

Computer graphics lab manual

Department of computer science and engineering
Year 2019-20

1. Program to implement Mid Point Line Algorithm. The line coordinates should be specified by the user.

```

#include <GL/glut.h>
#include <stdio.h>

int x00,y00,x01,y01;

void init()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-500,500,-500,500);
}

void writepixel(int x,int y)
{
    glPointSize(5);
    glBegin(GL_POINTS);      //WRITE PIXEL
    glColor3f(0,0,0);
    glVertex2f(x,y);
    glEnd();
    glFlush();
}
void display()
{
    int i,j;

    float dx=x01-x00 , dy = y01-y00;
    float d = 2*dy-dx;

    float incrE = dy;
    float incrNE = dy - dx;

    int x=x00,y=y00;

    writepixel(x,y);

    while(x<x01)
    {
        if(d<=0)

```

Department of Computer Science & Engineering

```

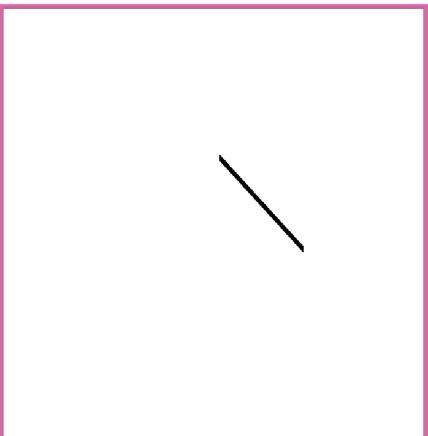
{
    d=d+2*incrE;
    x=x+1;
}
else
{
    d=d+2*incrNE;
    x=x+1;
    y=y+1;
}
writepixel(x,y);
}

int main(int argc,char *argv[])
{
    printf("Enter the values \n");
    printf("x0="); scanf("%d",&x00);
    printf("y0="); scanf("%d",&y00);
    printf("x1="); scanf("%d",&x01);
    printf("y1="); scanf("%d",&y01);

    glutInit(&argc,argv);
    glutInitWindowSize(500,500);
    glutCreateWindow("MIDPOINT LINE ALGORITHM");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}

```

OUT PUT:



2. Program to implement Mid Point Circle Algorithm. The radius should be specified by the user.

```
#include <GL/glut.h>
#include <stdio.h>
float x ;
float y ;
float r;
void init()
{
    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-900,900,-900,900);
}

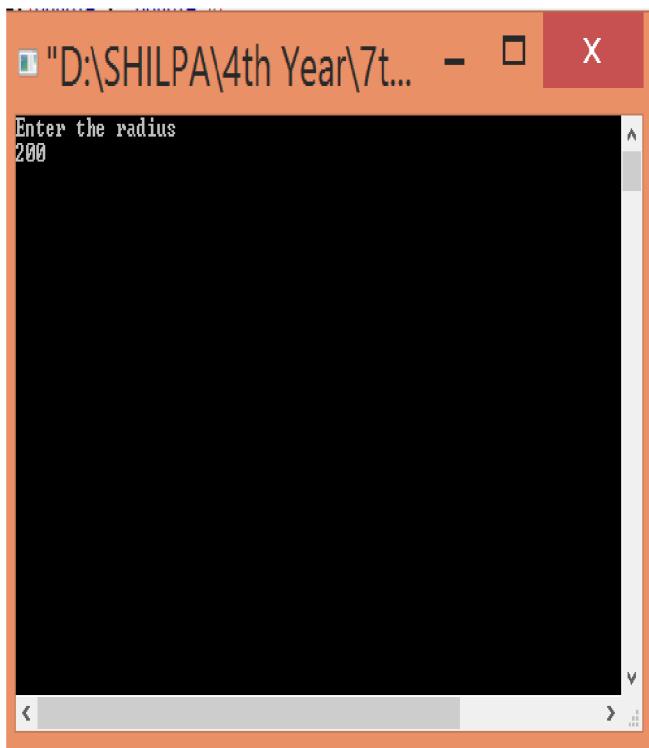
void writepixel(float x,float y)
{
    glPointSize(5);
    glBegin(GL_POINTS);
    glColor3f(1.0,0.0,0.0);
    glVertex2f(x,y);
    glEnd();
    glFlush();
}
void midpointcircle( )
{
    x=0;
    y=r;
    double d=5.0/4.0-r;
    writepixel(x,y);
    while(y>x)
    {
        if(d<0)
            d=d+2.0*x+3.0;
        else
        {
            d=d+2.0*(x-y)+5.0;
            y--;
        }
        x++;
        writepixel(x,y);
        writepixel(y,x);
        writepixel(y,-x);
        writepixel(x,-y);
    }
}
```

```
writepixel(-x,-y);
writepixel(-y,-x);
writepixel(-y,x);
writepixel(-x,y);
glFlush();
}
}

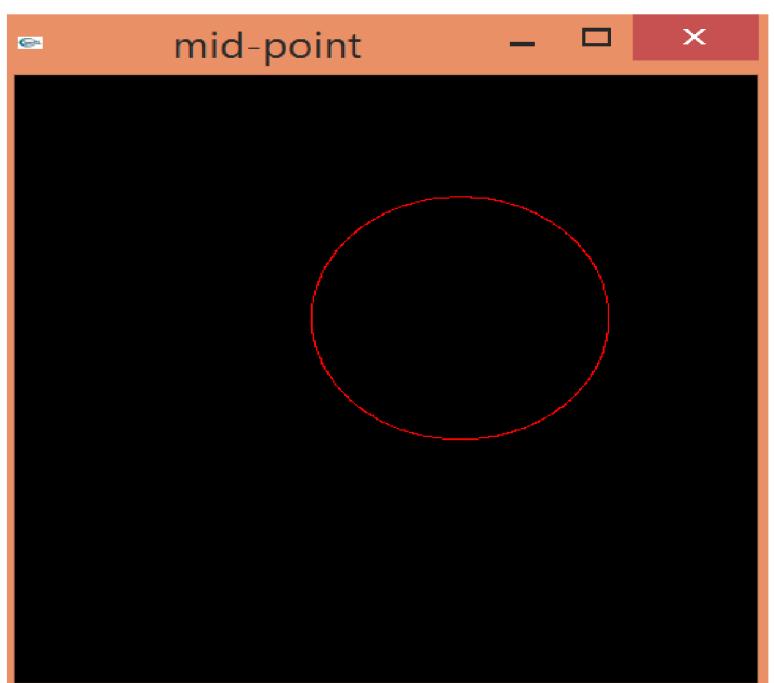
int main(int argc,char *argv[])
{
printf("Enter the radius of the circle\n");
scanf("%f",&r);

glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutCreateWindow("Midpoint Circle Algorithm");
init();
glutDisplayFunc(midpointcircle);
glutMainLoop();
return 0;
}
```

OUT PUT



```
"D:\SHILPA\4th Year\7t... - X
Enter the radius
200
```



3. Program to implement Liang-Barsky line clipping algorithm. Make provision to specify the input line, window for clipping and viewport for displaying the clipped image.

```

#include<windows.h>
#include<stdio.h>
#include<GL/glut.h>
float xmin,ymin,xmax,ymax;
double xvmin=600,yvmin=600,xvmax=900,yvmax=900;
int x0,x1,y0;

int cliptest(double p,double q,double *t1,double *t2)
{
    double t=q/p;
    if(p<0.0)
    {
        if(t>*t1) *t1=t;
        if(t>*t2) return(false);
    }
    else
        if(p>0.0)
    {
        if(t<*t2) *t2=t;
        if(t<*t1) return(false);
    }
    else
        if(p==0.0)
    {
        if(q<0.0) return(false);
    }
    return(true);
}

void LiangBarskyLineClipAndDraw(double x0,double y0,double x1,double y1)
{
    double dx=x1-x0,dy=y1-y0,te=0.0,t1=1.0;
    if(cliptest(-dx,x0-xmin,&te,&t1))
        if(cliptest(dx,xmax-x0,&te,&t1))
            if(cliptest(-dy,y0-ymin,&te,&t1))
                if(cliptest(dy,ymax-y0,&te,&t1))
                {
                    if(t1<1.0)
                    {
                        x1=x0+t1*dx;
                        y1=y0+t1*dy;
                    }
                }
}

```

Department of Computer Science & Engineering

```

        }
        if(te>0.0)
        {
            x0=x0+te*dx;
            y0=y0+te*dy;
        }
        double sx=(xvmax-xvmin)/(xmax-xmin);
        double sy=(yvmax-yvmin)/(ymax-ymin);
        double vx0=xvmin+(x0-xmin)*sx;
        double vy0=yvmin+(y0-ymin)*sy;
        double vx1=xvmin+(x1-xmin)*sx;
        double vy1=yvmin+(y1-ymin)*sy;
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xvmin,yvmin);
        glVertex2f(xvmax,yvmin);
        glVertex2f(xvmax,yvmax);
        glVertex2f(xvmin,yvmax);
        glEnd();
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINES);
        glVertex2d(vx0,vy0);
        glVertex2d(vx1,vy1);
        glEnd();
    }
}

void drawrect()
{
    glColor3f(0.0,0.0,1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin,ymin);
    glVertex2f(xmax,ymin);
    glVertex2f(xmax,ymax);
    glVertex2f(xmin,ymax);
    glEnd();
}

void drawline()
{
    glColor3f(1.0,0.0,1.0);
    glBegin(GL_LINES);
    glVertex2d(x0,y0);
    glVertex2d(x1,y1);
    glEnd();
}

void display()
{
}

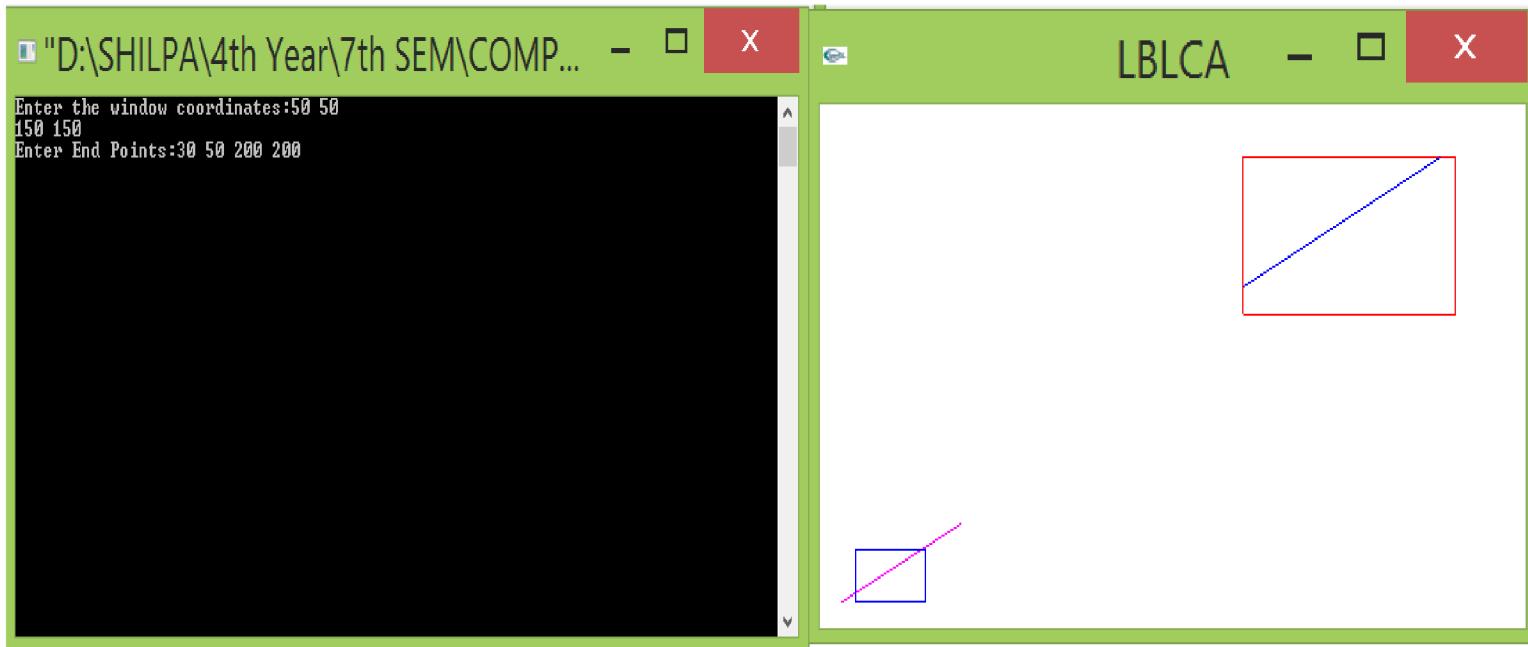
```

```

{
    glClear(GL_COLOR_BUFFER_BIT);
    drawline();
    drawrect();
    LiangBarskyLineClipAndDraw(x0,y0,x1,y1);
    glFlush();
}
void myinit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(1.0,0.0,0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1000.0,0.0,1000.0);
}
int main(int argc,char** argv)
{
    printf("Enter the window coordinates:");
    scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);
    printf("Enter End Points:");
    scanf("%d%d%d%d",&x0,&y0,&x1,&y1);
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("LBLCA");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
    return 0;
}// Try for input values 100,100,500,500;

```

OUT PUT:



4. Program to implement the Cohen-Sutherland line-clipping algorithm. Make provision to specify the input line, window for clipping and viewport for displaying the clipped image.

```
#include<GL/glut.h>
#include <stdio.h>
#include <stdbool.h>

typedef int Outcode;

const int INSIDE=0;
const int LEFT=1;
const int RIGHT=2;
const int BOTTOM=4;
const int TOP=8;
double xmin=50,ymin=50,xmax=100,ymax=100;

void init()
{
    glClearColor(1,2,3,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0,500,0,500);
}
```

Department of Computer Science & Engineering

```

Outcode ComputeCode( double x, double y)
{
    Outcode code=INSIDE;

    if(x<xmin)
        code|=LEFT;
    else if(x>xmax)
        code|=RIGHT;
    else if(y<ymin)
        code|=BOTTOM;
    else if(y>ymax)
        code|=TOP;

    return code;
}

void CohenSutherlandline(double x0,double a,double x1,double b)
{
    bool accept=false;
    Outcode outcode0=ComputeCode(x0,a);
    Outcode outcode1=ComputeCode(x1,b);
    double x,y;

    while(true)
    {
        if(!(outcode0|outcode1)){
            accept=true;
            break;
        }
        else if(outcode0 & outcode1)
        {
            break;
        }
        else
        {

            Outcode outcodeout =outcode0?outcode0:outcode1;
            if(outcodeout & TOP)
            {
                x=x0 +(x1-x0)*(ymax-a)/(b-a);
                y=ymax;
            }
            else if(outcodeout & BOTTOM)

```

Department of Computer Science & Engineering

```

{
x=x0+(x1-x0)*(ymin-a)/(b-a);
y=ymin;
}
else if(outcodeout & RIGHT)
{
y=a +(b-a)*(xmax-x0)/(x1-x0);
x=xmax;
}
else if(outcodeout & LEFT)
{
y=a +(b-a)*(xmin-x0)/(x1-x0);
x=xmin;
}
if(outcodeout == outcode0)
{
x0=x;
a=y;
outcode0=ComputeCode(x0,a);
}
else
{
x1=x;
b=y;
outcode1=ComputeCode(x1,b);
}
}
}
}
if(accept)
{
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINE_LOOP);
 glVertex2f(4*xmin,4*ymin);
 glVertex2f(4*xmax,4*ymin);
 glVertex2f(4*xmax,4*ymax);
 glVertex2f(4*xmin,4*ymax);
 glEnd();
glBegin(GL_LINES);
 glVertex2f(4*x0,4*a);
 glVertex2f(4*x1,4*b);
 glEnd();
}
}

void display()

```

Department of Computer Science & Engineering

```
{
double x0=60,a=20,x1=80,b=120;
glBegin(GL_LINE_LOOP);
	glColor3f(1.0,0.0,0.0);
	glVertex2f(xmin,ymin);
	glVertex2f(xmax,ymin);
	glVertex2f(xmax,ymax);
	glVertex2f(xmin,ymax);
	glEnd();

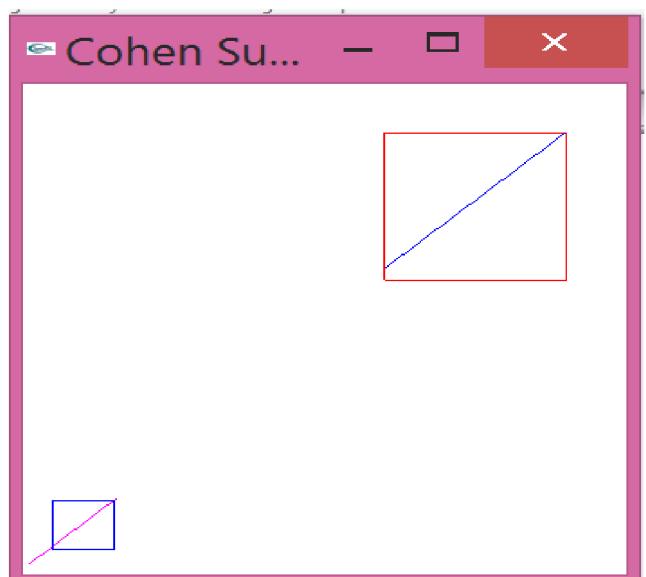
glBegin(GL_LINES);
 glVertex2f(x0,a);
 glVertex2f(x1,b);
glEnd();

CohenSutherlandline(x0,a,x1,b);
glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);
	glutInitWindowSize(800,800);
	glutCreateWindow("Cohen Sutherland line clipping window");
	init();
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

OUT P

```
D:\SHILPA\4th Year\7th SEM\COMP...
Enter the window coordinates:50 50
150 150
Enter End Points:30 50 200 200
```



5. Program to fill any given polygon using scan-line area filling algorithm. vertices for the polygon should be specified by the user.

```
#include<windows.h>
#define BLACK 0
#include<stdlib.h>
#include<stdio.h>
#include<GL/glut.h>
float x1,x2,x3,x4,y1,y2,y3,y4;
void edgedetect(float x1,float y1,float x2,float y2,int *le,int *re)
{
    float mx,x,temp;
    int i;
    if((y2-y1)<0)
    {
        temp=y1;
        y1=y2;
        y2=temp;
        temp=x1;
        x1=x2;
        x2=temp;
    }
    if((y2-y1)!=0)
        mx=(x2-x1)/(y2-y1);
    else
        mx=x2-x1;
    x=x1;
    for(i=y1;i<=y2;i++)
    {
        if(x<(float)le[i])
            le[i]=(int)x;
        if(x>(float)re[i])
            re[i]=(int)x;
        x+=mx;
    }
}

void draw_pixel(int x,int y)
{
    glColor3f(0.0,1.0,0.0);
    glPointSize(3.0);
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
}
```

Department of Computer Science & Engineering

```

void scanfill(float x1,float y1,float x2,float y2,float x3,float y3,float x4,float y4)
{
  int le[500],re[500];
  int i,y;
  for(i=0;i<500;i++)
  {
    le[i]=500;re[i]=0;
  }
  edgedetect(x1,y1,x2,y2,le,re);
  edgedetect(x2,y2,x3,y3,le,re);
  edgedetect(x3,y3,x4,y4,le,re);
  edgedetect(x4,y4,x1,y1,le,re);
  for(y=0;y<500;y++)
  {
    if(le[y]<=re[y])
      for(i=(int)le[y];i<(int)re[y];i++)
    {
      draw_pixel(i,y);
      Sleep(1);
      glFlush();
    }
  }
}

void display()
{
  x1=20.0;y1=20.0;x2=10.0;y2=30.0;x3=20.0;y3=40.0;x4=30.0;y4=30.0;
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0.0,0.0,1.0);
  glBegin(GL_LINE_LOOP);
  glVertex2f(x1,y1);
  glVertex2f(x2,y2);
  glVertex2f(x3,y3);
  glVertex2f(x4,y4);
  glEnd();
  scanfill(x1,y1,x2,y2,x3,y3,x4,y4);
  glFlush();
}

void myinit()
{
  glClearColor(1.0,1.0,1.0,1.0);
}

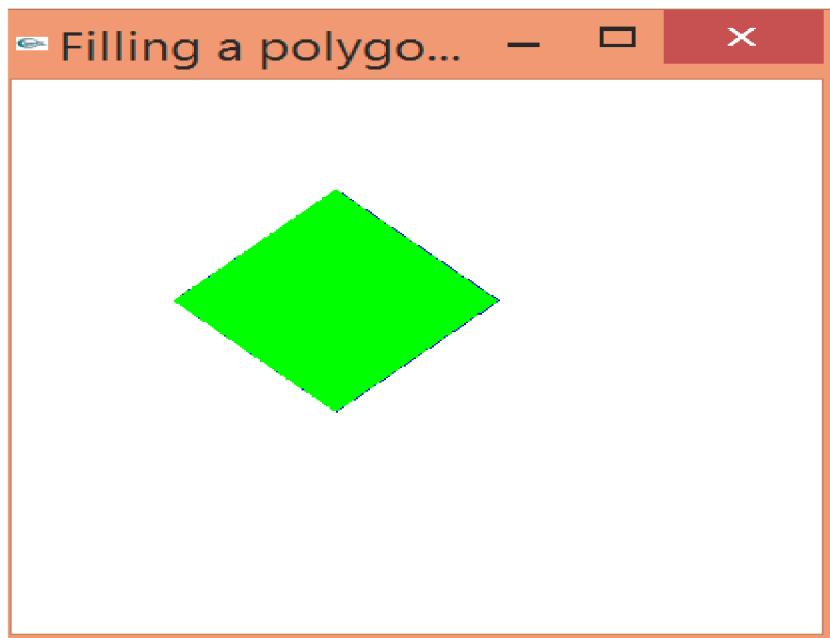
```

```
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,50.0,0.0,50.0);

}

int main(int argc,char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Filling a polygon using Scan-Line Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

OUT PUT



6. Program to recursively subdivide a triangle to form 2D Sierpinski gasket. The number of recursive steps is to be specified by the user

```
#include <windows.h>
#include <GL/glut.h>
#include <stdio.h>
int n;

void init()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 10, 0, 10);
}

void triangle(float *a, float *b, float *c)
{
    glVertex2f(a[0],a[1]);
    glVertex2f(b[0],b[1]);
    glVertex2f(c[0],c[1]);
}

void draw_triangle(float *a, float *b, float *c, int k)
{
    float ab[2], ac[2], bc[2];
    int i;
    if(k>0)
    {
        for(i=0;i<2;i++)
            ab[i]=(a[i]+b[i])/2;
        for(i=0;i<2;i++)
            bc[i]=(b[i]+c[i])/2;
        for(i=0;i<2;i++)
            ac[i]=(a[i]+c[i])/2;

        draw_triangle(a,ab,ac,k-1);
        draw_triangle(b,bc,ab,k-1);
        draw_triangle(c,ac,bc,k-1);
    }
    else
    {
        triangle(a,b,c);
    }
}
```

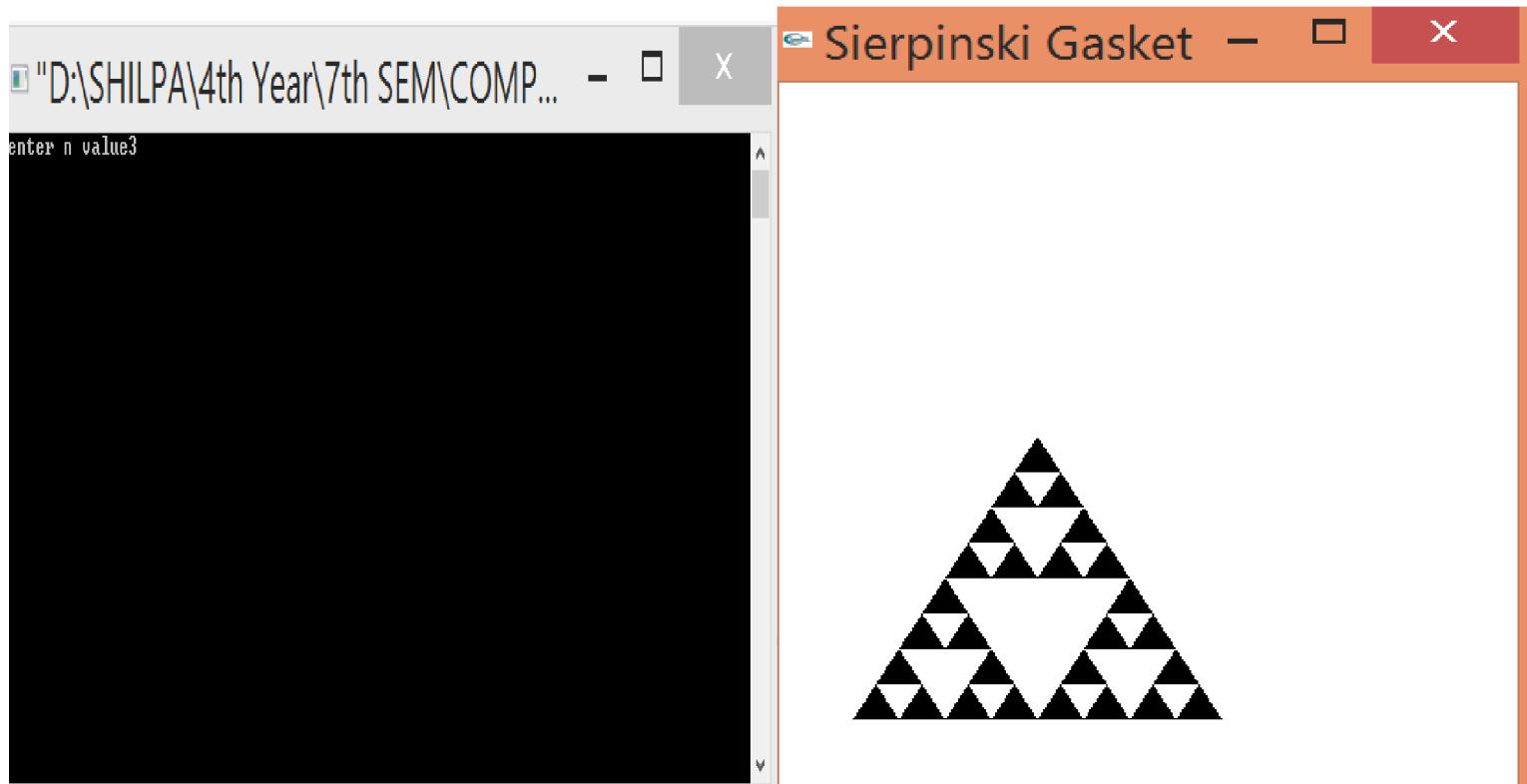
```

void display()
{
    float a[2]={1,1};
    float b[2]={6,1};
    float c[2]={3.5,5};

    glBegin(GL_TRIANGLES);
    glColor3f(0,0,0);
    draw_triangle(a,b,c,n);
    glEnd();
    glFlush();
}
int main(int argc, char** argv)

{
    printf("enter n value");
    scanf("%d",&n);
    glutInit(&argc,argv);
    glutInitWindowSize(500,500);
    glutCreateWindow("Sierpinski Gasket");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
    
```

OUT PUT



7. Program to display a set of values {f(i, j)} as a rectangular mesh. Number of rows and columns for the mesh generation must be taken from the user.

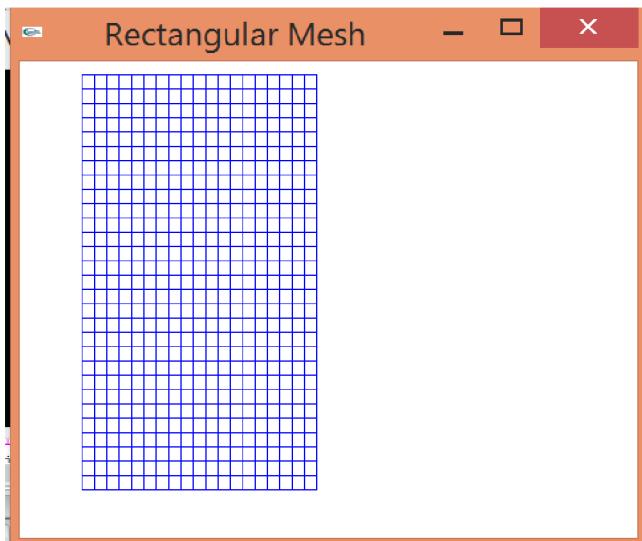
```
#include<windows.h>
#include<stdlib.h>
#include<GL/glut.h>
#define maxx 20
#define maxy 30
#define dx 10
#define dy 15
GLfloat x[maxx]={0.0},y[maxy]={0.0};
GLfloat x0=50,y0=50;
GLint i,j;
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(1.0,0.0,0.0);
    glPointSize(5.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,499.0,0.0,499.0);
    glutPostRedisplay();
}
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0,0.0,1.0);
    for(i=0;i<maxx;i++)
        x[i]=x0+i*dx;
    for(j=0;j<maxy;j++)
        y[j]=y0+j*dy;
    glColor3f(0.0,0.0,1.0);
    for(i=0;i<maxx-1;i++) // Filling mesh from bottom to top and left to right
        for(j=0;j<maxy-1;j++)
    {
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(x[i],y[j]);
        glVertex2f(x[i],y[j+1]);
        glVertex2f(x[i+1],y[j+1]);
        glVertex2f(x[i+1],y[j]);
        glEnd();
        glFlush();
    }
    glFlush();
}
```

```

}
int main(int argc,char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,400);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Rectangular Mesh");
    glutDisplayFunc(display);
    init();
    glutMainLoop();
}

```

OUT PUT



8. Program to create a random figure and rotate it about a given fixed point using transformation matrices. Make provision for the user to enter pivot point for the rotation.

```

#include<windows.h>
#include<stdio.h>
#include<math.h>
#include<GL/glut.h>
GLfloat house[2][9]={{100.0,100.0,175.0,250.0,250.0,150.0,150.0,200.0,200.0},
{100.0,300.0,400.0,300.0,100.0,100.0,150.0,150.0,100.0}};
GLfloat theta;
GLfloat h=100.0;
GLfloat k=100.0;
void drawhouse()
{
    glColor3f(1.0,0.0,0.0);

```

Department of Computer Science & Engineering

```

glBegin(GL_LINE_LOOP);
glVertex2f(house[0][0],house[1][0]);
glVertex2f(house[0][1],house[1][1]);
glVertex2f(house[0][3],house[1][3]);
glVertex2f(house[0][4],house[1][4]);
glEnd();
	glColor3f(1.0,0.0,1.0);
	glBegin(GL_LINE_LOOP);
	glVertex2f(house[0][5],house[1][5]);
	glVertex2f(house[0][6],house[1][6]);
	glVertex2f(house[0][7],house[1][7]);
	glVertex2f(house[0][8],house[1][8]);
	glEnd();
	glColor3f(1.0,0.0,0.0);
	glBegin(GL_LINE_LOOP);
	glVertex2f(house[0][1],house[1][1]);
	glVertex2f(house[0][2],house[1][2]);
	glVertex2f(house[0][3],house[1][3]);
	glEnd();
}
void display()
{
    int i;
    GLfloat m[16],p,q;

    p=-h*(cos(theta)-1)+k*(sin(theta));
    q=-k*(cos(theta)-1)-h*(sin(theta));
    for(i=0;i<15;i++)
        m[i]=0.0;
    m[0]=cos(theta);
    m[1]=sin(theta);
    m[4]=-sin(theta);
    m[5]=cos(theta);
    m[12]=p;
    m[13]=q;
    m[10]=1;
    m[15]=1;
    glMatrixMode(GL_MODELVIEW);
    glClear(GL_COLOR_BUFFER_BIT);
    drawhouse();
    glPushMatrix();
    glMultMatrixf(m);
    drawhouse();
    glPopMatrix();
}

```

Department of Computer Science & Engineering

```

    glFlush();
}
void myinit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(1.0,1.0,0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,499.0,0.0,499.0);
    glMatrixMode(GL_MODELVIEW);
}
int main(int argc,char **argv)
{
    printf("enter the rotation angle\n");
    scanf("%f",&theta);
    theta=theta*3.141/180;
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("rotated house");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
    return 0;
}

```

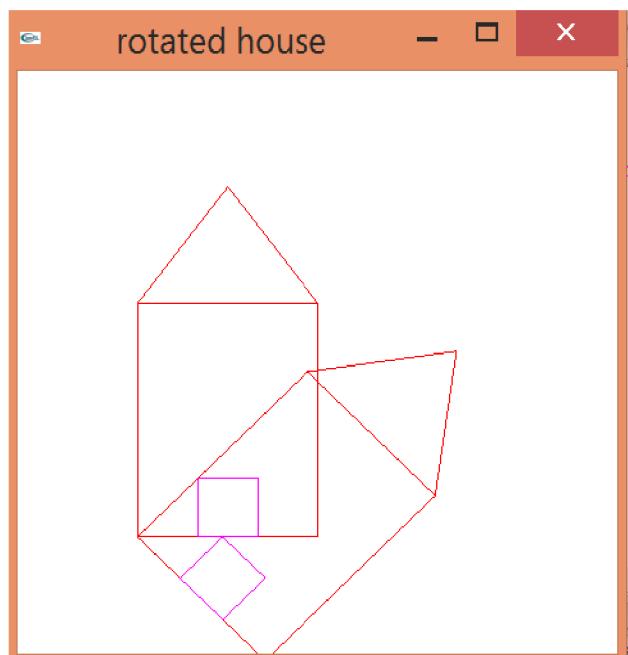
OUT PUT

"D:\SHILPA\4th Year\7... - □ X

```

enter the rotation angle
-45

```



9.Program to create a random object and to implement the suggested mouse and keyboard interactions through OpenGL function.

```
#include<windows.h>
#include<GL/glut.h>
#include<stdio.h>
#include<math.h>
GLfloat wh=500, ww=500;
GLfloat size=10.0;
```

```
void init()
{
```

```
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0,1,0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,ww,0,wh);
```

```
}
```

```
void myReshape(GLint w, GLint h)
{
```

```
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,double(w),0,double(h));
```

```
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0,0,w,h);
    ww=w;
    wh=h;
```

```
}
```

```
void draw(int x, int y)
{
```

```
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    y=wh-y;
    glBegin(GL_POLYGON);
```

```

glVertex2f(x+size, y+size);
glVertex2f(x-size, y+size);
glVertex2f(x-size, y-size);
glVertex2f(x+size, y-size);
glEnd();
glFlush();
}

void mymouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_RIGHT_BUTTON && state== GLUT_DOWN)
        draw(x,y);
}

```

```

void display()
{
    glClearColor(1,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    glFlush();
}

```

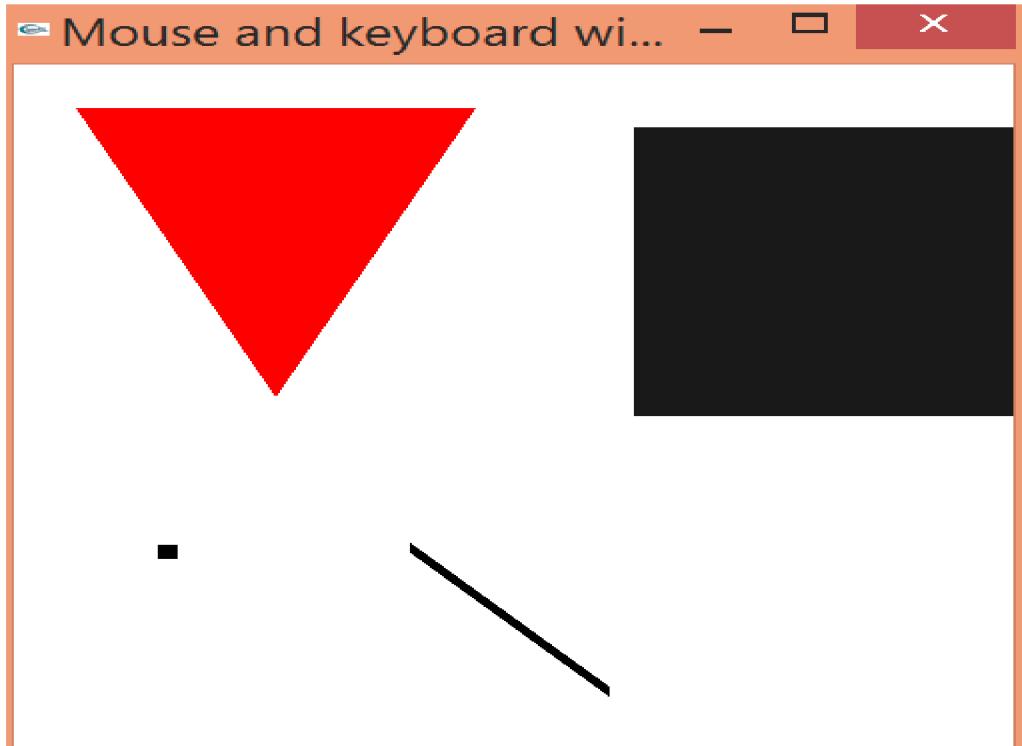
```

int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(wh,ww);
    glutInitWindowPosition(0,0);
    glutCreateWindow("NITTE");
    init();
    glutDisplayFunc(display);

    glutMouseFunc(mymouse);
    glutReshapeFunc(myReshape);
    glutMainLoop();
}

```

OUT PUT



10. Program to draw a color cube and spin it using OpenGL transformation matrices along x, y and z axes.

```

#include<windows.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<stdio.h>
GLfloat vertices[][3]={{{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},{1.0,1.0,-1.0},
{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},{1.0,-1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};
GLfloat normals[][3]={{-1.0,-1.0,-1.0},{1.0,-1.0,1.0},{1.0,1.0,-1.0},
{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},{1.0,-1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};
GLfloat colors[][3]={{0.0,0.0,0.0},{1.0,0.0,0.0},{1.0,1.0,0.0},{0.0,1.0,0.0},
{0.0,0.0,1.0},{1.0,0.0,1.0},{1.0,1.0,1.0},{0.0,1.0,1.0}};

void polygon(int a,int b,int c,int d)
{
    glBegin(GL_POLYGON);
    glColor3fv(colors[a]);
    glNormal3fv(normals[a]);
    glVertex3fv(vertices[a]);
    glColor3fv(colors[b]);
    glNormal3fv(normals[b]);
    glVertex3fv(vertices[b]);
    glColor3fv(colors[c]);
    glNormal3fv(normals[c]);
    glVertex3fv(vertices[c]);
    glColor3fv(colors[d]);
    glNormal3fv(normals[d]);
    glVertex3fv(vertices[d]);
}
    
```

Department of Computer Science & Engineering

```

glVertex3fv(vertices[b]);
glColor3fv(colors[c]);
glNormal3fv(normals[c]);
glVertex3fv(vertices[c]);
glColor3fv(colors[d]);
glNormal3fv(normals[d]);
glVertex3fv(vertices[d]);
glEnd();
}
void colorcube(void)
{
    polygon(0,3,2,1);
    polygon(2,3,7,6);
    polygon(0,4,7,3);
    polygon(1,2,7,6);
    polygon(4,5,6,7);
    polygon(0,1,5,4);
}
static GLfloat theta[]={0.0,0.0,0.0,0.0};
static GLint axis=2;
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(theta[0],1.0,0.0,0.0);
    glRotatef(theta[1],0.0,1.0,0.0);
    glRotatef(theta[2],0.0,0.0,1.0);
    colorcube();
    glFlush();
    glutSwapBuffers();
}
void spinCube()
{
    theta[axis]+=1.0;
    if(theta[axis]>360.0) theta[axis]=-360.0;
    glutPostRedisplay();
}
void mouse(int btn,int state,int x,int y)
{
    if(btn==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)axis=0;
    if(btn==GLUT_MIDDLE_BUTTON&&state==GLUT_DOWN)axis=1;
    if(btn==GLUT_RIGHT_BUTTON&&state==GLUT_DOWN)axis=2;
}
void myReshape(int w,int h)
{

```

```

glViewport(0,0,w,h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if(w<=h)
    glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-
10.0,10.0);
else
    glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-
10.0,10.0);
glMatrixMode(GL_MODELVIEW);

}
int main(int argc,char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(500,500);
    glutCreateWindow("Rotating a color cube");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutIdleFunc(spinCube);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}

```

OUT PUT

