Develop a program to demonstrate basic geometric operations on the 3D object

**Design, develop and implement recursively subdivide a tetrahedron to form 3D sierpinski gasket. The number of recursive steps is to be specified by the user.(LAB PROGRAM**)

#include <stdlib.h>

#include <stdio.h>

#include <GL/glut.h>

typedef GLfloat point[3]; //Three coordinates x, y,z values

point v[]={{-1.0,-0.5,0.0},{1.0,-0.5,0.0},{0.0,1.0,0.0}, {0.0,0.0,1.0}}; // 4 vertices

GLfloat colors[4][3]={{1.0,0.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0},{1.0,1.0,0.0}};// color for four faces of tetrahedron

int n;// no of divisions

void triangle(point a,point b,point c) // creating a face of tetrahedron

{

glBegin(GL\_POLYGON);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glEnd();

}

void tetra(point a,point b,point c,point d) // representation of the tetrahedron

{

glColor3fv(colors[0]);

triangle(a,b,c); // face-1

glColor3fv(colors[1]);

triangle(a,c,d); //face-2

glColor3fv(colors[2]);

triangle(a,d,b); // face-3

glColor3fv(colors[3]);

triangle(b,d,c); // face-4

}

void divide\_tetra(point a,point b,point c,point d,int m) // perform division

{

point mid[6];

int j;

if(m>0)

{

for(j=0;j<3;j++) // generating the bisetors for each edge

{

mid[0][j]=(a[j]+b[j])/2.0; // mid0 between a and b

mid[1][j]=(a[j]+c[j])/2.0; // mid1 between a and c

mid[2][j]=(a[j]+d[j])/2.0; // mid2 between a and d

mid[3][j]=(b[j]+c[j])/2.0; //// mid3 between b and c

mid[4][j]=(c[j]+d[j])/2.0; // mid4 between c and d

mid[5][j]=(b[j]+d[j])/2.0; // mid5 between b and d

}

divide\_tetra(a,mid[0],mid[1],mid[2],m-1); // apex tetrahedron

divide\_tetra(mid[0],b,mid[3],mid[5],m-1); // left corner

divide\_tetra(mid[1],mid[3],c,mid[4],m-1); // front

divide\_tetra(mid[2],mid[5],mid[4],d,m-1); // right

}

else

tetra(a,b,c,d);// creating the tetrahedron

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glClearColor(1.0,1.0,1.0,1.0);

divide\_tetra(v[0],v[1],v[2],v[3],n);

glFlush();

}

void myReshape(int w,int h)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

//glOrtho(-1,1,-1,1,-1,1);

glOrtho(-1.0,1.0,-1.0\*((GLfloat)h/(GLfloat)w), 1.0\*((GLfloat)h/(GLfloat)w),-1.0,1.0);

else

//glOrtho(-1,1,-1,1,-1,1);

glOrtho(-1.0\*((GLfloat)w/(GLfloat)h),1.0\*((GLfloat)w/(GLfloat)h),-1.0,1.0,-1.0,1.0);

glMatrixMode(GL\_MODELVIEW);

glutPostRedisplay();

}

void main(int argc,char \*\* argv)

{

printf( "No of Division?: ");

scanf("%d",&n);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(500,500);

glutCreateWindow( "3D gasket" );

glutDisplayFunc(display);

glutReshapeFunc(myReshape);

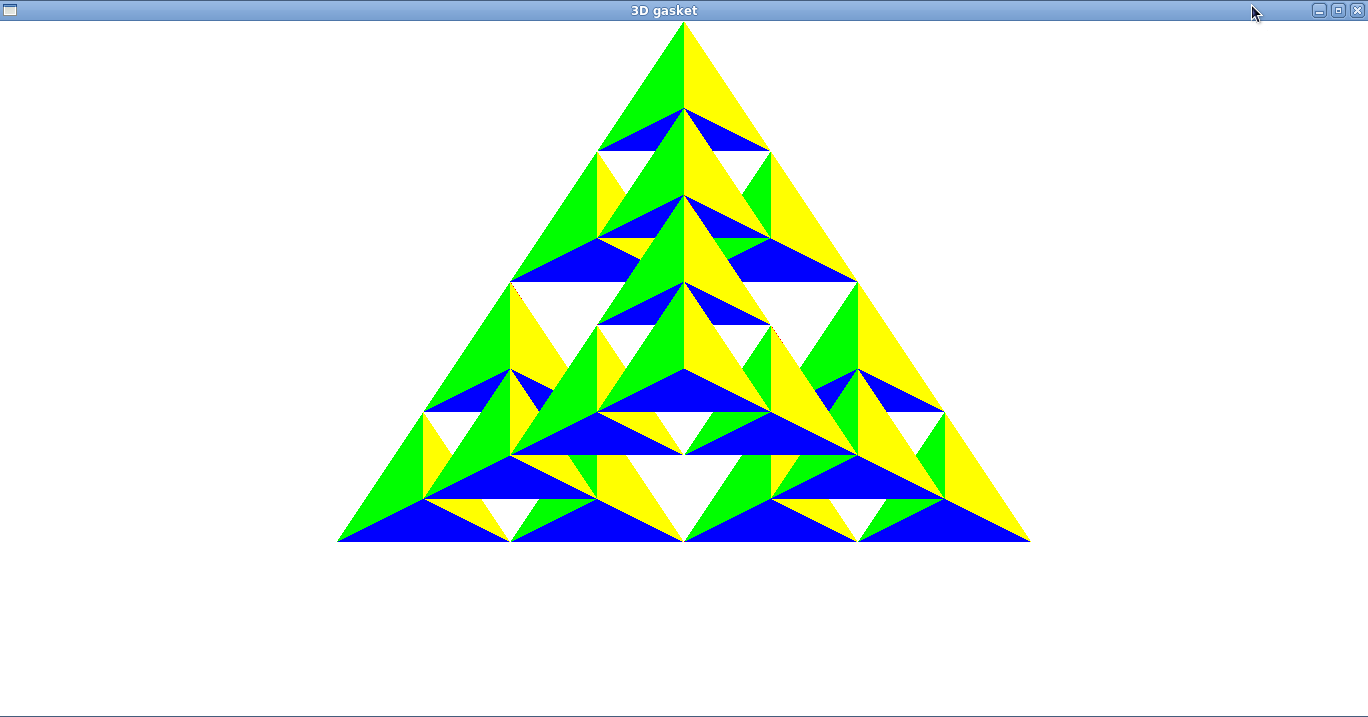
glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

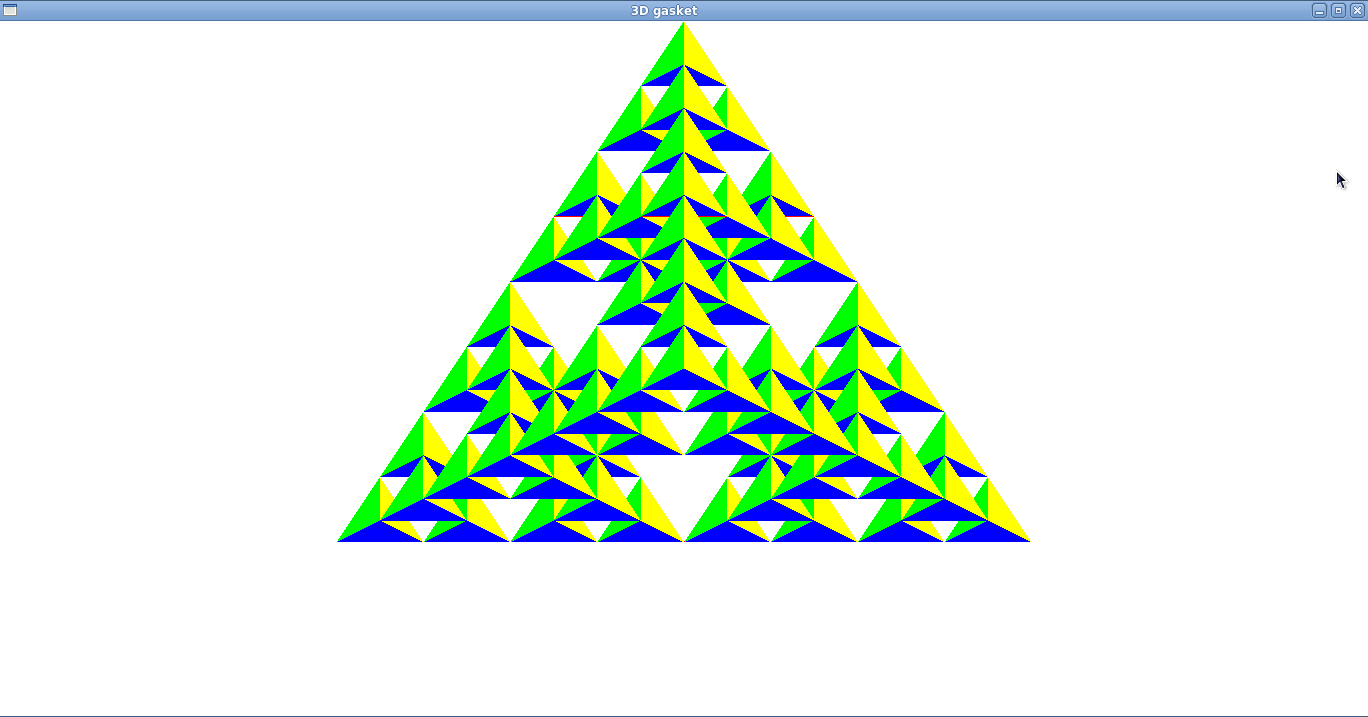
}

**\*\*\*\* Output \*\*\***

No. of Divisions ? 2



No. of Divisions ? 3



Develop a program to demonstrate basic geometric operations on the 2D object

**2D RECURSIVE SIERPINSK**I

#include<stdio.h>

#include<GL/glut.h>

int n;

typedef GLfloat point2[2];

point2 v[3]= {{-2,-1}, {2,-1},{0,1}};

void triangle( GLfloat \*a, GLfloat \*b, GLfloat \*c)

/\* display one triangle \*/

{

glVertex2fv(a);

glVertex2fv(b);

glVertex2fv(c);

}

void divide\_triangle(GLfloat \*a, GLfloat \*b, GLfloat \*c, int m)

{

/\* triangle subdivision using vertex numbers \*/

point2 v0, v1, v2;

int j;

if(m>0)

{

for(j=0; j<2; j++) v0[j]=(a[j]+b[j])/2;

for(j=0; j<2; j++) v1[j]=(a[j]+c[j])/2;

for(j=0; j<2; j++) v2[j]=(b[j]+c[j])/2;

divide\_triangle(a, v0, v1, m-1);

divide\_triangle(c, v1, v2, m-1);

divide\_triangle(b, v2, v0, m-1);

}

else(triangle(a,b,c));

/\* draw triangle at end of recursion \*/

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

divide\_triangle(v[0], v[1], v[2], n);

glEnd();

glFlush();

}

void myinit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-2.0, 2.0, -2.0, 2.0);

glMatrixMode(GL\_MODELVIEW);

glClearColor (1.0, 1.0, 1.0,1.0);

glColor3f(1.0,0.0,0.0);

}

int main(int argc, char \*\*argv)

{

printf("Enter the number of divisions\n");

scanf("%d",&n);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("2D Gasket");

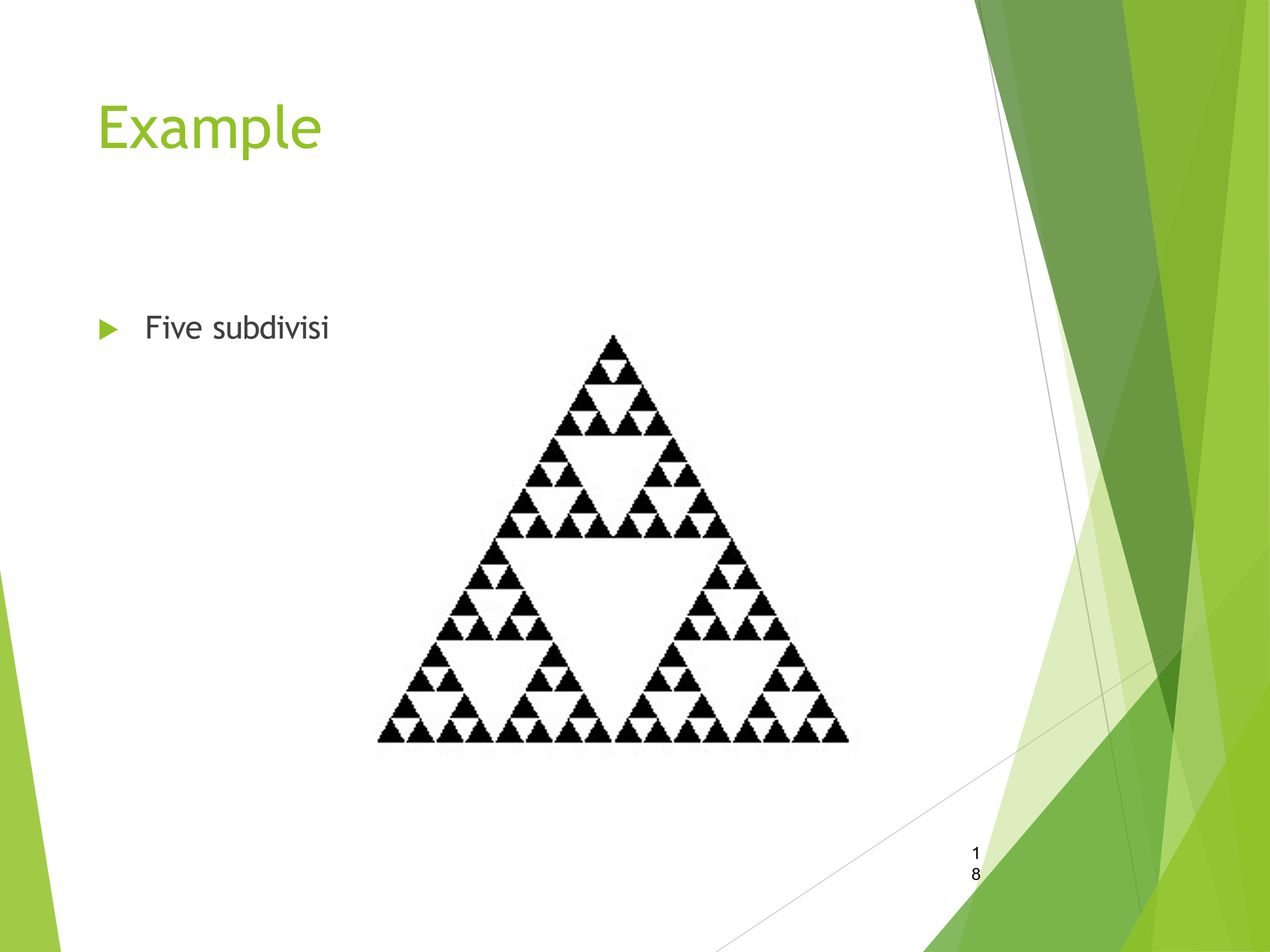
glutDisplayFunc(display);

myinit();

glutMainLoop();

return 0;

}



**2D RANDOM SIERPINSKI GASKET**

#include<stdio.h>

#include<GL/glut.h>

#include<stdlib.h>

int n;

void display()

{

GLfloat vertices[3][3]={{0.0,0.0,0.0},{25.0,50,0.0},{50.0,0.0,0.0}};

GLfloat p[3]={7.5,5.0,0.0};

int j, k;

int rand();

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POINTS);

for(k=0;k<n;k++)

{ j=rand()%3;

p[0]=(p[0]+vertices[j][0])/2;

p[1]=(p[1]+vertices[j][1])/2;

glVertex3fv(p);

}

glEnd();

glFlush();

}

void myinit()

{ glClearColor (1.0, 1.0, 1.0,1.0);

glColor3f(0.0,0.0,0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 60.0, 0.0, 60.0);

//glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char \*\*argv)

{

printf("Enter the number of divisions\n");

scanf("%d",&n);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("2D Gasket");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}

