of the polygon to be clipped

Let edge E, determined by endpoint A and B, be any edge of the positive convex polygon. (positive oriented, তাই left → ভিতর হিসেবে count হবে

We clip each edge in turn against the edge E of the clipping polygon forming a new polygone whose vertices are determined as follow



Vertex output list = intersection point + destination



Chapter 06

3-Dimensional Transformation

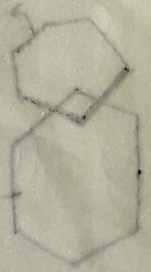
Translation:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

scaling

$$S_{sx,sy,sz} \cdot \begin{cases} x' = sx.x \\ y' = sy.y \\ z' = sz.z \end{cases}$$

$$S_{sx,sy,sz} = \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & sz \end{pmatrix}$$



symmetric, asymmetric scaling.

Rotation

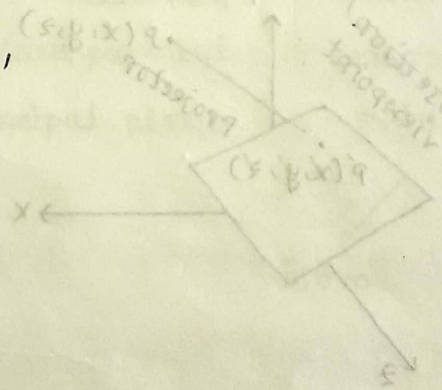
Rotation about z axis:

$$R_{\theta,k}: \begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \\ z' = z \end{cases}$$

$$R_{\theta,j}: \begin{cases} x' = x\cos\theta + z\sin\theta \\ y' = y \\ z' = -x\sin\theta + z\cos\theta \end{cases}$$

$$R_{\theta,i} = \begin{cases} x' = x \\ y' = y\cos\theta - z\sin\theta \\ z' = y\sin\theta + z\cos\theta \end{cases}$$

calculation করতে নিতে হবে।



composite transformation: self + study

OpenGL

glut

gl headerfile

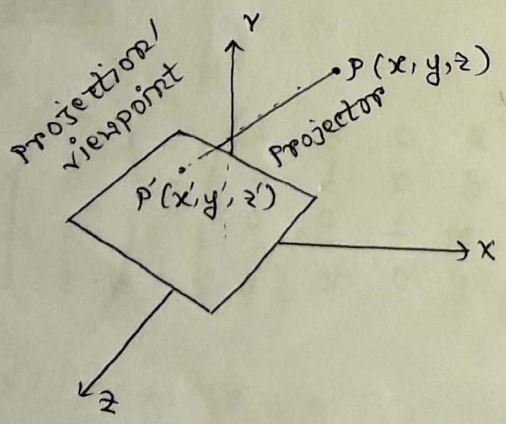
**

Chapter 07 (Mathematics of Projection)

(8 A)

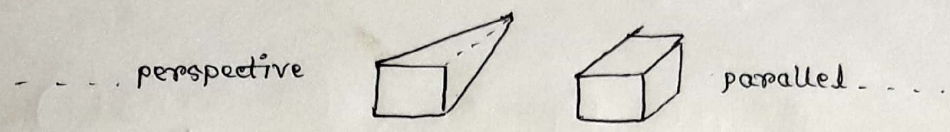
3D view কে 2D তে project (২ প্রকারের)

1. Perspective Projection
2. Parallel



Projection can be defined as an mapping of point P(x, y, z) on its image P'(x', y', z') in the projection plane or view plane.

The mapping is determined by a projection line called the project that passes through p and intersects the view plane.



**

Projection

Perspective
(Converging projects)

one point
(one principal vanishing point)

two point
(two principal vanishing point)

three point
(three principal vanishing point)

Parallel
(parallel projects)

Orthographic
(Projectors Perpendicular to view plane)

Oblique
(Projectors not perpendicular to viewplane)

Multiview
(view plane parallel to principal plane)

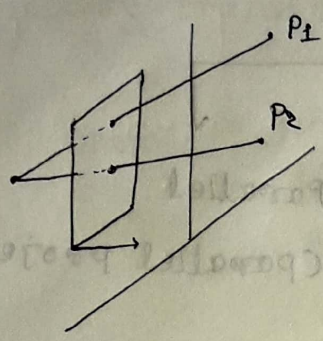
Axonometric
(viewplane not parallel to principal plane)

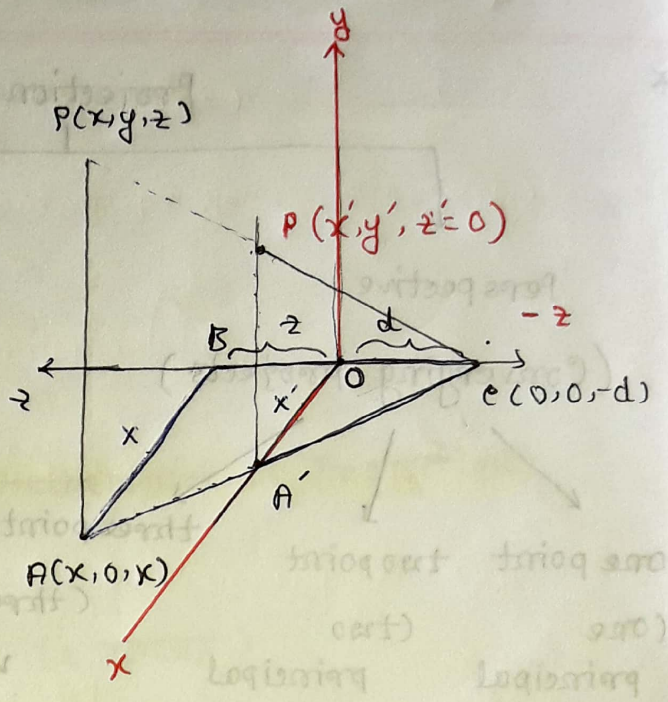
isom

Dimetric

trimetri

8B





View plane is the xy plane and center of projection, is taken as the point C (0, 0, -d) on the negative z axis.

Using similar triangle ABC and A'OC, we find

$$\frac{x'}{d} = \frac{x}{z+d}, \quad \frac{y'}{d} = \frac{y}{z+d}$$

and, $z' = 0$

So, $x' = \frac{d \cdot x}{z+d}$, $y' = \frac{d \cdot y}{z+d}$

and, $z' = 0$

*

Perspective Anomalies

1. Perspective foreshortening
2. Vanishing point
3. View confusion
4. Topological distribution

Graphics OpenGL (slide OPENGL1)

API → variety of function (120 functions)

OpenGL is · an 3D API (Application program interface)
· platform independent
· Operating system independent.

API → Application Program Interface

graphics Process

geometric primitive
(background + image primitives → Rendering → Frame Buffer

OpenGL Library

GLU (OpenGL Utility Library)
GLUT (interaction facility provide করে)

events : interaction with mouse, keyboard, window resize.

GLU (provides more complex scenario, such as curves and surfa

(Device Independence) rectangle windows on physical display, into which OpenGL draws graphics.
· events কে handle করার জন্য call back fn.

OpenGL Frame Buffer : → OpenGL doesn't draw directly to window

সবকিছু frame-buffer এ draw করে (an array of pixel)

flush করলে display তে চলে যাবে,

parallel buffer ও

glutInit

glutInitDisplay (constant) GLUT single ← single buffer

glutInitWindowSize (640, 480) [প্রথমটি window র size]

glutInitPosition (100, 150) [window টা main screen এর কত]

glutCreateWindow ("Window

callback
initialize
wait

main()

initialize

wait in infinite loop

Register glutMouseFunc (onMouseClick);

callbacks for all events in your program will react
↳ event handle.

OpenGL not object oriented (
↳ polymorphism, overloading

main()

init()

↳ default environment setup

```
#include <GL/glut.h>
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT)
    glBegin(GL_POLYGON)
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();
    glFlush();

int main(int argc, char** argv){
    glutCreateWindow("simple");
    glutDisplayFunc(mydisplay);
    glutMainLoop();

}
```

The Opengl Pipeline

rasterization → fragment

frame buffer ক flush করলেই display → screen stage

Depth information : 3D object.

Vertex stage : Transformation matrics

Rasterization : কত smoothly draw করা যায়।

Fragment : attribute (color) final elements of the drawing stage

Framebuffer

Screen stage

OpenGL Main points :

* 3D pipeline, Opengl pipeline

3D Viewing Pipeline vs OpenGL Pipeline

Modeling → viewing → projection

Normalization → viewport

OpenGL vertex Transformation:

vertex Data → ModelView Matrix → Projection Matrices (clipping)

MView

Normalization

Divide by

w

Order of transformation is important :

glline, glbeginner and glend এর মধ্যে যে basic shape গুলো

~~Assignment~~ Basic Polygon → assignment.

* include <GL/gl.h>

* include <GL/glut.h>

$ void display (void)

glclear (GL_COLOR_BUFFER_BIT) আগের আঁকা shape এর bit clear করার জন্য

↳ floating

glColor3f (1.0, 1.0, 1.0) RGB color (0-1)

↳ 3টা parameter 1 → white

glBegin (GL_QUADS)

↳ চারকোণ shape

glVertex3f (0.05f, 0.05f, 0.0f)

x y z

↳ z zero, so 2D shape

glVertex2i (

Opengl Command syrt (Slide 44.2)

* States

Capabilities:

glEnable(capability)
glDisable
GLboolean glIsEnabled

GL_BLEND
GL_DEPTH_TEST
GL_FOG
GL_LIGHTING

Quering states

glGet*()
↳ query

Drawing Geometry

glBegin()

glEnd()

Vertex ~~Attributes~~:

- glcolor
- gl Normal 3f (Lighting position normal)
-

Graphics draw করার আরো ২টি উপায় :

✓Vertex Array

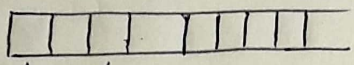
✓Display List

* problem of glBegin() , glEnd()

→ অনেকগুলো function call করতে হয়। Function call বড় overhead reduce করতে পারি by vertex array and Display list.

Vertex array

vertex গুলো array তে রাখা হয়।



1টা shape (triangle)

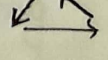
এখানে structure আকারে

* How to call vertex array?

* Display list কিভাবে লেখা হয়?

Rendering এর ক্ষেত্রে valid concave

culling mode on করে দিলে smoothly polygon কে draw



front, back (3D dim)

clockwise, anticlockwise

2D

winding mode GL_CCW or GL_CW

Culling → The process of removal of hidden surfaces is termed as culling related to visible surface determination.

Lookup table (4 bit → 16 bit color)

	Red	Green	Blue
0	0	0	0
1	255	0	0

Processing of color

RGBA

Lighting → Blending

Texture

glPointsize

glLineWidth

glLinestipple (dotted line)

Perspective

glFrustum

glOrtho

gluPerspective

gluOrtho

Camera

gluLookAt

Order is very important

Light and shading

Light : ambient, diffuse, specular
↓ ↓ └→ highlight an object
no source indirect

Ambient + diffuse :

Hidden surface

Texture mapping

How to use texturing automatically

* Atmospheric Effects

* Antialiasing

Blending

→ Alpha Blending

Blue + red → magenta

14.07.20

v.v. imp

3D viewing and OpenGL **Pipeline**

Object coordinates → eye coordinate

$$eye = Modelview \cdot \begin{pmatrix} obj \\ coord \end{pmatrix}$$

projection এ একটা vanishing point থাকবে।
অথবা
parallel ও হতে পারে।

clipping হয় এখানে।

Normalised : [-1, 1] range

Viewport : নিজের device এ কিভাবে দেখাতে চাচ্ছি।

Window coordinate

bitmap and vector graphics

·bmp ·svg ↘
object and primitives
Adobe illustrator

smallest unit
pixel

Adobe photoshop

1st assignment এ box এ add

void init() → set up basic opengl state

flush করলে screen এ যায়।

glEnable (GL_DEPTH)

↳ overlap

Perspective

aspect ratio and angle

Orthographic

parallel

gluLookAt (eye, center, up-direction)

OpenGl 1

22 (Pipeline) +

33

33.2

✓ Basic Program structure

simple.c

Viewport

Lab এর সব primitives

et " " " function

Fog and Blending

lab final (graphics + opengl), এর viva ও time এ হবে।

vanishing point , CRT, LED, LCD diss

projection

clipping

Blending: The colour of the surfaces can be combined, if the surface in front is not completely opaque.

Ex. glBlendFunc(GL_SRC_ALPHA...

glEnable(GL_BLEND)

Fog: Objects far away would not be as sharp and clear as objects close by. We can simulate