**PRATHAM KANDARI**

**500097663**

**Batch -7**

**Lab Experiment -6**

Q:1. WAP to draw Concentric circle.

Ans:

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

int radius1,radius2;

// Center of the cicle = (320, 240)

int xc = 320, yc = 240;

// Plot eight points using circle's symmetrical property

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-y, yc-x);

glVertex2i(xc-x, yc+y);

glVertex2i(xc-y, yc+x);

glEnd();

}

// Function to draw a circle using bresenham's

// circle drawing algorithm

void bresenham\_circle(int r)

{

int x=0,y=r;

float pk=(5.0/4.0)-r;

/\* Plot the points \*/

/\* Plot the first point \*/

plot\_point(x,y);

int k;

/\* Find all vertices till x=y \*/

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();

}

// Function to draw two concentric circles

void concentric\_circles(void)

{

/\* Clears buffers to preset values \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

bresenham\_circle(radius1);

bresenham\_circle(radius2);

}

void Init()

{

/\* Set clear color to white \*/

glClearColor(1.0,1.0,1.0,0);

/\* Set fill color to black \*/

glColor3f(0.0,0.0,0.0);

/\* glViewport(0 , 0 , 640 , 480); \*/

/\* glMatrixMode(GL\_PROJECTION); \*/

/\* glLoadIdentity(); \*/

gluOrtho2D(0 , 640 , 0 , 480);

}

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

{ printf("enter radius 1 : ");

radius1=50;

printf("enter radius 2: ");

radius2=100;

/\* Initialise GLUT library \*/

glutInit(&argc,argv);

/\* Set the initial display mode \*/

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

/\* Set the initial window position and size \*/

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

/\* Create the window with title "DDA\_Line" \*/

glutCreateWindow("bresenham\_circle");

/\* Initialize drawing colors \*/

Init();

/\* Call the displaying function \*/

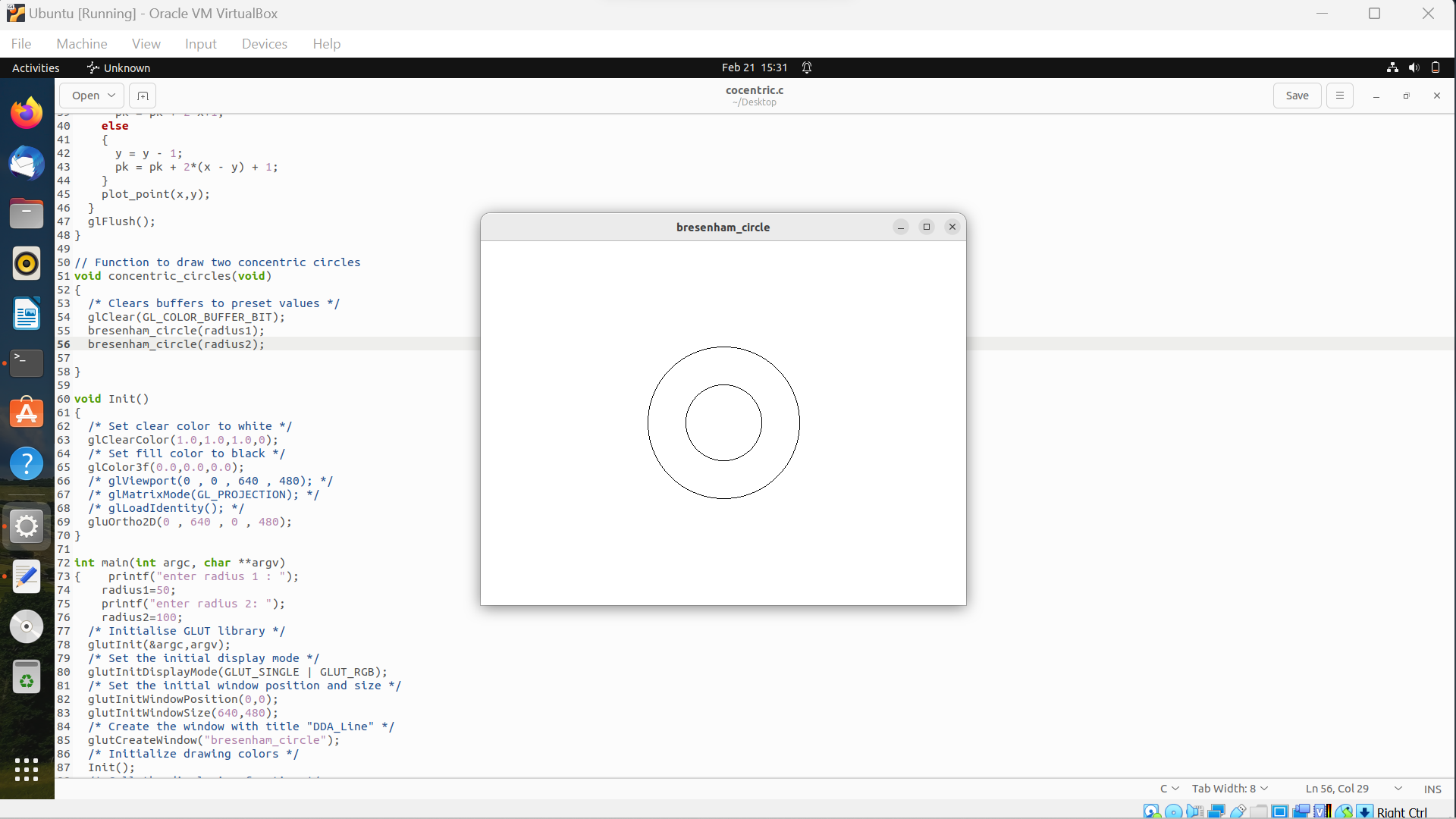
glutDisplayFunc(concentric\_circles);

/\* Keep displaying untill the program is closed \*/

glutMainLoop();

return 0;

}



Q:2. WAP to draw line using Bressenhems Algo and DDA in a single window where user takes the input.

Ans:

#include <GL/glut.h>

#include <windows.h>

#include <stdio.h>

GLint x0,y0,xEnd,yEnd;

float x1,x2,y1,y2;

void init()

{

glClearColor(1.0,1.0,1.0,0.0);

glColor3f(1.0f,0.0f,0.0f);

glPointSize(1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-50.0,50.0,-50.0,50.0);

}

void setPixel(GLint xcoordinate, GLint ycoordinate)

{

glBegin(GL\_POINTS);

glVertex2i(xcoordinate,ycoordinate);

glEnd();

glFlush();

}

void lineBA(GLint x0,GLint y0,GLint xEnd,GLint yEnd)

{

GLint dm = xEnd-x0;

GLint dn = yEnd-y0;

GLint steps,k;

steps=dm;

GLint m,n,p0=(2\*dn)-dm;

setPixel(x0,y0);

m=x0;

n=y0;

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

{

if(p0<0)

{

p0=p0+(2\*dn);

m+=1;

}

else

{

p0=p0+(2\*dn)-(2\*dm);

m+=1;

n+=1;

}

setPixel(m,n);

}

}

void DDA(void)

{{

float dy,dx,step,x,y,k,Xin,Yin;

dx=x2-x1;

dy=y2-y1;

if(abs(dx)> abs(dy))

{

step = abs(dx);

}

else

step = abs(dy);

Xin = dx/step;

Yin = dy/step;

x= x1;

y=y1;

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

for (k=1 ;k<=step;k++)

{

x= x + Xin;

y= y + Yin;

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

glFlush();

}

}

void readInput()

{

printf("\nEnter for Bresenham,s \n x0, y0, xEnd, yEnd(resp): ");

scanf("%i %i %i %i",&x0,&y0,&xEnd,&yEnd);

}

void Display(void)

{

lineBA(x0,y0,xEnd,yEnd);

DDA();

}

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

{

printf("Enter for DDA \n");

printf("Enter the value of x1 : ");

scanf("%f",&x1);

printf("Enter the value of y1 : ");

scanf("%f",&y1);

printf("Enter the value of x2 : ");

scanf("%f",&x2);

printf("Enter the value of y2 : ");

scanf("%f",&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(600,600);

glutInitWindowPosition(0,0);

glutCreateWindow("Breshnam's 2 Line Drawing Algorithm");

readInput();

glutDisplayFunc(Display);

init();

glutMainLoop();

return 0;

}

