***TRIANGLE:-***

#include <GL/glut.h>

void displayMe(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POLYGON);

glVertex3f(0.5, 0.0, 0.0);

glVertex3f(0.0, 0.5, 0.0);

glVertex3f(0.0, 0.0, 0.5);

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(400, 300);

glutInitWindowPosition(100, 100);

glutCreateWindow("Hello world!");

glutDisplayFunc(displayMe);

glutMainLoop();

return 0;

}

//$ g++ main.cpp -o firstOpenGlApp -lglut -lGLU -lGL

//$ ./firstOpenGlApp

***DDA ALGORITHM:-***

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

using namespace std;

double a=0,b=0,c=0,d=0;

float round\_value(float v)

{

return floor(v + 0.5);

}

void LineDDA(double X1,double Y1,double X2,double Y2)

{

double dx=(X2-X1);

double dy=(Y2-Y1);

double steps;

float xInc,yInc,x=X1,y=Y1;

steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));

xInc=dx/(float)steps;

yInc=dy/(float)steps;

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_POINTS);

glVertex2d(x,y);

int k;

for(k=0;k<steps;k++) {

x=x+xInc;

y=y+yInc;

glVertex2d(round\_value(x), round\_value(y));

}

glEnd();

glFlush();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

LineDDA(a,b,c,d);

}

void Init()

{

glClearColor(1.0,1.0,1.0,0);

glColor3f(1.0,0.0,0.0);

glViewport(0 , 0 , 640 , 480);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0 , 640 , 0 , 480);

}

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

{

printf("\nEnter values of First endpoint");

scanf("%lf%lf",&a,&b);

printf("\nEnter values of Second endpoint");

scanf("%lf%lf",&c,&d);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(0,0);

glutCreateWindow("Assignment 1 DDA");

Init();

glutDisplayFunc(display);

glFlush();

glutMainLoop();

}

/\*

ubuntu@ubuntu:~$ g++ dda.c -lglut -lGL -lGLEW -lGLU

ubuntu@ubuntu:~$ ./a.out

Enter values of First endpoint 100.0 100.0

Enter values of Second endpoint 600.0 600.0

\*/

***BREZERS CIRCLE:-***

#include<GL/glut.h>

#include<iostream>

using namespace std;

int r;

void E\_way(int x, int y){

glBegin(GL\_POINTS);

glVertex2i(x+320,y+240);

glVertex2i(y+320,x+240);

glVertex2i(y+320, -x+240);

glVertex2i(x+320, -y+240);

glVertex2i(-x+320,-y+240);

glVertex2i(-y+320,-x+240);

glVertex2i(-y+320,x+240);

glVertex2i(-x+320,y+240);

glEnd();

glFlush();

}

void B\_circle(){

float d;

d = 3 - 2\*r;

int x,y;

x = 0 ;

y = r ;

do{

E\_way(x,y);

if(d<0){

d=d+4\*x+6;

}

else{

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

y=y-1;

}

x=x+1;

}while(x<y);

}

void init(){

glClearColor(1,1,1,0);

glColor3f(1,0,0);

gluOrtho2D(0,640,0,480);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

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

cout<<"\n Enter Radius \t ";

cin>>r;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(100,100);

glutInitWindowSize(640,480);

glutCreateWindow("Circle");

init();

glutDisplayFunc(B\_circle);

glutMainLoop();

return 0;

}

OUTPUT

g++ filename.cpp -lGL -lGLU -lglut

./a.out

***RECTANGLE:-***

#include <GL/glut.h>  
  
void Init(void) {  
 glClearColor(1.0f, 1.0f, 1.0f, 0.0f); glMatrixMode(GL\_PROJECTION);   
 gluOrtho2D(-400.0f, 400.0f, -400.0f, 400.0f);  
}  
  
void Draw\_colourful\_triangle(void) {  
 glClear(GL\_COLOR\_BUFFER\_BIT);   
 glColor3f(0.0f, 0.7f, 0.3f);   
 glBegin(GL\_POLYGON);   
 glVertex2i(-200, -200);  
 glVertex2i(200, -200);  
 glVertex2i(200, 200);  
 glVertex2i(-200, 200);  
 glEnd();   
  
 glFlush();   
}  
  
int main(int argc, char\*\* argv) {  
 glutInit(&argc, argv);   
 glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);   
 glutInitWindowPosition(50, 100);   
 glutInitWindowSize(800, 800); glutCreateWindow("A GLUT window canves for CG");  
  
 Init(); glutDisplayFunc(Draw\_colourful\_triangle);  
 glutMainLoop();  
}

***MAIN MENU:-***

#include <iostream>

#include <math.h>

#include <time.h>

#include <GL/glut.h>

using namespace std;

float R=0,G=0,B=0;

int Algo;

void init(){

glClearColor(1.0,1.0,1.0,0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,640,0,480);

}

void mouse(int btn, int state, int x, int y){

y = 480-y;

if(btn==GLUT\_LEFT\_BUTTON)

{

if(state==GLUT\_DOWN)

{

float bcol[] = {1,0,0};

float newCol[] = {R,G,B};

float oldcol[] = {1,1,1};

}

}

}

void goMenu(int value){

switch(value){

case 1:

R = 0, G = 1, B=0;

break;

case 2:

R = 1, G = 1, B=0;

break;

case 3:

R = 1, G = 0, B=1;

break;

case 4:

R = 0, G = 0, B=1;

break;

}

glutPostRedisplay();

}

void world()

{

glLineWidth(3);

glPointSize(2);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(150,100);

glVertex2i(300,300);

glVertex2i(450,100);

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(200,200);

glutCreateWindow("Polygon filling Methods");

init();

glutCreateMenu(goMenu);

glutAddMenuEntry("Color 1 Green",1);

glutAddMenuEntry("Color 2 Yellow",2);

glutAddMenuEntry("Color 3 Pink",3);

glutAddMenuEntry("Color 4 Blue",4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMouseFunc(mouse);

glutMainLoop();

return 0;

}

***KOACH CURVE:-***

#include <GL/glut.h>

#include <math.h>

GLfloat oldx=-0.7,oldy=0.5;

void drawkoch(GLfloat dir,GLfloat len,GLint iter)

{

GLdouble dirRad = 0.0174533 \* dir ;

GLfloat newX = oldx + len \* cos(dirRad);

GLfloat newY = oldy + len \* sin(dirRad);

if (iter==0)

{

glVertex2f(oldx, oldy);

glVertex2f(newX, newY);

oldx = newX;

oldy = newY;

}

else

{

iter--;

drawkoch(dir, len, iter);

dir += 60.0;

drawkoch(dir, len, iter);

dir -= 120.0;

drawkoch(dir, len, iter);

dir += 60.0;

drawkoch(dir, len, iter);

}

}

void display()

{

glClearColor(1.0,1.0,1.0,0);

glColor3f(0.0, 0.0, 0.0);

glClear( GL\_COLOR\_BUFFER\_BIT );

glBegin(GL\_LINES)

drawkoch(0.0,0.05,3);

drawkoch(-120.0, 0.05, 3);

drawkoch(120.0,0.05,3);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Koch Curves");

glutDisplayFunc(display);

glutMainLoop();

}