Title: - Write C-H- program to draw a concave polygon and fill it with desired color using fill algorithm.

Class: -SE Computer

Sub:-OOPL & CGL

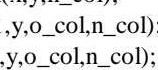


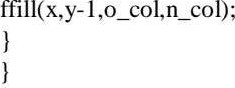
# Program-

#include<graphics.h>

#include<iostream> #include<stdlib.h> using namespace std; void ffill(int x,int y,int o\_col,int n\_col)

int current — getpixel(x,y); if(current==o\_col)

delay(l); putpixel(x,y,n\_col); ffill(x+l ; ffill(x- 1 ffill(x,y+l,o\_col,n\_col);



int main()

int xl,y1,x2,y2,x3,y3,xavg,yavg; int gdriver = DETECT,gmode; initgraph(&gdriver,&gmode,NULL); cout << " Enter the points of triangle"; setcolor(l ); cin xl >> yl >> x2 Y2 x3 >> y3; xavg — (int)(x1+x2+x3)/3; yavg = (int)(yl +Y2+Y3)/3; line(xl,y1,x2,y2); line(x2,y2,x3,y3); line(x3,y3,x1,yl);

ffill(xavg,yavg,O,3); getch(); return O;

# /\*Output:-





Title: - Write program to generate Hilbert curve using concept of fractals.

Roll No:-

Class:-SE Computer

Sub:-OOPL & CGL



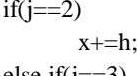
# Program-

#include<iostream>

#include<graphics.h>

#include<math.h> #include<cstdlib> using namespace std; void move(int j, int h, int &x,int &y)

y-=h; else



else

if(j==3)

else x-=h; lineto(x,y);

void hilbert(int r,int d,int I , int u,int i,int h,int &x,int &y)

i--; hilbert(d,r,u,l,i,h,x,y); move(r,h,x,y); hilbert(r,d,l,u,i,h,x,y); move(d,h,x,y); hilbert(r,d,l,u,i,h,x,y); move(l,h,x,y); hilbert(u,l,d,r,i,h,x,y); int main()

int int cout<<"Give the value ofn=";

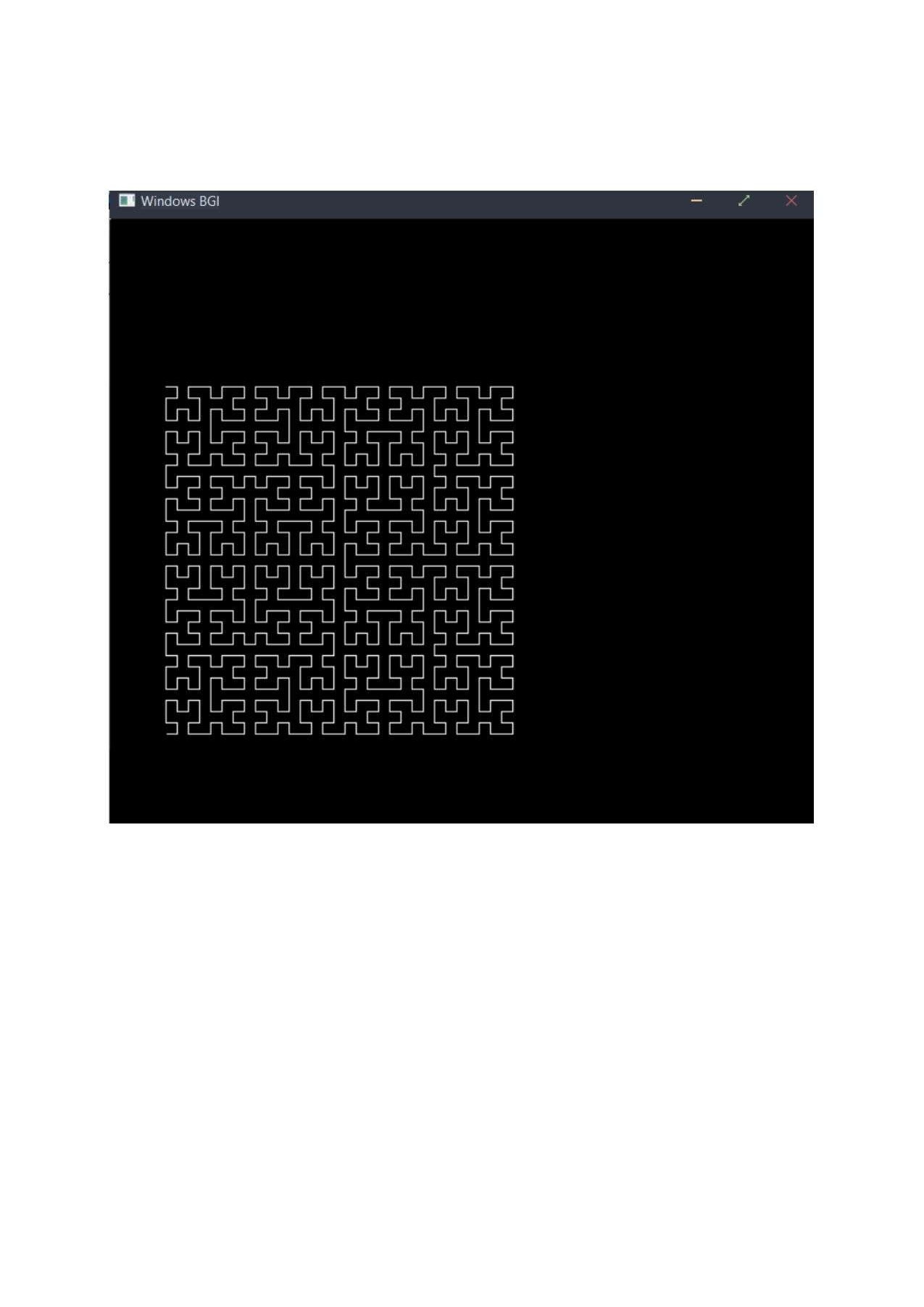


x=xO; y=yo; int driver=DETECT,mode=O; initgraph(&driver,&mode,NULL); moveto(x,y); hilbert(r,d,l,u,n,h,x,y);

delay(10000)•, closegraph(); return O;

# /\*Output:-

|  |  |  |
| --- | --- | --- |
|  |  | Pract b2\CGL Practb 2exe |
| Give the value of | | |
|  | | |
| Process exited after 20.65 seconds with return value | | |
| Press any key to continue . | | |



Title: - Write OpenGL Program to draw Sunrise and Sun Set.

Roll No:-

Class:-SE Computer

Sub:-OOPL & CGL

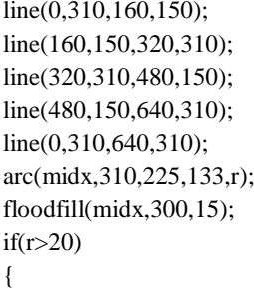


# Program-

#include<graphics.h> int main()

int gd = DETECT, gm; initgraph(&gd, &gm,NULL); int midx,midy,r=l(); midx=getmaxx()/2; while(r<=50)

cleardevice(); setc010r(WHITE);



setcolor(7);



setcolor(6);



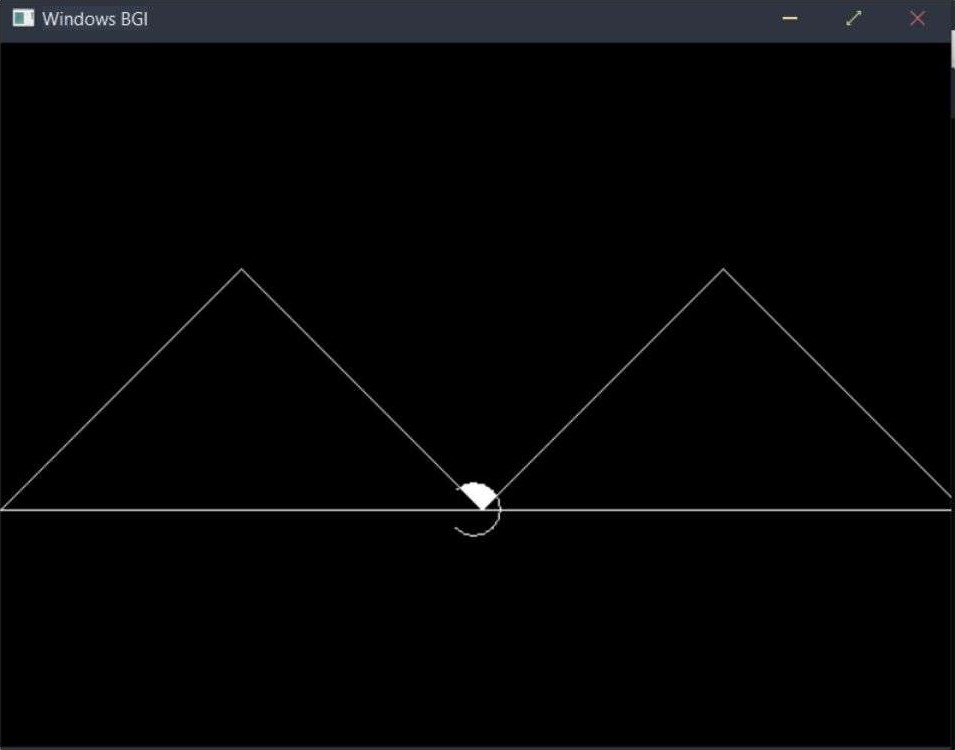
setcolor(2);



delay(1000);

getch(); closegraph();

# /\*Output:-



Mini Project

Title: - Write a C++ program to draw a man walking in rain with an umbrella.

Roll No:-

Class:-SE Computer

Sub:-OOPL & CGL



## Program-

#include<stdio.h>

#include<graphics.h>

#define ScreenWidth getmaxx()

#define ScreenHeight getmaxy() #define GroundY ScreenHeight\*O.80 int Idisp=O; void DrawManAndUmbrella(int x,int Idisp)

//Man's head



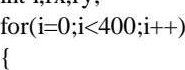
line(x,GroundY-80,x,GroundY-30); //Man's hand line(x,GroundY-70,x+lO,GroundY-60); line(x, GroundY-65 10, GroundY-55 ) ; line(x+l O, line(x+lO,GroundY-55,x+20,GroundY-70);

//Man's legs line(x,GroundY-30,x+ldisp,GroundY); line(x,GroundY-30,x-ldisp,GroundY); //umbrella

O, 180, 40); line(x+20,GroundY-120,x+20,GroundY-70);

void Rain(int x)

int i,rx,ry;



rx—rand() % ScreenWidth;

ry=rand() % ScreenHeight; if(ry<GroundY-4)

if(rY<GroundY-120 Il (ry>GroundY-120 && (rx<x-20 I l rx>x+6()))) line(rx,ry,rx+O.5,ry+4);

int main()

int initgraph(&gd,&gm,NULL); while( !kbhit())

//Draw Ground

,ScreenWidth,GroundY);

Rain(x);

Idisp=(ldisp+2)%20;

DrawManAndUmbrella(x,ldisp); delay(75); cleardevice(); x=(x+2)%ScreenWidth;

getch();

## / \*Output:-



#include<iostream>

#include<dos.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

/\* Defining structure for end point of line \*/ using namespace std; typedef struct coordinate

int x;

int y;

char code[4];

void drawwindow();

void drawline (PT pl,PT p2,int cl);

PT setcode(PT p);

int visibility (PT pl,PT p2);

PT resetendpt (PT pl,PT p2);

void check\_line(PT pl,PT p2);

int main()

initwindow(800,800); //int gd=DETECT, gm;

PT pi,p2;

END-POINT 1 (x,y):

cin>>pl.x>>pl.y;

END-POINT 2 (x,y): 'i;

cin>>p2.x>>p2.y;

//initgraph(&gd,&gm,"\\Turboc3\\bgi");

drawwindow();

drawline(p1,p2,15);

check\_line(p1,p2);

return(O);

void check\_line(PT pl,PT p2)

int v;

pl=setcode(pl);

p2=setcode(p2);

v=visibility(p1,p2);

switch(v)

case O: cleardevice(); /\* Line conpletely visible \*/

drawwindow();

drawline(p1,p2,15);

break;

case 1: cleardevice(); /\* Line completely invisible \*/

drawwindow();

break;



case 2: cleardevice(); /\* line partly visible \*/

pl=resetendpt (pl,p2);

p2=resetendpt(p2,p1);

check\_line(p1,p2);

break;

delay(2000);

/\* Function to draw window \*/

void drawwindow()

setcolor(RED);









delay(2000);

/\* Function to draw line between two points

void drawline (PT pl,PT p2,int cl)

setcolor(cl);

line(p1.x,p1.y,p2.x,p2.y);

delay(2000);

 Function to set code of the coordinates



PT setcode(PT p)

PT ptemp;



/\* TOP \*/

else





ptemp.code[l]='l'; /\* BOTTOM \*/

else

ptemp.code[1]='O';

if (p.x>450)

ptemp.code[2]='1'; /\* RIGHT \*/

else



if (p.x<150) /\* LEFT \*/

ptemp.code[3]='1';

else



ptemp.x=p.x;

ptemp.y=p.y;

return(ptemp);

/\* Function to determine visibility of line



int visibility (PT pl,PT p2)

int



if((pl.code[i] I I (p2.code[i]

flag=2;



if((p1.code[i]==p2.code[i])

flag=l;

return (O);

if(flag==l)

return(l);

if(flag==2)

return(2);

/\* Function to find new end points



PT resetendpt (PT pl,PT p2)

PT temp;

int x,y,i;

float m,k;

if( pl.code[3]=='1') /\* Cutting LEFT Edge \*/

x=150;

if(p1.code[2]=='1') /\* Cutting RIGHT Edge \*/

x=450;



m=(float) (p2.y-p1.y)/(p2.x-p1.x);

k=(pl.y+(m\*(x-pl.x)));

temp.y=k;

temp.x=x;



return(temp);

/\* Cutting TOP Edge \*/

y=100;

if(pl.code -='1') /\* Cutting BOTTOM Edge \*/

y=350;



m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(float)pl.x+(float)(y-pl.y)/m;

temp.x=k;

temp.y=y;

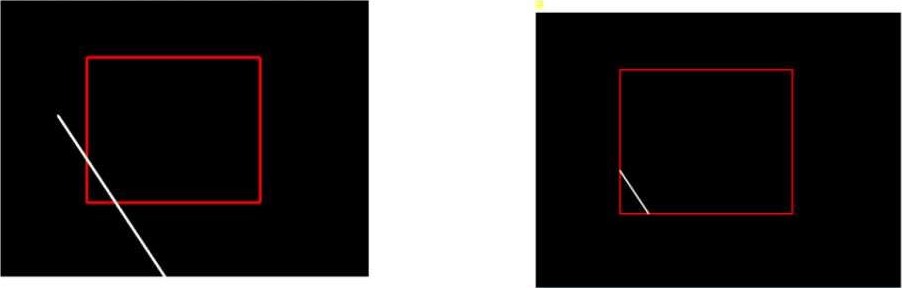
if(temp.X<=450&&temp.x>=150)

return(temp);

else

return(pl);

|  |
| --- |
| ЕНТЕВ ЕНО—РОIНТ 1 (х,ч) : 100 200  ЕНТЕВ END—POIHT 2 (х,у): зоо 500\_ |



#include<conio.h> #include<iostream> #include<graphics.h> #include<math.h> using namespace std; class drawpattern

private:

float dx,dy,i ,length; float count; public:

int xl,y1,x2,y2; int xmid,ymid; void getdata(); void ddaline(int xl,int x2,int yl, int y2); int xc,yc,r; void bdrawcircle(int xc,int yc,int r);

void drawpattern::getdata()



cout<<"Enter xl cin>>xl; cout<<"Enter yl" ; cin>>yl;

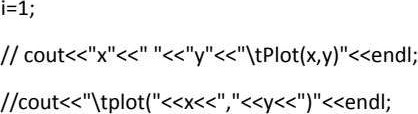
x2 "; cin>>x2; cout<<"Enter y2" ; void drawpattern::ddaline(int xl, int x2, int yl, int y2)

float x,y; dx = (x2-x1); dy = (y2-y1);

//cout<<"value of dx:"<<dx<<endl;

// cout<<"value of if(abs(dx)>=abs(dy)) length = abs(dx); else length = abs(dy);

// cout<<"length:"<<length<<endl; dx = dx/length; dy = dy/length; x=xl; y=yl;



while(i<=length)

x = x + dx;

y=y+ dy; // cout<<x<<"



putpixel(x,y,15);

void drawpattern::bdrawcircle(int xc,int yc,int r) //xc=320; //yc=240;

int x,y,d;

putpixel(xc+x,yc-y,15); // initialize the decision variable

do

putpixel(xc+x,yc+y,15); putpixel(xc-x,yc-y,15); putpixel(xc+x,yc-y,15); putpixel(xc-x,yc+y,15); putpixel(xc+y,yc-x,15); putpixel(xc-y,yc-x,15); putpixel(xc+y,yc+x,15); putpixel(xc-y,yc+x,15); if(d<O)

d=d+4\*x+6;

else

d=d+4\*(x-y)+10; y=y-l;

X=X+I

while(x<=y);

int main()

//clrscr();



//int gdriver= DETECT, gmode;

//initgraph(&gdriver,&gmode, "c://Turboc3//BGl cleardevice(); drawpattern d;

d.getdata();

d.ddaline(d.x1,d.y1,d.x2,d.y1);// (xl,yl) and (x2,y1)

d.ddaline(d.x2,d.y1,d.x2,d.y2);

d.ddaline(d.x2,d.y2,d.x1,d.y2);

d.ddaline(d.xI,d.y2,d.xI,d.y1);

d.xmid=abs((d.x1+d.x2))/2;

d.ymid=abs((d.y1+d.y2))/2;

d.ddaline(d.xmid,d.y1,d.x2,d.ymid);// (xl,yl) and (x2,y1)

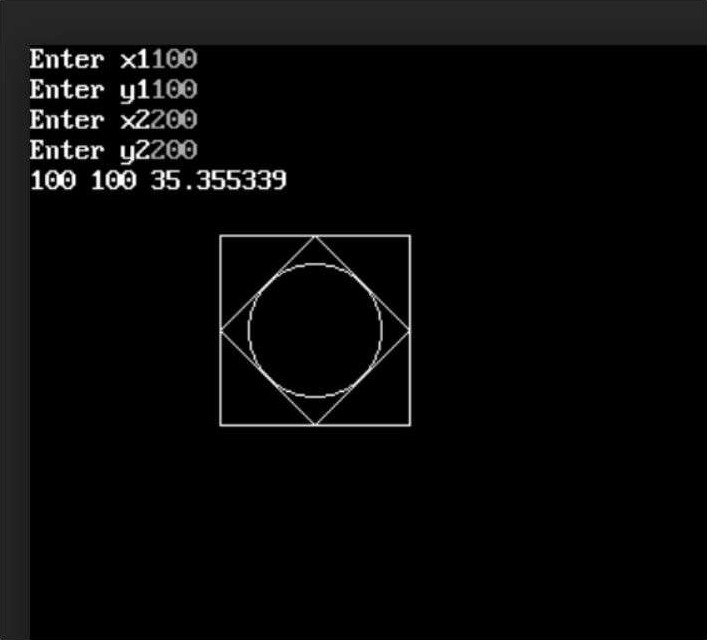
d.ddaline(d.x2,d.ymid,d.xmid,d.y2);

d.ddaline(d.xmid,d.y2,d.x1,d.ymid);

d.ddaline(d.xl,d.ymid,d.xmid,d.yl); float rad,cal,sidex,sidey; sidex=abs(d.x2-d.x1); sidey=abs(d.y2-d.y1); cal=pow(sidex,2)+pow(sidey,2); cal=2\*sqrt(caI); rad=(sidex\*sidey)/cal; cout<<sidex<< <<sidey;

cout<<" "<<rad;

d.bdrawcircle(d.xmid,d.ymid,rad); getch(); closegraph(); // getch(); return O;



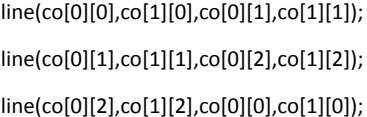
#include<iostream> #include<conio.h> #include<graphics.h> #include<stdlib.h> #include<stdio.h> #include<math.h> using namespace std; class trans

public:

float transco[3] [3]; // float orico[3][3]; float scalco[3][3]; float rotco[3] [3]; void drawtri(float [3][3]); void translation(int,int,float [3] [3]); void scaling(float,float,float [3][3]); void rotation(float,float [3][3]);

void trans::drawtri(float co[3][3])

//clrscr();



void trans::translation(int tx,int ty,float orico[3][3])

cout<<"Enter Translation Factor"<<endl; cin>>tx>>ty;

int i,j; for(i=O;i<3;i++)

transco[O][i]=orico[O][i]+tx; transco[l][i]=orico[l][i]+ty; transco[2][i]=1;

for(i=O;i<3;i++)





cout<<endl;

void trans::scaling(float sx,float sy,float orico[3][3])

cout<<"Enter Scaling cin>>sx>>sy;

int i,j; for(i=O;i<3;i++)



scalco[l][i]=orico[l][i]\*sy; scalco[2][i]=1;

for(i=O;i<3;i++)

for(j=O;j<3;j++)



cout<<endl;

void trans::rotation(float theta,float orico[3]

Rotation Angle"<<endl; cin>>theta; cout<<theta<<endl; theta: theta\*(3.14/180); cout<<"theta in radious"<<theta<<endl;

int i,j,refx,refy; for(i=O;i<3;i++)



rotco[i][j]=O;

for(i=O;i<3;i++)

rotco[O][i]=orico[O][i]\*cos(theta)-orico[1][i]\*sin(theta); rotco[11[i]=orico[O][i]\*sin(theta)+orico[1][i]\*cos(theta); int main()

//clrscr();



int c;

//int DETECT, gm;

//initgraph(&gd,&gm,"C:\\TurboC3\\BGl"); trans t; int tx,ty; float sx,sy; float theta; float for(int i=O;i<3;i++)

for(int j=O;j<3;j++)

<<j<<"->"<<orico[i][j]<< cout<<endl;

t.drawtri(orico); cout<<"Enter your choice"<<endl; cout<<"l. Translation"<<endl; cout<<"2. Scaling"<<endl; cout<<"3. Rotation"<<endl; cin>>c; switch(c)

case 1:

t.translation(tx,ty,orico);

t.drawtri(t.transco); break; case 2:

t.scaling(sx,sy,orico);

t.drawtri(t.scalco); break; case 3:

t.rotation(theta,orico);

t.drawtri(t.rotco); break; default:

cout<<("You have written wrong Choice");

getch(); return O;

