NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 01

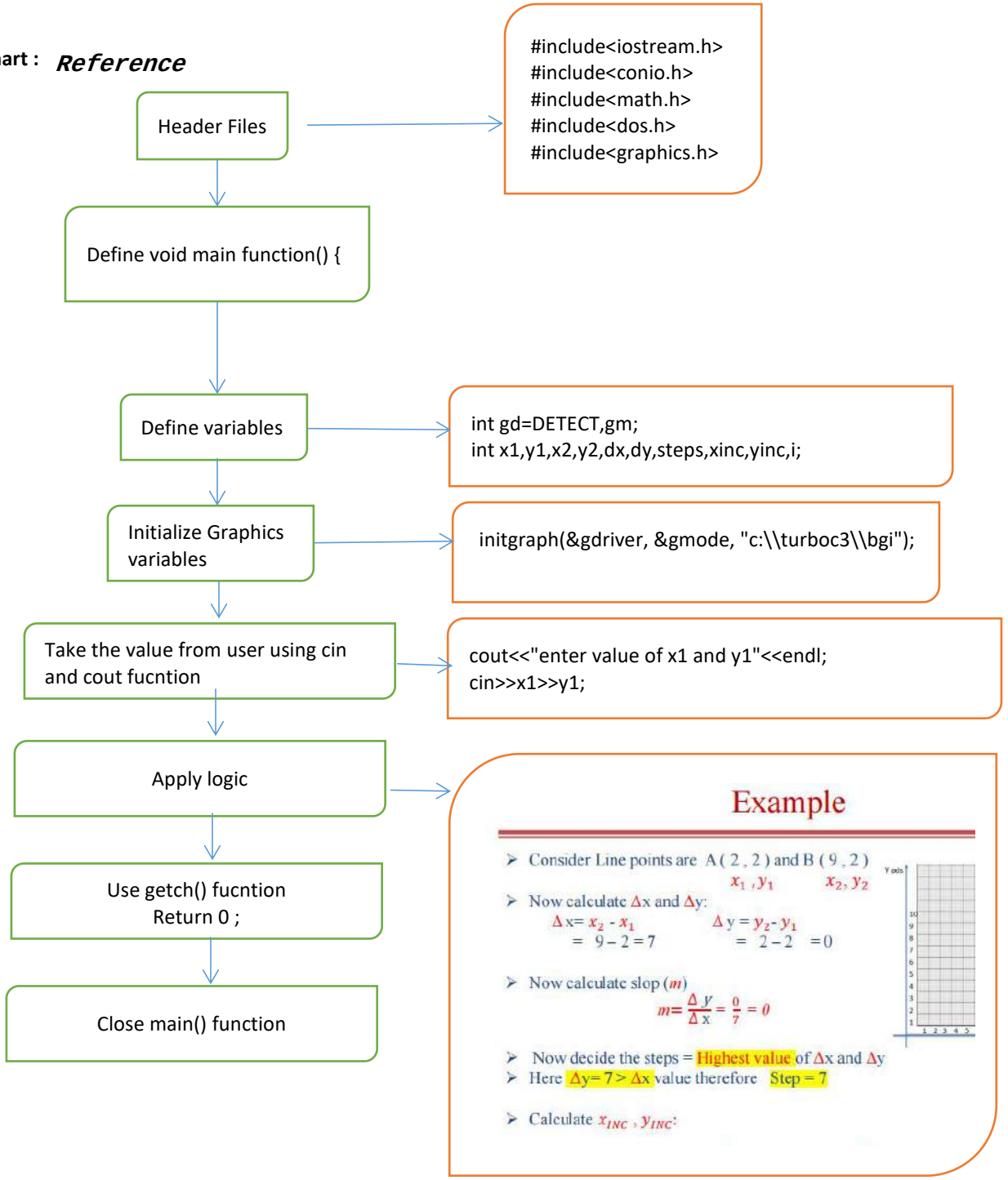
Experiment No : 1

Aim : Aim : Write a program in c for DDA line drawing algorithm.

i) To draw a thin line

Software used : Turbo C++

Flow Chart : ***Reference***



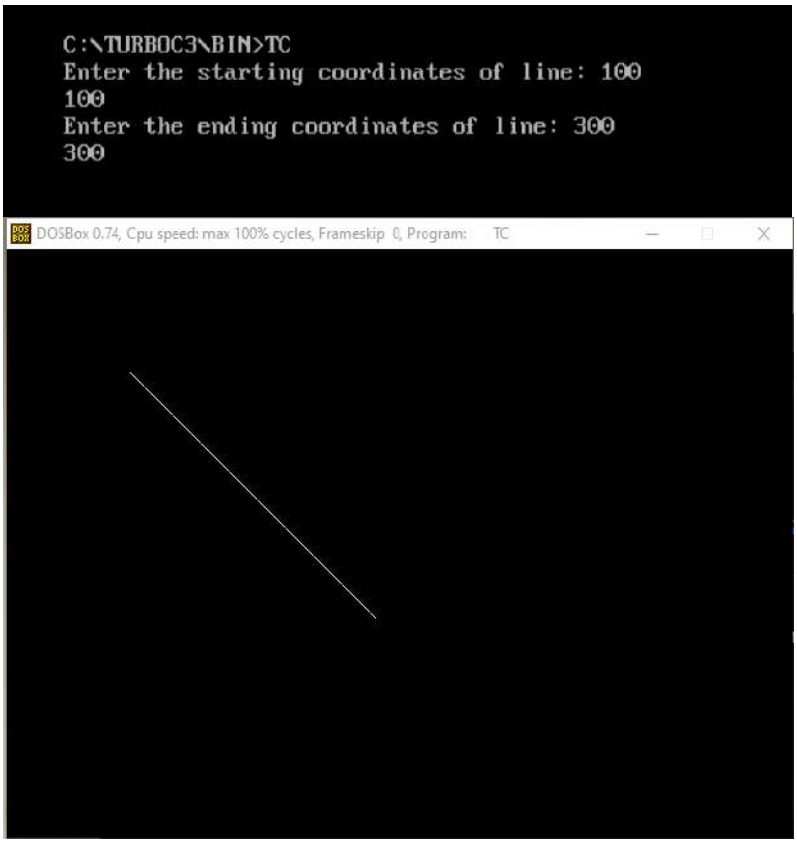
Source Code:

*DDAThinLine - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
int main(void)
{
        int gdriver=DETECT,gmode;
        int i,a,b,a1,b1,a2,b2,da,db,length,aincr,bincr;
        printf("Enter the starting coordinates of line: ");
        scanf("%d%d",&a1,&b1);
        printf("Enter the ending coordinates of line: ");
        scanf("%d%d",&a2,&b2);
        initgraph(&gdriver,&gmode,"c:\\TURBOC3\\BGI");
        da=abs(a2-a1);
        db=abs(b2-b1);
        if(da>db)
        length=da;
        else
        length=db;
        aincr=da/length;
        bincr=db/length;
        putpixel(a1,b1,15);
        a=a1;
        b=b1;
        for(i=1;i<=length;i++)
        {
                a=a+aincr;
                b=b+bincr;
                putpixel(a,b,15);
        }
        getch();
        closegraph();
        return 0;
}
```

Output:



Conclusion: Thin line is drawn using DDA line drawing algorithm.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 02

Experiment No : 2

Aim : Write a program in c/ c++ for DDA line drawing algorithm.
ii) To draw a dotted line
iii) To draw a dashed line

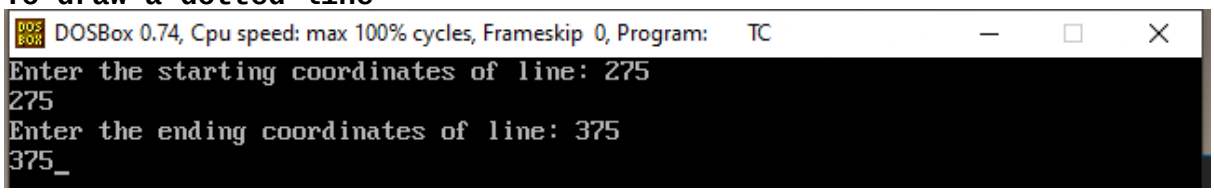
Software used : Turbo C++
Source Code:
ii)To draw a dotted line

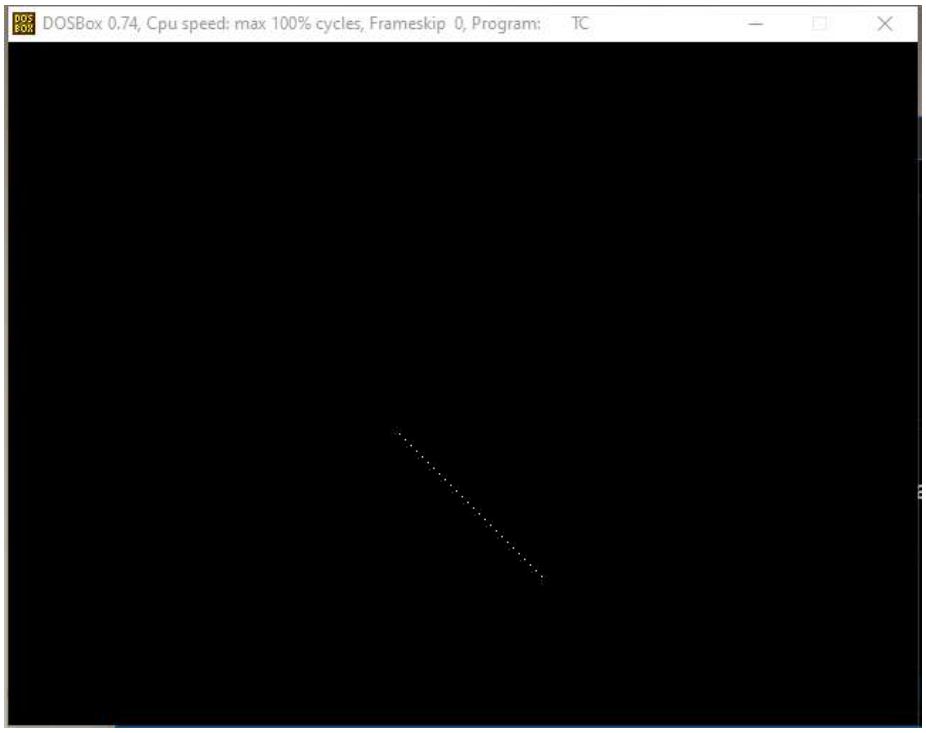
DDADottedLine - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
int main(void)
{
        int gdriver=DETECT,gmode;
        int i,a,b,a1,b1,a2,b2,da,db,length,aincr,bincr;
        printf("Enter the starting coordinates of line: ");
        scanf("%d%d",&a1,&b1);
        printf("Enter the ending coordinates of line: ");
        scanf("%d%d",&a2,&b2);
        initgraph(&gdriver,&gmode,"c:\\TURBOC3\\BGI");
        da=abs(a2-a1);
        db=abs(b2-b1);
        if(da>db)
        length=da;
        else
        length=db;
        aincr=da/length;
        bincr=db/length;
        putpixel(a1,b1,15);
        a=a1;
        b=b1;
        for(i=1;i<=length;i++)
{
                a=a+aincr;
                b=b+bincr;
                if(i%4==0)
                {
                        putpixel(a,b,15);
                }
        }
        getch();
        closegraph();
        return 0;
}
```

Output:
ii)To draw a dotted line





Source Code:

iii) To draw a dashed line

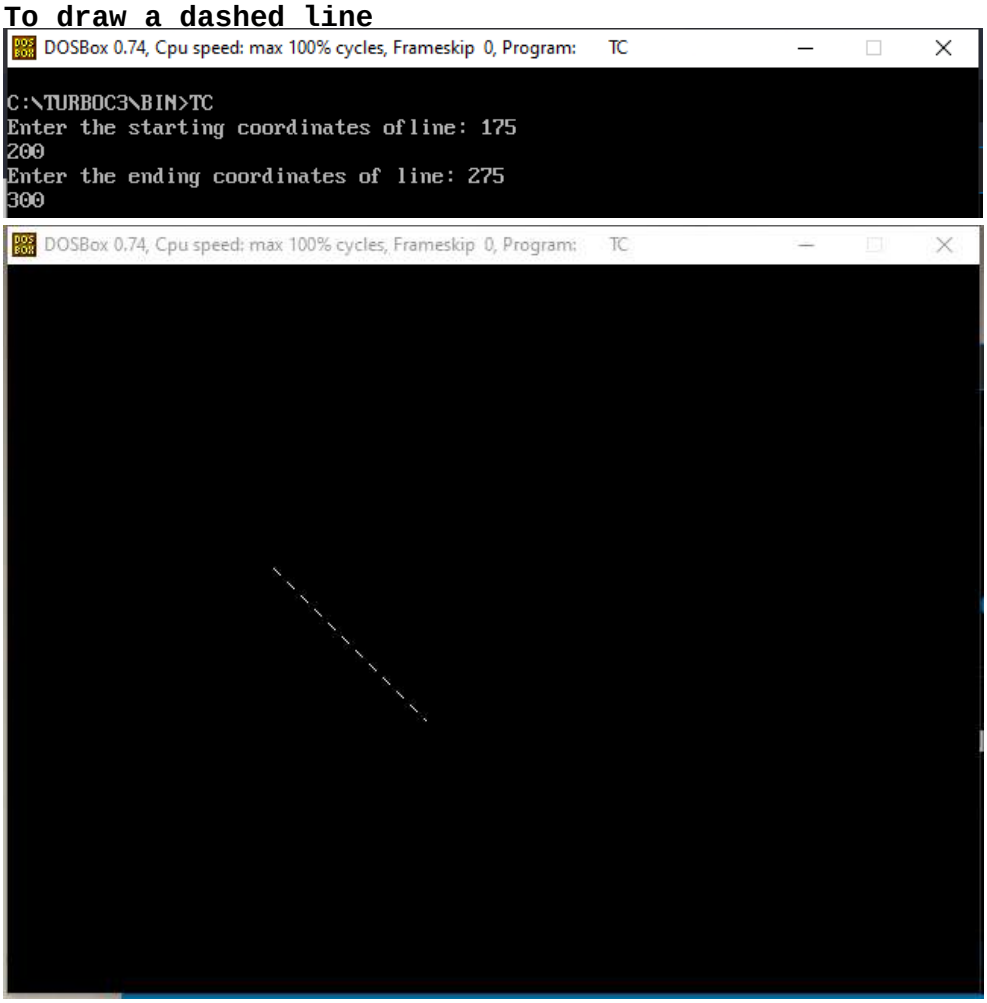
DDADashedLine - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
int main(void)
{
        int gdriver=DETECT,gmode;
        int i,a,b,a1,b1,a2,b2,da,db,length,aincr,bincr;
        printf("Enter the starting coordinates ofline: ");
        scanf("%d%d",&a1,&b1);
        printf("Enter the ending coordinates of line: ");
        scanf("%d%d",&a2,&b2);
        initgraph(&gdriver,&gmode,"c:\\TURBOC3\\BGI");
        da=abs(a2-a1);
        db=abs(b2-b1);
        if(da>db)
        length=da;
        else
        length=db;
        aincr=da/length;
        bincr=db/length;
        putpixel(a1,b1,15);
        a=a1;
        b=b1;
        for(i=1;i<=length;i++)
        {
                a=a+aincr;
                b=b+bincr;
                if((i%9>4)==0)
                {
                        putpixel(a,b,15);
                }
        }
        getch();
        closegraph();
        return 0;
}
```

Output:

iii) To draw a dashed line



Conclusion: Dotted and Dashed line are drawn successfully using DDA line drawing algorithm.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 03

Experiment No : 3

Aim : Write a program in c/ c++ to implement **Bresenham's line drawing** algorithm.

i) To draw a thin line

Software used : Turbo C++

Flow Chart :

Source Code:

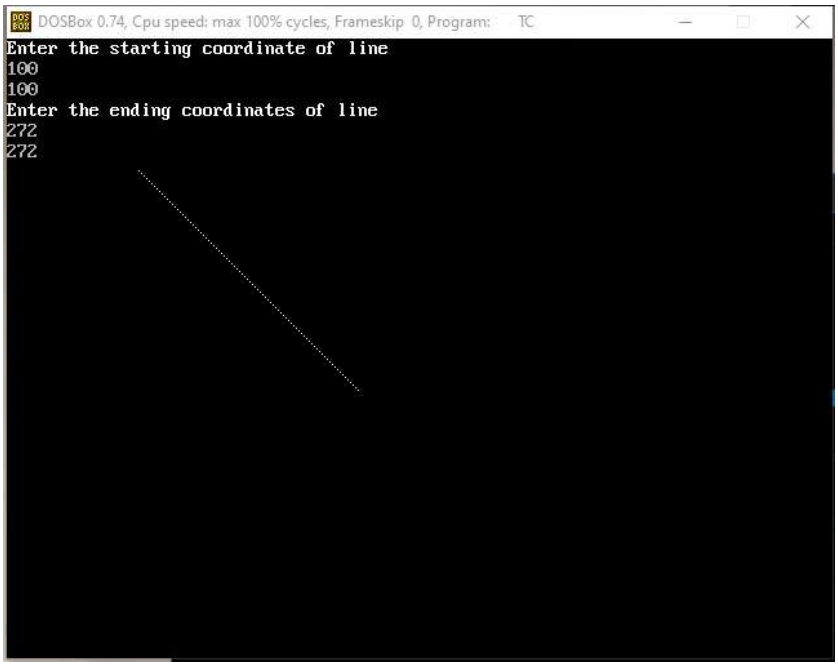
i) To draw a dotted line

Bresenham'sDottedLine - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main()
{
        int gd=DETECT,gm,x,y,x1,y1,x2,y2,dx,dy,i,e;
        float xinc,yinc;
        initgraph(&gd,&gm, "C:\\TURBOC3\\BGI");
        printf("Enter the starting coordinate of line\n");
        scanf("%d%d",&x1,&y1);
        printf("Enter the ending coordinates of line\n");
        scanf("%d%d",&x2,&y2);
        dx=x2-x1;
        dy=y2-y1;
        if(x1<x2)
        xinc=1;
        else
        xinc=-1;
        if(y1<y2)
        yinc=1;
        else
        yinc=-1;
        x=x1;
        y=y1;
        if(dx>=dy)
        {
                e=(2*dy)-dx;
                while(x!=x2)
                {
                        if(e<0)
                        e=e+(2*dy);
                        else
                        {
                                e=e+(2*(dy-dx));
                                y=y+yinc;
                                y=y+yinc;
                        }
                        x=x+xinc;
                        x=x+xinc;
                        putpixel(x,y,WHITE);
                }
        }
        else
        {
                e=(2*dx)-dy;
                while(y!=y2)
                {
                        if(e<0)
                        e=e+(2*dx);
                        else
                        {
                                e=e+(2*(dx-dy));
                                x=x+xinc;
                                x=x+xinc;
                        }
                        y=y+yinc;
                        y=y+yinc;
                        putpixel(x,y,WHITE);
                }
        }
        getch();
        closegraph();
}
```

Output - i) To draw a dotted line



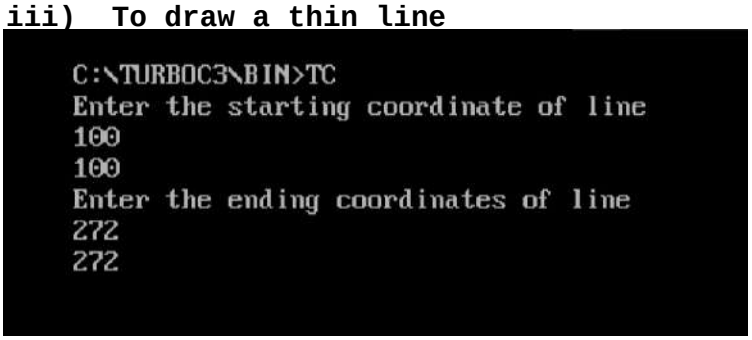
Source code - iii) To draw a thin line

Bresenham'sThinLine - Notepad

File Edit Format View Help

```
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
int main(void)
{
    int gdriver=DETECT,gmode;
    int i,x,y,x1,y1,x2,y2,dx,dy,length,P0;
    printf("Enter the starting coordinate of line\n");
    scanf("%d%d",&x1,&y1);
    printf("Enter the ending coordinates of line\n");
    scanf("%d%d",&x2,&y2);
    initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");
    dx=abs(x2-x1);
    dy=abs(y2-y1);
    if(dx>dy)
        length=dx;
    else
        length=dy;
    putpixel(x1,y1,2);
    x=x1;
    y=y1;
    P0=2*dy- dx;
    for(i=0;i%6<4;i++)
        for(i=0;i<=length;i++)
        {
            if(P0<0)
            {
                x=x+1;
                y=y+1;
                putpixel(x,y,2);
                P0=P0+2*y;
            }
            else
            {
                x=x+1;
                y=y+1;
                putpixel(x,y,2);
                P0=P0+2*y-2*dx;
            }
        }
    getch();
    closegraph();
    return 0;
}
```

Output - **iii) To draw a thin line**



Conclusion - Dotted and Thin line are drawn successfully using Bresenham’s line drawing algorithm.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

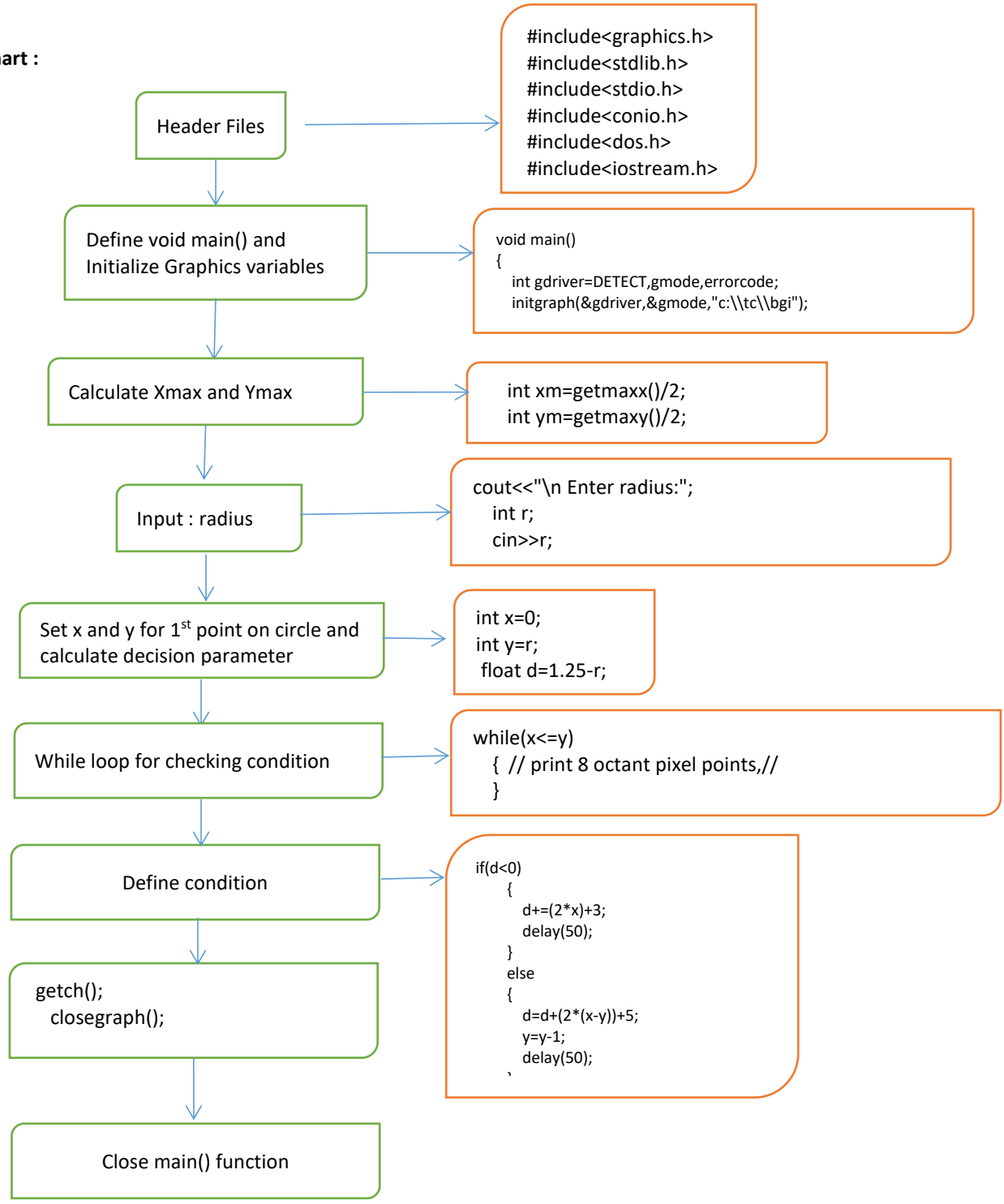
EXPERIMENT 04

Experiment No : 4

Aim : Write a program in c to implement Mid-Point circle generating algorithm.

Software used : Turbo C++

Flow Chart :



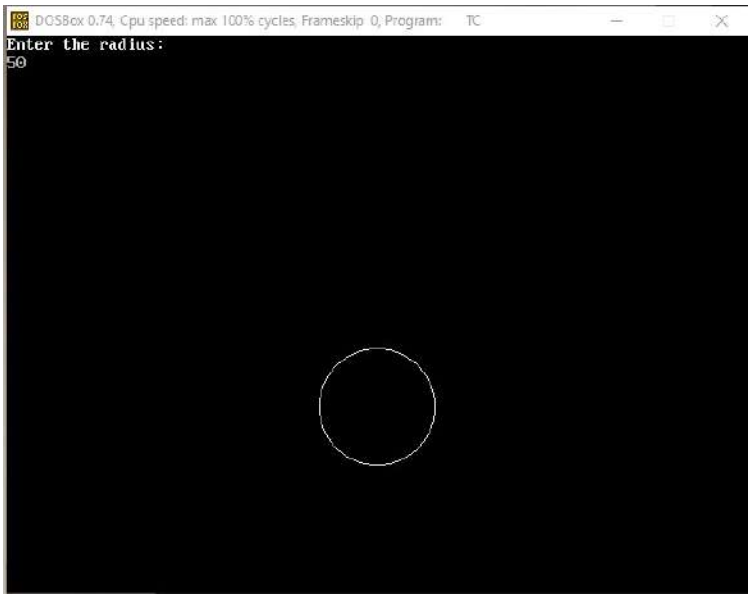
Source Code:

Mid-PointCircle - Notepad

File Edit Format View Help

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<iostream.h>
void main()
{
        int gdriver=DETECT,gmode,errorcode;
        initgraph(&gdriver,&gmode,"c:\\TURBOC3\\BGI");
        int xm=getmaxx()/2;
        int ym=getmaxx()/2;
        int r;
        printf("Enter the radius: \n");
        scanf("%d",&r);
        int x=0;
        int y=r;
        float d=1.25-r;
        while(x<=y)
        {
                putpixel(xm+x,ym-y,15);
                putpixel(xm+y,ym-x,15);
                putpixel(xm+y,ym+x,15);
                putpixel(xm+x,ym+y,15);
                putpixel(xm-x,ym+y,15);
                putpixel(xm-y,ym+x,15);
                putpixel(xm-y,ym-x,15);
                putpixel(xm-x,ym-y,15);
                if(d<0)
                {
                        d+=(2*x)+3;
                        delay(50);
                }
                else
                {
                        d=d+(2*(x-y))+5;
                        y=y-1;
                        delay(50);
                }
                x=x+1;
        }
        getch();
        closegraph();
}
```

Output:



Conclusion: We have learnt to implement Mid-Point circle generating algorithm.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 05

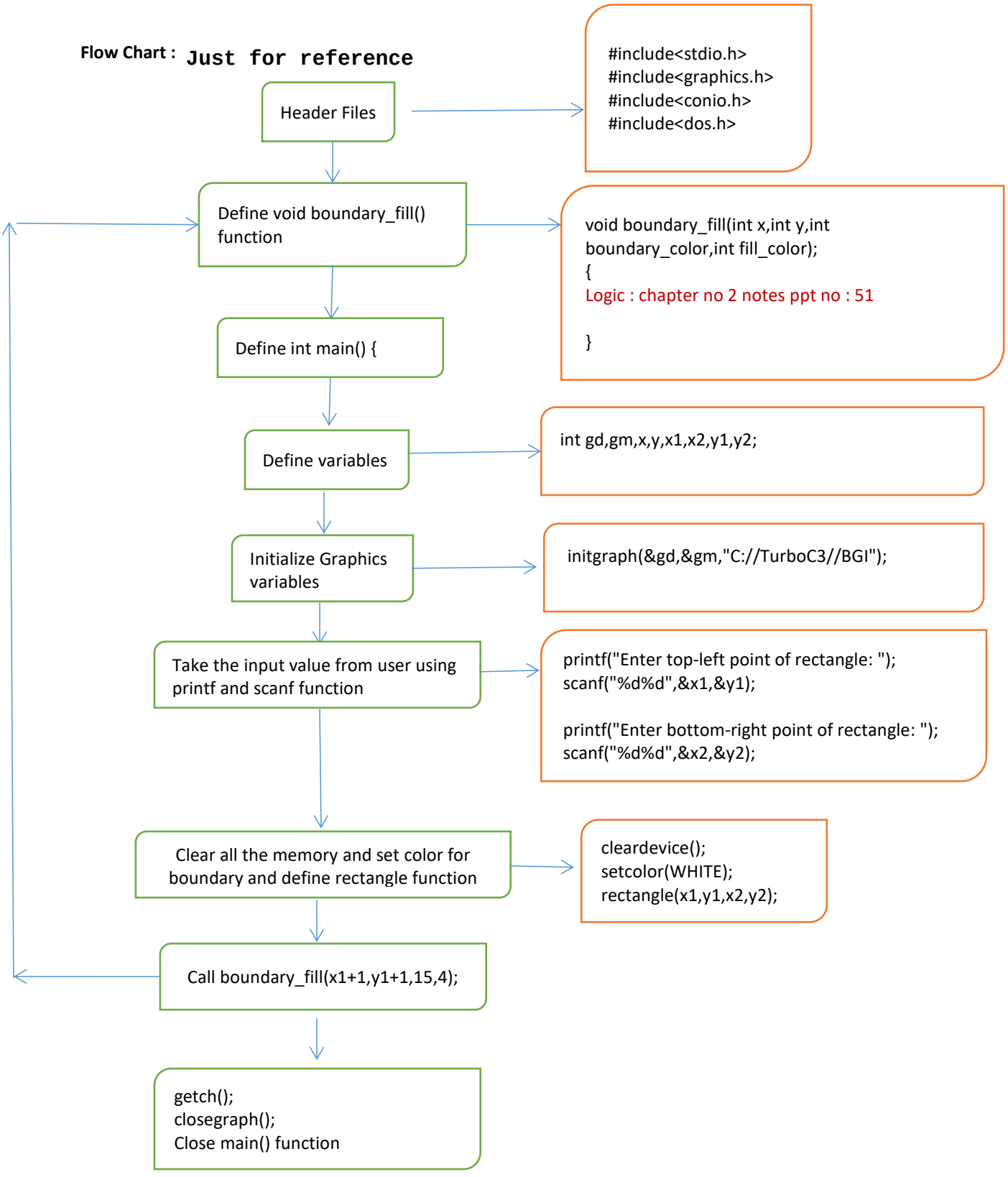
Experiment No : 5

Aim : Write a program in c to implement

i) Boundary Fill algorithm ii) Flood Fill algorithm.

Software used : Turbo C++

Flow Chart : `Just for reference`



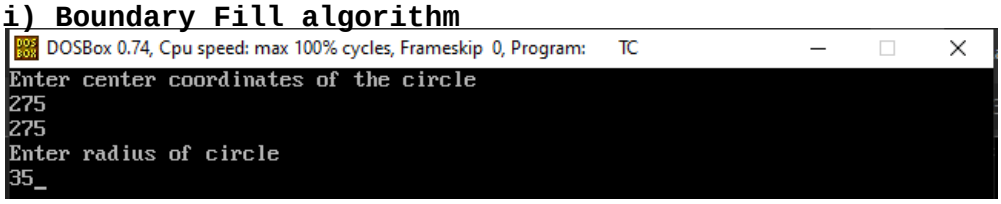
Source Code: i) Boundary Fill algorithm

BoundaryFillAlgorithm - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
void boundaryfill(int p,int q,int f_color,int b_color)
{
        if(getpixel(p,q)!=b_color && getpixel(p,q)!=f_color)
        {
                putpixel(p,q,f_color);
                delay(1);
                boundaryfill(p+1, q,f_color,b_color);
                boundaryfill(p, q+1,f_color,b_color);
                boundaryfill(p-1, q,f_color,b_color);
                boundaryfill(p, q-1,f_color,b_color);
        }
}
int main()
{
        int gm,gd=DETECT,radius;
        int p,q;
        printf("Enter center coordinates of the circle\n");
        scanf("%d%d",&p,&q);
        printf("Enter radius of circle\n");
        scanf("%d",&radius);
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");
        circle(p,q,radius);
        boundaryfill(p,q,5,15);
        getch();closegraph();
        return 0;
}
```

Output: i) Boundary Fill algorithm



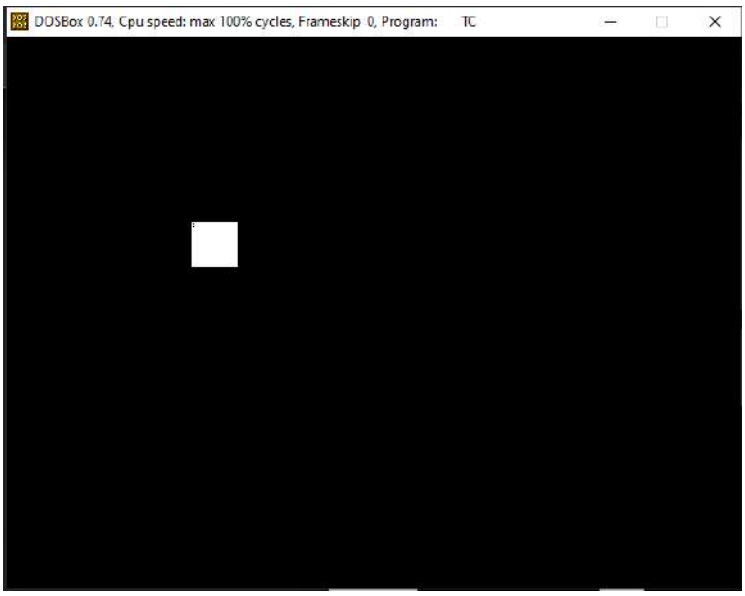
Source Code: ii) Flood Fill algorithm.

FloodFillAlgorithm - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
void flood_fill(int p,int q,int old_c,int new_c)
{
        int c;
        c=getpixel(p,q);
        if(c==old_c)
        {
                putpixel(p,q,new_c);
                flood_fill(p+1,q,old_c,new_c);
                flood_fill(p-1,q,old_c,new_c);
                flood_fill(p+1,q+1,old_c,new_c);
                flood_fill(p-1,q-1,old_c,new_c);
                flood_fill(p,q+1,old_c,new_c);
                flood_fill(p,q-1,old_c,new_c);
                flood_fill(p-1,q+1,old_c,new_c);
                flood_fill(p+1,q-1,old_c,new_c);
        }
}
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");
        rectangle(161,161,200,200);
        flood_fill(170,170,0,15);
        getch();
}
```

Output: ii) Flood Fill algorithm.



Conclusion : We learned to perform Boundary Fill algorithm & Flood Fill algorithm through this experiment successfully.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 06

Experiment No : 06

Aim: **Write a program in c for character generation -bitmap method.**

Software used : Turbo C++

Flowchart:

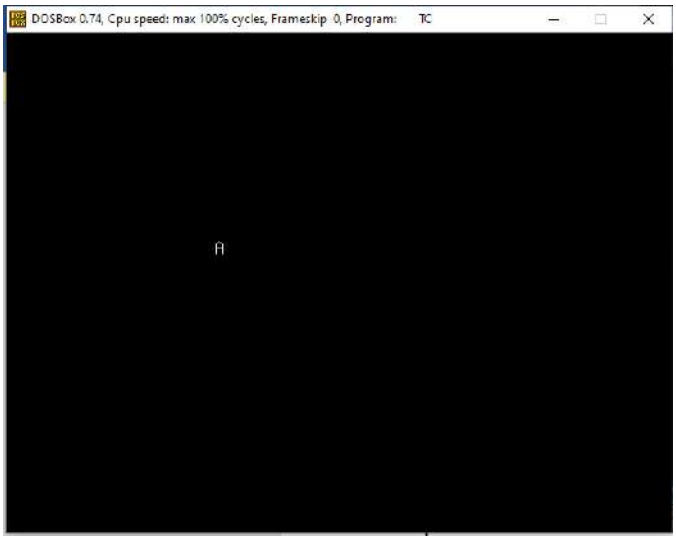
Code:

bitMap - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int gd,gm,i,j;
    int a[13][9]=
        {
                {0,0,0,0,1,0,0,0,0},
                {0,0,0,1,0,1,0,0,0},
                {0,0,1,0,0,0,1,0,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,1,1,1,1,1,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
                {0,1,0,0,0,0,0,1,0},
        };
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm, "C:\\TURBOC3\\BGI");
        for(i=0;i<13;i++)
        {
                for(j=0;j<9;j++)
                {
                        putpixel(200+j,200+i,15*a[i][j]);
                }
        }
        getch();
        closegraph();
}
```

Output:



Conclusion: We successfully generated a character generation -bitmap method through this experiment.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

EXPERIMENT 07

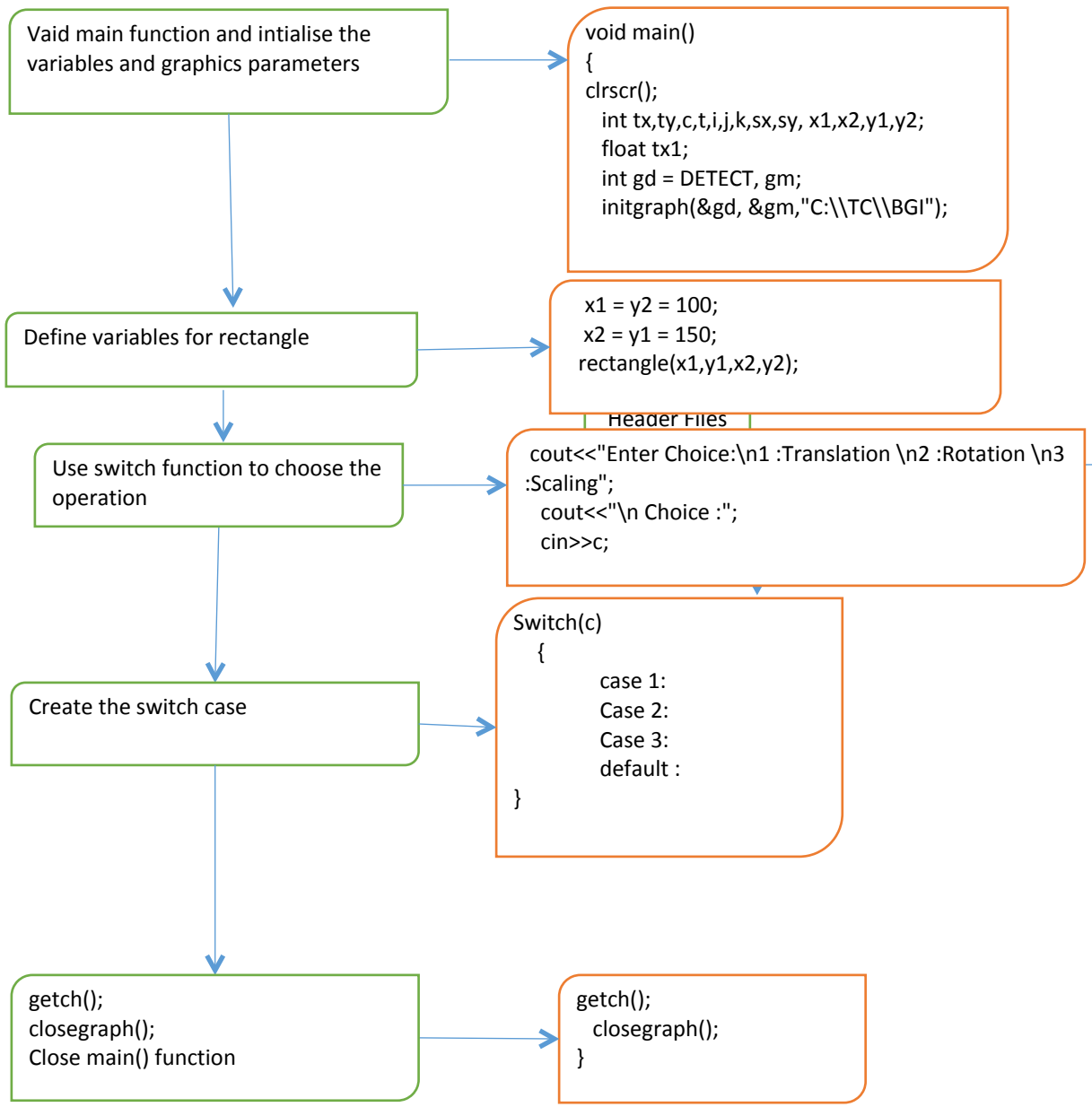
Experiment No : 7

Aim : Write a program in c to implement 2D transformations

i) Translation ii) Rotation iii) Scaling

Software used : Turbo C++

Flow Chart :



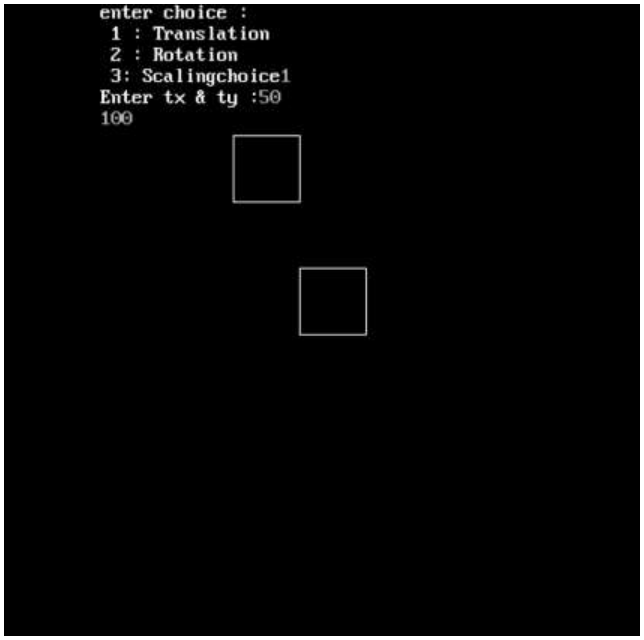
Source Code:

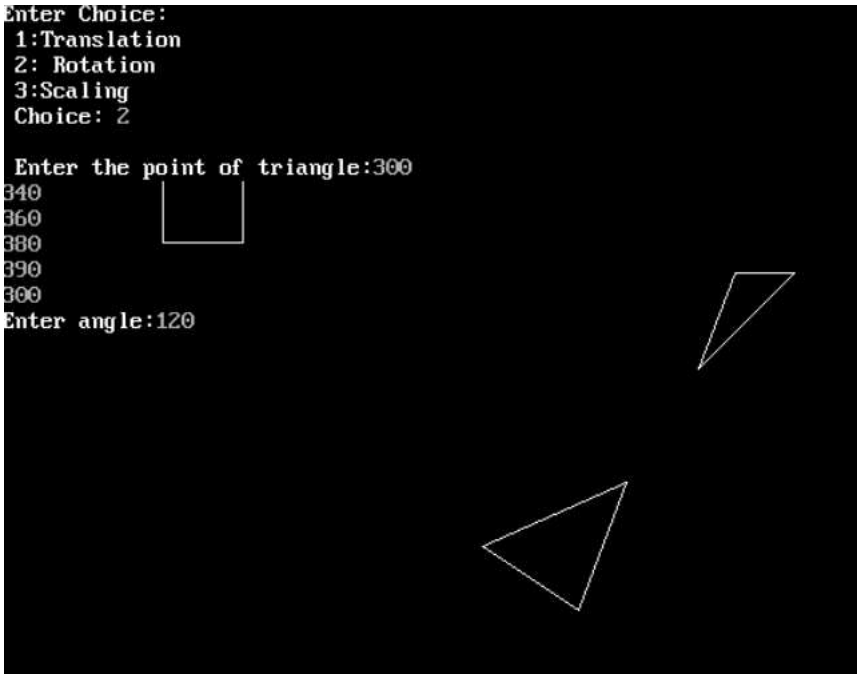
Experiment07 - Notepad

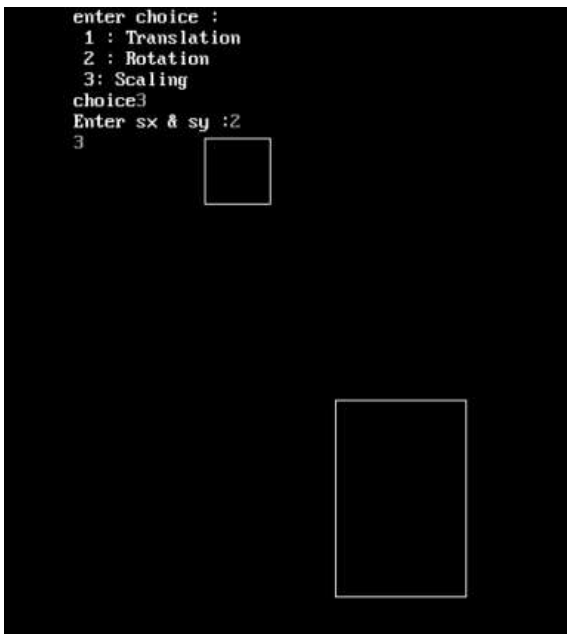
File Edit Format View Help

```
#include<graphics.h>
#include<conio.h>
#include<iostream.h>
#include<math.h>
#include<stdlib.h>
void main()
{
        clrscr();
        int tx,ty,c,t,i,j,k,sx,sy,x1,x2,y1,y2;
        float tx1;
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C://Turboc3//BGI");
        x1=y2=100;
        x2=y1=150;
        rectangle(x1,y1,x2,y2);
        cout<<"Enter choice :\n 1 : Translation\n 2 : Rotation\n 3: Scaling";
        cout<<"\nchoice";
        cin>>c;
        switch(c)
        {
                case 1:
                        int t1,t2,t3,t4;
                        cout<<"Enter tx & ty :";
                        cin>>tx>>ty;
                        t1=x1+tx;
                        t2=y1+ty;
                        t3=x2+tx;
                        t4=y2+ty;
                        rectangle(t1,t2,t3,t4);
                        break;
                case 2 :
                        int X1,Y1,X2,Y2,X3,Y3,r;
                        int rx,nx1,ny1,nx2,ny2,nx3,ny3;
                        cout<<" \n Enter the point of triangle:";
                        cin>>X1>>Y1>>X2>>Y2>>X3>>Y3;
                        line(X1,Y1,X2,Y2);
                        line(X2,Y2,X3,Y3);
                        line(X3,Y3,X1,Y1);
                        cout<<"Enter angle : ";
                        cin>>r;
                        rx=r*(3.14/180);
                        nx1=abs(X1*cos(rx)-Y1*sin(rx));
                        ny1=abs(Y1*cos(rx)+X1*sin(rx));
                        nx2=abs(X2*cos(rx)-Y2*sin(rx));
                        ny2=abs(Y2*cos(rx)+X2*sin(rx));
                        nx3=abs(X3*cos(rx)-Y3*sin(rx));
                        ny3=abs(Y3*cos(rx)+X3*sin(rx));
                        line(nx1,ny1,nx2,ny2);
                        line(nx2,ny2,nx3,ny3);
                        line(nx3,ny3,nx1,ny1);
                        break;
                case 3 :
                        cout<<"Enter sx & sy : ";
                        cin>>sx>>sy;
                        rectangle(x1*sx,y1*sy,x2*sx,y2*sy);
                        break;
                default :
                        cout<<"Not a valid choice";
        }
        getch();
        closegraph();
}
```

Output:







Conclusion: We have learned the algorithm for scaling, translating and rotating a 2-Dimensional object through this experiment.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

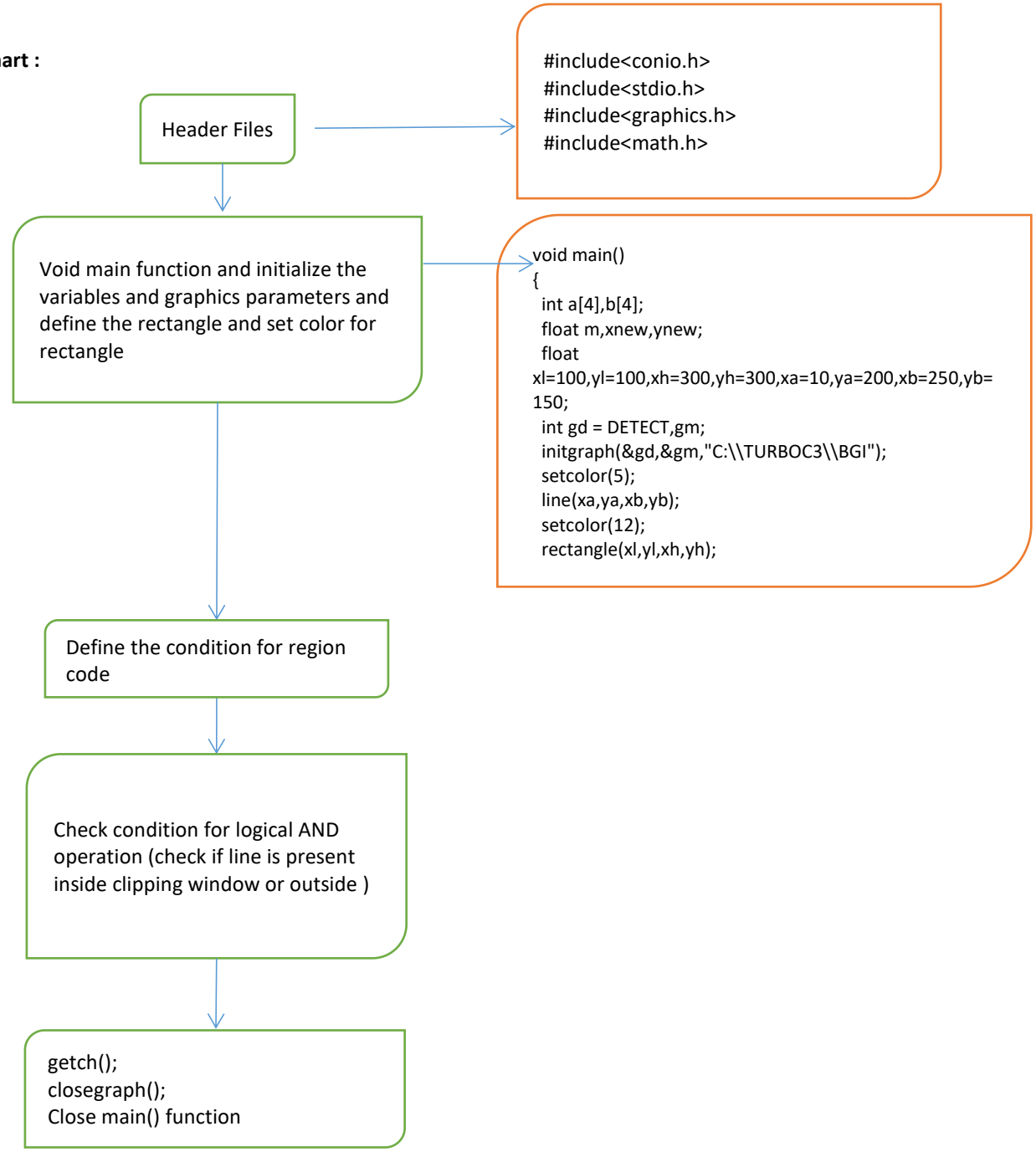
EXPERIMENT 08

Experiment No : 8

Aim : Write a program in c to implement Line clipping algorithm Cohen Sutherland

Software used : Turbo C++

Flow Chart :



Source Code :

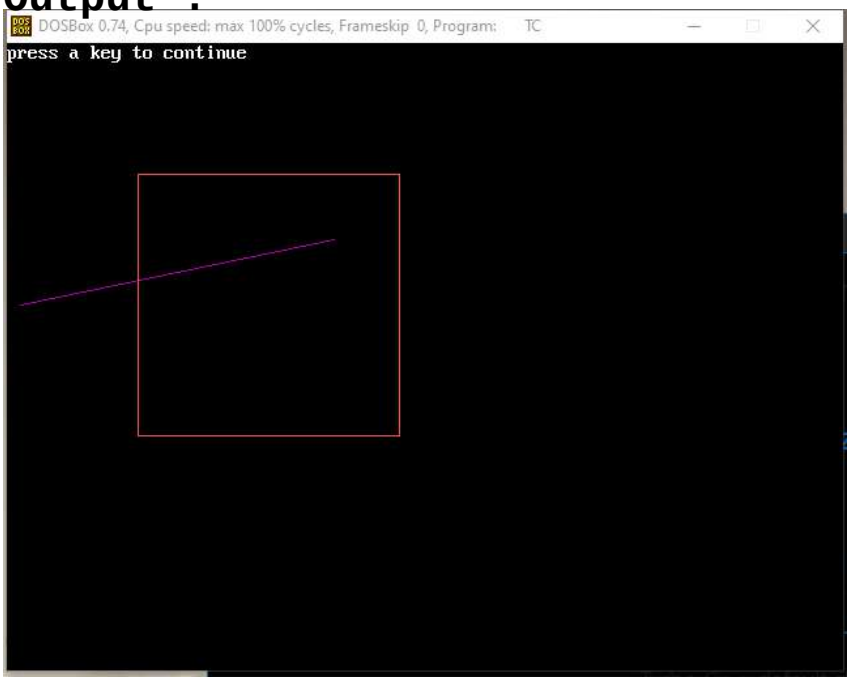
COHENALG.CPP - Notepad

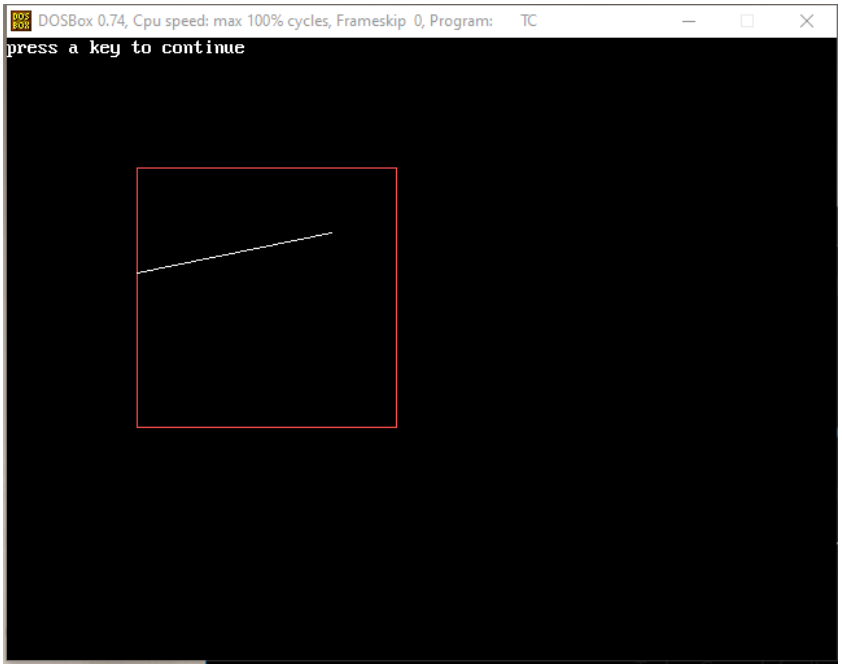
File Edit Format View Help

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

void main()
{
  int a[4],b[4];
  float m,xnew,ynew;
  float xl=100,yl=100,xh=300,yh=300,xa=10,ya=200,xb=250,yb=150;
  int gd = DETECT,gm;
  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
  setcolor(5);
  line(xa,ya,xb,yb);
  setcolor(12);
  rectangle(xl,yl,xh,yh);
  m = (yb-ya)/(xb-xa);

  if(xa < xl)
    a[3] = 1;
  else a[3] = 0;

  if(xa>xh)
   a[2] = 1;
  else a[2] = 0;

  if(ya < yl)
   a[1] = 1;
  else a[1] = 0;

  if (ya > yh)
   a[0] = 1;
   else a[0] = 0;

  if(xb < xl)
    b[3] = 1;
  else b[3] = 0;

  if(xb>xh)
   b[2] = 1;
  else b[2] = 0;

  if(yb < yl)
   b[1] = 1;
  else b[1] = 0;

  if (yb > yh)
   b[0] = 1;
   else b[0] = 0;

  printf("press a key to continue");
  getch();
  if(a[0] == 0 && a[1] == 0 && a[2] == 0 && a[3] == 0 && b[0] == 0 && b[1] == 0 && b[2] == 0 && b[3] == 0 )
  {

   printf("no clipping");
   line(xa,ya,xb,yb);
  }
  else if(a[0]&&b[0] || a[1]&&b[1] || a[2]&&b[2] || a[3]&&b[3])
  {
   clrscr();
   printf("line discarded");
   rectangle(xl,yl,xh,yh);
  }
  else
  {
   if(a[3] == 1 && b[3]==0)
   {
    ynew  = (m * (xl-xa)) + ya;
    setcolor(12);
    rectangle(xl,yl,xh,yh);
    setcolor(0);
    line(xa,ya,xb,yb);
    setcolor(15);
    line(xl,ynew,xb,yb);
   }
```

```
    else if(a[2] == 1 && b[2] == 0)
    {
     ynew  = (m * (xh-xa)) + ya;
     setcolor(12);
     rectangle(xl,yl,xh,yh);
     setcolor(0);
     line(xa,ya,xb,yb);
     setcolor(15);
     line(xl,ynew,xb,yb);
    }
    else if(a[1] == 1 && b[1] == 0)
    {
     xnew = xa + (yl-ya)/m;
     setcolor(0);
     line(xa,ya,xb,yb);
     setcolor(15);
     line(xnew,yh,xb,yb);
    }

    else if(a[0] == 1 && b[0] == 0)
    {
     xnew = xa + (yh-ya)/m;
     setcolor(0);
     line(xa,ya,xb,yb);
     setcolor(15);
     line(xnew,yh,xb,yb);
    }
   }
   getch();
   closegraph();
}
```

Output :





Conclusion : We successfully implement Line clipping algorithm using Cohen Sutherland.

NAME : PRATHAMESH CHIKANKAR
ROLL NO. AND YEAR : AIMLD08 AND SE
BRANCH AND DIV. : CSE(AI&ML) AND D
SUBJECT : COMPUTER GRAPHICS (CG)

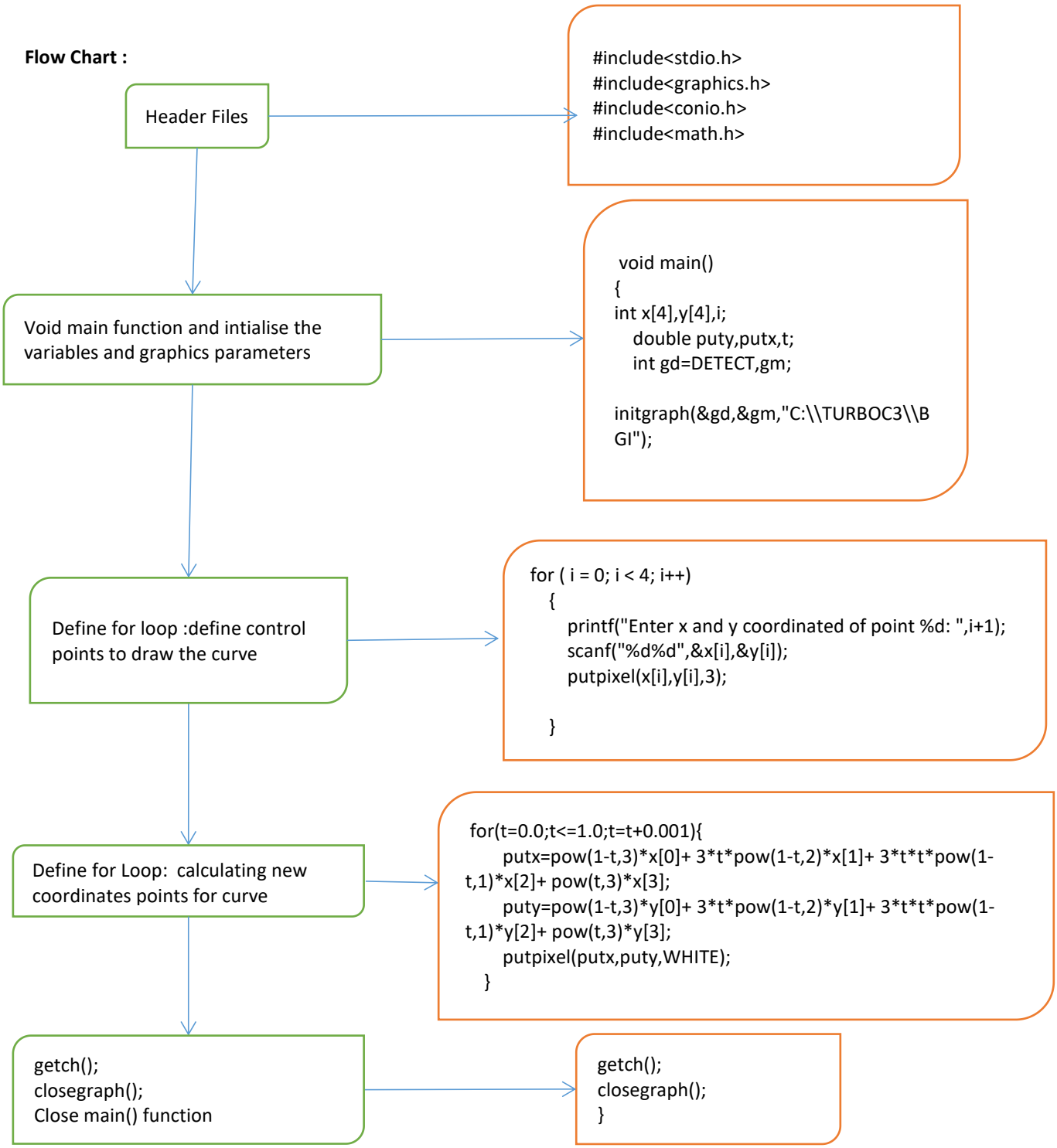
EXPERIMENT 09

Experiment No : 9

Aim : Write a program in c to implement Bezier curve.

Software used : Turbo C++

Flow Chart :



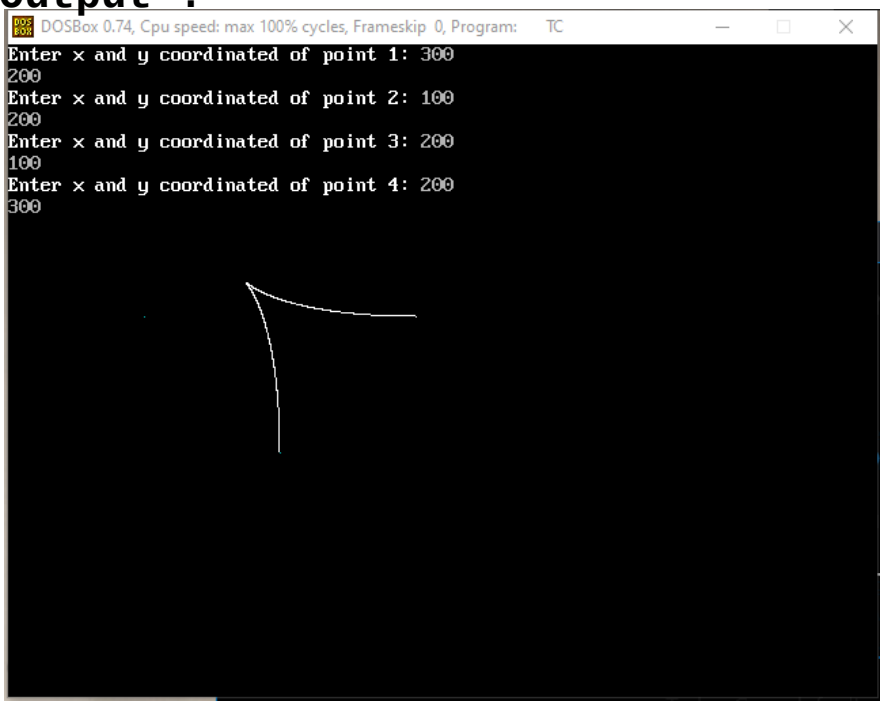
Source Code:

B_CURVE.CPP - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main(){
    int x[4],y[4],i;
    double puty,putx,t;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    for ( i = 0; i < 4; i++)
    {
        printf("Enter x and y coordinated of point %d: ",i+1);
        scanf("%d%d",&x[i],&y[i]);
        putpixel(x[i],y[i],3);

    }
    for(t=0.0;t<=1.0;t=t+0.001){
        putx=pow(1-t,3)*x[0]+ 3*t*pow(1-t,2)*x[1]+ 3*t*t*pow(1-t,1)*x[2]+ pow(t,3)*x[3];
        puty=pow(1-t,3)*y[0]+ 3*t*pow(1-t,2)*y[1]+ 3*t*t*pow(1-t,1)*y[2]+ pow(t,3)*y[3];
        putpixel(putx,puty,WHITE);
    }
    getch();
    closegraph();

}
```

Output :



Conclusion : We successfully implement Bezier curve through this experiment.

Computer Graphics Mini Project

- GUIDED BY VAIBHAV PALAV

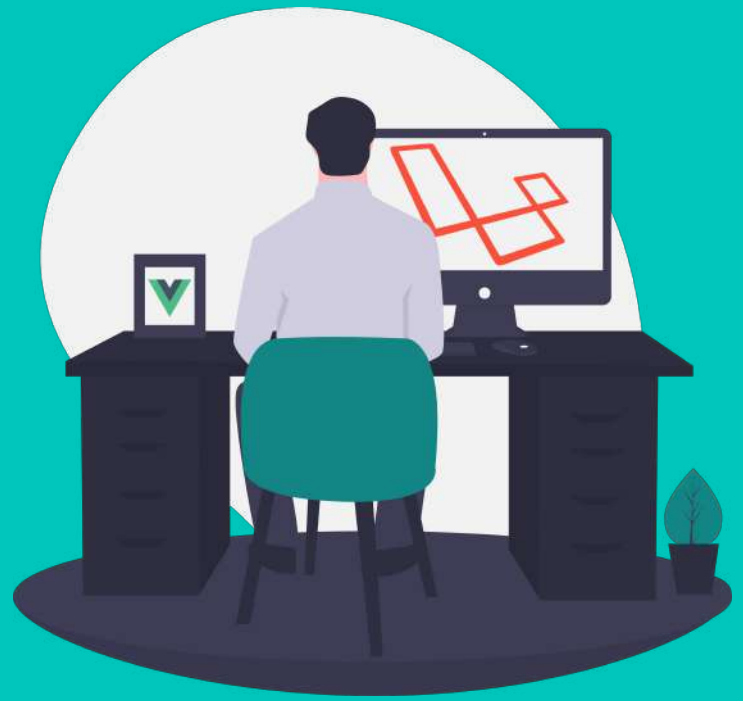
Team Members :

1. AIMLD50_SINGH SUDHAM DHARMENDRA
2. AIMLD03_ANSARI HANZALA MOHD IMAMUDDIN
3. AIMLD41_RUPARELIYA AKSHAR JAYANTI
4. AIMLD08_CHIKANKAR PRATHAMESH SHIVAJI

Visualization Of Signal Transmission

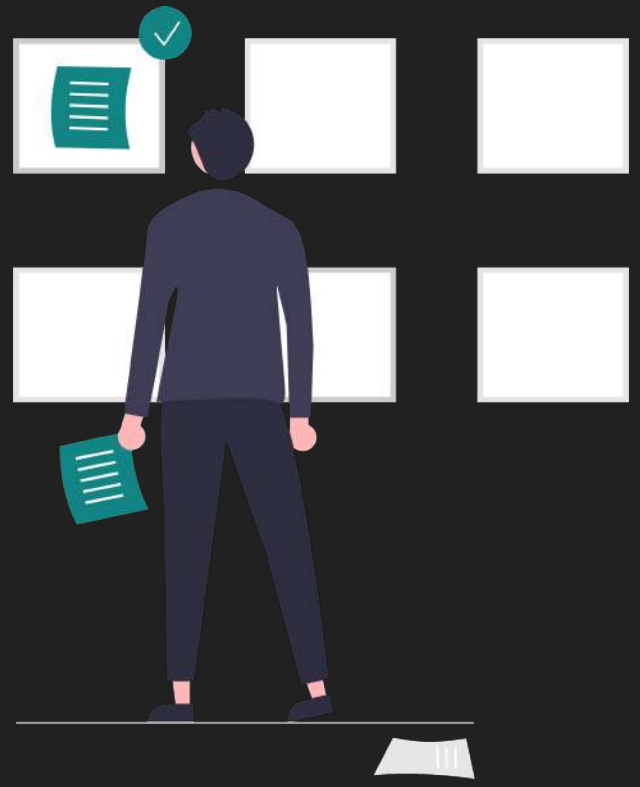
OUTLINE :

1. Introduction
2. Program
3. Output
4. Conclusion



Introduction

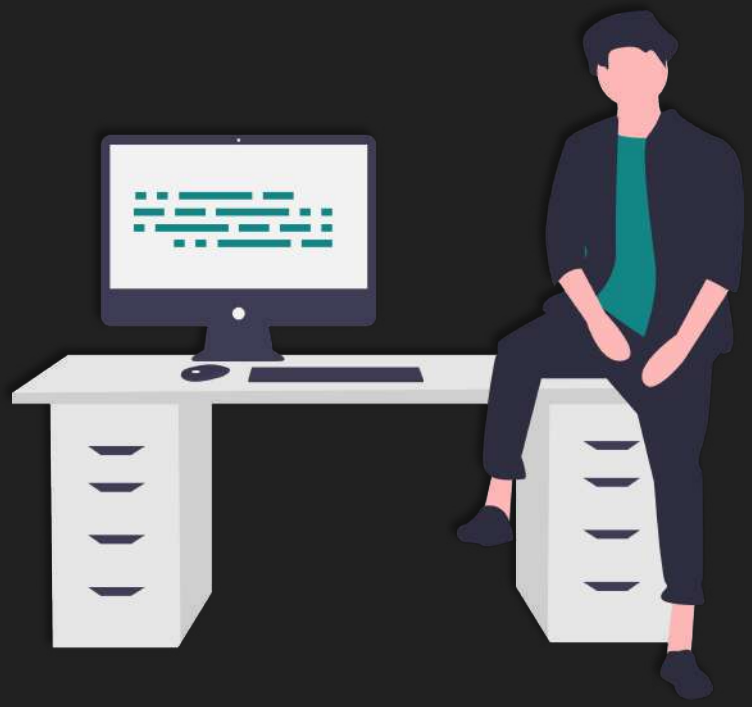
1. In this Mini Project, We are going to develop an Animation using Computer Graphics for Visualization of Signal Transmission .
2. Wireless Communication is the fastest growing and most vibrant technological areas in the communication field.
3. Wireless Communication is a method of transmitting information from one point to other, without using any connection like wires, cables or any physical medium.
4. Antennas are electrical devices that transform the electrical signals to radio signals in the form of Electromagnetic (EM) Waves and vice versa.
5. These Electromagnetic Waves propagates through space. Hence, both transmitter and receiver consists of an antenna.



Introduction

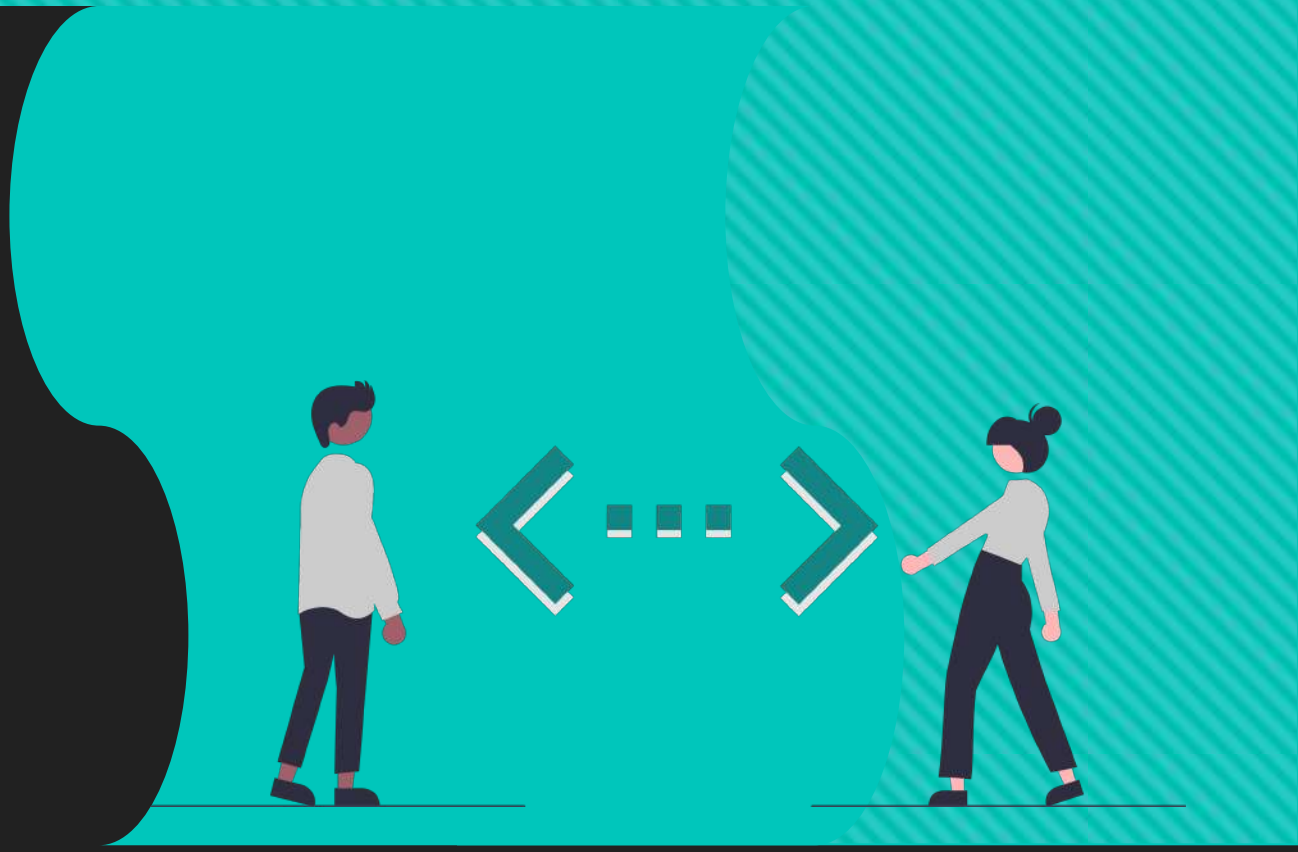
Some Built-in Functions of Graphics :

1. **initgraph() :** initgraph initializes the graphics system by loading a graphics driver from disk and putting the system into graphics mode.
2. **graphresult() :** graphresult returns the error code for the last graphics operation that reported an error and resets the error level to grOk.
3. **getmaxx() :** getmaxx returns the maximum x screen coordinate for the current graphics driver and mode. getmaxx is invaluable for centering, determining the boundaries of a region onscreen, and so on.
4. **getmaxy() :** getmaxx returns the maximum x screen coordinate

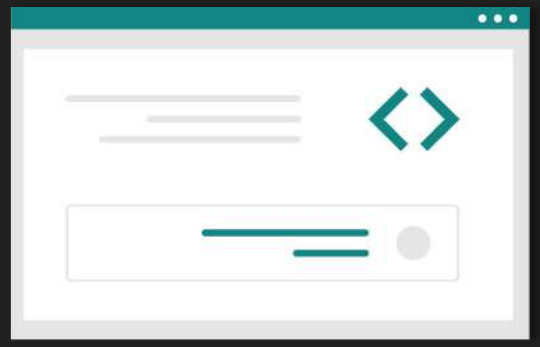


Program

```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
int main()
{
    int gdriver = DETECT, gmode, err;
    int midx, y, radius = 5;
    int k = 0, stangle, endangle;

    initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

    err = graphresult();
    if (err != grOk)
    {   /* error occurred */
        printf("Graphics Error: %s\n",grapherrormsg(err));
        getch();
        return 0;
    }
    /* mid position in x-axis */
    midx = getmaxx() / 2;

    /* calculating the position of y */
    y = (3 * getmaxy()) / 4;

    /* start and end angles of signals */
    stangle = 120;
    endangle = 200;
```

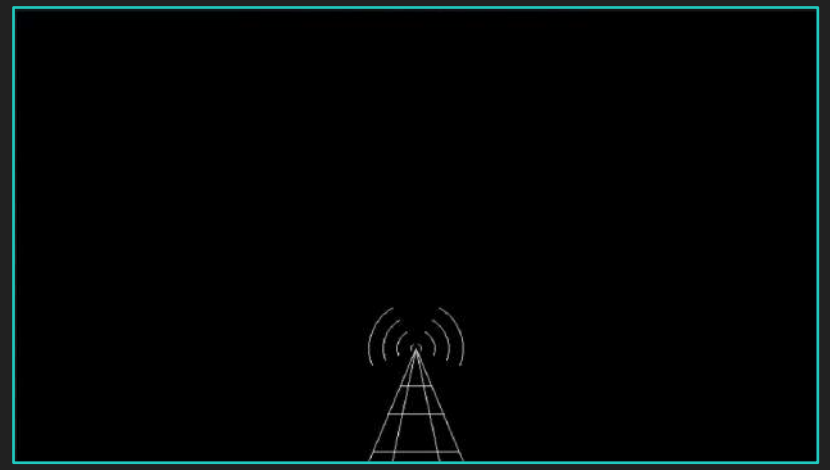


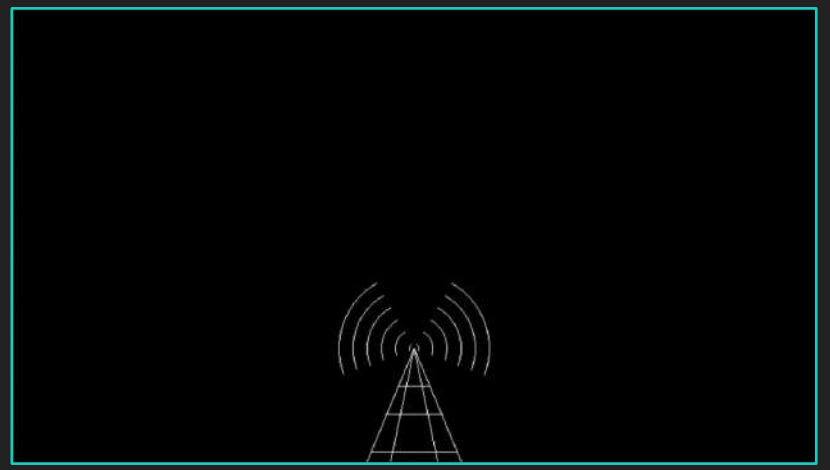
Program

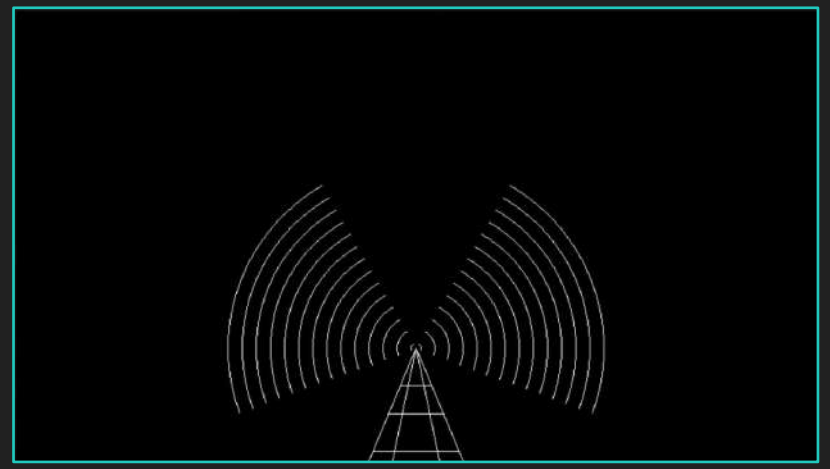


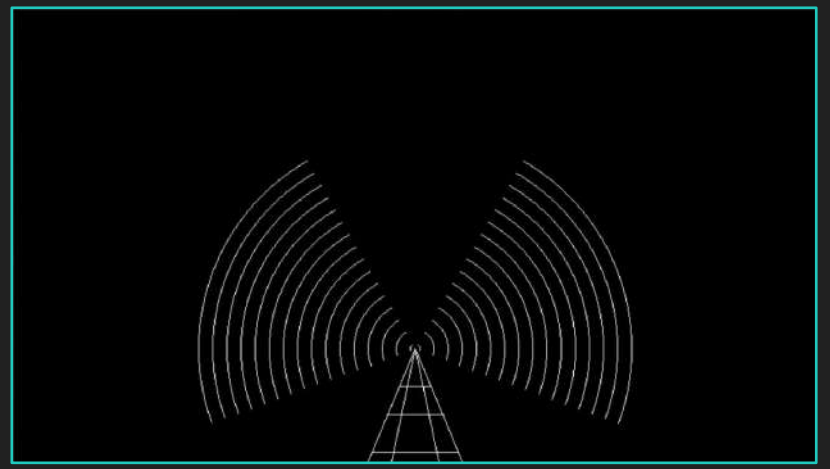
```
    while (!kbhit())
    {
        k = 0, radius = 5;
        /* clears graphic screen */
        cleardevice();

        /* construct antenna using lines */
        line(midx - 50, getmaxy(), midx, y);
        line(midx + 50, getmaxy(), midx, y);
        line(midx - 25, getmaxy(), midx, y);
        line(midx + 25, getmaxy(), midx, y);
        line(midx - 45, getmaxy() - 10, midx + 45, getmaxy() - 10);
        line(midx - 30, getmaxy() - 50, midx + 30, getmaxy() - 50);
        line(midx - 16, getmaxy() - 80, midx + 16, getmaxy() - 80);
        /* signals from the antenna */

        while (k < 18)
        {
            /* signal at the left side */
            arc(midx, y, stangle, endangle, radius);
            /* signal at the right side */
            arc(midx, y, 0, 60, radius);
            arc(midx, y, 340, 360, radius);
            radius = radius + 15;
            delay(50);
            k++;
        }
    }
    getch();

    /* deallocate memory allocated for graphic screen */
    closegraph();

    return 0;
}
```

Output



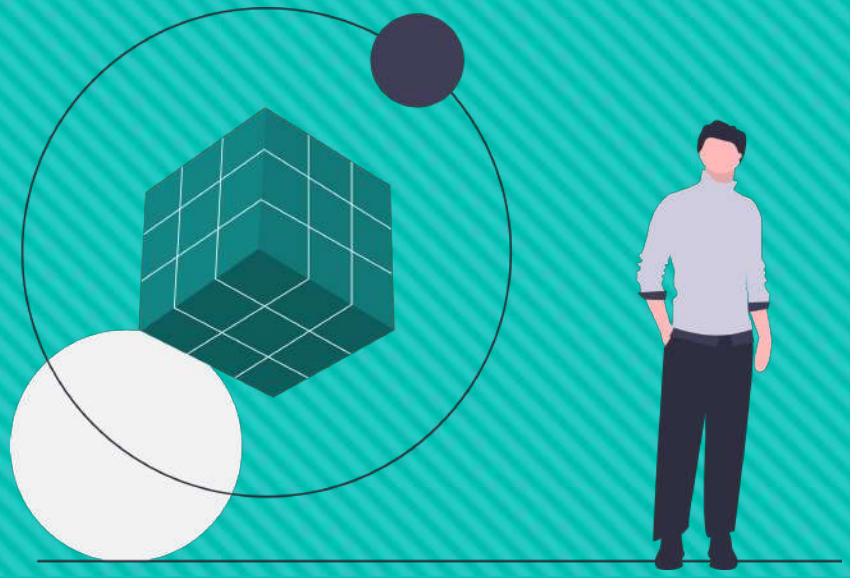






Conclusion

- ✔ From the above program and output, we conclude that the project is able to perform Visualization of Signal Transmission i.e. the Transmission of Electromagnetic Waves from an antennas or Satellites .
- ✔ Aim for the Project is Successfully Achieved !!!



Thank You

PAGE: a)
DATE: / /

Name: Pratham Chikankar

Roll No. and Year: AIML008 and SE

Branch and Div.: CSE (AI & ML) and D

Subject: Computer Graphics (CG)

Assignment No. 1

last date of submission: 20/11/21

AIMLD08

PAGE : 01
DATE : / /

Q.1. Differentiate between Raster Scan and Random Scan devices

Ans:-

Raster Scan	Random Scan
Resolutn of raster scan is lesser or lower than random scan	while the resolution of random scan is higher than Raster Scan
Alltratn is not so easy	Any alltratn is easy in comparison of Raster scan.
Interlacing is used	Interlacing is not used.
Electron beam is directed from top to bottom & one row at time on screen. It is directed to whole screen.	Electron beam is directed to only that part of screen where picture is required to be drawn, one line at a time.
It stores picture definitn a set of intensity values of the pixels in the frame buffer.	It stores picture definitn as a set of line commands in the Refresh buffer
example: pen plotter	eg:- TV sets

Q.2. Explain DDA line drawing algorithm and plots the points for line AB A(10, 15) and B(5, 25) using it.

Ans:- • DDA is an incremental method of scan conversion of line. In this method calculation is performed at each step but by using results of previous steps.

Suppose at step i, the pixels is $(x, y) = (x_i, y_i)$

the line of equation for step i

$y_i = mx_i + b \rightarrow$ eqn 1)

Next value will be

$y_{i+1} = mx_{i+1} + b \rightarrow$ eqn 2)

$\therefore m = \Delta y / \Delta x$ where $\Delta y = y_{i+1} - y_i \rightarrow$ eqn 3)

and $\Delta x = x_{i+1} - x_i \rightarrow$ eqn 4)

$\therefore \Delta y = m\Delta x \Rightarrow \Delta x = \Delta y / m$

$\therefore x_{i+1} = x_i + \Delta x$

$x_{i+1} = x_i + \Delta y / m$

case i : when $|m| < 1$ then

$x = x$, $y = y$, set $\Delta x = 1$

$y_{i+1} = y_i + m$, $x = x + 1$

until $x = x_2$

case ii : when $|m| > 1$ then

$x = 1$, $y = y$, set $\Delta y = 1$

$x_{i+1} = 1/m$, $y = y + 1$,

until $y \rightarrow y_2$

PTO

- Let $P_1(10,15)$ $P_2 = (5,25)$

$\Rightarrow x_1 = 10$ $y_1 = 15$

$x_2 = 5$ $y_2 = 25$

$dx = 5-10 = -5$

$dy = 25-15 = 10$

Since $|dy| > |Dx|$ the line is of step slope category

$\therefore$ steps $= (dy)_{abs} = 10$

$x_{increment} = dx/steps = -5/10 = -0.5$

$y_{increament} = dy/step = 10/10 = 1$

we know 1st point, lets plot as (10, 15)

$\therefore x_{new} = x_{old} + x_{increment} = 10 - 0.5 = 9.5$

$y_{new} = y_{old} + y_{increment} = 15 + 1 = 16$

the same way we will calculate points till y_{new} becomes equal to $y_2 = 2.5$

$\therefore$

x	y	plot
10	15	(10,15)
9.5	16	(9.5, 16)
9	17	(9, 17)
8.5	18	(8.5, 18)
8	19	(8, 19)
7.5	20	(7.5, 20)
7	21	(7, 21)
6.5	22	(6.5, 22)
6	23	(6, 23)
5.5	24	(5.5, 24)
5	25	(5, 25)

Q.3. Derive the composite transformation matrix to scale an object with respect to a fixed point.

Ans:- To determine the general form of the scaling matrix with respect to a fixed point P(h,k) we have to perform three steps:

1. Translate point P(h,k) at the origin by performing translation (T1).
2. Scale the point or object by performing Scaling (S)
3. Translate the origin back by performing reverse translation (T2)

therefore, the general form of the scaling matrix is a composition of T1 S T2 matrices. It can be given as

$$S_p = T_1 S T_2$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ -h & -k & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ h & k & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -h & -k & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ h & k & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ -S_x h + h & S_y k + k & 1 \end{bmatrix}$$

Q. 4. Derieve the 2D transformation matrix to reflect a figure about the line $y = mx$

Ans:- equation of line $y = mx + c$

Slope $= m$ & intercept $= c$

we can relate slope m to angle θ by equation $m = \tan\theta$

$\therefore \theta = \tan^{-1} m$ ... where θ is the angle of line with x-axis

Translation matrix can be given as

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

Rotation matrix to match given line with x-axis can be obtained as

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

Reflection matrix about x-axis

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

Inverse transformation matrices,

$$R_z^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

$\therefore$ final transformation matrix can be obtained

$$R_T = T \cdot R_z \cdot M \cdot R_z^{-1} \cdot T^{-1}$$

PTO

$$\sin\theta = \frac{m}{\sqrt{m^3+1}} \qquad \cos\theta = \frac{m}{\sqrt{m^3+1}}$$

... as we have $\tan\theta = m$, using trigonometric identities we obtained.

$$\therefore R_T = \begin{bmatrix} \cos 2\theta & \sin 2\theta & 0 \\ \sin 2\theta & -\cos 2\theta & 0 \\ -c\sin 2\theta & c(1+\cos 2\theta) & 1 \end{bmatrix}$$

By substituting values of $\sin\theta$ and $\cos\theta$ we have

$$R_T = \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & 0 \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & 0 \\ \frac{-2cm}{m^2+1} & \frac{2c}{m^2+1} & 1 \end{bmatrix}$$

Q. 5. Specify midpoint circle algorithm using the same plot circle whose radius is 5 units and centre is at (10, 10).

Ans:- The circle has radius as 5 units with centre at (10, 10)

$\therefore r = 5, \; x_c = 10, \; y_c = 10$

plot 1st point as (15, 10)

similarly by symmetry

PTO

plot (10, 15)
(5, 10)
(10, 5)

Now find $p = 1 - r = 1 - 5 = -4$

Till $(x < y)$ we have to perform the following

if $(p < 0)$

$p = -3$

so, we have to increment x by 1

Modify p as

$p = p + 2x + 1$

i.e $p = -2 + 2 \times 10 + 1 = -1$

Here we are not increment y plot $(x+1, y)$ i.e. (1, 5)

and if $p > 0$ then $x = x + 1$, $y = y - 1$ &

put $p = p + 2(x - y) + 1$

plot $(x+1, y+1)$.

∴

x	y	d	pixel
0	5	-4	(10, 15)
1	5	$-4+2\times1+1$	(11, 15)
2	4	$-1+2\times2+1$	(12, 14)
3	3	$14+2\times3+1$	(13, 13)
4	2	$11+2\times4+1$	(14, 12)
5	1	$20+2\times5+1$	(15, 11)

PAGE: a)
DATE: / /

Name : Pratamesh chikankar

Roll No. and year : AIML008 and SE

Branch and Div. : CSE-(AI & ML) and D

Subject : Computer Graphics (CG)

Assignment No. 2

last date of submission : 25/11/21

AIM1D08
PAGE: 01
DATE: / /

1. What is the purpose of Inside-outside Test, explain any one method

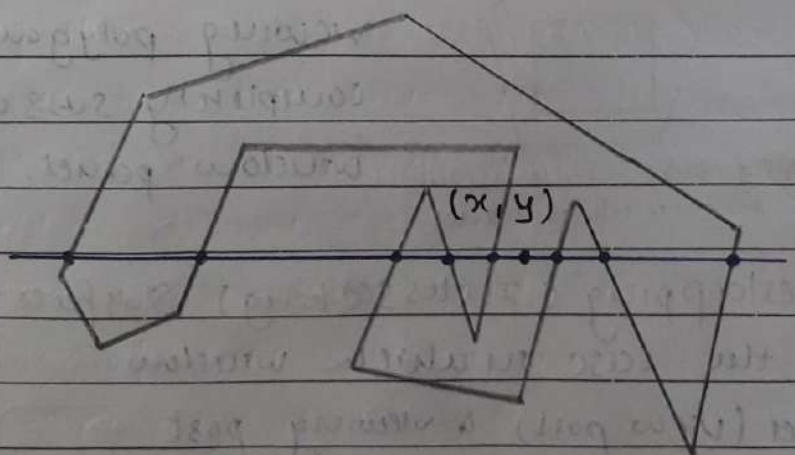
Ans :-

Inside - outside test:

This method is also known as counting number method. While filling an object, we often needs to identify whether particular point is inside the object or outside it. There are two methods by which we can identify whether particular point is inside an object or outside. 1. odd-Even Rule
2. Non-zero winding number rule

Odd-Even rule:

In this technique, we will count the edges crossing along the line from any point (x,y) to infinity. If the number of interactions is odd, then the point (x,y) is an interior point; and if the number of interactions is even, then the point (x,y) is an exterior point. The following example depicts this concept.



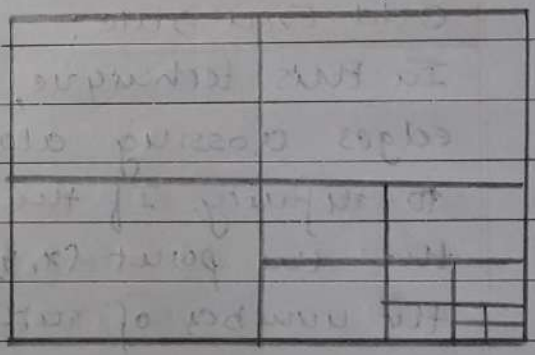
From the above figure, we can see that from the point (x,y) the number of interactions point

on the left side is 5 and on the right side is 3. from the both ends, the number of intersection point is odd, so that the point is considered within object.

---

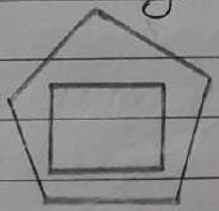
2. Explain Area subdivision algorithm for hidden surface removal.

Ans:- The area subdivision algorithm is based on the divide and conquers methods where the visible (viewing) area is successively divided into smaller and smaller rectangles until the simplified area is detected. considers given with the window panel, where the polygon will be projected, is as follows:



when we subdivided the window panel against the polygon we may come through the following cases which are as follows:

1. surrounding surface:

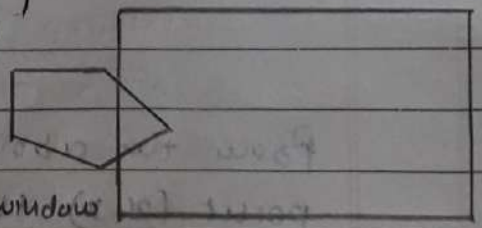


polygon has circumscribed the window panel

It's the case in which the viewing polygon surface completely surrounds the whole window panel.

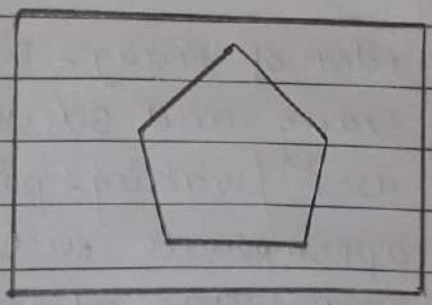
2. overlapping (Intersecting) surface:

It's the case in which window panel (view port) & viewing port polygon surface both intersects each other.



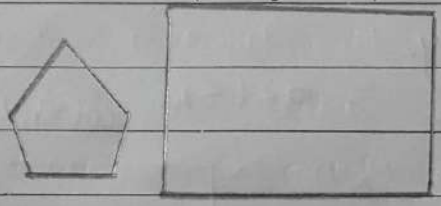
Both window panel & polygon are overlapping each others surface.

3. Inside (contained) surface:
In this case, in which the whole polygon surface is inscribed inside the window panel. This case is just opposite to the first (surrounding surface) case.



complete polygon has inscribed inside the window panel.

4. Outside (disjoint) surface:



In this case, the whole polygon surface is completely outside the window panel.

polygon is completely outside of the window panel.

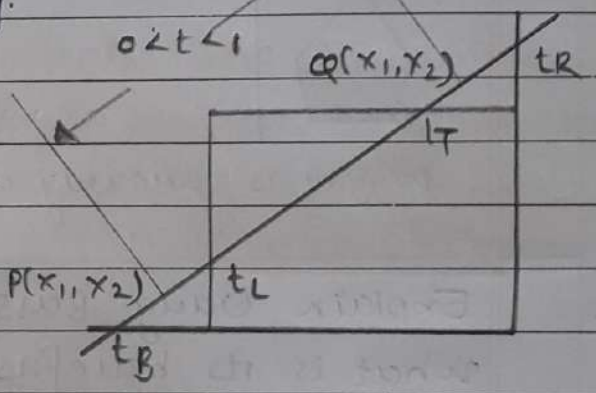
---

3. Explain Liang Barsky line clipping algorithm, what is its benefits over cohen Sutherland algorithm? clip the line with co-ordinates (5,10) and (35,30) against the window $(x_{min}, y_{min}) = (10,10)$ and $(x_{max}, y_{max}) = (20,20)$.

Ans:- • The Liang-Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping box to determine the intersections b/w the line and clipping box. with these intersections it knows which portion of the line should be drawn. This algorithm is more efficient than cohen-sutherland. The idea is for clipping line of Liang-Barsky and Cyrus-Beck are the same. The only difference is Liang-Barsky algorithm has been optimized for an upright rectangular clip window. So we will study only the

idea of Liang-Barsky.

- Liang and Barsky have created an algorithm that uses floating-point arithmetic but finds the appropriate ends points with at most four computations. This algorithm uses the parametric equations for a line and solves for inequalities to find the range of the parameter for which the line is in viewport.
- Let $P(x_1, y_1)$, $Q(x_2, y_2)$ be the line which we want to study.



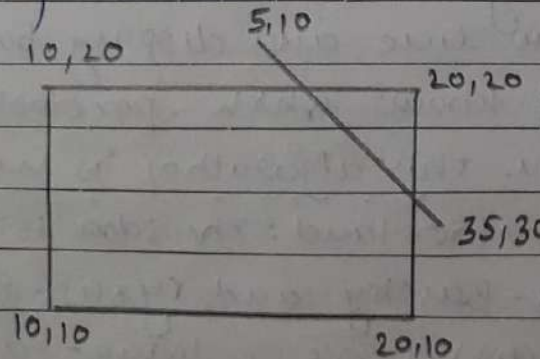
The parametric equation of line segment form gives x-value and y-value for every point in terms of a parameter that ranges from 0 to 1. The equations are:

$$x = x1 + (x2 - x1) * t = x1 + dx * t$$

and $$y = y1 + (y2 - y1) * t = y2 + dx * t$$

we can see that when $(t=0)$ the point computed is $P(x_1, y_1)$ and when $t=1$, the point computed is $Q(x_2, y_2)$.

- Lets first draw the window and line as shown in fig.



Given things are

$x_L = 10$, $y_B = 10$, $x_R = 20$

$y_T = 20$

Lets call a line AB with its coordinates as $x1 = 5$, $y_1 = 10$, $x_2 = 35$, $y_2 = 30$

Now we have to find D_x and D_y as

$$D_x = x_2 - x_1 = 35 - 5 = 30$$

$D_y = Y_2 - Y_1 = 30 - 10 = 20$

lets calculate the values of parameters P and Q as

$P_1 = -D_x = -30$

$P_2 = D_x = 30$

$P_3 = -D_y = -20$

$P_4 = D_Y = 20$

Now, $Q_1 = X1 - X_L = 5 - 10 = 5$

$Q_2 = X_R - X_2 = 20 - 15 = 5$

$Q_3 = Y_1 - Y_B = 10 - 10 = 0$

$Q_4 = Y_T - Y_2 = 20 - 10 = 10$

Now lets find P_i

$P_1 = Q_1 / P_1 = -5/(-30) = (1/6)$

$P_2 = Q_2 / P_2 = 5/30 = (1/2)$

$P_3 = Q_3 / P_3 = 0/(-20) = 0$

$P_4 = Q_4 / P_4 = 10/20 = (1/2)$

Now $t_1 = Max(1/6, 0, 0) = 1/6$

$t_2 = Min(1/2, 1/2, 0) = 1/2$

Now $x_1' = x1 + Dx * t1 \Rightarrow 5 + 30 * (1/6) \Rightarrow 10$

$Y_1' = Y1 + DY * t1 \Rightarrow 10 + (20) * (1/6) \Rightarrow 13.33$

And with t_2 it will be

$x_2' = x1 + Dx * t_2 \Rightarrow 5 + 30 * (1/2) \Rightarrow 20$

$Y_2' = Y1 + Dy * t_2 \Rightarrow 10 + (20) * (1/2) \Rightarrow 20$

from this will come to know that a point (20,20) is an intersection point with respect to the edge of the window boundary. so we need to discard the line from point (5,10) to (5, 83.33) and consider line (20,20) to (35,30).

4. Define window, viewport, object in object space, Image in image space, world coordinate system, physical device coordinate system, obtain viewing transformation matrix.

Ans:- **window**: A world-coordinate area selected for display is called a window. In computer Graphics a window is a graphical control element.

viewport: An area on a display device to which a window is mapped is called a viewport. A viewport is a polygon viewing region in computer Graphics.

Object in object space: space where object model resides. Compares objects and its part of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.

Image in image space: the optical space co-ordinating the visual representation or component of scene.

world coordinate system: Once the individual object shapes have been specified, we can place the objects into appropriate position within the scene using a reference frame called world coordinate. object space contains the dimensions as actual.

physical device coordinate system: The corresponding coordinate system on the display device where the image of the picture is displayed.

Obtain viewing transformation matrix: Mapping of coordinates of points and lines that form the picture into appropriate coordinates on the display device.

5. What is meant by parallel and perspective projection? Derive matrix for perspective projection.

Ans:- Parallel projection:

i) In this projection, z-coordinates is discarded & lines from each vertex on the object are extended until they intersect the view plane.
ii) The point of intersection is the projection of vertex.
iii) We connect the projected vertices by line segment which correspond to connections on the original object.
iv) A line projection preserves relative proportion of objects
v) accurate views of the various sides of an object are obtained with a line projection. But not a realistic program representation.

Perspective projection:

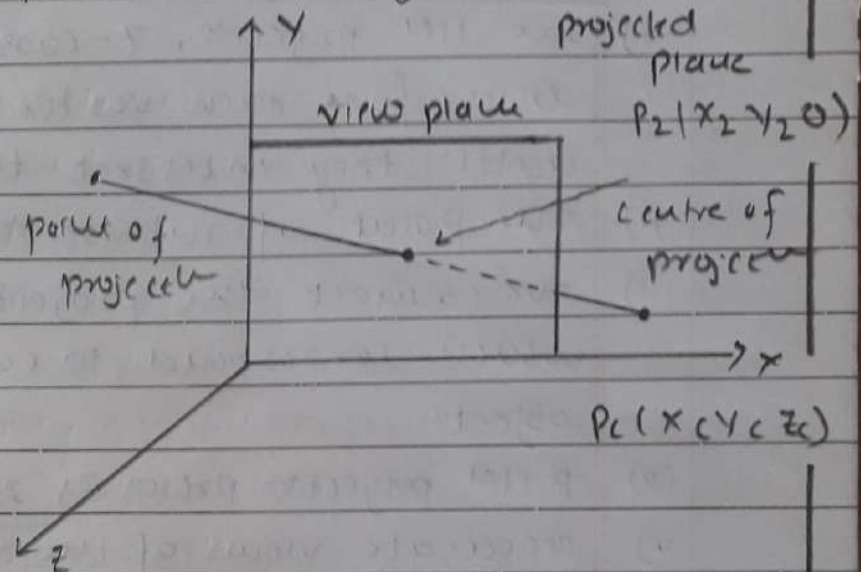
i) In perspective projection, the lines of projection aren't line.
ii) perspective projection transforms object position to the view plane while converging to centre point of projection.
iii) In this all the projection are converge at a single point called "centre of projection" or "projection reference point".
iv) perspective projection produces realistic views but does not preserve relative proportions.
v) projection of distant objects are smaller than the projection of objects of the same size that are closer to the projection plane.

Matrix for perspective projection:

Let us consider the centre of projection is at $P_c(x_c, y_c, z_c)$

$P_c(x_c, y_c, z_c)$ and the point on object is $P_1(x_1, y_1, z_1)$ $P_1(x_1, y_1, z_1)$, then the parametric eqn for the line containing these points can be given as

$$x_2 = x_c + (x_1 - x_c)U$$
$$y_2 = y_c + (y_1 - y_c)U$$
$$z_2 = z_c + (z_1 - z_c)U$$



for projected point z_2 is 0, therefore the third equation can be written as

$$0 = z_c + (z_1 - z_c)U$$
$$U = -z_c / z_1 - z_2$$

Substituting value of U in first two eqns we get,

$$x_2 = (x_c - z_c) * (x_1 - x_c) / (z_1 - z_c)$$
$$= x_c z_1 - x_c z_c - x_1 z_c + x_c z_c / z_1 - z_c$$
$$= x_c z_1 - x_1 z_c / z_1 - z_c$$
$$y_2 = (y_c - z_c) * (y_1 - y_c) / (z_1 - z_c)$$
$$= y_c z_1 - y_c z_c - y_1 z_c + y_c z_c / z_1 - z_c$$

The above eqn can be represented in the homogenous matrix form as given below,

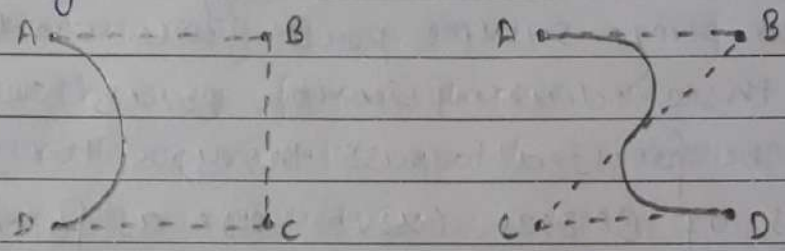
$$[x_2 y_2 z_2 1] = [x, y, z, 1] \begin{bmatrix} -z_c & 0 & 0 & 0 \\ 0 & -z_c & 0 & 0 \\ x_c & y_c & 0 & 1 \\ 0 & 0 & 0 & -z_c \end{bmatrix} \begin{bmatrix} -z_c & 0 & 0 & 0 \\ 0 & -z_c & 0 & 0 \\ x_c & y_c & 0 & 1 \\ 0 & 0 & 0 & -z_c \end{bmatrix}$$

Here we have taken the centre of projection as $P_c(x_c y_c z_c)$ if we take the centre of projection on the -ve z axis such that $x = 0$, $y = 0$ and $z = -z_c$ //

6. Explain Bezier curve and Bspline curve? State the various properties of Bezier curve.

ANS:- Bezier curve:

- It is a different way of specifying a curve, so the same shapes can be represented by B-spline and Bezier curve. The cubic Bezier curve require four sample points, these points completely specify the curve.

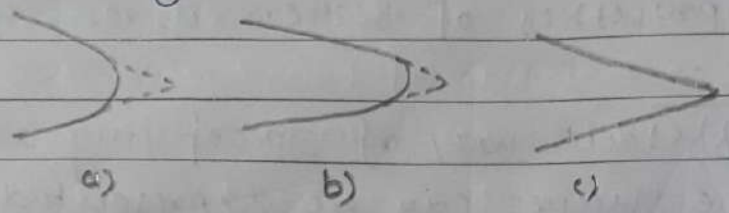


- The curve begins at the first sample point and ends at fourth point. If we need another Bezier curve then we need another four sample points. But if we need two Bezier curves connected to each other, then with six sample points we can achieve it. For this, the third and fourth point of first curve should be made same as first & second point of curve.

B spline curve:

- A set of blending functions which takes this approach is called B-splines. Basically spline means a strip, which we have to move around the sample points. Generally the B-spline blending functions were designed to eliminate sharp corners in the curve and the curve does not usually pass through the sample points. But if we need sharp corners we can produce it by

using many identical sample points.



- Fig a) shows the curve using one sample point. Fig b) shows the curve using two identical sample points. It means we use calling the blending function for same sample point twice. and fig. c) shows if these identical sample points are used the curve will be focused to pass through the sample points.

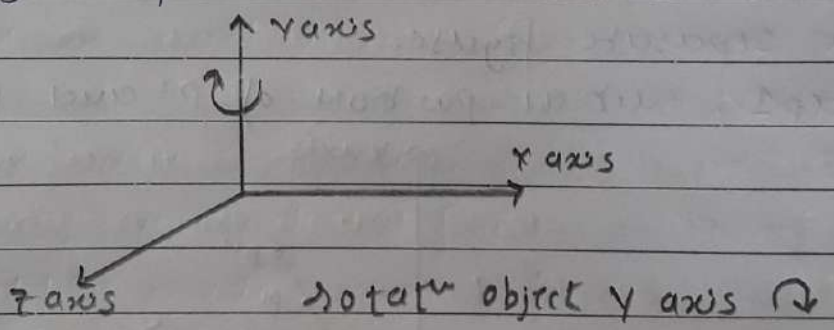
Properties of Bezier curve are as follows:

1. The basic functions are real in nature
2. Bezier curve always passes through the first and last control points i.e. curve has same end points as the guiding polygon.
3. The degree of polynomial defining the curve segment is one less than the number of defining polygon point.
4. The curve generally follows the shape of the defining polygon.
5. The curve lies lies entirely within the convex hull formed by four control points.
6. The curve is invariant under an affine transform
7. The direction of tangent vector at the end point is same as that of the vector determined by first and last segments.
8. The curve exhibits variation diminishing property.

7. Explain 3D rotation about an arbitrary axis.

Ans:- When the object is rotated about an axis that is not ||el to any one of coordinate axis, i.e. x, y, z. Then additional transformation are required. First of all, alignment is needed, and then the object is being back to the original position. Following steps are required.

1. Translate the object to the origin.
2. Rotate object so that axis of object coincide with any of coordinate axis.
3. Perform rotation about co-ordinate axis with whom coinciding is done.
4. Apply inverse rotation to bring rotation back to the original position.

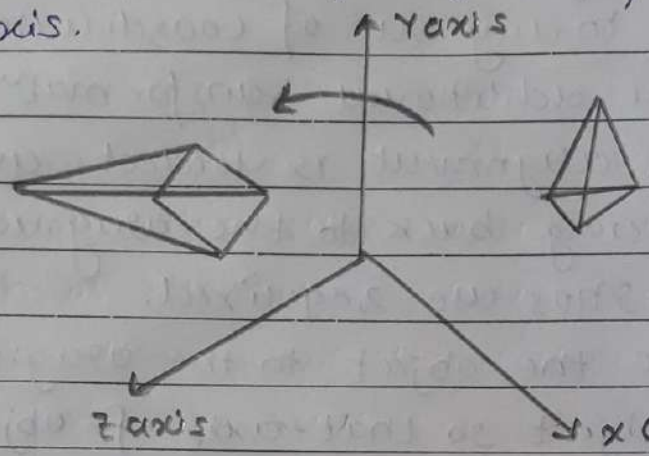


Matrix for representing 3-D rotation about

Z-axis | X-axis | Y-axis

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

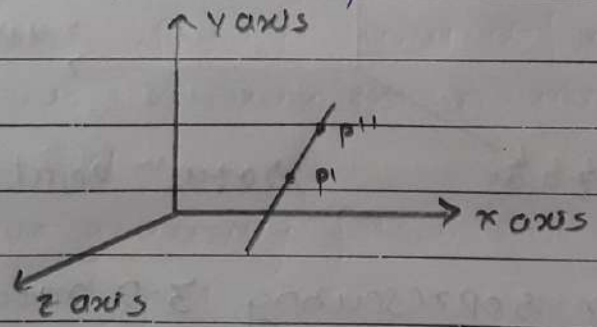
Following figure show the original position of object and position of object after rotation about the x-axis.



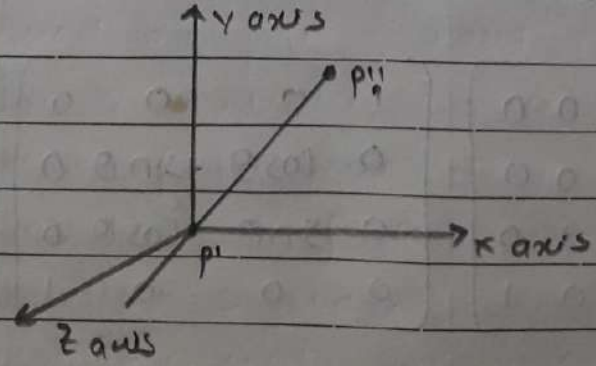
5. Apply inverse translate to bring rotation axis to the original position.

For such transformations, composite transformations are required. All the above steps are applied on points P' and P''. Each step is explained using a separate figure.

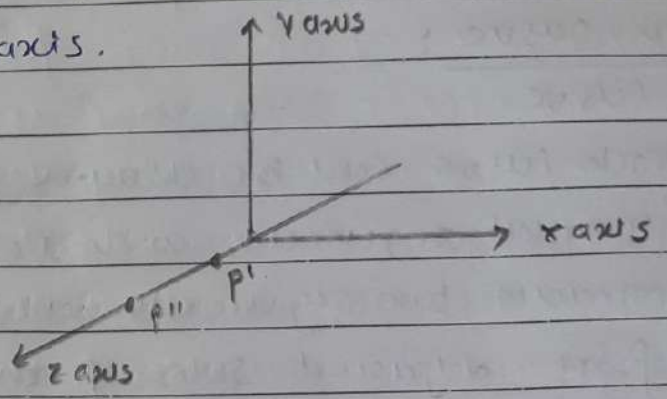
Step 1: Initial position of P' and P'' is shown



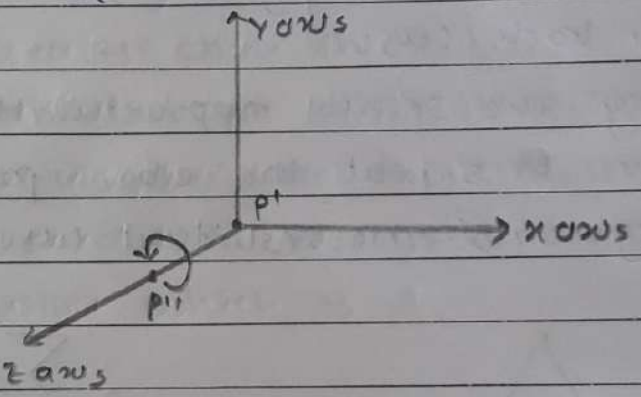
Step 2: Translate object P' to origin



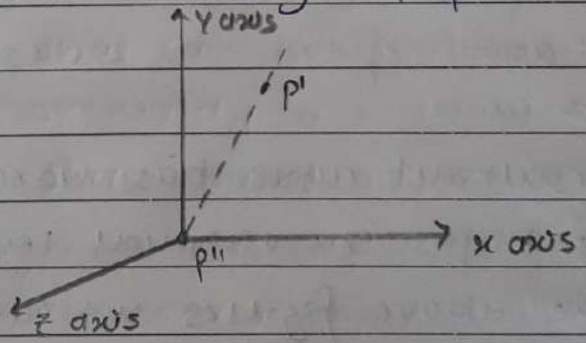
Step 3: Rotate P'' to z axis so that origin aligns along the z axis.



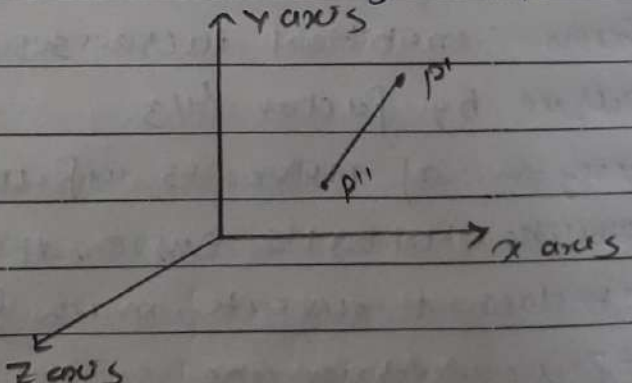
Step 4: Rotate about around z-axis



Step 5: Rotate axis to the original position



Step 6: Translate axis to the original posit

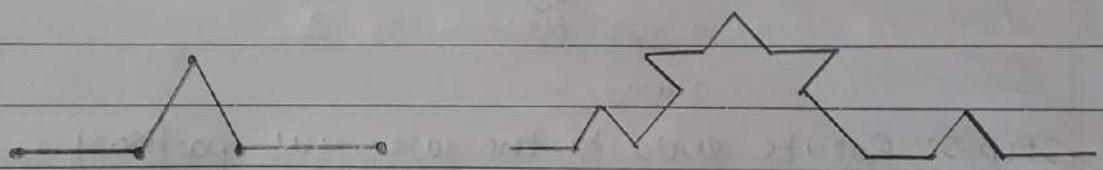


8. a) Explain Koach curve and factals

Ans:- **Koach curve :**

Koch curve

- The Koch curve can be drawn by dividing line into 4 equal segments with scaling factor 1/3 and middle two segments are so adjusted that they form adjacent sides of an equilateral triangle as shown in the figure a). This is the first approx. to the koch curve.
- To apply the second approximation to the Koch curve we have to repeat the above process for each of the four segments - The resultant curve is shown in fig. b)



a) first approx. of the koch curve

b) the second approx. to Koch curve.

- The resultant curve has more wiggles and its length is 16/9 times the original length.
- From the above figures we can easily note following points about the Koch curve:
  - > Each repetition increases the length of the curve by factor 4/3.
  - > length of curve is infinite
  - > Unlike Hibegtis curve, it doesn't fill an area.
  - > It doesn't deviate much form its original shape.
  - > If we reduce the scale of the curve by 3 we find the curve that looks just like the original one; but we must assemble 4 such curves to make the

originals, so we have

$4 = 3^D$

Solving for D we get

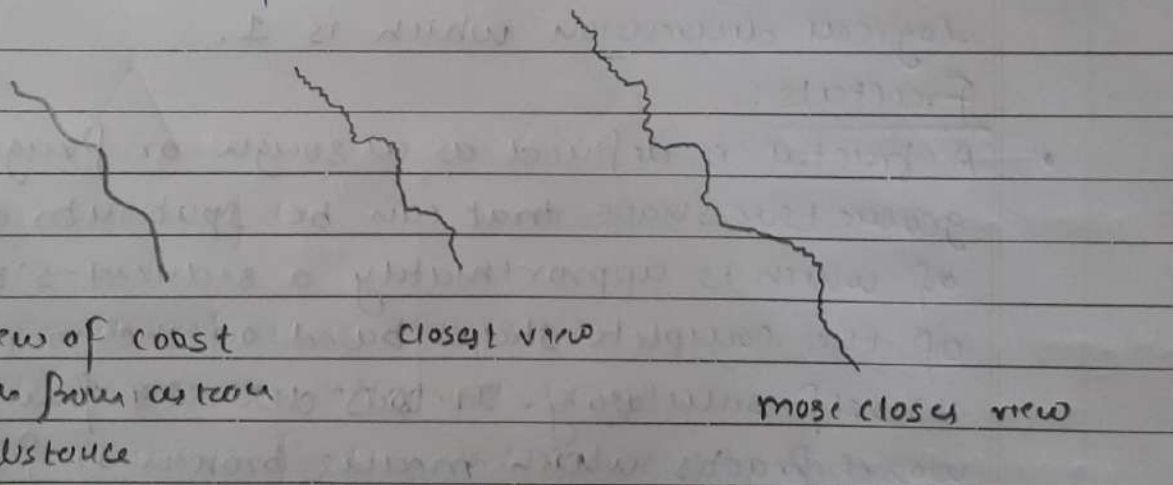
$D = \log_3 4 = \log 4 / \log 3 = 1.2618$

- Therefore for Koch curve topological dimensions is 1 but fractal dimension is 1.2618
- From the above discussion we can say that point sets, curve and surfaces which gave a fractal dimensions greater than the topological dimensions are called fractals. The Hilbert's curve and Koch curves are fractals, because their fractal dimensions (respectively, 2 and 1.2618) are greater than their topological dimension which is 1.

Fractals:

- A fractal is defined as a rough or fragmented geometric shape that can be split in to parts, each of which is approximately a reduced-size reproduction of the complete shape based on the property known as self-similarity. It was derived from the latin word fractus which means broken or fractured. Natural objects can be realistically described using fractal geometry methods. Example - cloud, mountains trees, stone, etc..
- fractal methods use procedures rather than equations to model objects. so it uses procedural modelling. The major characteristics of any procedural model is that the model is not based on data, but rather on the implementation of the procedure following a particular set of rules.

- A fractal combines the following characteristics
  - > Its part having the same form or structures as a whole, except that they are at a different scale and may be slightly deformed.
  - > Its form is extremely irregular or fragmented, and remains so, whatever the scale of examination.
  - > It is formed by iteration i.e. the procedure is used repeatedly (recursively)
  - > Example: if $P0 = (x0\ y0\ z0)$ is a selected initial position the successive levels $P1 = F(P0)$, $P2 = F(P1) \ldots Pn = F(Pn-1)$ are generated by a transformation function F.
  - > Fractional dimensions.



  - > Imagine that one object is made up of clay. If we break that objects in terms of line or true segments, then we will give the dimensions $D_t = 1$. If the object is broken into a plane, then we will give the dimension $D_t = 2$. And if the object is broken into 3D object like cube, sphere etc., then we will give the transformation as $D_t = 6$. Here the variables D_t is called topological dimensions.