**CSE 4200 Lab 2 – Spencer Wallace**

**Part 1) Modifying window settings**

**Graphical user interface

Description automatically generated with low confidence**

- This picture/window uses a display window that is 1280x720

- The window is set to be displayed at 100x0.

- The world settings are changed to 1280x200.

- The viewport uses (150, 200, 250, 100)

**Code**

#define SCREENWIDTH 1280.0

#define SCREENHEIGHT 720.0

glutInitWindowSize(SCREENWIDTH, SCREENHEIGHT); //set window size on screen

glutInitWindowPosition( 100, 0 ); //set window position on screen

glViewport(150, 200, 250, 100);

gluOrtho2D( 0.0, SCREENWIDTH, 0.0, 200 );

**Part 2) Bresenham**

A picture containing graphical user interface

Description automatically generated

The line is drawn from 0, 0 to 200, 100 with a point size of 1 pixel. The circle is drawn with an origin in the center of the window, and a radius of 20.

**Code**

#include <GL/glut.h>

using namespace std;

/\*

Try the Bresenham's Line and Circle algorithms discussed in class.

Use the line algorithm to draw a line with end points (0, 0 ) and ( 200, 100 ).

Use the circle algorithm to draw a circle with radius 20.

\*/

void drawPoint(float x, float y, float size)

{

glPointSize( size );

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

//x1 and y1 are coordinates of start point, x2 and y2 are coordinates of end point

void bLine(float x1, float y1, float x2, float y2)

{

float dx = x2 - x1;

float dy = y2 - y1;

float pk = 2\*dy - dx;

float x = x1; float y = y1;

for(int k = 0; k < (int)(dx-1); ++k)

{

if(pk < 0)

pk = pk + 2\*dy;

else

{

pk = pk + 2\*dy - 2\*dx;

++y;

}

++x;

drawPoint(x, y, 1.0);

}

}

//r is radius, x is x-coordinate of origin, y is y-coordinate of origin

void bCircle(float r, float x, float y)

{

float yCircle = r;

float xCircle = 0;

float d = (3.0/2.0) - r;

while(xCircle<=yCircle)

{

drawPoint(x+xCircle,y+yCircle, 2.0);

drawPoint(x+yCircle,y+xCircle, 2.0); //find other points by symmetry

drawPoint(x-xCircle,y+yCircle, 2.0);

drawPoint(x+yCircle,y-xCircle, 2.0);

drawPoint(x-xCircle,y-yCircle, 2.0);

drawPoint(x-yCircle,y-xCircle, 2.0);

drawPoint(x+xCircle,y-yCircle, 2.0);

drawPoint(x-yCircle,y+xCircle, 2.0);

if (d<0)

d += (2\*xCircle)+3;

else

{

d += (2\*(xCircle-yCircle))+5;

yCircle -= 1;

}

xCircle++;

}

}

//initialization

void init( void )

{

glClearColor( 1.0, 1.0, 1.0, 0.0 ); //get white background color

// glColor3f( 0.0f, 1.0f, 0.0f ); //set drawing color

glPointSize( 1.0 ); //specifies dot size

glMatrixMode( GL\_PROJECTION );

glLoadIdentity(); //replace current matrix with identity matrix

gluOrtho2D( 0.0, 500.0, 0.0, 500.0 );

// gluOrtho2D( 0.0, 5000.0, 0.0, 5000.0 );

}

void display( void )

{

// glViewport( 100, 100, 300, 300);

glClear( GL\_COLOR\_BUFFER\_BIT ); //clear screen

glColor3f ( 0.0, 0.0, 0.0 );

bLine(0,0, 200, 100);

bCircle(20, 250, 250);

glFlush(); //send all output to screen

}

**Summary**

All parts completed successfully. Because all parts are, to my knowledge, fully and correctly completed - I am giving myself 20/20.