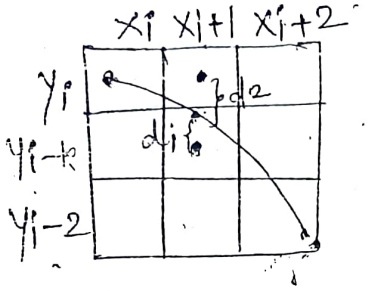
3) Bresenham's Circle generating Algorithm:-

Consider circle center ~~at (x,y)~~ (x_c, y_c) at origin $(0,0)$, then calculate pixel position (x,y) by moving $x=0$ to $x=y$ by unit step. Starting pixel position of circle $(x_i, y_i) = (0, r)$.



Assume that pixel (x_i, y_i) is plotted and need to determine which pixel whether (x_i+1, y_i) or (x_i+1, y_i-1) is closer to circular path from the figure. Circle is sampled at (x_i+1).

~~eqn of circle is~~

There is a decision parameter d_i to determine this.

$$d_i = 3 - 2r$$

if $(d_i \leq 0)$

then select pixel at position (x_i+1, y_i)

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

else (x_i+1, y_i-1)

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

$y = y - 1$.

Bresenham Circle generating Algorithm:-

Step 1: Accept circle center (x_c, y_c) and radius r as input.

(1) Shapes
(2) DDA
(3) Brosen
(4) Circle
(5) ~~Ellipse~~
(6) Area fil
(7) 2D
(8) Colli
(9) Polyg
(10) 3D
(11) Bezier
(12) OpenGL
(13) OpenGL
(14) Animat

Step 2 : To setup initial parameters.
$x = 0$, $y = r$,
$di = 3 - 2r$.

Step 3: Move the value of x from $x=0$ to $x=y$ by unit step.

Step 4: Display circle pixels using eight way symmetry.
(Write 8 putpixel commands here)

Step 5: if $di \leq 0$
then select pixel at position (x_i+1, y_i)
$di = di + 4x_i + 6$.
else
Select pixel at position (x_i+1, y_i-1)
$di = di + 4x_i - 4y_i + 10$.
$y = y - 1$.

Step 6: Repeat the procedure through step 3

Step 7: End.

C program for Bresenham's Circle Algorithm :-

```
#include <stdio.h>
#include <conio.h>
#include<graphics.h>
void main()
{
int gd= DETECT, gm;
clrscr();
initgraph (&gd,&gm,"C:/TC/BGI");
int x,y,di,r,xc,yc;
printf("Enter circle center & radius");
scanf("%d %d %d",&xc,&yc,&r);
x=0;
y=r;
```

8 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector - 1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.... (3)

To be filled in by the candidate

No 002467

Subject :
Course : CM / IF / EJ / EX
Year : I / II / III
Date :

Sr. No.:
Roll No. :
Signature of the Invigilator

Main Answer book	No. fo Suppliments	Total
1		

Que. No.	1	2	3	4	5	6	Total Marks	Sign. Of Examiner
Marks								

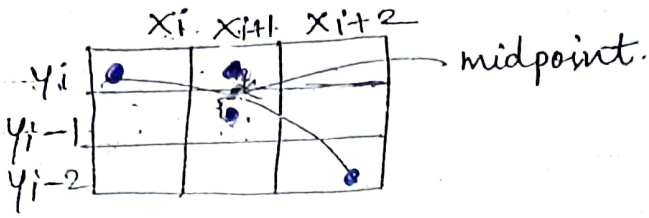
(Please start writing from here)


```
di = 3 - 2*r;
for (x=0 ; x <= y ; x++)
  {
  putpixel (x+xc , y+yc , 15);
  putpixel (y+xc, x+yc , 15);
  putpixel (y+xc, -x+yc, 15);
  putpixel (x+xc, -y+yc, 15);
  putpixel (-x+xc , -y+yc, 15);
  putpixel (-y+xc , -x+yc, 15);
  putpixel (-y+xc , x+yc, 15);
  putpixel (-x+xc , y+yc, 15);
 if (di<=0)
   {
   di = di + 4 * x + 6;
   }
 else
   {
   di = di + 4 * x - 4 * y + 10;
   y = y-1 ;
   }
}
getch();
closegraph();
}
```

x, y

Plot circle wh... = radi...
is 5 b cent... in (0,0)

4) Mid-point Circle generating Algorithm:-



Starting co-ordinate for circle (0, r).
Move the value of x from x=0 to x=y by unit step.
Decision parameter di is used to decide which pixel to select.

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

if (di <= 0)
then
$di+1 = di + 2 * xi + 3$ $x = x+1, y = y$
else
$di+1 = di + 2xi - 2yi + 5$ $x = x+1, y = y-1$
$y = y-1$

Midpoint Circle generating Algorithm:-

Step 1:
Accept circle center (xc, yc) and radius r as input.

Step 2:
To setup initial parameter
$x = 0$
$y = r$
$di = \frac{5}{4} - r$ (0, r)

Step 3:
Move the value of x from x=0 to x=y by unit step.

Step 4:
Display circle using 8 way symmetry. (0, r)
(Write 8 putpixel commands here).

Step 5:
if (di ≤ 0)
then select pixel (xi+1, yi)
di+1 = di + 2xi + 3
else select pixel (xi+1, yi-1)
di+1 = di + 2xi - 2yi + 5
y = y - 1

Step 6: Repeat procedure through Step 3.
Step 7: End.

C Program for Midpoint Circle Algorithm is same as Bresenham Circle Algorithm only. Just change the formulas or equations.

5) DDA Circle generation Algorithm:

The equation of the circle when circle center is at origin.

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

Construct the circle using incremental x value; $\Delta x = \epsilon \cdot y$ & incremental y value; $\Delta y = -\epsilon \cdot x$ where ϵ is calculated from radius of the circle.

$$2^{n-1} <= r < 2^n.$$

$$\epsilon = 2^{-n}.$$

Hence, to get the correct circle by applying incremental steps, the eqn. are

are,

$x_{n+1} = x_n + \epsilon \cdot y_n$ and

$y_{n+1} = y_n - \epsilon \cdot x_{n+1}$.

DDA Algorithm for circle:

Step 1:

Accept circle center (x_c, y_c) and radius r as input.

Step 2:

Setup $x = 0, y = r;$

Step 3:

$x_1 = x, y_1 = y$

Step 4:

do

$x_2 = x_1 + \epsilon \cdot y_1$

$y_2 = y_1 - \epsilon \cdot x_2$ and

plot the pixel for circle using

putpixel $(x_2, y_2, 15);$

Step 5:

Initialize the co-ordinate $x_1 = x_2$, $y_1 = y_2$.

while $((y_1 - y) \leq \epsilon \;||\; (x - x_1) > \epsilon)$

Step 6: Stop.

C program for DDA Circle Algorithm :-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void main()
{
int gd, gm, r, val, i, xc, yc; i = 0;
float e, x, y, x1, y1, x2, y2;
clrscr();
printf("Enter the radius"); // Take i/p for (xc & yc) also.
scanf("%d", &r);
initgraph(&gd, &gm, "c:/TC/BGI");
x=0;
y=r;
/* calculate e */
do
 {
 val = pow(2,i);
 i++;
 }while (val<r)
e = 1/pow(2,i-1);
do
{
 x2= x1 + e.y1;
 y2 = y1 - e.x2;
 putpixel (x2+xc, y2+yc, 4);
 x1 = x2;
 y1 = y2;
}while((y1-y) <= e || (x-x1)>e)
getch();
closegraph();
}
```

* Polygon :-

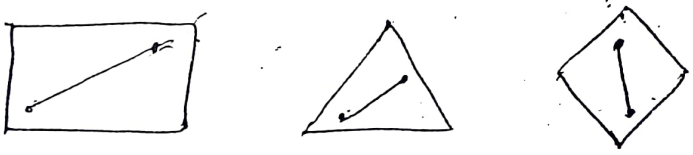
Polygon may be represented as a number of line segments connected end to end to form a closed figure. The line segments which form the boundary of polygon are called the polygon ~~vertices~~ edges.

- To form a polygon we need minimum three vertices and three edges & that too must form a closed loop. Triangle is the simplest form of polygon.
- We can divide polygon into two types :-

1) Convex :-

Convex polygon is a polygon in which if you take any two points which are inside the polygon and if you draw a line passing through two ~~lines~~ points & if all the points on this line lies inside the polygon, then such a polygon is called as convex polygon.

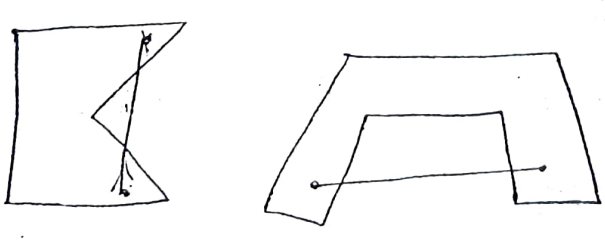
eg :-



2) Concave :-

Polygons on which if line drawn from two points inside of polygon passes from outside the polygon. is called as concave polygon.

eg :-



* Polygon Representation :-

Basic approaches to represent polygons are :-

1) Polygon drawing primitive :-

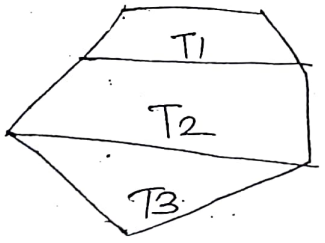
Some graphics devices supports polygon drawing approach, where directly basic polygon shapes are drawn. Here, one polygon is saved as one unit.

eg. Standard command for drawing triangle could be,

triangle $(x_1, y_1, x_2, y_2, x_3, y_3)$.

2) Trapezoid primitive :-

Some graphics devices support trapezoid primitive, where a polygon is represented as a set of trapezoids. Trapezoids are formed from two scan lines & two line segments (edges of polygon) as shown in fig.



3) Line and Point :-

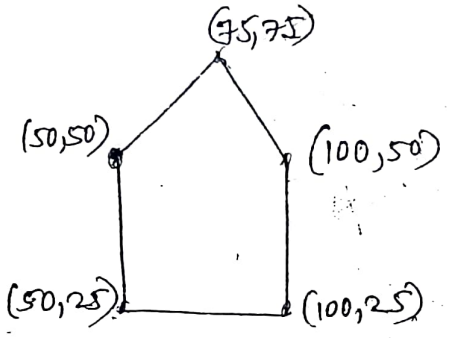
Here, polygons are represented using series of only lines and points. This series of lines and points is represented as a single unit and is stored in display file. Display file stores data in following format as array.

op	x	y

where op = opcode. It is 2 to draw line
x, y = co-ordinates of vertex.

Any number 'n' greater than 2 in op field indicates number of line segments in polygon and thus indicate next 'n' number of comman to use to draw a single polygon.

Following figure shows a polygon and its representation using lines & points in a display file.



Representation of above polygon in Display file

	op	x	y
no. of vertices & 5 ids commands more to come	5	50	50
	2	75	75
	2	100	50
	2	100	25
	2	50	25
	2	50	50

drawpoly (6, a[]);

int a[] = {

8 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector - 1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.... (4)

To be filled in by the candidate

Subject :

Course : CM / IF / EJ / EX Sr. No.: Nº 002468

Year : I / II / III

Date :

Roll No. :

Signature of the Invigilator

Main Answer book	No. fo Suppliments	Total
1		

Que. No.	1	2	3	4	5	6	Total Marks	Sign. Of Examiner
Marks								

(Please start writing from here)

* Entering Polygons :-

We will see how to actually enter this polygon data to the graphics system.

How to enter polygon into display file. It consists of three columns, where,
first column is opcode number,
second column is x-coordinate,
third column is y-coordinate.

These three columns can be treated as three different arrays.

Following is the stepwise procedure to enter a polygon.

Step 1 :

Enter N as number of polygon edges.

Step 2 :

Enter x & y co-ordinates of all vertices of polygon in array Ax and Ay respectively.

Step 3 :

Initialize counter i = 0.

op[i] = N
x[i] = Ax[i]
y[i] = Ay[i]
i = i+1

It loads first row of display file i.e. polygon command with start point.

Step 4:

op[i] = 2
x[i] = Ax[i]
y[i] = Ay[i]
i = i+1

Step 5:

Repeat step 4 'N-1' times where step 4 enters a line to command.

Step 6:

op[i] = 2
x[i] = Ax[0]
y[i] = Ay[0]

This step enters last line command with start point as end point so to close the polygon.

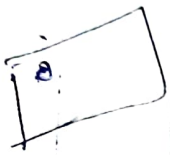
Step 7:

Stop

* Inside- Outside Test:

In standard polygon shapes like triangle, rectangl its component edges are joined only at vertices and edges do not have common points in plane.

edges are non-intersecting. But some graphics appl. produces polygon shapes which produces intersecting edges. For such shapes to find out if a particular point is inside or outside polygon is difficult and need some rule or test to follow.



There are two tests to find out if a point is inside or outside of polygon.

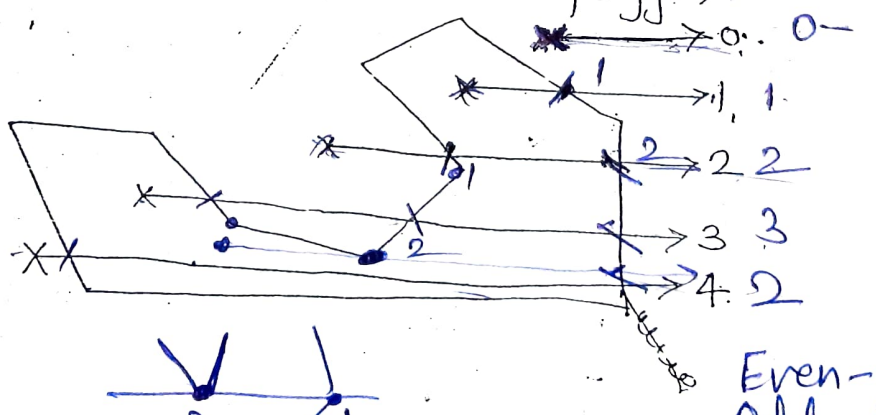
1) Odd-Even Rule
2) Non-Zero Winding Number Rule.

1) Odd-Even Test:

Consider a polygon made up of N vertices (x_i, y_i) where i ranges from 0 to N-1. The last vertex is assumed to be the same as first vertex i.e. the polygon is closed.

To determine status of a point (x_p, y_p) to be tested whether inside or outside, consider a horizontal ray (scanline) drawn from (x_p, y_p) and to the right.

a) If the number of times this ray intersects the line segments (edges of polygon) is even then the point is outside the polygon.
b) If the number of intersections are odd then the point (x_p, y_p) lies inside the polygon.



edges are non-intersecting. But some graphics appl. produces polygon shapes which produces intersecting edges. For such shapes to find out if a particular point is inside or outside polygon is difficult and need some rule or test to follow.

There are two tests to find out if a point is inside or outside of polygon.

1) Odd-Even Rule
2) Non-Zero Winding Number Rule.

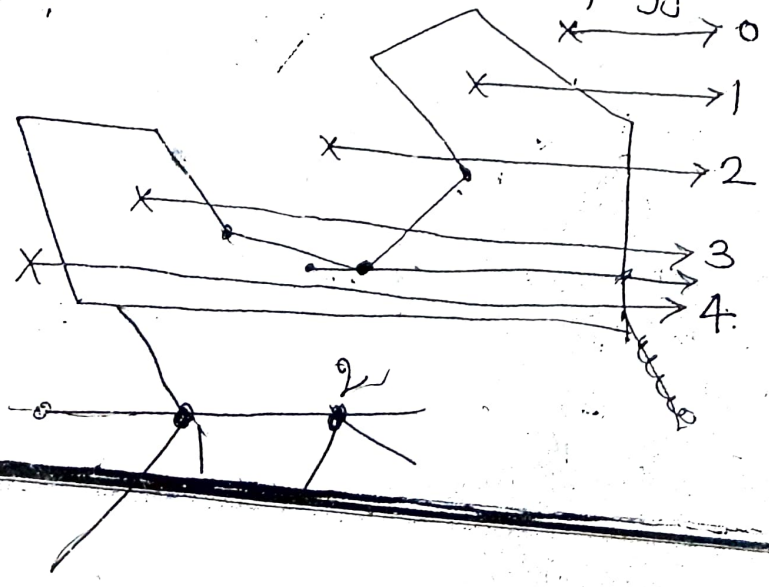
1) Odd-Even Test:

Consider a polygon made up of N vertices (x_i, y_i) where i ranges from 0 to N-1. The last vertex is assumed to be the same as first vertex i.e. the polygon is closed.

To determine status of a point (x_p, y_p) to be tested whether inside or outside, consider a horizontal ray (scanline) drawn from (x_p, y_p) and to the right.

a) If the number of times this ray intersects the line segments (edges of polygon) is even then the point is outside the polygon.

b) If the number of intersections are odd then the point (x_p, y_p) lies inside the polygon.

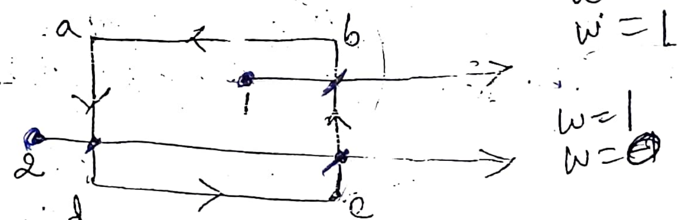


$w \neq 0 \Rightarrow$ interior
$w = 0 \Rightarrow$ exterior

2) Non-Zero Winding Number Test :-

- Counts the number of times the object winds around the point in the anti-clockwise direction.
- From the given point, again draw a straight line to infinity. Initialize winding number $w = 0$.
- For each edge crossing line from right to left add 1 i.e. $w = w + 1$.
- For each edge crossing the line from left to right subtract 1 i.e. $w = w - 1$.
- If $w \neq 0$ then the point is an interior point otherwise it is exterior.

eg



cb - R-L - $w = w+1 = 1$
ad C-R - $w = w-1 = 0$

For point 1, it intersects only one edge (bc) which is moving from right to left.

$w = 0$ earlier and

now $w = w + 1$

$\therefore \boxed{w = 1}$

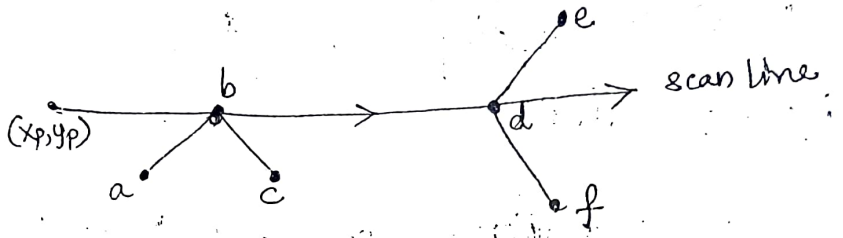
As $w \neq 0$, the point is an interior point.

For Point 2, it intersects two edges. $w = 0$.
1st edge (ad) which moves from ~~right to left~~ left to right $\therefore w = w - 1$ $\therefore \boxed{w = -1}$

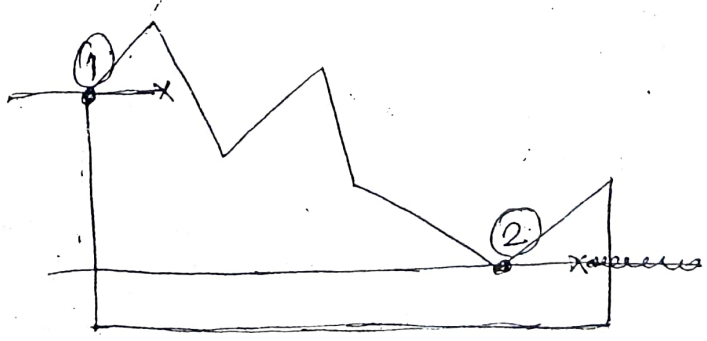
2nd edge (bc) which moves from right to left
$\therefore w = w + 1 = -1 + 1 = 0$ $\therefore \boxed{w = 0}$
if $w = 0$, then it is an exterior point.

Zero intersection will be considered as even. The test for even or odd will be based on modulus 2, ie., if no. of intersections modulus 2 is 0 then the number is even, if it is 1 then it is odd.

What happens when edge or vertex of polygon lies on the ray from (x_p, y_p).
Possible situations are illustrated below.



Here, edge (b d) of polygon is ignored as it lies on the scanline. When intersection point is vertex of polygon, here b and d, then we have to look at other end points of two segments which meet at this vertex.
If these points lie in same side of the scanline, then a point counts as even number of intersections (ie. 2).
If these points lie on opposite sides of scanline, then a point is counted as a single intersection (ie. odd & 1). It is illustrated in fig. below.



* Polygon Filling :-

Filling a polygon means highlighting all the pixels which lie inside the polygon with any color other than background color.

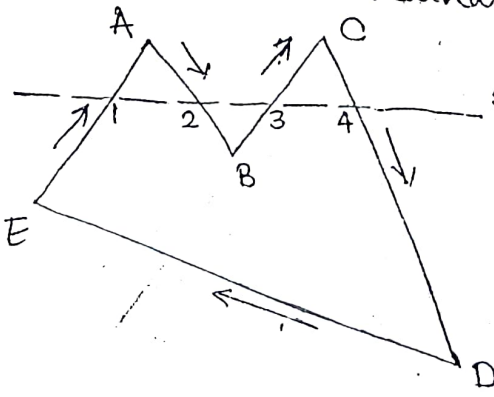
Scan-line Polygon fill Algorithm :-

Scan-line polygon fill algorithm proceeds scanline by scanline. For each scanline crossing a polygon, the area fill algorithm locates the intersection points with the polygon edge. These intersection points are then sorted on increasing order of their x-values. The corresponding locations in frame buffer are set to the specified fill color.

Algorithm :-

Step 1 :

Accept the number of vertices in the polygon and the co-ordinates of vertices.



Step 2 :

Active Edge List (AEL) is prepared for each scanline intersecting the polygon that contains the list of edges intersecting the scan line.

Div C3 & C4

for eg :-

AEL →

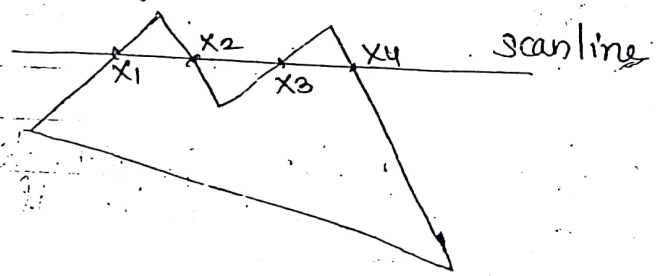
AB
BC
CD
EA

Step 3 :

For each polygon edge in AEL, the intersecting points are calculated.

Step 4 :

Sort all the edges intersecting points in increasing order of x-values.



x1	left
x2	right
x3	left
x4	right

The first x value gives left boundary and second x value gives right boundary.

Step 5 :

Fill all the pixel positions between left and right boundary with the specified color value.

Step 6 :

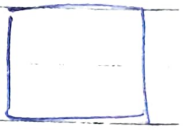
Repeat the steps (2) through (5) for all the scanlines.

3) Flood fill Algorithm :-

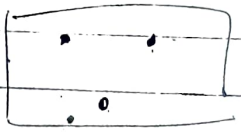
If boundary is defined by several different color regions then we can point such areas by replacing specified interior color to fill color; instead of searching boundary color. This approach is called as flood-fill algorithm.

Flood fill algorithm :-

Step 1 : Start
Step 2 : Read any seed point position (x, y)
Step 3 : Check to see if pixel (x, y) has old interior color.
If old color, then set it to new fill color
Step 4 : Repeat steps 3 & 4 for 4 neighbouring pixel of (x, y)



Flood fill Algo. accepts
① Co-ordinate of interior point (x, y)
② fill-color
③ old interior color.



Program for flood-fill Algorithm :-

```
#include <stdio.h>
#include <graphics.h>
main()
{
int gd=DETECT, gm;
initgraph(&gd, &gm, "C:\TURBOC3\BGI");
rectangle (100, 100, 200, 200);
flood (105, 105, 4, 15);
getch();
closegraph();
}
```

line (100, 100, 300, 300);
line (300, 100, 150, 300)
line (100, 100, 150, 300)

```
flood (int x, int y, int fore_color, int back_color).
{
  if (getpixel(x,y) != back_col && getpixel(x,y) != fore_col)
  {
    putpixel (x, y, fore_col);
    flood (x+1, y, fore_col, back_col);
    flood (x-1, y, fore_col, back_col);
    flood (x, y+1, fore_col, back_col);
    flood (x, y-1, fore_col, back_col);
  }
}.
```

8-way connected.

(x+1, y+1)
(x-1, y-1)
(x+1, y-1)
(x-1, y+1).

2) Boundary fill Algorithm :-

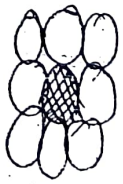
Boundary fill Algorithm starts at a point inside a region and paint the interior portion towards the boundary.

If the boundary is specified in a single color the algorithm proceeds outward pixel by pixel until the boundary color is encountered.

A Boundary fill algorithm accepts ——
1) Co-ordinate of an interior point (x, y)
2) a fill color
3) a boundary color.

Starting from (x, y), the algorithm tests the neighbouring pixels to determine if they are of the boundary color.
If not, they are painted with the fill color and their neighbours are tested. This process continues until all pixels upto the boundary color for the area are tested.
Neighbouring pixels are tested using either 4-connected or 8-connected method as shown below.

four-connected.



eight-connected

(x, y)

4 Pages

(1)

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

34159

To be filled in by the candidate

Subject :

Sr. No.

Course : CM / IF / EJ / EX

Year : I / II / III

Roll No. :

Date :

Signature of the Invigilator

Main Answer book	No. of Supplments	Total
1		

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)

Ch.3 TRANSFORMATION.

In many applications, there is a need for storing or manipulating displays. A graphic system allows the programmers to define the pictures that include variety of transformation.

Basic transformations are :-

1) Translation
2) Scaling
3) Rotation.

There are two complementary points of view for describing the object movements.

1) Geometric transformation :-

In this case, object is itself moved relative to a stationary co-ordinate system or background.

eg. Motion of an automobile against scenic background.

2) Co-ordinate Transformation :-

Background is moved relative to

stationary object.
eg. Car-race in video-games.

* Transformation :-

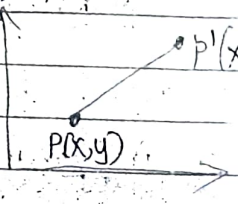
Transformation is the procedure for calculating new co-ordinate position for the points, as required by a specific change in size and orientation for the objects.

1) Translation :-

Movement of an object from one position to another.

It repositions object along a straight line path from one co-ordinate location to another.

Consider a point P(x, y) in two-dimensional co-ordinate system. By adding translation distance tx and ty, point P can be moved to a new position P'(x', y').

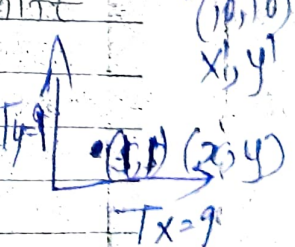


$\therefore x' = x + tx$

$y' = y + ty$

The translation distance pair (tx, ty) is called as translation vector or shift vector.

Let. $P = \begin{bmatrix} x \\ y \end{bmatrix}$ $T = \begin{bmatrix} tx \\ ty \end{bmatrix}$



then. $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} + \begin{bmatrix} tx \\ ty \\ 1 \end{bmatrix}$$

3x3

$(x, y) \rightarrow (xh, yh, h) \rightarrow (x_0h, y_0h, h)$

$x = \frac{xh}{h}$ $\quad y = \frac{yh}{h}$ $\quad (x, y, 1)$

$xh = x_0h$ $\quad yh = y_0h$

5

4 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

$(x, y) \rightarrow (x, y, 1)$

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

34150

To be filled in by the candidate

Sr. No. 2

Subject :

Course : CM / IF / EJ / EX

Year : I / II / III

Roll No. :

Date :

Signature of the Invigilator

Main Answer book	No. of Suppliments	Total
1		

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)

Ir homogeneous co-ordinate representation,

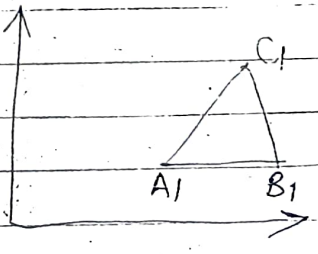
$$\rightarrow R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ Anticlockwise ✓

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

T: Rotate A Polygon :-

Consider a polygon $A_1B_1C_1$.

I. Initial position of Polygon $A_1B_1C_1$.



II. Rotate A_1 to A_2
B_1 to B_2
C_1 to C_2 by an angle θ
such that,

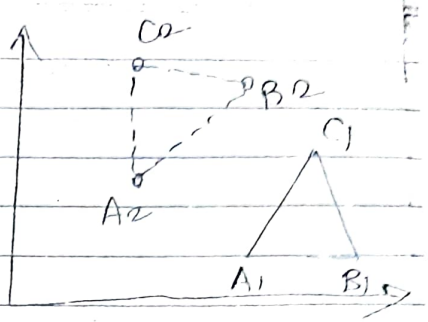
$A_2 = R \cdot A_1$

$B_2 = R \cdot B_1$

$C_2 = R \cdot C_1$

where R is rotation matrix.

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



III. Draw the polygon $A_2 B_2 C_2$

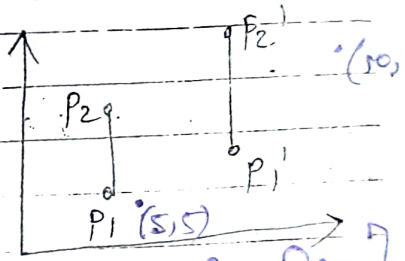
3) Scaling :-

A scaling transformation alters the size of an object.

Consider a line segment P_1P_2.
Scaling operation is obtained by multiplying values (x, y) of each point by scaling factors S_x and S_y to produce transformed co-ordinates such that,

$x' = S_x \cdot x$

$y' = S_y \cdot y$



$S_x = S_y =$

where,
S_x = scaling factor in x direction.
S_y = scaling factor in y direction.

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

Then,

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

$P_1' = S \cdot P_1$ where $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$P_2' = S \cdot P_2$

$$P' = P + T$$

$$= \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

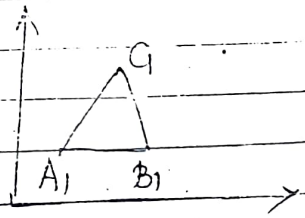
The homogeneous co-ordinate transformation matrix T is,

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

To Translate a polygon :-

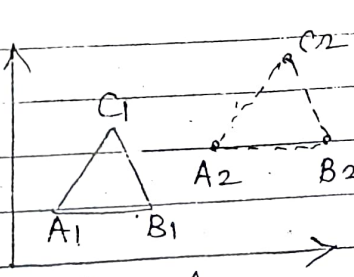
Consider a polygon $A_1B_1C_1$.

I. Initial position of polygon $A_1B_1C_1$



II. Translate A_1 to A_2
B_1 to B_2
C_1 to C_2

Such that $A_2 = T \cdot A_1$
$B_2 = T \cdot B_1$
$C_2 = T \cdot C_1$.



The required translation matrix is,

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

$\therefore A_2B_2C_2 = T \cdot A_1B_1C_1$.

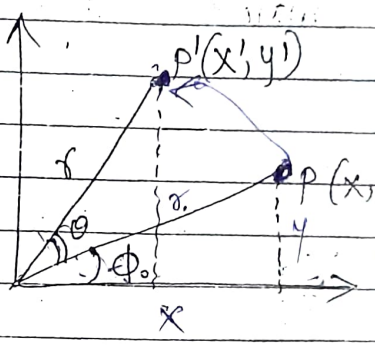
III. Draw polygon $A_2B_2C_2$.

3x1 3x3

2) Rotation :—

A rotation is applied to an object by repositioning it along a circular path.

Consider a point P(x, y) in a 2D Co-ordinate system.



$\cos\phi = \frac{x}{r}$

$\sin\phi = \frac{y}{r}$

$\therefore x = r \cdot \cos\phi$

$y = r \cdot \sin\phi$

Rotation of point P by an angle θ about origin in anticlockwise direction gives new position P'(x', y')

In polar co-ordinate system,

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

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

$x' = r \cdot \cos(\phi + \theta)$ $\quad r \cdot (\cos\phi\cos\theta - \sin\phi\sin\theta)$

$= r \cdot \cos\phi \cdot \cos\theta - r \sin\phi \cdot \sin\theta$

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

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

$= r \cdot \cos\phi \sin\theta + r \sin\phi \cos\theta$

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

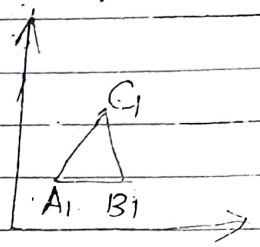
In matrix form;

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

To scale a polygon :-

Consider a polygon $A_1B_1C_1$.

I. Initial position of polygon $A_1B_1C_1$



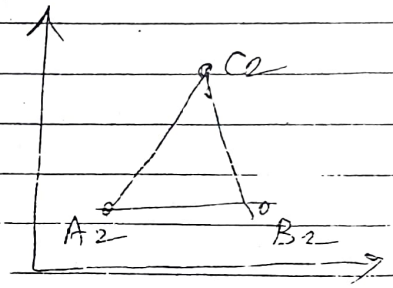
II. Scale A_1 to A_2
B_1 to B_2
C_1 to C_2

such that

$A_2 = S \cdot A_1$

$B_2 = S \cdot B_1$

$C_2 = S \cdot C_1$



where,

S is scaling matrix.

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

III. Draw the polygon $A_2B_2C_2$.

Example:- Scale the triangle $A_1B_1C_1$ for the following given values of Sx and Sy.

I. When $Sx = 1$, $Sy = 1$.

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

Object does not change.

II. When $Sx = 2$, $Sy = 1$,

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

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2x_1 \\ y_1 \\ 1 \end{bmatrix}$$

III. When $S_x = 1$, $S_y = 2$

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ 2y_1 \\ 1 \end{bmatrix}$$

I. When $S_x = 0.5$, $S_y = 0.5$

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5x_1 \\ 0.5y_1 \\ 1 \end{bmatrix}$$ Half its size

V. When $S_x = 2$, $S_y = 2$

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

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2y_1 \\ 1 \end{bmatrix}$$ Double its size

The following conclusions can be drawn:-

1) $S_x = S_y < 1$ then a) reduces the size of object.
   b) moves the object closer to origin

2) $S_x = S_y > 1$ then a) increases size of object.
   b) moves object away from the origin.

(9)

4 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

34181

To be filled in by the candidate

Subject :

Sr. No. (3)

Course : CM / IF / EJ / EX

Year : I / II / III

Roll No. :

Date :

Signature of the invigilator

Main Answer book	No. of Suppliments	Total
1		

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)

Examples :- (8 ums).

1) Translate a triangle ABC by 5 units in X direction where the co-ordinates are A(5,5), B(10,5) and C(10,10).

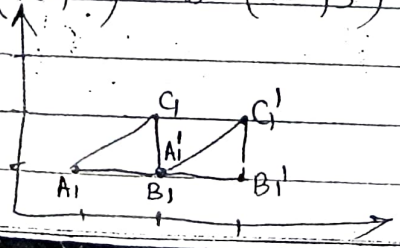
Sol^n : Let A'B'C' be the new triangle.

$\therefore$ A'B'C' = T. ABC. where T is translation matrix and $t_x = 5$ & $t_y = 0$.

$$\therefore A'B'C' = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} A \\ B \\ C \end{matrix} \begin{bmatrix} 5 & 10 & 10 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{matrix} A' & B' & C' \end{matrix} \begin{bmatrix} 10 & 15 & 15 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore$ A'(10,5) B'(15,5) & C'(15,10).



2) Scale the triangle ABC to reduce it to half of its size where co-ordinates of ABC are : A(5,5) B(10,5) C(10,10).

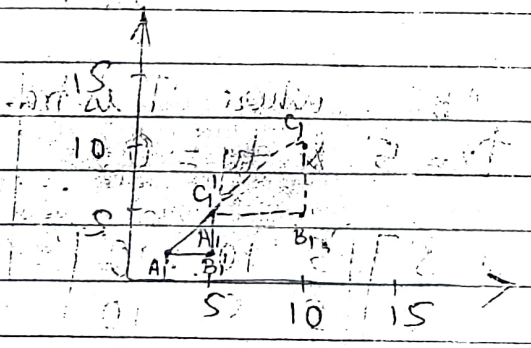
Soln :- let A'B'C' be the new triangle.
Scaling factors are
$S_x = 0.5$, $S_y = 0.5$.

$$\therefore A'B'C' = S \cdot ABC$$

$$= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 10 & 10 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.5 & 5 & 5 \\ 2.5 & 2.5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

A'(2.5, 2.5) B'(5, 2.5), C'(5, 5).



3) Scale the same triangle ABC to extend it to double its size only in x direction.

Soln :- let A'B'C' be the new triangle
Scaling factors are,
$S_x = 2$, $S_y = 1$

2) Scale the triangle ABC to reduce it to half of its size where co-ordinates of ABC are : A(5,5) B(10,5) C(10,10).

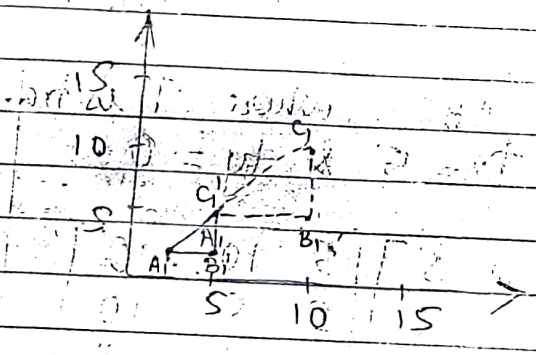
Soln :- let A'B'C' be the new triangle.
Scaling factors are.
$S_x = 0.5$, $S_y = 0.5$.

$$\therefore A'B'C' = S \cdot ABC$$

$$= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 10 & 10 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.5 & 5 & 5 \\ 2.5 & 2.5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

A'(2.5, 2.5) B'(5, 2.5), C'(5, 5).



3) Scale the same triangle ABC to extend it to double its size only in x direction.

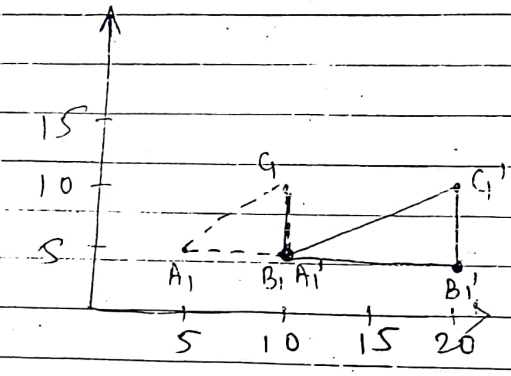
Soln :- let A'B'C' be the new triangle
Scaling factors are,
$S_x = 2$, $S_y = 1$

$$\therefore A'B'C' = S \cdot ABC$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 10 & 10 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 20 & 20 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore A'(10,5)$ $B'(20,5)$ $C'(20,10)$.



4) Rotate a line AB in anticlockwise direction for 45° angle about origin where. A(5,5) and B(20,5).

Soln :- let A'B' be the new line after rotation..

$$\therefore A'B' = R \cdot AB.$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} A \\ \\ \\ \end{matrix} \begin{matrix} B \\ \\ \\ \end{matrix} \begin{bmatrix} 5 & 20 \\ 5 & 5 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 20 \\ 5 & 5 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 20 \\ 5 & 5 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 15/\sqrt{2} \\ 10/\sqrt{2} & 40/\sqrt{2} \\ 1 & 1 \end{bmatrix}$$

$\therefore A'(0, 10/\sqrt{2})$ & $B'(15/\sqrt{2}, 40/\sqrt{2})$.

5) A rectangle A(2,2), B(5,2), C(5,3), D(2,3) is rotated by 90° about origin in the anticlockwise direction. Find new co-ordinates of rectangle after rotation.

→ Let A'B'C'D' be the new co-ordinates of rectangle.

$$A'B'C'D' = R \cdot ABCD$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} A & B & C & D \\ \end{matrix} \begin{bmatrix} 2 & 5 & 5 & 2 \\ 2 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 90(0) & -\sin 90(+1) & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 5 & 2 \\ 2 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

$\therefore A'(-2, 2)$ $B'(-2, 5)$, $C'(-3, 5)$ $D'(-3, 2)$

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

13

4 Pages

Subject :

To be filled in by the candidate

34162

Course : CM / IF / EJ / EX

Sr. No. 4

Year : I / II / III

Date

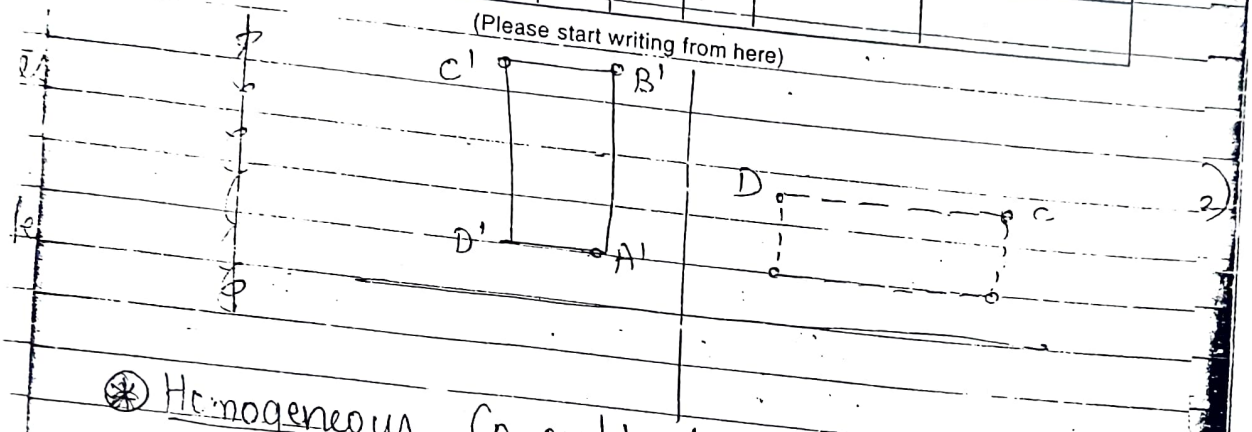
Roll No.:

Main Answer book	No. of Supplments	Total
1		

Signature of the Invigilator

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)



✱ Homogeneous Co-ordinates :-

To express any two dimensional transformation as a matrix multiplication, each cartesian co-ordinate position (x, y) is represented with the homogeneous co-ordinate tripple (xh, yh, h) where,

$$X = \frac{x'h}{h} \quad , \quad Y = \frac{yh}{h}$$

Thus, a general homogeneous co-ordinate representation can be written as $(x.h, y.h, h)$. The homogeneous parameter h may be any non-zero value. Better choice for h is ~~For~~ each two dimensional co-ordinate

is then represented with homogeneous co-ordinates $(x, y, 1)$.

Expressing the co-ordinates in homogeneous co-ordinate values allows us to represent all geometric transformation equations as matrix multiplications.

For translation,

$$P' = T \cdot P$$

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

For rotation,

$$P' = R \cdot P$$

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

For Scaling,

$$P' = S \cdot P$$

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

* Composite transformation :-

If the one or more transformations are performed ; then it is termed as composite transformation.

1) Two successive Translation vectors (tx_1, ty_1) and (tx_2, ty_2) are applied to co-ordinate position P.

then,

$$P' = T_2 \cdot T_1 \cdot P$$
$$= T_M \cdot P \quad \text{where } T_M = T_2 \cdot T_1$$

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

$$= \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix} \quad T_M = T(tx_1 + tx_2, ty_1 + ty_2)$$

T_M is composite transformation matrix

2) Two successive rotation.

$$P' = R(\theta_2) \cdot R(\theta_1) \cdot P$$
$$= T_M \cdot P \quad \text{where } T_M = R(\theta_2) \cdot R(\theta_1)$$

$$T_M = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\therefore T_M = R(\theta_1 + \theta_2)$.

3) Two successive scaling.

$$P' = S_2 \cdot S_1 \cdot P$$
$$= T_M \cdot P \quad \text{where } T_M = S_2 \cdot S_1$$

$$\therefore Tm = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore Tm = S(Sx_1 \cdot Sx_2, Sy_1 \cdot Sy_2)$$

* Matrix Representations :-

Every graphics applications involve the sequence of geometric transformation.

Consider a picture construction application where the object is to be translated, rotated and then scaled to fit the picture components into their proper positions.

To produce a sequence of transformations, we have to calculate the transformed co-ordinates one step at a time. First, co-ordinate positions are scaled, then these scaled co-ordinates are rotated and finally the rotated co-ordinates are translated.

For Point $P(x, y)$ on object we get $P'(x', y')$ such that,

$$x' = x + tx$$

$$y' = y + ty$$

17

4 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

34163

To be filled in by the candidate

Subject :

Sr. No. 5

Course : CM / IF / EJ / EX

Year : I / II / III

Roll No.:

Date :

Signature of the Invigilator

Main Answer book	No. of Suppliments	Total
1		

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)

P'(x', y') will get rotated to P''(x'', y'') such that

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

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

P''(x'', y'') will get scaled to P'''(x''', y''') such that,

$x''' = S_x \cdot x''$

$y''' = S_y \cdot y''$

P''' is the final position of point P.

Matrix Method eliminates the calculation of intermediate values.

All transformation equations can be expressed as matrix multiplication.

i.e. $P' = T \cdot P$ — (1)

$P'' = R \cdot P'$ — (2)

$P''' = S \cdot P''$ — (3)

where T = translation, R = Rotation, S = Scaling

Substituting P'' in eq (3),

$P''' = S \cdot R \cdot P'$.

Now, substituting P', it becomes

$$P''' = S \cdot R \cdot T \cdot P.$$

$$P''' = ((S \cdot R) \cdot T) \cdot P.$$

$$= T_M \cdot P.$$

where

$T_M = S \cdot R \cdot T$ is the composite transformation matrix.

=

* Other Transformations:-

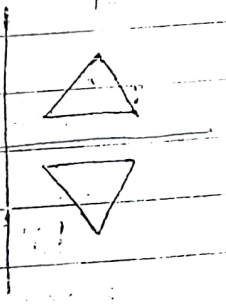
1) REFLECTION

Reflection is a transformation that produces a mirror image of an object relative to an axis of reflection.

1) Reflection about x-axis (ie. line y=0)

The matrix is

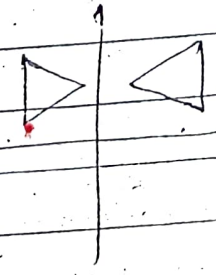
$$Rf_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



2) Reflection about y-axis

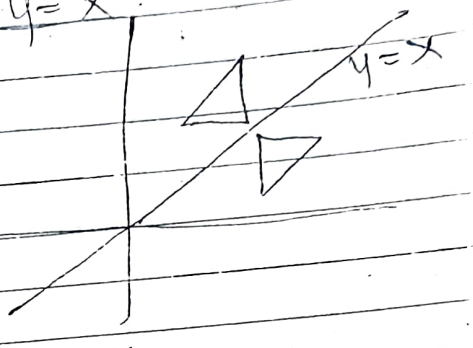
The matrix is

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



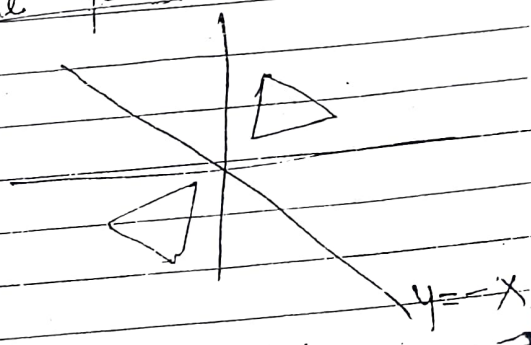
3) Reflection about line $y = x$
Matrix is,

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



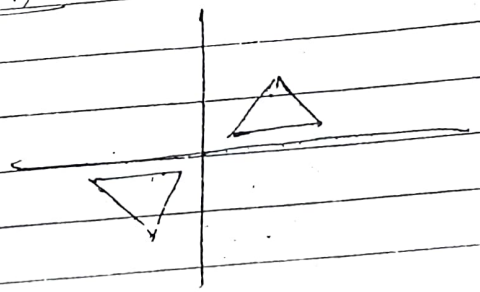
4) Reflection about line $y = -x$
matrix is,

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



5) Reflection about origin
matrix is,

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



2) SHEARING :—

A transformation that slants the shape of object is called as shear transformation.
Types of shear transformation are —
i) x- shear — it shifts x co-ord. values
ii) y- shear — it shifts y co-ord. values

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ Shx & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (20)$$

1) x-shear :-

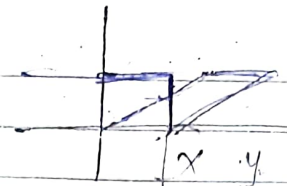
Matrix is,

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

where Shx is x-shear factor.

$x' = x + Shx \cdot y$

$y' = y$



2) y-shear :

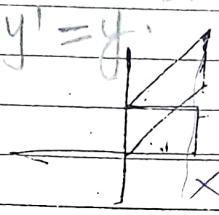
Matrix is,

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

where Shy is y-shear factor.

$x' = x$

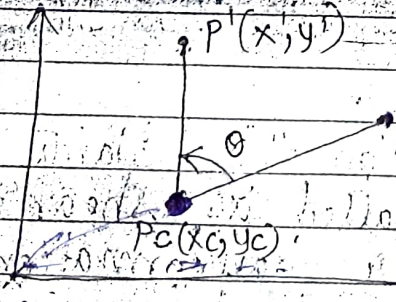
$y' = x \cdot Shy + y$



* Rotation about an arbitrary point :-

Consider a pt. P (x, y).

Rotation about Pc (xc, yc) by an angle θ in anticlockwise direction will move P(x, y) to P' (x', y') as shown below.



The required transformation is accomplished by performing the foll. sequence of operation

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

$x' = x + y \cdot Shx$

$y' = y$

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

$x' = x$

$y' = x \cdot Shy + y$

$x' = x \cdot Shx + y$

$y' = x + y \cdot Shy$

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

(21)

4 Pages

TERNA PUBLIC CHARITABLE TRUST'S

TERNA POLYTECHNIC

Sector-1, Koparkhairane, Navi Mumbai - 400 709.

PROGRESSIVE THEORY TEST I / II / 20.....

(6)

34164

To be filled in by the candidate

Sr. No.

Subject :

Course : CM / IF / EJ / EX

Year : I / II / III

Roll No. :

Date :

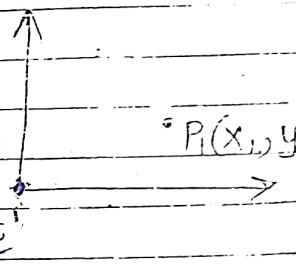
Signature of the Invigilator

Main Answer book	No. of Suppliments	Total
1		

Que No.	1	2	3	4	5	6	Total Marks	Sign. of Examiner
Marks								

(Please start writing from here)

I. Translate pt $P_C(x_c, y_c)$ to the origin. Correspondingly, find the new position of point P.



Let the new position of point P be P_1.

The required translation matrix is,

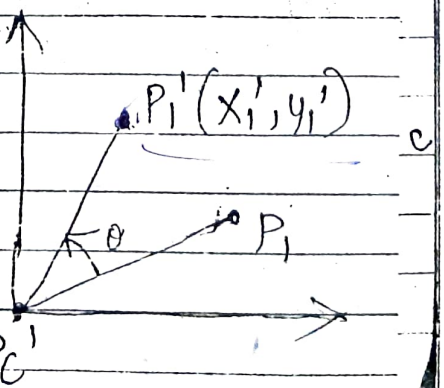
$$T_1 = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$

$P_C' = T_1 \cdot P_C$ moves P_C to origin.

Then

$P_1 = T_1 \cdot P$ moves pt. P to P_1 such that P_C moves to origin.

II. Rotate P_1 to P_1' by an angle θ in anti-clockwise direction.

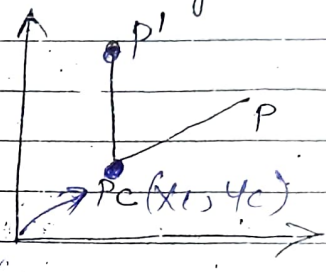


The required Rotation matrix is,

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

$\therefore P_1' = R \cdot P_1$

III. Translate the point Pc' to its original position (x_c, y_c)



Correspondingly, find new position of P_1'.

let the new position of P_1' be P'.

Required Translation matrix is,

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

$\therefore P' = T_2 \cdot P_1'$

$P_c = T_2 \cdot P_c'$

$T_2 \cdot R \cdot T_1 \cdot P$

$R(\theta) \cdot P$

TM.

The overall expression can be written as

$P' = T_2 \cdot R \cdot T_1 \cdot P$

$= R(\theta) \cdot P$

$P_1' = T_2 P_1'$

$= T_2 \cdot R P_1 = T_2 \cdot R \cdot T_1 P$

where $R(\theta)$ is composite transformation matrix,

$R''(\theta) = ((T_2 \cdot R) \cdot T_1)$