**Name:** Nishan Paul

**ID:** 1604085

**Course Title:** Computer Graphics (Sessional)

**Course No:** CSE-458

**Level 4, Term 1**

**Assignment 09**

**Title:** Draw a given shape

**Code:**

#include<windows.h>

#include<GL/glut.h>

#include<bits/stdc++.h>

#define PI 3.1416

void BoundaryFill(int x, int y, float\* fill\_color, float\* boundary\_color) {

float color[3];

glReadPixels(x, y, 1.0, 1.0, GL\_RGB, GL\_FLOAT, color);

if((color[0] != boundary\_color[0] || color[1] != boundary\_color[1] || color[2] !=

boundary\_color[2] ) &&

(color[0] != fill\_color[0] || color[1] != fill\_color[1] || color[2] != fill\_color[2])) {

glColor3f(fill\_color[0], fill\_color[1], fill\_color[2]);

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

BoundaryFill(x + 1, y, fill\_color, boundary\_color);

BoundaryFill(x - 1, y, fill\_color, boundary\_color);

BoundaryFill(x, y + 1, fill\_color, boundary\_color);

BoundaryFill(x, y - 1, fill\_color, boundary\_color);

}

}

void c\_curve\_(float x, float y, float len, float alpha, int n) {

if(n>0) {

float offset = (60\*PI)/180;

c\_curve\_(x,y,len,alpha+offset,n-1);

x=x+len\*cos(alpha+offset);

y=y+len\*sin(alpha+offset);

c\_curve\_(x,y,len,alpha-offset,n-1);

}

else {

glBegin(GL\_LINES);

glVertex2d(x,y);

glVertex2d(x+(len\*cos(alpha)),y+(len\*sin(alpha)));

glEnd();

glFlush();

}

}

void c\_curve(void) {

glColor3f(0.0,1.0,0.0);

float x, y, len, x1, y1, x2, y2, x3, y3, radian, degree = 0;

x = 200;

y = 300;

len = 100;

x1 = x;

y1 = y;

radian = (degree\*PI)/180;

c\_curve\_(x,y,len,radian,1);

x = x+(len\*cos(radian));

y = y+(len\*sin(radian));

x2 = x;

y2 = y;

degree = 180+108;

radian = (degree\*PI)/180;

c\_curve\_(x,y,len,radian,1);

x = x+(len\*cos(radian));

y = y+(len\*sin(radian));

degree = 216;

radian = (degree\*PI)/180;

c\_curve\_(x,y,len,radian,1);

x = x+(len\*cos(radian));

y = y+(len\*sin(radian));

x3 = x;

y3 = y;

degree = 144;

radian = (degree\*PI)/180;

c\_curve\_(x,y,len,radian,1);

x = x+(len\*cos(radian));

y = y+(len\*sin(radian));

degree = degree-72;

radian = (degree\*PI)/180;

c\_curve\_(x,y,len,radian,1);

}

void algo\_ellipse\_(int x, int y) {

int center\_x = 250, center\_y = 243;

glVertex2i(center\_x + x, center\_y + y);

glVertex2i(center\_x - x, center\_y + y);

glVertex2i(center\_x + x, center\_y - y);

glVertex2i(center\_x - x, center\_y - y);

}

void algo\_ellipse() {

glColor3f(0,0,0);

glBegin(GL\_POINTS);

int a = 70, b = 143;

float x = 0;

float y = b;

float p = b\*b - (a\*a\*b) + (a\*a\*0.25) ;

float dx = 0;

float dy = a\*a\*2\*b;

while(dx < dy) {

algo\_ellipse\_(x, y);

x++;

dx += (b\*b\*2);

if(p < 0)

p = p + dx + (b\*b);

else {

y--;

dy -= (a\*a\*2);

p += dx - dy +(b\*b);

}

}

float p2 = (b\*b\*(x + 0.5)\*(x + 0.5)) + (a\*a\*(y - 1)\*(y - 1)) - (a\*a\*b\*b);

while(y > 0) {

algo\_ellipse\_(x, y);

y--;

dy -= (a\*a\*2);

if(p2 >= 0)

p2 -= dy + (a\*a);

else {

x++;

dx += (2\*b\*b);

p2 += dx - dy + (a\*a);

}

}

glEnd();

glFlush();

}

void rectangle(float x1, float y1, float x2, float y2) {

glColor3f(0,0,0);

float x = x1;

float y = y1;

float dx = x2-x1;

float dy = y2-y1;

float m = dy/dx;

int limit = m<=1? abs(dx) : abs(dy);

if(m<=1)

dx = 1;

else

dy = 1;

for(int i=0; i<limit; i++){

x += dx;

y += dy;

glBegin(GL\_POINTS);

glVertex2f(x,y);

}

glEnd();

glFlush();

}

void init(void) {

glClearColor(1,1,1,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 640, 0, 480);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(100,100);

glutCreateWindow("1604085 / lab final");

init();

glEnable(GL\_POINT\_SMOOTH);

glutDisplayFunc([]() {

algo\_ellipse();

rectangle(100, 100, 400, 100);

rectangle(400, 100, 400, 386);

rectangle(100, 386, 400, 386);

rectangle(100, 100, 100, 386);

c\_curve();

float boundary[] = {0, 1.0, 0};

float fillcolor[] = {0, 1.0, 0};

BoundaryFill(200, 200, fillcolor, boundary);

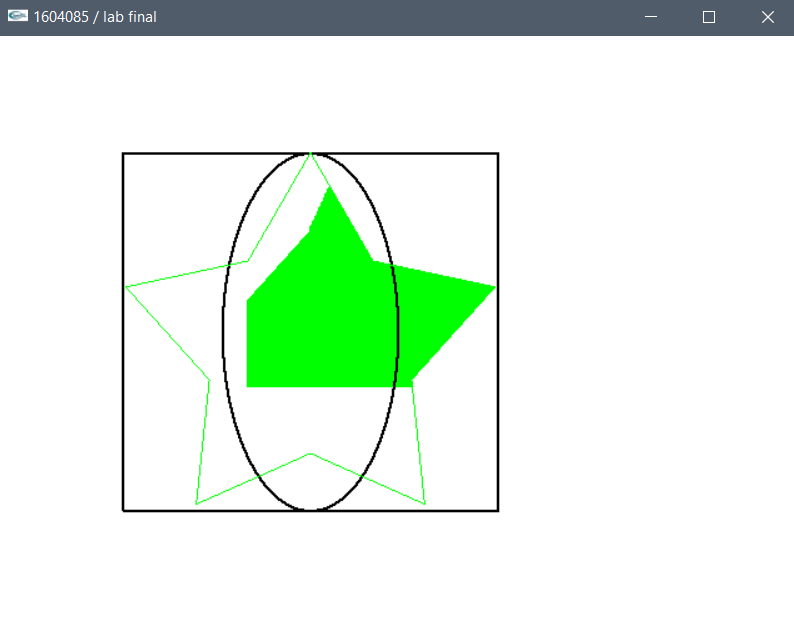
});

glutMainLoop();

return 0;

}

**Output:**

****