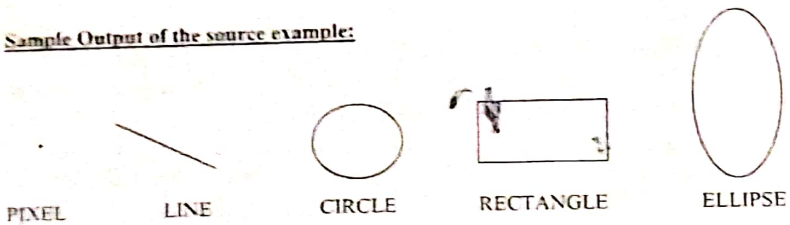
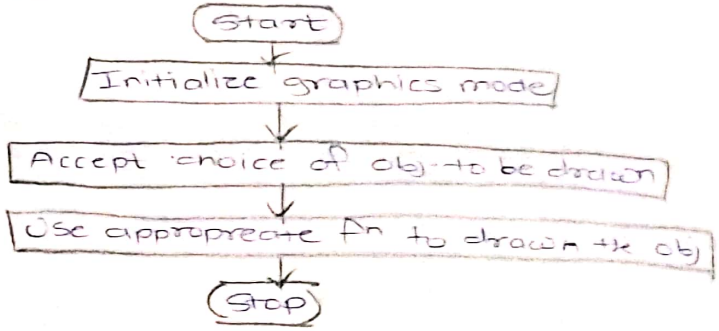
Sample Output of the source example:



VIII. Algorithm

Step 1: Start
Step 2: Declare and detect graphic drive
Step 3: Declare Variables
Step 4: Initialize graphics obj mode.
Step 5: Accept choice of object to be known.
Step 6: Select the proper fn to be the respect object.
Step 7: if(choice == pixel, use putpixel(int x, int y, int color)
Step 8: if (choice == liner, use fn line(int x, int y, int x, int y)
Step 9: if(choice == circle, use circle(int x, int y, int radius)
Step 10: if (choice == rectangle, use rectangle(int left, int top, int right, int bottom)
Step 11: if(choice == elipse, use elipse(int x, int y, strangle, end angle, x radius, y radius)
Step 12: Stop.

IX. Flow Chart



X. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XI. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in C to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety/ethical practices.

XII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) Core (TM) i3 RAM 8 GB
2	Software	Turbo C
3	Any other resource used	

XIII. Result (Output of the Program)

We understood & wrote programme to draw basic graphics objects

XIV. Conclusion(s)

In this practical we learnt that basic graphics mode function and how to draw basic graphic objects using graphics mode function.

XV. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Find the minimum and maximum coordinates of screen.
2. What is graphics driver and graphics mode?
3. What is path of graphics driver?
4. Find error of the following code

```
{
putpixel(20,20);
outtextxy(25,25,"PIXEL");
}
```

5. List four applications of Computer graphics.
6. Define terms pixel, line and raster scan graphics.

(Space for answers)

Q.2] Graphics driver :- This arguments Specifies the graphics driver to be used and it interfaces with display adapter. Some of available graphics drivers are CGA, EGA, VGA, etc.
Graphics mode :- Each graphic adapter can use several difference possible graphics mode. The most argument is used to select particular mode.

Q.3] To start the graphics System, first call the initgraph(); initgraph initializes the graphics system by loading a graphics driver from disk, and putting the system into graphics mode.
Syntax :- void initgraph(int *gd, int *gm, char path);

Q.4] Error :-

Q.5] Applications of Computer graphics as follows :-
i) Digital art, ii) Special effects, iii) Visual effects, iv) Video games, v) Computer aided design, vi) Medical imaging

Q.6] i) Pixel :- A pixel is the smallest addressable screen element
ii) Line :- A line is correction between two end points.
iii) Raster scan graphics :- In addition between two and the CPU a Special purpose processor called the video Controller or display controller is used to controll the operation of the display device.

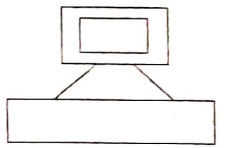
XVI. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:

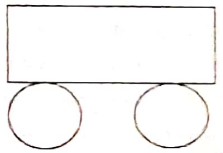
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C program to display following objects

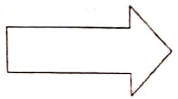
a.



b.



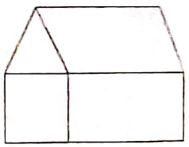
c.



d.

e.

f.



VIII. Algorithm

Step 1: Start

Step 2: Read the coordinates of the two end points, i.e. (x_1, y_1) (x_2, y_2)

Step 3: Calculate dx & dy

dx = abs $(x_2 - x_1)$

dy = abs $(y_2 - y_1)$

Step 4: Calculate length

if (dx >= dy)

length = dx;

else

length = dy;

Step 5: Calculate the increament factor

$\Delta x = (x_2 - x_1) /$ length

$\Delta y = (y_2 - y_1) /$ length

Step 6: Initialize the initial point on the line

x = x1 + 0.5 * sign (Δx)

y = y1 + 0.5 * sign (Δy)

Step 7: Initialize i to 1

while (i <= length)

{

plot (int (x) int (y))

x = x + Δx

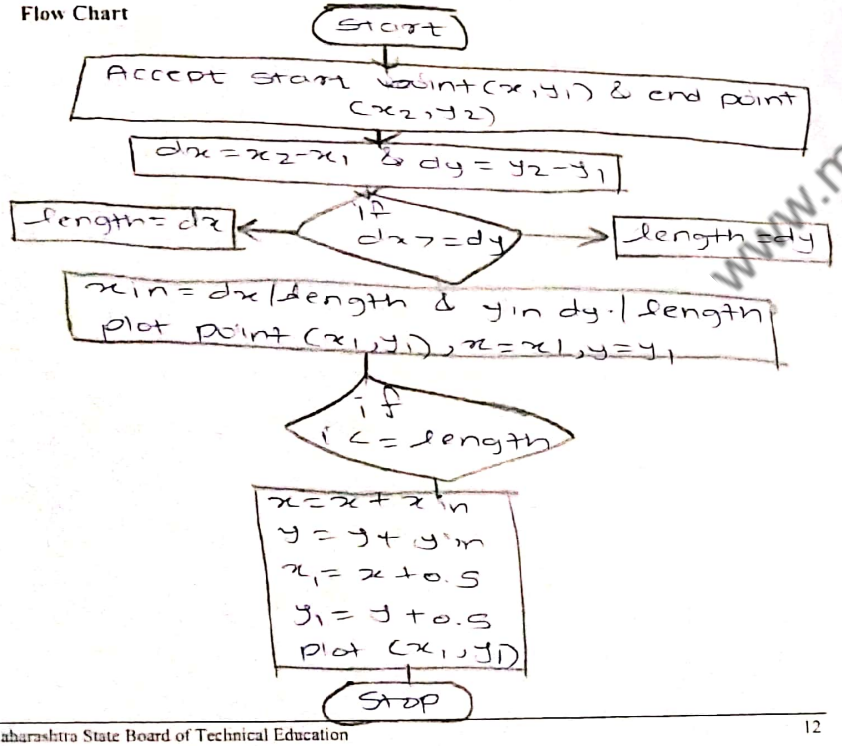
y = y + Δy

i = i + 1

}

Step 8: Stop.

IX. Flow Chart



X. 'C' Program Code

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main()
{
    float x,y,x1,y1,x2,y2,dx,dy,length;
    int i,gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    printf("Enter the value of x1 and y1 : ");
    scanf("%f%f",&x1,&y1);
    printf("\n Enter the value of x2 and y2 : ");
    scanf("%f%f",&x2,&y2);
    dx=abs(x2-x1);
    dy=abs(y2-y1);
    if(dx>=dy){
    length=dx;
    }
    else{
    length=dy;
    }
    dx=dx/length;
    dy=dy/length;
    x=x1;
    y=y1;
    i=1;
    while(i<=length)
    {
        putpixel(x,y);
        x=x+dx;
        y=y+dy;
        i=i+1;
    }
    getch();
    closegraph();
    }
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety/ethical practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i3 RAM 4 GB
2	Software	Turbo C
3	Any other resource used	

XIV. Result (Output of the Program)

We understood & wrote programme to draw line using DDA algorithm

XV. Conclusion(s)

In this practical we learnt that how to draw line using DDA line algorithm

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Define the term Rasterization.
2. Write slope intercept form of a line.
3. Write advantages and disadvantages of DDA algorithm.

(Space for Answers)

Q.1] The process of determining the appropriate pixels for representing pictures or object is called as the 'rasterization'.

Q.2] The Slope intercept form of ~~th~~ a line is

$y = mx + b$

Where 'y' is y cordinate 'm' is slope
'x' is x cordinate 'b' is y intercept

Q.3] Advantages & Disadvantages of DDA algorithm:-

Advantages	Disadvantages
1> It is simple type of algorithm.	1>Floating point arethmetic in DDA algorithm is time Consuming.
2> It is faster than the other methods	2> Accumulation of round off error in successive additions of floating point increament can cause the calculated pixel positions to drift away from the true line path for long line segments

XVII. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

i. Give following values for every iteration of DDA algorithm to draw a line from(3,4) to(6,8)

dx	dy	step	xin	yin

ii. Give following values for every iteration of DDA algorithm to draw a line from (-5,-5) to(-12,-12)

dx	dy	step	xin	yin

(Space for Answers)

Q.1] Step1: Read end points of line

$(x_1, y_1) = (3, 4)$

$(x_2, y_2) = (6, 8)$

Step2: Calculate dx & dy

$dx = abs(x_2 - x_1)$

$= abs(6 - 3)$

$dx = 3$

$dy = abs(y_2 - y_1)$

$= abs(8 - 4)$

$dy = 4$

Step3: if $(dx >= dy)$ AS $(3 >= 4)$

else

length = dy = 4

Step4: Increament factor:-

$\Delta x = (x_2 - x_1) / length$

$= (6 - 3) / 4$

$= 3.14$

$\Delta x = 0.75$

$\Delta y = (y_2 - y_1) / length$

$= (8 - 4) / 4$

$= 4/4$

$\Delta y = 1$

Step5: Initial point

$x = x_1 + 0.5 * sign(\Delta x)$

$= 3 + 0.5 * sign(0.75)$

$x = 3$

$y = y_1 + 0.5 * sign(\Delta y)$

$= 4 + 0.5 * sign(1)$

$= 4 + 0.5$

$y = 4.5$

Step6: Now, i=1

i) If i=1

plot(x,y) = plot(3,4)

$x = x + \Delta x = 3 + 0.75$

$x = 3.75$

$y = y + \Delta y = 4.5 + 1$

$y = 5.5$

ii) If i=2

plot(x,y) = plot(3,5)

$x = x + \Delta x = 3.75 + 0.75$

$x = 4.5$

$y = y + \Delta y = 5.5 + 1$

$y = 6.5$

iii) if i=3

plot(x,y) = plot(4,6)

$x = x + \Delta x$

$= 4.5 + 0.75$

$x = 5.25$

$y = y + \Delta y = 7.5 + 1$

$y = 8.5$

PAGE NO.
DATE / /

Value of (i)	plot pixel (x,y)	x	y
	plot (3,4)		
1	plot (3,5)	5.75	5.5
2	plot (4,6)	4.5	6.5
3	plot (5,7)	5.25	7.5
4	plot (6,8)	8.5	8.5

2] Step 1: Read end points of line.

$(x_1, y_1) = (-5, -5)$

$(x_2, y_2) = (-12, -12)$

Step 2: Calculate dx & dy

$dx = abs(x_2 - x_1)$

$= abs(-12 - (-5))$

$= abs(-12 + 5)$

$dx = 7$

$dy = abs(y_2 - y_1)$

$= abs(-12 - (-5))$

$= abs(-12 + 5)$

$dy = 7$

Practical no: 2

Name:- Sumit Tanaji Biranje.

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int x,y,x1,y1,x2,y2,dx,dy,length;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("enter the value of x1,y1 and x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
dx=x2-x1;
dy=y2-y1;
if(dx>=dy)
  {
      length=dx;
  }
else {
length=dy;
}
dx=dx/length;
dy=dy/length;
  x=x1;
  y=y1;
i=1;
while(i<=length)
  {
putpixel(x,y,15);
  x=x+dx;
  y=y+dy;
```

Practical No. 3: Program to draw line using Bresenham's algorithm.

I. Practical Significance

Bresenham's line algorithm is an algorithm that determines the points of an *n*-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is commonly used to draw line primitives in a bitmap image.

II. Relevant Program Outcomes (POs):

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problems.
- **Discipline knowledge:** Apply Computer Programming knowledge to solve broad-based Computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering related problems.
- **Engineering tools:** Apply relevant Computer programming technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III. Competency and Practical skills

This practical is expect to develop the following skills in you

Develop 'C' programs to draw basic graphics objects:

1. Write syntax for graphics functions.
2. Write and Save a simple C program.
3. Setup graphics drivers, graphics mode and directory to run graphics program.
4. Compile the C program using Turbo C.
5. Debug and execute the program.

IV. Relevant Course Outcome(s):

- Implement standard algorithms to draw various graphics objects using C program.

V. Practical Outcome (POs):

a) Implement Bresenham's algorithm to draw line.

VI. Relevant affective domain related Outcome(s)

a. Handle command prompt environment.
b. Experiment with graphics environment.
c. Follow safety/ethical practices.

VII. Minimum Theoretical Background

Bresenham's line algorithm

Bresenham's line algorithm determines the points of an *n*-dimensional raster that is selected to form a close approximation to a straight line between two points. It is used to draw line primitives in a bitmap image , as it uses only integer addition, subtraction and bit shifting. It is an incremental error algorithm. It is one of the earliest algorithms developed in the field of Computer graphics.

Consider a line with initial point (x1, y1) and terminal point (x2, y2) in device space. If $\Delta x = x2 - x1$ and $\Delta y = y2 - y1$, we define the driving axis (DA) to be the x-axis if $|\Delta x| \geq |\Delta y|$, and the y-axis if $|\Delta y| > |\Delta x|$. The DA is used as the "axis of control" for the algorithm and is the axis of maximum movement. Within the main loop of the algorithm, the coordinate corresponding to the DA is incremented by one unit. The coordinate corresponding to the other axis (usually denoted the passive axis or PA) is only incremented as needed.

Procedure

1. Input the two line end-points, storing the left end-point in (x_0, y_0)
2. Plot the point (x_0, y_0)
3. Calculate the constants Δx, Δy, $2\Delta y$, and $(2\Delta y - 2\Delta x)$ and get the first value for the decision parameter as: $P_0 = 2\Delta y - 2\Delta x$
4. At each x_k along the line, starting at $k=0$, perform the following test:
   If $p_k < 0$, the next point to plot is (x_k+1, y_k) and $p_{k+1} = p_k + 2\Delta y$
   Otherwise, the next point to plot is (x_k+1, y_k+1) and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Repeat step 4 (Δx) times.

Sample Output of the source example:

VIII. Algorithm

Step 1 :- Start

Step 2 :- Read the end points (x_1, y_1) & (x_2, y_2)

Step 3 :- Calculate

$\Delta x = |x_2 - x_1|$

$\Delta y = |y_2 - y_1|$

Step 4 : Initialize starting point

$x = x_1$

$y = y_1$

Step 5 : plot (x, y);

Step 6 : Initial value of decision parameter P_k,

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

Step 7 :- if $(P_k < 0)$ {

$x = x + 1$;

$y = y$;

$P_k = P_k + 2\Delta y$; }

if $(P_k >= 0)$ {

$x = x + 1$;

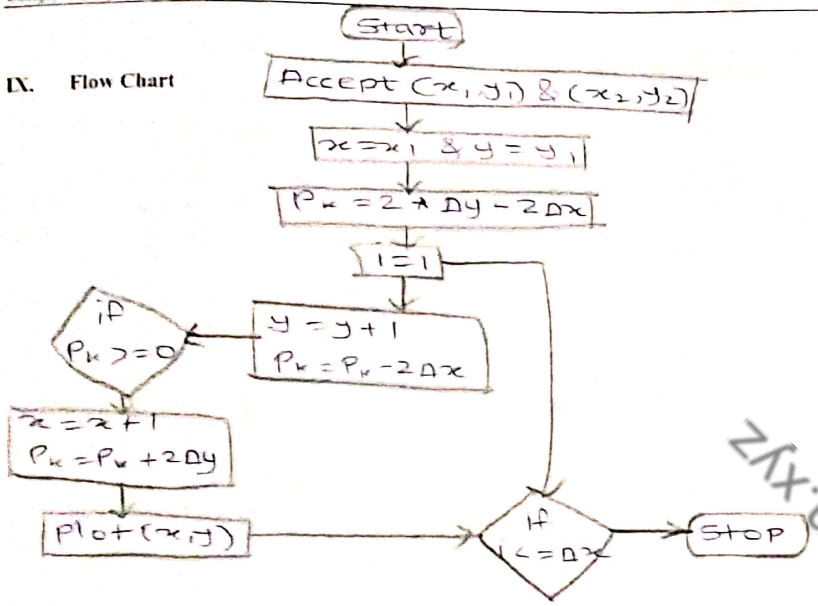
$y = y + 1$;

$P_k = P_k + 2\Delta y - 2\Delta x$; }

Plot (x, y)

Step 8 : Stop.

IX. Flow Chart



X. 'C' Program Code

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void drawline(int x1, int y1, int x2, int y2)
{
    int dx, dy, P, x, y;
    dx = x2 - x1;
    dy = y2 - y1;
    x = x1;
    y = y1;
    P = 2 * dy - dx;
    while(x < x1)
    {
        if(P >= 0)
        {
            putpixel(x, y, 7);
            y = y + 1;
            P = P + 2 * dy - 2 * dx;
        }
```


XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;)
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety/ethical practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i3 RAM - 4 GB
2	Software	Turbo C
3	Any other resource used	

XIV. Result (Output of the Program)

We understood & wrote programe to draw line using Bresenham's algorithm

XV. Conclusion(s)

In this practical we learnt that how to draw line using bresenhams line algorithm

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Explain the term decision parameter.
2. Write advantages of Bresenham's algorithm over DDA.

(Space for Answers)

Q.1] Decision parameter is a criteria based on which further calculation made.
for eg. :-
In Bresenham algorithm, if decision factor is > 0 then one set of actions takes place and if decision factor is <= 0 then same other steps are to followed.

Q.2] Following are the advantages of Bresenham's algorithm over DDA :-
i) This algorithm is faster than DDA
ii) Uses only addition (+) & substraction (-) operations.
iii) No error component is introduced as like in DDA.

Practical no: 3

Name: Sumit Tanaji Biranje.

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

void main()
{
intgd=DETECT,gm, x1=100,y1=100,x2=300,y2=300,dx,dy,x,y,pk,i;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

dx=(x2-x1);
dy=(y2-y1);

  x=x1;
  y=y1;

pk=2*dy-dx;
i=1;
while(i<=dx)
  {
putpixel(x,y,15);
if(pk<0)
  {
    x=x1+1;
    y=y1;
pk=pk+2*dy;
  }

else
  {
```

XVII. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

i. Give following values for every iteration of Bresenham's algorithm to draw a line from(3,4) to(6,8)

Δx	Δy	2Δx	2Δy	P

ii. Give following values for every iteration of Bresenham's algorithm to draw a line from(-6,-6) to(-14,-14)

Δx	Δy	2Δx	2Δy	P

(Space for Answers)

Q.1] Step 1: $x_1 = 3$ $x_2 = 6$

$y_1 = 4$ $y_2 = 8$

Step 2: $\Delta x = |x_2 - x_1| = |6 - 3| = 3$

$\Delta x = 3$

$\Delta y = |y_2 - y_1| = |8 - 4| = 4$

$\Delta y = 4$

Step 3: $x = x_1 = 3$

$y = y_1 = 4$

Step 4: plot (x, y)

i.e. plot (3, 4) Initial point

Step 5: Value of decision parameter

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

$= 2(4) - 3$

$= 8 - 3$

$P_k = 5$

PAGE NO.:
DATE : / /

Step 6 : Now i=1

i) As, Pk = 0.5

if (Pk >= 0)

x = x + 1 = 3 + 1 = 4

y = y + 1 = 4 + 1 = 5

Pk = Pk + 2Δy − 2Δx

= 5 + (2 × 4) − (2 × 3)

Pk = 7

ii) As, Pk = 7

if (Pk >= 0)

x = x + 1 = 4 + 1 = 5

y = y + 1 = 5 + 1 = 6

Pk = Pk + 2Δy − 2Δx

= 7 + (2 × 4) − (2 × 3)

= 7 + 8 − 6

Pk = 9

iii) As, Pk = 9

if (Pk >= 0)

x = x + 1 = 5 + 1 = 6

y = y + 1 = 6 + 1 = 7

Pk = Pk + 2Δy − 2Δx

= 9 + (2 × 4) − (2 × 3)

= 9 + 8 − 6

Pk = 11

PAGE NO.:
DATE : / /

Value of (i)	plot pixel (x, y)	x	y	Pk
		3	4	5
1	(3,4)	4	5	7
2	(4,5)	5	6	9
3	(5,6)	6	7	11

Q.2] Step1: $x_1 = -6$ $x_2 = -14$

$y_1 = -6$ $y_2 = -14$

Step 2: $\Delta x = |x_2 - x_1| = |-14 - (-6)| = 8$

$\Delta x = 8$

$\Delta y = |y_2 - y_1| = |-14 - (-6)|$

$\Delta y = 8$

Step 3: $x = x_1 = -6$

$y = y_1 = -6$

Step 4: plot (x, y)

i.e. plot (-6, -6) initial point

Step 5: Value of decision parameter

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

$= 2(8) - 8$

$= 16 - 8$

$P_k = 8$

Step 6: Now i = 1.

i) As $P_k = 8$

if ($P_k >= 0$)

$x = x + 1 = -6 + 1 = -5$

$y = y + 1 = -6 + 1 = -5$

$P_k = P_k + 2\Delta y - 2\Delta x$

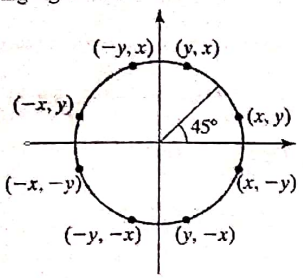
$= 8 + (2 \times 8) - (2 \times 8)$

$= 8 + 16 - 16$

$P_k = 8$

VII. Minimum Theoretical Background

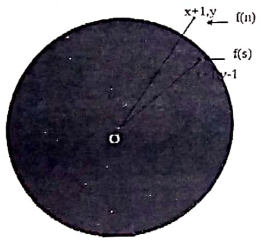
The equation of circle is $x^2+y^2=r^2$, where (x, y) are coordinates of a center and r is radius. To draw circle in computer graphics we use Bresenham's circle drawing algorithm and Mid Point circle drawing algorithm.



It's not easy to display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.

Bresenham's Circle algorithm:

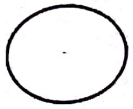
Bresenham's algorithm is based on the idea of determining the subsequent pixels required to draw the circle. x+1,y is to fins the next pixel to draw the circle.



Procedure:

1. Set initial values of center of the circle (xc, yc) and (x, y)
2. Set decision parameter d to d = 3 – (2 * r).
3. Repeat steps 4 to 8 until x <= y
4. Call drawCircle(int xc, int yc, int x, int y) function.
5. Increment value of x.
6. If d < 0, set d = d + (4*x) + 6
7. Else, set d = d + 4 * (x – y) + 10 and decrement y by 1.
8. Call drawCircle(int xc, int yc, int x, int y) function.

Sample Output of the source example:



VIII. Algorithm

Step 1 : Start
Step 2 : Declare and ~~D~~etect graphics ~~mode~~drive
Step 3 : Calculate initial decision variable $Pi = 3 - 2 * r$
Step 4 : $x = 0$ & $y = r$
Step 5 : if $(Pi < 0)$
$x = x + 1$
$Pi = Pi + 4x + 6$
else if $(Pi >= 0)$
$y = y - 1$
$Pi = Pi + 4(x - y) + 10$
Step 6 : Plot pixels in all octants as
Plot (x, y)
Plot (y, x)
Plot (−y, x)
Plot (−x, y)
Plot (−x, −y)
Plot (−y, −x)
Plot (x, −y)
Plot (y, −x)
Step 7 : Stop

IX. Flow Chart

Start

Initialize graphics mode

Declare xc, yc, radius R, P

Accept xc, yc & R

$P = 1 - R$, $x = 0$, $y = R$

$x <= y$

$P < 0$

$x = x + 1$
$y = y$
$P = P + 2x + 3$

End

$P > 0$

$x = x + 1$
$y = y - 1$
$P = P + 2x - 2y + 5$

X. 'C' Program Code

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\TC\\BGI");
        int x,y,xc,yc,r,d;
        clrscr();
        Printf("Enter the radius of circle:");
        Scanf("%d",&r);
        Printf("Enter circle of circle:");
        Scanf("%d%d",&xc,&yc);
        d=3-2*r;
        x=0;
        y=r;
        while(x<=y)
        Put pixel(xc+x,yc+y,RED);
        Put pixel(xc-y,yc-x,RED);
        getch(); Printf("Enter the x & y position for circle");
        closegraph(); Scanf("%d%d",&x,&y);
        Printf("\n Enter radius");
        Scanf("%d",&radius);
        circle(x,y radius);
        flood Fill(x,y,white);
}
```

```
delay(100);
getch();
closegraph();
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i3 RAM -4GB
2	Software	Turbo C
3	Any other resource used	

XIV. Result (Output of the Program)

We understood & wrote programe to draw circle

XV. Conclusion(s)

In this experiment we understood the how to draw circle using Bresenham's circle algorithm.

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write equation of a circle.
2. Write the algorithm to draw 8-way symmetry of a circle.
3. How the value of decision parameter (d) is calculated.

(Space for Answers)

Q.1] The equation of a circle.

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

Q.2] Algorithm to draw 8-way symmetry of a circle.

Step 1: Get the coordinates of the center of the circle & radius & store them in x, y, & R respectively. Set $p=0$ & $a=R$

Step 2: Set decision parameter $D=3-2R$

Step 3: Repeat through Step 8 while plus than equal to R.

Step 4: Call draw circle (x, y, P, Q)
Step 5: Increament the Value of P
Step 6: If D less than 0 then D=D+4P+6
Step 7: else set P=P-1, D=D+4(P-Q)+10
Step 8: call draw circle (x, y, P, Q)

Q.3] The value of decision parameter is calculated by
d=3-(2*r)

XVII. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Calculate pixels for a circle with radius 10 using Bresenham's algorithm.

d	x	y	Plot(x,y)

2. Draw a circle with center (50,50) and radius 20 by using Bresenham's algorithm.

(Space for Answers)

Name:- Sumit Tanaji Biranje.

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
intgd= DETECT,gm,r=100,pi,x,y;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
 x=0;
 y=r;
pi=3-2*r;
do
 {
putpixel(200+x,200+y,15);
putpixel(200-x,200+y,15);
putpixel(200+x,200-y,15);
putpixel(200+y,200+x,15);
putpixel(200-x,200-y,15);
putpixel(200-y,200+x,15);
putpixel(200+y,200-x,15);
 putpixel(200-y,200-x,15);
if(pi<0)
  {
   x=x+1;
pi=pi+4*x+6;
  }
else
  {
   x=x+1;
```

VI. Relevant Affective domain related Outcome(s)

a. Experiment with graphics environment.
b. Follow safety/ethical practices.
c. Maintain tools and equipment.

VII. Minimum Theoretical Background

Flood Fill Algorithm

In Flood Fill algorithm we start with some seed and examine the neighboring pixels, however pixels are checked for a specified interior color instead of boundary color and is replaced by a new color. It can be done using 4 connected or 8 connected region method.

Procedure:

Flood-fill (node, target-color, replacement-color):
1. If target-color is equal to replacement-color, return.
2. If the color of node is not equal to target-color, return.
3. Set the color of node to replacement-color.
4. Perform Flood-fill (one step to the south of node, target-color, replacement-color).
Perform Flood-fill (one step to the north of node, target-color, replacement-color).
Perform Flood-fill (one step to the west of node, target-color, replacement-color).
Perform Flood-fill (one step to the east of node, target-color, replacement-color).
5. Return.

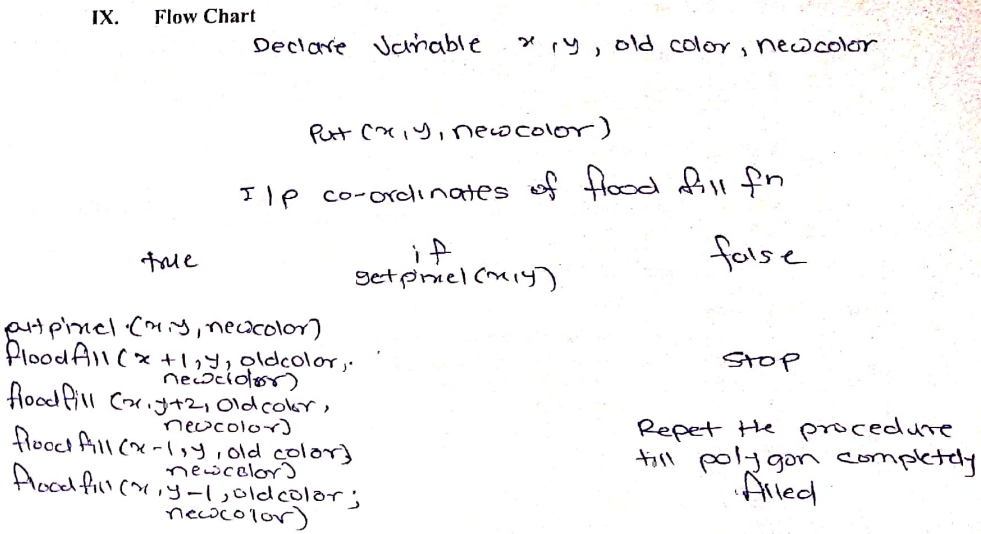
Example :
circle(100,100,50);
floodfill(100,100,RED);
Fills the circle with red colour.

VIII. Algorithm

Step 1: Start
Step 2: Declare and detect graphics driver
Step 3: plot (x,y,newcolor);
Step 4: Read x, y, radius.
Step 5: Input point for floodfill
Step 6: if (getpixel (x,y) = old color)
Step 7: putpixel (x,y, newcolor);
floodfill (x-1, y, oldcolor, newcolor);
floodfill (x-1, y, oldcolor, newcolor);
floodfill (x, y-1, oldcolor, newcolor);
Step 8: Repet until polygon is completely filled.
Step 9: Stop.

Start

IX. Flow Chart



X. 'C' Program Code

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void floodfill (int x, int y, int o, color, int newcolor)
{
        int gd=DETECT, gm;
        initgraph (&gd, &gm, "C:\\TC\\BGI");
        if (get pixel (x, y) == o, color)
        {
                putpixel (x, y, n.color)
                floodfill (x+2, y, n.color, ocolor);
                floodfill (x-1, y, n.color, o.color);
                floodfill (x, y-1, n.color, o.color);
        }
}
void main()
{
        interscr();
        getch();
        closegraph();
}
```

XI Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XI. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety practices.

XII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i3 RAM-4 GB
2	Software	Turbo C
3	Any other resource used	

XIII. Result (Output of the Program)

We understood & wrote programe to fill polygon

XIV. Conclusion(s)

In this practical we learnt that how to fill polygon ser using floodfill algorithm.

XV. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

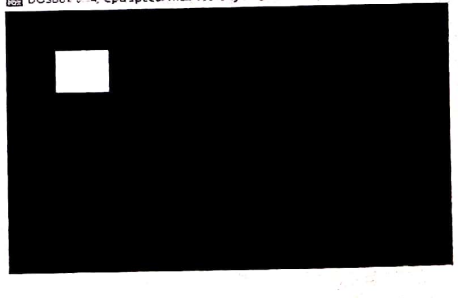
1. Define polygon.
2. Explain types of polygon.

Practical no: 5

Name:- Sumit Tanaji Biranje.

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
 rectangle(50,50,100,100);
 flood(55,55,15,4);
 getch();
 closegraph();
}
flood(x,y,foreground_col,background_col)
{
 if(getpixel(x,y)!=foreground_col&& getpixel(x,y)!=background_col)
 {
  putpixel(x,y,foreground_col);
   flood(x+1,y,foreground_col,background_col);
   flood(x-1,y,foreground_col,background_col);
   flood(x,y+1,foreground_col,background_col);
   flood(x,y-1,foreground_col,background_col);
   flood(x+1,y+1,foreground_col,background_col);
   flood(x-1,y-1,foreground_col,background_col);
   flood(x-1,y+1,foreground_col,background_col);
   flood(x+1,y-1,foreground_col,background_col);
}}
```



PAGE NO.:
DATE : / /

Q.4] List cordinates of neighbouring pixels in 4 connected method for seed pixel with cordinates (x,y)

The coordinates of neighbouring pixels in 4 connected method for seed pixel with coordinates (x,y) are: (x+1,y) (x-1,y)
(x,y+1) (x,y-1)

Q5] Inside - Outside test of polygon.

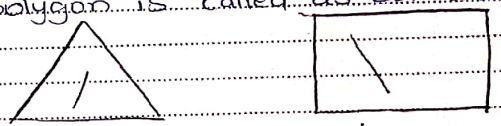
- Given a polygon & a point 'p' to find 'p' whether it iles inside the polygon are not. The points lying on the border are constidered as inside.
- The simple approach to check whether a point & extend it to inifinity.

3. List Coordinates of neighboring pixels in 8-connected method for seed pixel with coordinates(x,y)
4. List Coordinates of neighboring pixels in 4-connected method for seed pixel with coordinates(x,y)
5. Explain inside-outside test of polygon.

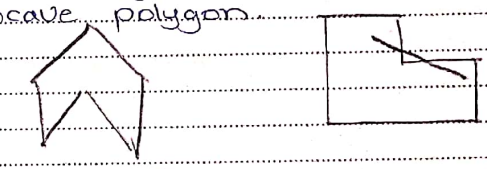
(Space for Answers)

Q.1] A closed plane figure made up of Serval line Segments that are joined together. The Sides do not close each other exactly two sides meet at every Vertex.

Q.2] A polygon in which the line segment joining any two points with polygon lies completely inside the polygon is called as convex polygon



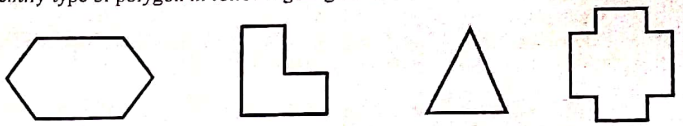
A polygon in which the line Segment joining any two points within polygon or may not lies completely inside the polygon is called as concave polygon.



Q.3] List coordinates of neighboring pixel in 8 connected method for seed pixel with coordinates (x,y)

The set of pixels of 4 neighbours of P and diagonal neighbours of p together are called as 8-neighbours of pixel: p(x,y) Co-ordinates of neighbouring pixels in 8-connected method for seed pixel with coordinates (x,y) are: (x+1,y), (x-1,y), (x,y+1), (x,y-1), (x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1)

2. Identify type of polygon in following diagrams.



3. Give the basic difference between flood fill and boundary fill.

(Space for Answers)

Q.1]

4 connected	8 connected
i) In this technique connected pixels we are putting the pixels above, below to the right & to the left side of the current pixels	i) In this technique 8 connected pixels we are putting pixels above, below height & left side of the current pixels.
ii) It requires huge amount of memory	ii) Memory consumption is relatively less in Boundry fill algorithm
iii) Flood fill algorithms are simple & efficient	iii) The complexity of Boundry fill algorithm is high

Q.2] i) Hexagon

ii)

iii) Triangle

iv)

Q.3]

Flood fill algorithm	Boundry fill algorithm
i) It can process the image containing more than one boundry colores	i) It can only process the image containing single boundry colour
ii) It is comparatively slower than the Boundry fill algorithm	ii) It is faster than the flood fill.

Let us assume that the original coordinates are (X, Y), the scaling factors are (S_X, S_Y), and the produced coordinates are (X', Y'). This can be mathematically represented as shown below –

$$\mathbf{X' = X \cdot S_X \text{ and } Y' = Y \cdot S_Y}$$

The scaling factor S_X, S_Y scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below –

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

OR

$$\mathbf{P' = P \cdot S}$$

Where S is the scaling matrix.

The scaling is shown in the following figure.

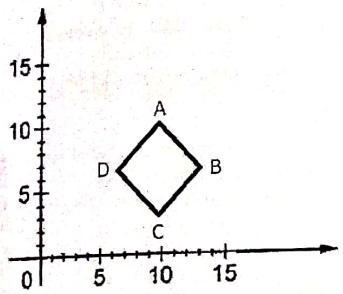


Figure: Scaling before

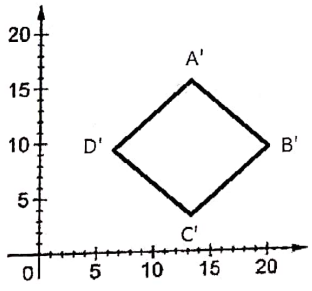


Figure: Scaling after

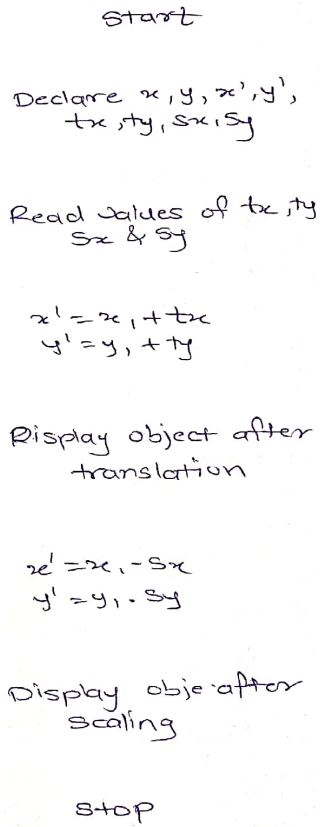
Procedure:

Step 1: Start the program.
Step 2: Input the object coordinates
Step 3: For Translation
a) Enter the translation factors tx and ty.
b) Move the original coordinate position (x,y) to a new position (x1,y1).ie. x=x+x1, y=y+y1.
c) Display the object after translation
Step 4: For Scaling
a) Input the scaled factors sx and sy.
b) The transformed coordinates (x1,y1) , x1=x.sx and y1=y.sy.
c) Display the object after scaling
Step 5: Stop the Program.

VIII. Algorithm

Step 1 : Start
Step 2 : Declare x, y, x', y', tx, ty, sx, sy
Step 3 : Enter & read values of tx & ty.
Step 4 : x' = x, + tx & y = y, + ty
Step 5 : Display object obtain translation
Step 6 : Enter & Read sx & sy
Step 7 : x' = x, - sx & y, = y1. sy
Step 8 : Display object after scaling
Step 9 : Stop.

IX. Flow Chart



XIV. Result (Output of the Program)

XV. Conclusion(s)

We understood how to implement a program for 2dimentional transformation (translation & scaing.)

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write the transformation matrix for 2D Translation.
2. Write the transformation matrix for 2D Scaling.
3. What does scaling transformation do?
4. Whether size of object remains same or changed in case of translation?

(Space for Answers)

Q.1] $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$

Q.2] $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$

Q.3] Scaling transmation changes the size of object along x or y-axis or both axis

Q.4] When size of object remains same or changed in case of translation the scaling occurs.

XVII. Exercise

Attempt Q1. And teacher shall allot Q. 2/Q.3 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Translate the polygon with co-ordinates A(2,5),B(7,10) and C(10,2) by 3 units in x direction and 4 units in y direction.
2. Scale the polygon with co-ordinates A(2,5),B(7,10) and C(10,2) by 2 units in x direction and 2 units in y direction.
3. Give a 3x3 homogeneous co-ordinate transformation matrix for each of the following translations.
   i. Shift the image to the right 3 units
   ii. Shift the image up 2 units
   iii. Move the image down ½ unit and right 1 unit
   iv. Move the image down 2/3 unit and left 4 units.
4. Find the transformation matrix that transforms the given square ABCD to half its size with center still remaining at the same position. The co-ordinates of the square are : A(1,1),B(3,1),C(3,3) and D(1,3) and center at (2,2).Also find the resultant co-ordinates of square.

(Space for Answers)

Q.1] A(2,5), B(7,10), C(10,2)

$t_x = 3$

$t_y = 4$

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \end{bmatrix} = \begin{bmatrix} 2+3 & 7+3 & 10+3 \\ 5+4 & 10+4 & 2+4 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 10 & 13 \\ 9 & 14 & 6 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2+0+3 & 7+0+3 & 10+0+3 \\ 0+5+4 & 0+10+4 & 2+0+4 \\ 0+0+1 & 0+0+1 & 0+0+4 \end{bmatrix} = \begin{bmatrix} 5 & 10 & 13 \\ 9 & 14 & 6 \\ 1 & 1 & 1 \end{bmatrix}$$

PAGE NO.
DATE : / /

Q. 2] A(2,5), B(7,10), C(10,2)

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

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 14 & 20 \\ 10 & 20 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Q.3] i) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

ii) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

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

iv) $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 2/3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

PAGE NO.:
DATE : / /

Q.4] $S_x = 0.5$, $S_y = 0.5$, $P = (7, 2)$

$t_x = 2$, $t_y = 2$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 & 2.5 & 2.5 & 2.5 \\ 1.5 & 1.5 & 2.5 & 2.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

XVIII. References / Suggestions for further Reading

1. https://books.google.co.in/books?isbn=8184317379
2. http://www.freebookcentre.net/ComputerScience-Books-Download/Basics-of-Computer-Graphics.html
3. http://www.freetechbooks.com/introduction-to-computer-graphics-t892.html
4. https://en.wikipedia.org/wiki/Book:Game_Devel_Book_2

XIX. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		**30%**
1.	Debugging ability	20%
2.	Follow ethical practices.	10%
Product related (15 Marks)		**70%**
3.	Correctness of algorithm	15%
4.	Correctness of Program codes	30%
5.	Quality of input/output messaging and output formatting	10%
6.	Timely Submission of report	5%
7.	Answer to sample questions	10%
Total (25 Marks)		**100%**

List of Students /Team Members

1. Sumit....Birange.....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

$$[X'Y'] = [XY]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} OR$$

$$P' = P \cdot R$$

Where R is the rotation matrix and is represented as

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

The rotation angle can be positive and negative (i.e. clockwise or counterclockwise) For positive rotation angle, we can use the above rotation matrix. However, for negative angle rotation, the matrix will change as shown below –

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

$$= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} (\because cos(-\theta) = cos\theta \text{ and } sin(-\theta) = -sin\theta)$$

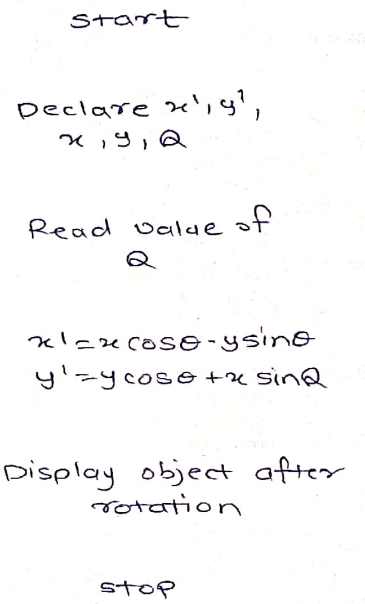
Procedure:

Step 1: Start the program.
Step 2: Input the object coordinates
Step 3: For Rotation
a) Enter the radian for rotation angle θ.
b) Rotate a point at position (x,y,z) through an angle θ about the origin
x1=xcos θ – ysin θ , y1=ycos θ + xsin θ.
c) Display the object after rotation
Step 4: Stop the Program.

VIII. Algorithm

Step 1: Start
Step 2: Declare x, y, Q
Step 3: Read value of Q
Step 4: $x_1 = x\cos\theta - y\sin\theta$
$y_1 = y\cos\theta + y\sin\theta$
Step 5: Display object after rotation
Step 6: Stop.

IX. Flow Chart



X. 'C' Program Code

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety/ethical practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i5 RAM - 4GB
2	Software	Turbo C++
3	Any other resource used	

XIV. Result (Output of the Program)

.....

.....

.....

XV. Conclusion(s)

We understood how to implement a program of 2 dimentional transformation (Rotation).....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write the transformation matrix for 2D Rotation.
2. Write rotation matrix for a rotation angle 30^0.

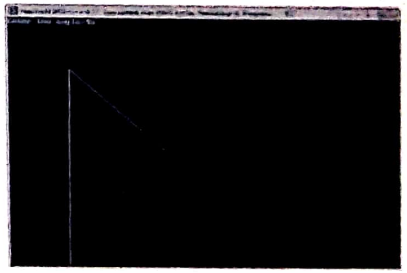
Experiment No. 8

Name: SumitTanajiBiranje

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        intgd=DETECT,gm;
        int x1=100,y1=100,x2=300,y2=300,x3,y3;
        double angle,c11,c12;
        initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
        line(x1,y1,x2,y2);
        printf("Enter the angle:");
        scanf("%d",&angle);
        c11=cos((angle*3.14)/100);
        c12=sin((angle*3.14)/100);
        x3=(x2*c11)-(y2*c12);
        y3=(x2*c12)+(y2*c11);
        line(x1,y1,x3,y3);
        getch();
        closegraph();
}
```

Output:



(Space for Answers)

Q.1] $R\theta = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Q.2] $R30 = \begin{bmatrix} \sin 30 & -\cos 30 \\ \cos 30 & \sin 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Q.1] $A = (4,3)$, $\theta = 45^\circ$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4/\sqrt{2} - 3/\sqrt{2} + 0 \\ 4/\sqrt{2} + 3/\sqrt{2} + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 5/\sqrt{2} \\ 1 \end{bmatrix}$$

$A' = (1/\sqrt{2}, 5/\sqrt{2})$

Q.2] $A(2,3)$, $B(5,5)$, $C(4,3)$ about point $(1,1)$

$P' = T_c \cdot R\theta \cdot T_c P$: $t_x = 1$ $t_y = 1$

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

XVII. Exercise

Attempt Q1. And teacher shall allot Q. 2/Q.3 from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. A point (4, 3) is rotated counterclockwise by an angle of 45^0. Find the rotation matrix and the resultant point.
2. Perform a counterclockwise 45^0 rotation of triangle A (2, 3), B (5, 5), C (4, 3) about point (1,1).
3. Find a transformation of triangle A(1,0),B(0,1),C(1,1) by
   i. Rotating 45^0about the origin and then translating one unit in x and y direction.
   ii. Translating one unit in x and y direction and then rotating 45^0about the origin.
4. Consider the square A(1,0),B(0,0),C(0,1),D(1,1).Rotate the square by 45^0 anticlockwise direction.

(Space for Answers)

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

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & \sqrt{2}/2 & -\sqrt{2}+1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 4 \\ 3 & 5 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ (3+3\sqrt{2}-\sqrt{2}+1) & -15 & (-44-\sqrt{2}) \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 4+2\sqrt{2} & -15 & (-44-\sqrt{2}) \\ 1 & 1 & 1 \end{bmatrix}$$

Q.3] i) $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

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

ii) $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

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

PAGE NO.:
DATE : / /

Q.4]

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 1 \\ \sin\theta & \cos\theta & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 6 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

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

$\therefore A' = (\sqrt{2}/2, \sqrt{2}/2)$

$B' = (0, 0)$

$C' = (-\sqrt{2}/2, \sqrt{2}/2)$

$D' = (0, \sqrt{2}/2)$

Procedure :

Step 1: Start the program.
Step 2: Input the object coordinates
Step 3: For Shearing
a) Input the shearing factors shx and shy.
b) Shearing related to x axis: Transform coordinates x1=x+shx*y and y1=y.
c) Shearing related to y axis: Transform coordinates x1=x and y1=y+shy*x.
d) Input the xref and yref values.
e) X axis shear related to the reference line y-yref is x1=x+shx(y-yref) and y1=y.
f) Y axis shear related to the reference line x=xref is x1=x
g) Display the object after shearing
Step 4: For Reflection
Reflection can be performed about x axis and y axis.
a) Reflection about x axis: The transformed coordinates are x1=a and y1=-y.
b) Reflection about y axis: The transformed coordinates are x1=x and y1=y.
c) Display the object after reflection
Step 5: Stop the Program.

VIII. Algorithm

Step 1 : Start
Step 2 : Declare x', y', x, y, shx, shy
Step 3 : Read the value of shx & shy
Step 4 : Shearing for x-axis
$x' = x_1 + shx * y_1$
Shearing for y-axis
$y' = y_1 + shy * x_1$
Shearing for x-ref
$x' = x_1 + shx (y_1 - y - y\text{-}ref)$
Shearing for x-ref
$x_1 = x\text{ref}$
Step 5 : For reflection about x-axis
$x_1 = x$ & $y_1 = -y$
reflection about y-axis
$x_1 = x$ & $y_1 = y$
Step 6 : Display object after reflection.
Step 7 : Stop

IX. Flow Chart

Start

Declare x', y', x, y, shx, shy

Shearing of x-axis

$x' = x_1 + shx + y$

Shearing for y-axis

$y' = y_1 + shy + x_1$

Reflection for x-axis & y-axis

$x_1 = x$ & $y_1 = y$
$x_1 = x$ & $y_1 = y$

Display object after shearing of reflection

Stop

X. 'C' Program Code

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core is RAM-4GB
2	Software	Turbo C++
3	Any other resource used	

XIV. Result (Output of the Program)

.....

.....

.....

XV. Conclusion(s)

We understood how to implement a program for 2 dimentional transformation (Reflection & Shearing)

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write the transformation matrix for 2D Reflection.
2. Write the transformation matrix for 2D Shear.
3. Differentiate between X-shear and Y-shear.
4. Define Reflection and Shearing.

(Space for Answers)

Q.1]
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Q.2] x-shear
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

y-share
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Q.4] Reflection :- It is a transformation that process miner image on an object exclusive to an axis of reflection.

Shearing :- A transformation that stand shapes of object is called Shearing.

XVII. Exercise

Attempt Q1. And teacher shall allot Q. 1/Q.2 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. A point (4, 3) is rotated counterclockwise by an angle of 45^0. Find the rotation matrix and the resultant point. Apply the Shearing transformation to square with A(0,0),B(1,0),C(1,1) and D(0,1) as given below :
   i. Shear parameter value of 0.5 relative to the line Yref= -1;
   ii. Shear parameter value of 0.5 relative to the line Xref= -1;
2. Apply shearing transformation to square with A(0,0),B(1,0),C(1,1) and D(0,1).If Sh_x =0.5 then find the resultant co-ordinates.

(Space for Answers)

Q.1] A=(4,3)
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4/\sqrt{2} - 2/\sqrt{2} + 0 \\ 4/\sqrt{2} + 3/\sqrt{2} + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \\ 1 \end{bmatrix}$$

$A' = A'(1/\sqrt{2}, 7/\sqrt{2})$

Q.2] Shy = 0.5

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

$A' = (0, 0)$
$B' = (1, 0.5)$
$C' = (1, 1.5)$
$D' = (0, 1)$

PAGE NO.:
DATE: / /

Q.3]

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Shx & 1 & 0 \\ -Shx * yref & 1 & 1 \end{bmatrix}$$

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

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

$$= \begin{bmatrix} 1 & 0.5 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1.5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 0, 0.5 \\ 1, 1 \\ 1, 2 \\ 0, 1.5 \end{bmatrix}$$

VIII. Algorithm

Step 1: Start

Step 2: Read coordinates of all vertices of the polygon.

Step 3: Read coordinates of the clipping window.

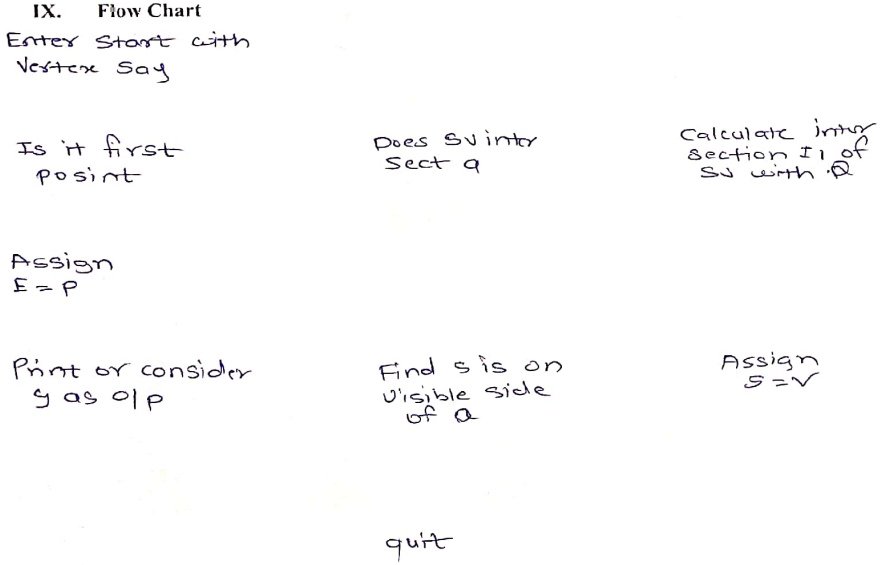
Step 4: Compare vertices of each of polygon individual with the clipping plane.

Step 5: Save the resulting intersections & vertices in the new list of vertices according to for possible relationships between the edges & the clipping boundary.

Step 6: Repete the steps 4 & 5 for remaining edges of clipping window. Each time resultiant list of vertices is successfully clipping window.

Step 7: Stop.

IX. Flow Chart



X. 'C' Program Code

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards but not limited to	As per batch size	For all Experiments
2	Operating system	Windows XP/Windows 7/LINUX version 5.0 or later		
3	Software	Turbo C /C++ Version 3.0 or later with DOSBOX		

XII. Precautions

1. Ensure that all C statements must end with a semicolon (;).
2. Use white spaces in c to describe blanks and tabs.
3. Ensure use of proper graphics function for relevant object.
4. Follow safety/ethical practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) i5 RAM - 4GB
2	Software	Turbo C
3	Any other resource used	

XIV. Result (Output of the Program)

XV. Conclusion (s)

In this practical we had learnt that polygon clipping algorithm using Sutherland hodgeman polygon clipping algorithm.

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

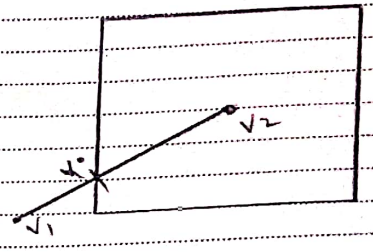
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

i. If the first vertex is outside the clipping window and second point is inside the clipping window, then write which points are added to output vertex list.

ii. Write the procedure to clip polygon using Sutherland Hodgeman Polygon Clipping algorithm.

(Space for Answers)

Q.i) If the first vertex of the polygon is outside & the second vertex is inside the window then the output will be the intersection point & second vertex.



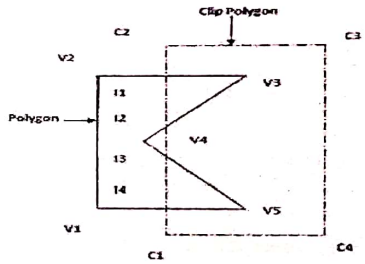
Output :- V1,V2

XVII. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Clip the following polygon using Sutherland Hodgeman Polygon Clipping algorithm.



(Space for Answers)

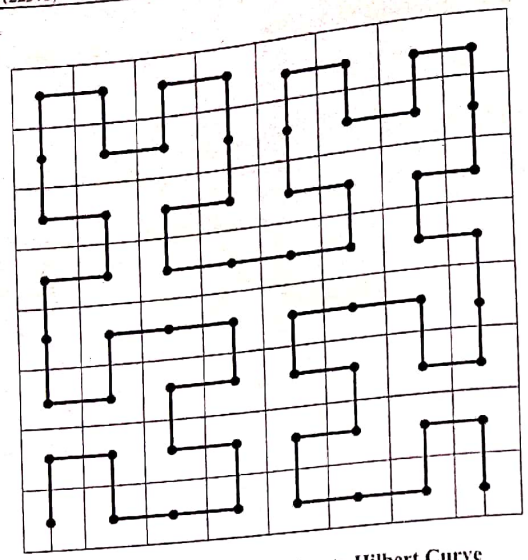


Figure 3: Third Approximation to Hilbert Curve

- The first approximation will divide square into 4 quadrant and draw curve which connect the center point (Figure 1).
- In second again divide each quadrant and again connects the center point (Figure 2).
- In the third approximation again subdivide the quadrant. If again connects the centers of the finest level before stepping to the next level (Figure 3).

Applying the process continuously, remember following things:

1. Curve never cross itself.
2. The curve is arbitrarily close to the every point in the square. There is no limit to subdivisions.
3. The curve fills the square.
4. The length of the curve is infinite, with each subdivision length increase by a factor of 4.
5. There is no limit of length.
6. The constructed curve is topologically equivalent to line Dt=1.
7. The fractal dimensions can be determined as at each subdivision the scale is changed by 2 but the length is changed by 4.
8. For square it takes 4 curves of the half scale to build the full sized object so dimension D can be given as $4 = 2^D$. It must be D=2.
9. The Hilbert curve has topological dimension 1 but fractal dimensions 2.

Procedure:

1. Hilbert subroutine draws the Hilbert curve.
2. It takes as parameters the depth of recursion, and dx and dy values that give the direction in which it should draw.
3. It recursively draws four smaller Hilbert curves and connects them with lines.

VIII. Algorithm

```
Step 1: Start
Step 2: enum ( ) {
        up, left, right, down
        } Void helbert (int level, int direction = up)
        { if (level == 1) {
        switch (direction)
        case left: move(right); move (down); move(left);
                   break;
        case up: move (down); move (right); move (up);
                   break;
        case right: move (left); move(up); move(right);
                   break;
        case down: move (up); move (left); move (down);
                   break;
        } }
        else {
        switch(direction){
        case left: hilbert (level + up); move (right);
                   hilbert (level + left); move (down);
                   hilbert (level + right); move (left);
                   hilbert (level + down);
                   break;
        case right: hilbert (level + down); move (left);
                   hilbert (level + , left); move (down);
                   hilbert (level + , left); move (right);
                   hilbert (level + , up);
                   break;
        case up: hilbert (level + , left); move (down);
                   hilbert (level + , up); move (right);
                   hilbert (level , right);
                   break;
        case down: hilbert (level + right); move (up);
                   hilbert (level + , down); move (left);
                   hilbert (down)
                   hilbert (level - 1 left);
                   break;
        }
Step 3: Stop.
```

IX. Flow Chart

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Intel (R) core (TM) is RAM- 4 GB
2	Software	Turbo C++
3	Any other resource used	

XIV. Result (Output of the Program)

.....

.....

.....

XV. Conclusion(s)

We understood to implement the hilbert curve program.

.....

.....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

i. Define Curve.

ii. Write topological and fractal dimension of Hilbert's curve.

(Space for Answers)

.....

Q-1] The curves can be drawn with two approches as follows

i.) First approch is to develop a curve generation algorithm such as a DDA. A true curve is generated with this approch.

ii.) Second method is approximate a curve by a number of straightline segments.

.....

Q.2] Topological = 1

Fractal = 2

The helbert curve have topological dimention 1 but fractual dimentional for

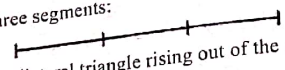
Procedure:

Koch curve:

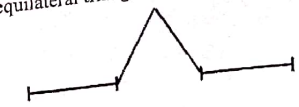
The Koch curve is a simple fractal that creates a pretty snowflake-like object. The iteration algorithm is very simple:

1. Start with a straight line:

2. Trisect the line into three segments:



3. Form an equilateral triangle rising out of the middle segment:



4. Repeat, with newly formed segment.

If you start with an equilateral triangle instead of a line, you get the lovely image shown at the top of the article after a few iterations.

Bezier Curve:

To each set of four points P_0, P_1, P_2, P_3 we associate a curve with the following properties:

1. It starts at P_0 and ends at P_3.
2. When it starts from P_0 it heads directly towards P_1, and when it arrives at P_3 it is coming from the direction of P_2.
3. The entire curve is contained in the quadrilateral whose corners are the four given points (their **convex hull**).

VIII. Algorithm

Step 1: Start

Step 2: Declare & initialize a variable.

Step 3: if (N=0) then
line (xb, yb, zb);
line (xc, yc, zc);
line (xd, yd, zd);

Step 4: if (N!=0) then
xab ← (pen-x + xb) / 2
yab ← (pen-x1 + yb) / 2
zab ← (pen-z + zb) / 2
xbc ← (xb+xc) / 2
ybc ← (yb+bc) / 2
zbc ← (zb+zc) / 2
xcd ← (yc+yd) / 2
zcd ← (zc+zd) / 2

PAGE NO :
DATE : / /

xabc ← (xab + ybc) / 2
yabc ← (xab + ybc) / 2
zabc ← (zab + zbc) / 2
xbcd ← (xbc + xcd) / 2
ybcd ← (ybc + ycd) / 2
zbcd ← (zbc + zcd) / 2
xabcd ← (xabc + ybcd) / 2
yabcd ← (yabc + ybcd) / 2
zabcd ← (zabc + ybcd) / 2

Step 5: bezier ABS (xab, yab, zab, xabc, yabc, zabc, xabcd, yabcd, zabcd, N-1)

Step 6: bezier ABS) (xbcd, ybcd, zbcd, xcd, ycd, zcd, xd, yd, zd, N-1)

Step 7: Stop