2DSCALING

#include <GL/glut.h>

#include <iostream>

#include <math.h>

using namespace std;

void findnewcoordinate(int s[][2], int p[][1]){

int temp[2][1] = {0};

for (int i = 0;i<2;i++)

for(int j = 0;j<1;j++)

for (int k = 0;k<2;k++)

temp[i][j] += (s[i][k] \* p[k][j]);

p[0][0] = temp[0][0];

p[1][0] = temp[1][0];

}

void scale(int x[], int y[], int sx, int sy){

glBegin(GL\_LINE\_LOOP);

glVertex2f(x[0],y[0]);

glVertex2f(x[1],y[1]);

glVertex2f(x[1],y[1]);

glVertex2f(x[2],y[2]);

glVertex2f(x[2],y[2]);

glVertex2f(x[0],y[0]);

glEnd();

int s[2][2] = {sx, 0, 0, sy};

int p[2][1];

for(int i = 0;i<3;i++){

p[0][0]=x[i];

p[1][0]=y[i];

findnewcoordinate(s,p);

x[i] = p[0][0];

y[i] = p[1][0];

}

glBegin(GL\_LINE\_LOOP);

glVertex2f(x[0],y[0]);

glVertex2f(x[1],y[1]);

glVertex2f(x[1],y[1]);

glVertex2f(x[2],y[2]);

glVertex2f(x[2],y[2]);

glVertex2f(x[0],y[0]);

glEnd();

glFlush();

}

void init(void){

glClearColor(0.0,0.0,0.0,0.0);

gluOrtho2D(0,500,0,500);

}

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

int x[] = {100,200,300};

int y[] = {200,100,200};

int sx = 2,sy = 2;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(0,0);

glutCreateWindow("2Dscaling");

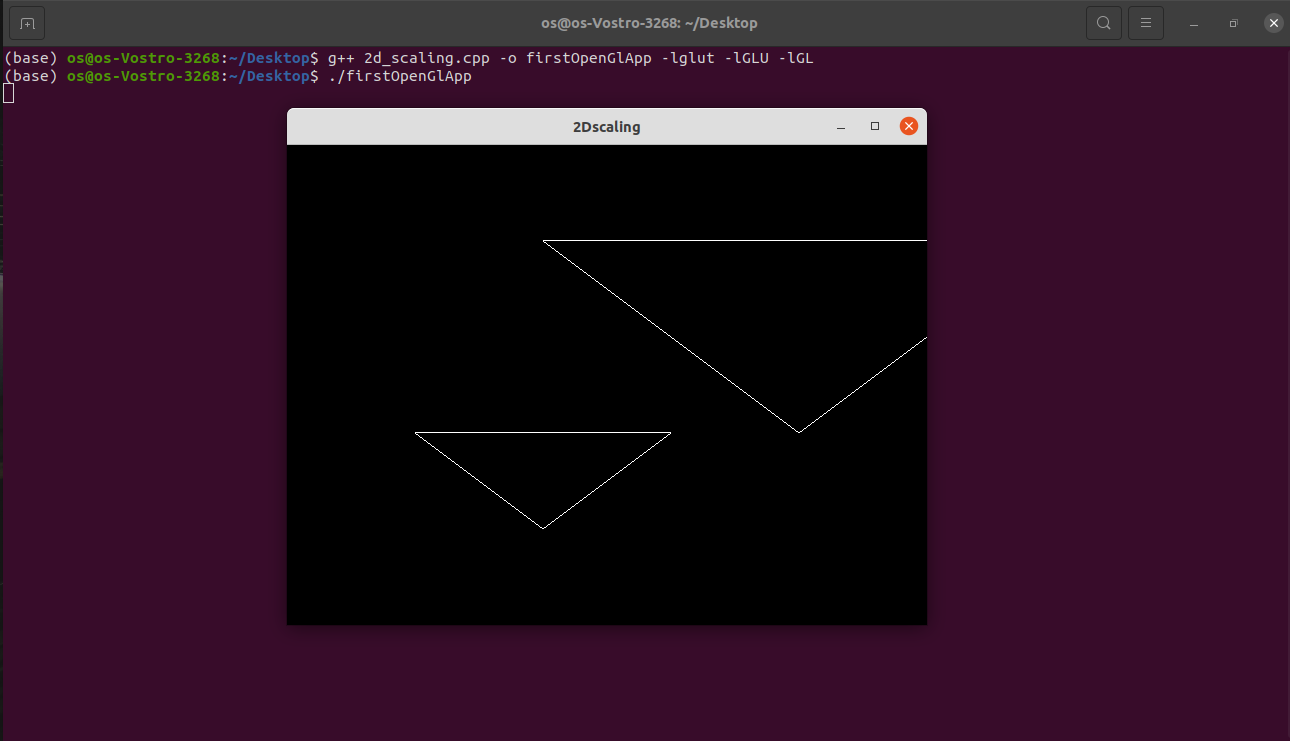
init();

scale(x,y,sx,sy);

glutMainLoop();

return 0;

}



2DTRANSLATION

#include<GL/glut.h>

#include <iostream>

#include <math.h>

using namespace std;

void translateRectangle(int P[][2], int T[]){

glBegin(GL\_LINE\_LOOP);

glVertex2f(P[0][0], P[0][1]);

glVertex2f(P[1][0], P[0][1]);

glVertex2f(P[1][0], P[1][1]);

glVertex2f(P[0][0], P[1][1]);

glEnd(); // calculating translated coordinates

P[0][0] = P[0][0] + T[0];

P[0][1] = P[0][1] + T[1];

P[1][0] = P[1][0] + T[0];

P[1][1] = P[1][1] + T[1];

glBegin(GL\_LINE\_LOOP);

glVertex2f(P[0][0], P[0][1]);

glVertex2f(P[1][0], P[0][1]);

glVertex2f(P[1][0], P[1][1]);

glVertex2f(P[0][0], P[1][1]);

glEnd();

glFlush();

}

void init(void){

glClearColor(0.0, 0.0, 0.0, 0.0);

gluOrtho2D(0, 500, 0, 500);

}

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

int P[2][2] = { 50, 80, 120, 180 };

int T[] = { 100, 100 }; // translation factor

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(0, 0);

glutCreateWindow("2D Transformaton Scaling ");

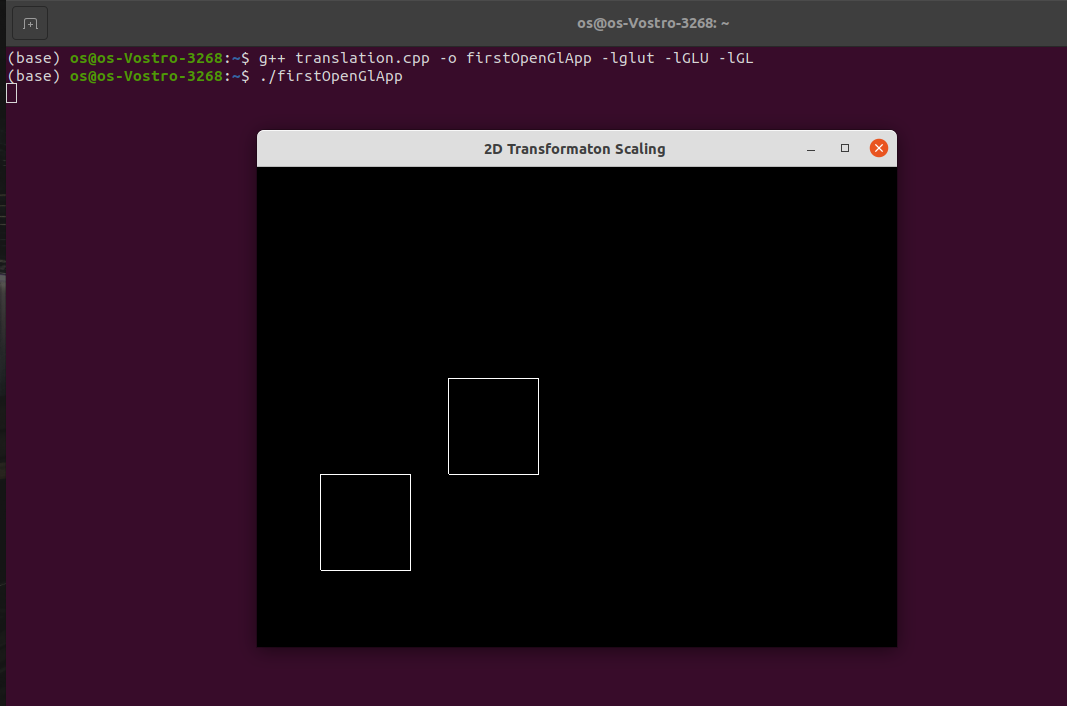
init();

translateRectangle(P, T);

glutMainLoop();

return 0;

}



BOUNDARY FILL

#include<GL/glut.h>

#include<iostream>

#include<math.h>

int ww = 600, wh = 500;

float fillCol[3] = {0.4,0.0,0.0};

float borderCol[3] = {0.0, 0.0,0.0};

void setPixel(int pointx, int pointy, float f[3])

{

glBegin(Gl\_POINTS);

glColor3fv(f);

glVertex2i(pointx,pointy);

glEnd();

glFlush();

}

void getPixel(int x, int y, float pixels[3])

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

}

void drawPolygon(int x1,int y1,int x2,int y2)

{

glColor3f(0.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(x1,y1);

glVertex2i(x1,y2);

glEnd();

glBegin(GL\_LINES);

glVertex2i(x2,y1);

glVertex2i(x2,y2);

glEnd();

glBegin(GL\_LINES);

glVertex2i(x1,y1);

glVertex2i(x2,y1);

glEnd();

glBegin(GL\_LINES);

glVertex2i(x1,y2);

glVertex2i(x2,y2);

glEnd();

glFlush();

}

void display(){

glClearColor(0.6,0.4,0.1,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

drawPolygon(150,250,200,300);

glFlush();

}

void boundaryFill4(int x,int y,float fillColor[3],float borderColor[3])

{

float interiorColor[3];

getPixel(x,y,interiorColor);

if ((interiorColor[0] != borderColor[0] && interiorColor[1] != borderColor[1] && interiorColor[2] != borderColor[2])&& (interiorColor[0] != fillColor[0] && interiorColor[1] != fillColor[1] && interiorColor[2] != fillColor[2]))

{

setPixel(x,y,fillColor);

boundaryFill4(x,y-1,fillColor,borderColor);

boundaryFill4(x,y+1,fillColor,borderColor);

boundaryFill4(x+1,y,fillColor,borderColor);

boundaryFill4(x-1,y,fillColor,borderColor);

}

}

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

{

if(btn == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

int xi = x;

int yi = (wh - y);

boundaryFill4( xi , yi , fillCol, borderCol);

}

}

void myinit()

{

glViewport(0,0,ww,wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(ww , wh);

glutCreateWindow("Boundry Fill Recursive");

glutDisplayFunc(display);

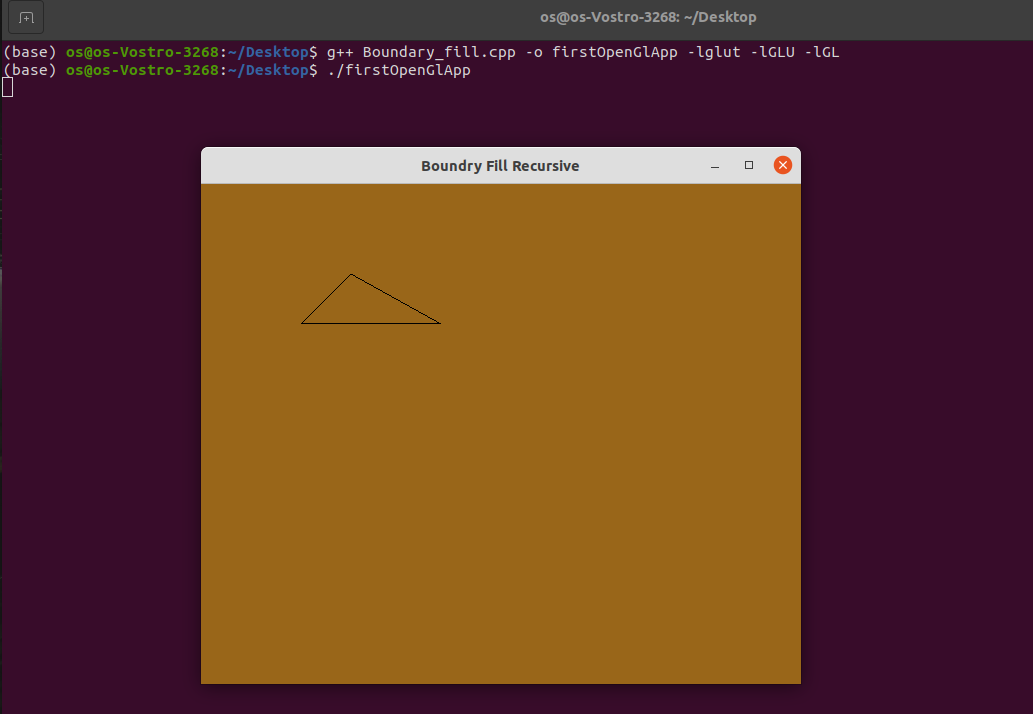
myinit();

glutMouseFunc(mouse);

glutMainLoop();

return 0;

}



BRESENHAM LINE

#include<iostream>

#include<GL/glut.h>

#include<math.h>

using namespace std;

float r,g,b,x,y;

float x\_1,x\_2,y\_1,y\_2;

bool flag = true;

void mouse(int button , int state, int mousex, int mousey){

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN){

flag = true;

x = mousex;

y = 480-mousey;

}

cout<<"mousex = "<<x;

cout<<"mousey = "<<y;

}

int sgn(float a){

if(a==0){

return 0;

}

if(a<0){

return -1;

}

else{

return 1;

}

}

void Line(){

cout<<"x\_1=" << x\_1 <<"y\_1=" << y\_1;

cout<<"x\_2=" << x\_2 <<"y\_2=" << y\_2;

float dx,dy,length,G;

//x\_2 = x;

//y\_2 = y;

dy = y\_2 - y\_1;

dx = x\_2 - x\_1;

G = (2\*dy)-dx;

if(abs(dx) >= abs(dy)){

length = abs(dx);

}

else{

length = abs(dy);

}

int j =0;

x = x\_1;

y = y\_1;

while(j <= length){

if(abs(dx) >= abs(dy)){

x = x+1;

if(G>=0){

y = y+1;

G = G+2\*(dy-dx);

}

else{

G = G + (2\*dy);

}

}

else{

y = y+1;

if(G>=0){

x = x+1;

G = G+2\*(dy-dx);

}

else{

G = G+ (2\*dy);

}

}

cout<< "\n x = " << x;

cout<< "y = " << y;

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

j++;

}

glFlush();

}

void init(void)

{

glClearColor(0,0,0,0);

glColor3f(1.0,1.0,0.0);

gluOrtho2D(0,640,0,640);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

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

{

cout<<"Enter x1,y1 point";

cin>>x\_1>>y\_1;

cout<<"Enter x2,y2 point";

cin>>x\_2>>y\_2;

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE| GLUT\_RGB);

glutInitWindowSize(0,600);

glutCreateWindow("DDA LINE ");

init();

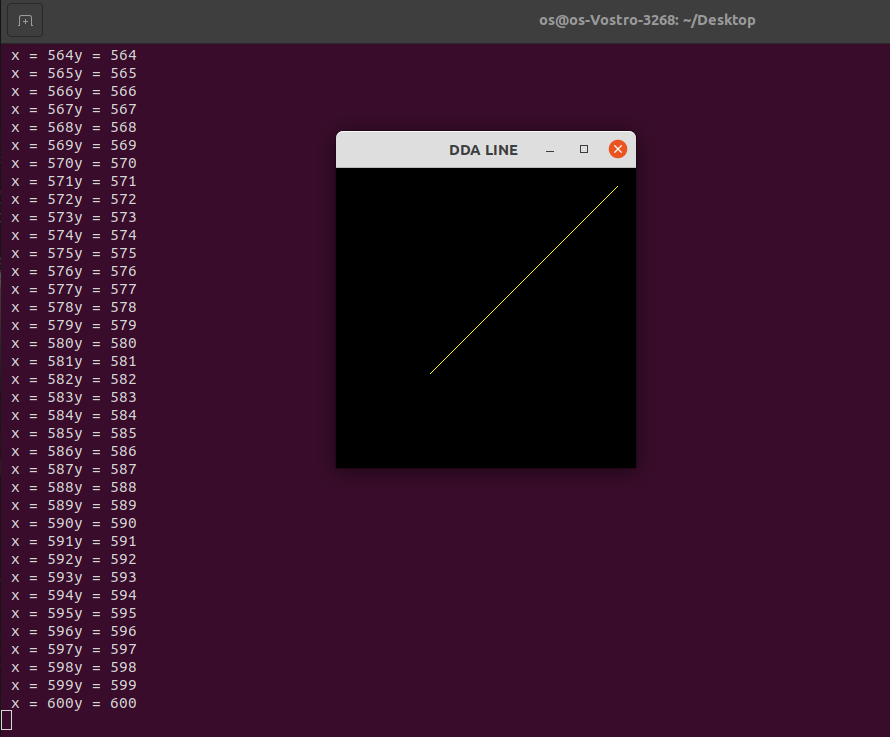
//glutMouseFunc(mouse);

glutDisplayFunc(Line);

glutMainLoop();

return 0;

}



BRESENHAM CIRCLE

#include<stdio.h>

#include<math.h>

#include<GL/glut.h>

int xc = 320, yc = 240;

void plot\_point(int x, int y){

glBegin(GL\_POINTS);

glVertex2i(xc+x, yc+y);

glVertex2i(xc+x, yc-y);

glVertex2i(xc+y, yc+x);

glVertex2i(xc+y, yc-x);

glVertex2i(xc-x, yc-y);

glVertex2i(xc-x, yc+y);

glVertex2i(xc-y, yc-x);

glVertex2i(xc-y, yc+x);

glEnd();

}

void bresenham\_circle(int r){

int x = 0, y = r;

float pk =(5.0, 4.0)-r;

plot\_point(x,y);

int k;

while (x<y){

x = x+1;

if(pk<0)

pk = pk + 2\*x+1;

else{

y = y-1;

pk = pk + 2\*(x-y)+1;

}

plot\_point(x,y);

}

glFlush();

}

void concentric\_circles(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

int radius = 200;

bresenham\_circle(radius);

}

void Init(){

glClearColor(1.0,1.0,1.0,0);

glColor3f(5.0,0.0,7.0);

gluOrtho2D(0,640,0,480);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(0,0);

glutInitWindowSize(640, 480);

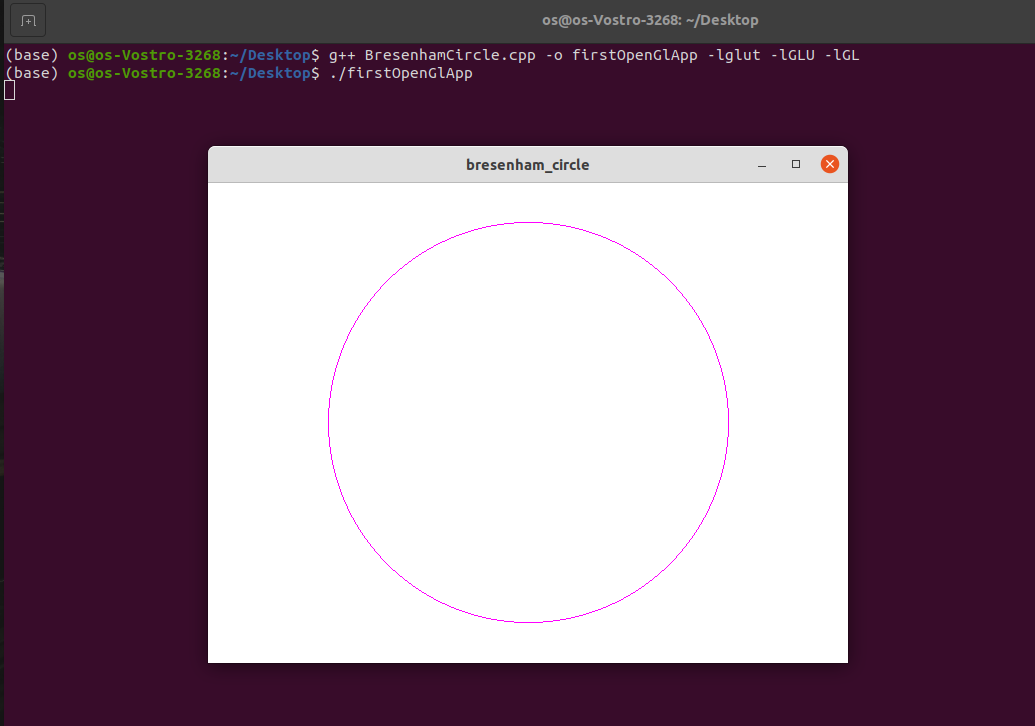
glutCreateWindow("bresenham\_circle");

Init();

glutDisplayFunc(concentric\_circles);

glutMainLoop();

}



DDA LINE

#include<GL/glut.h>

#include<iostream>

#include<math.h>

using namespace std;

float r, g, b, x, y;

float x\_1,x\_2,y\_1,y\_2;

float xin, yin,length;

bool flag = true;

void mouse(int button,int state,int mousex,int mousey)

{

if(button == GLUT\_LEFT\_BUTTON

&& state == GLUT\_DOWN){

flag = true;

x = mousex;

y = 640 - mousey;

}

}

int sgn(float a){

if(a == 0){

return 0;

}

if(a < 0){

return -1;

}

else

return 1;

}

void Line(){

cout<< "x\_1="<<x\_1<<"y\_1="<<y\_1;

cout<< "x\_2="<<x\_2<<"y\_2="<<y\_2;

float dy, dx, length;

x\_2 = x;

y\_2 = y;

dy = y\_2 - y\_1;

dx = x\_2 - x\_1;

if(abs(dx)>=abs(dy)){

length = abs(dx);

}

else{

length = abs(dy);

}

float xin, yin;

xin=(x\_2 - x\_1)/length;

yin=(y\_2 - y\_1)/length;

float x, y;

x=x\_1+0.5\*sgn(xin);

y=y\_1+0.5\*sgn(yin);

int i=0;

while(i<=length)

{

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

x=x+xin;

y=y+yin;

i++;

}

glFlush();

}

void init(void){

glClearColor(0,0,0,0);

glColor3f(1.0,1.0,0.0);

gluOrtho2D(0,640,0,640);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

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

cout<<"Enter x1,y1 point";

cin>>x\_1>>y\_1;

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE| GLUT\_RGB);

glutInitWindowSize(0,640);

glutCreateWindow("DDA LINE DRAWING TECHNIQUE");

init();

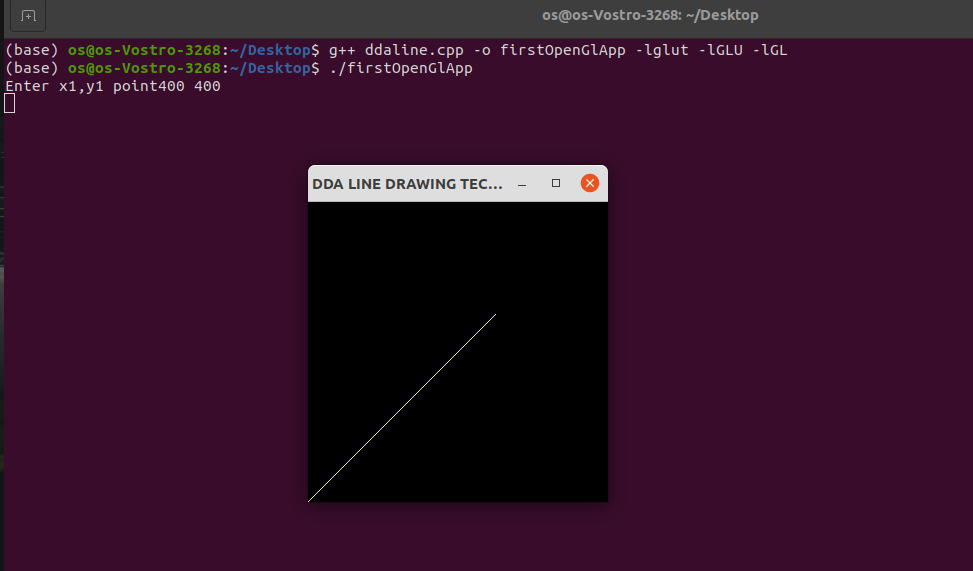
glutMouseFunc(mouse);

glutDisplayFunc(Line);

glutMainLoop();

return 0;

}



FLOOD FILL

#include<GL/glut.h>

#include<iostream>

#include<math.h>

int ww = 800, wh = 700;

float bgCol[3]={0.2,0.4,0.0};

float intCol[3]={0.0,0.0,1.0};

float fillCol[3] = {0.4 ,0.0,0.0};

void setPixel(int pointx, int pointy, float f[3])

{

glBegin(GL\_POINTS);

glColor3fv(f);

glVertex2i(pointx, pointy);

glEnd();

glFlush();

}

void getPixel(int x, int y, float pixels[3])

{

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

}

void drawPolygon(int x1, int y1, int x2, int y2)

{

glColor3f(0.0,0.0,1.0);

glBegin(GL\_POLYGON);

glVertex2i(x1,y1);

glVertex2i(x1,y2);

glVertex2i(x2,y2);

glVertex2i(x2,y1);

glEnd();

glFlush();

}

void display()

{

glClearColor(0.2,0.4,0.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

drawPolygon(150,250,200,300);

glFlush();

}

void floodfill4(int x, int y, float oldColor[3], float newcolor[3])

{

float color[3];

getPixel(x, y, color);

if(color[0]==oldColor[0]&&(color[1])==oldColor[1]&&(color[2])==oldColor[2])

{

setPixel(x, y, newcolor);

floodfill4(x+1, y,oldColor,newcolor);

floodfill4(x-1, y,oldColor,newcolor);

floodfill4(x, y+1,oldColor,newcolor);

floodfill4(x, y-1,oldColor,newcolor);

}

}

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

{

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

int xi = x;

int yi = (wh - y);

floodfill4(xi, yi, intCol, fillCol);

}

}

void myinit()

{

glViewport(0,0,ww,wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(ww , wh);

glutCreateWindow("Flood Fill Recursive");

glutDisplayFunc(display);

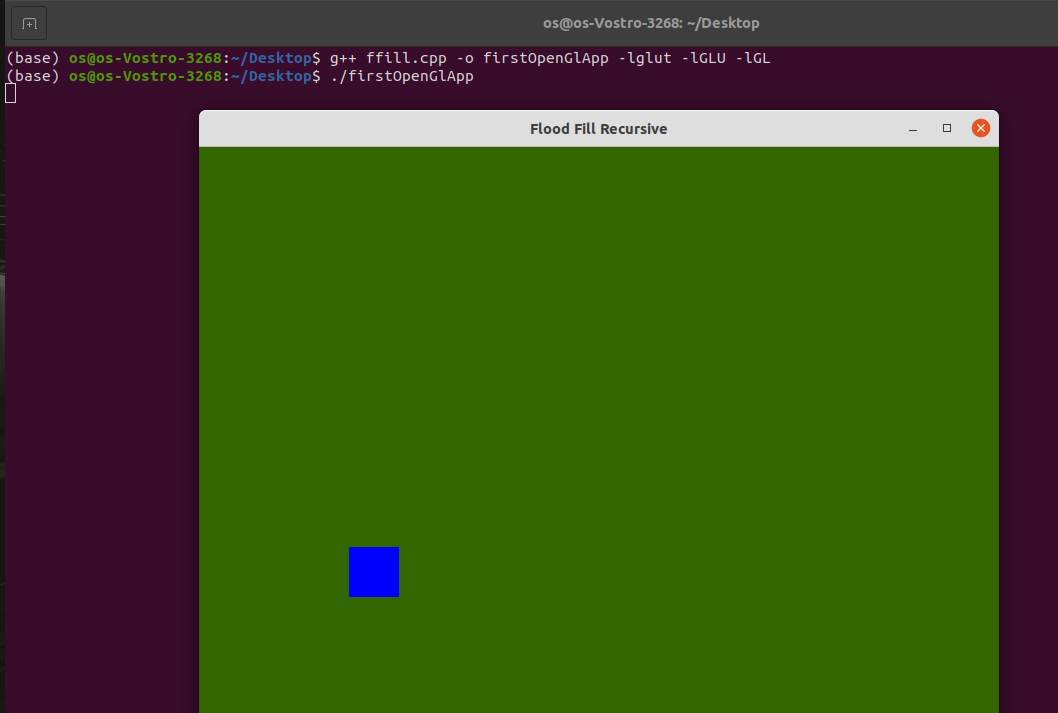
myinit();

glutMouseFunc(mouse);

glutMainLoop();

return 0;

}



SQUARE

#include <GL/glut.h>

void init() {

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);

glLoadIdentity();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINE\_LOOP);

glVertex3f(0.25,0.25,0.75);

glVertex3f(0.75,0.25,0.25);

glVertex3f(0.75,0.75,0.25);

glVertex3f(0.25,0.75,0.75);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(0, 600);

glutCreateWindow("DDA Line");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

