Assignment No.1

# Name: Omkar Manohar Hepat

# Class: CS-B

# Roll No: 31 PRN No: 12211509

# Subject: CGAVR

## Q1. Draw a line using DDA Algorithm.

#include <Windows.h>

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <iostream>

#include <math.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

float x0 = 50, y0 = 50, xn = 3000, yn = 4000;

float dy = (yn - y0), dx = (xn - x0);

float m = (yn - y0) / (xn - x0);

float x = x0, y = y0;

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

glColor3f(255.0, 0, 0);

glBegin(GL\_POINTS);

if (m < 1.0) {

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

glVertex2f(x, y);

x = x + 1;

y = y + m;

}

}

else {

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

glVertex2f(x, y);

y = y + 1;

x = x + (1 / m);

}

}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(800, 600);

glutCreateWindow("Assignment1\_31");

glViewport(0, 0, 800, 600);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 800, 0, 600);

glutDisplayFunc(display);

glutMainLoop();

return 0;

Output:A screen shot of a computer

Description automatically generated

## Q2. Draw a line using Bresenham’s Algorithm

#include <Windows.h>

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <iostream>

#include <math.h>

void drawLineBresenham(float x0, float y0, float xn, float yn) {

float dx = abs(xn - x0);

float dy = abs(yn - y0);

float m = dy / dx;

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

glColor3f(0, 155, 1.0);

glBegin(GL\_POINTS);

float x = x0, y = y0;

if (m < 1.0) {

float p = 2 \* dy - dx;

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

glVertex2f(x, y);

if (p >= 0) {

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

y += 1;

}

else {

p += 2 \* dy;

}

x += 1;

}

}

else {

float p = 2 \* dx - dy;

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

glVertex2f(x, y);

if (p >= 0) {

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

x += 1;

}

else {

p += 2 \* dx;

}

y += 1;

}

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawLineBresenham(50, 50, 300, 400);

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(800, 600);

glutCreateWindow("Assignment1\_31\_BresenhamLine");

glViewport(0, 0, 800, 600);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 800, 0, 600);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output:** **A screen shot of a computer

Description automatically generated**

**Q3. Draw a circle using Midpoint Algorithm**

#include <GL/glut.h>

#include <cmath>

int centerX = 250, centerY = 250, radius = 100;

void drawPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

void drawMidpointCircle() {

int x = radius, y = 0;

int p = 1 - radius;

while (x >= y) {

drawPixel(centerX + x, centerY + y);

drawPixel(centerX - x, centerY + y);

drawPixel(centerX + x, centerY - y);

drawPixel(centerX - x, centerY - y);

drawPixel(centerX + y, centerY + x);

drawPixel(centerX - y, centerY + x);

drawPixel(centerX + y, centerY - x);

drawPixel(centerX - y, centerY - x);

y++;

if (p <= 0)

p = p + 2 \* y + 1;

else {

x--;

p = p + 2 \* y - 2 \* x + 1;

}

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(255, 255, 0);

drawMidpointCircle();

glFlush();

}

void init() {

glClearColor(0.0, 255, 255,50);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 500, 0, 500);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Midpoint Circle Algorithm");

init();

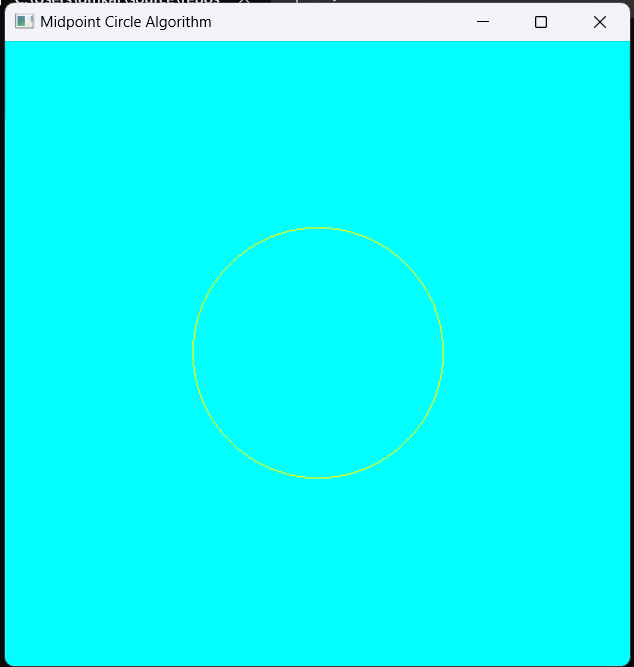
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Output:



## **Q4. Draw a Circle using Bresenham’s Algorithms.**

#include <GL/glut.h>

#include <cmath>

int centerX = 250, centerY = 250, radius = 100;

void drawPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

void drawCircle() {

int x = 0, y = radius;

int d = 3 - 2 \* radius;

while (x <= y) {

drawPixel(centerX + x, centerY + y);

drawPixel(centerX - x, centerY + y);

drawPixel(centerX + x, centerY - y);

drawPixel(centerX - x, centerY - y);

drawPixel(centerX + y, centerY + x);

drawPixel(centerX - y, centerY + x);

drawPixel(centerX + y, centerY - x);

drawPixel(centerX - y, centerY - x);

if (d < 0) {

d = d + 4 \* x + 6;

}

else {

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

y--;

}

x++;

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

drawCircle();

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 500, 0, 500);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Bresenham's Circle Algorithm");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Output:

A screenshot of a computer

Description automatically generated