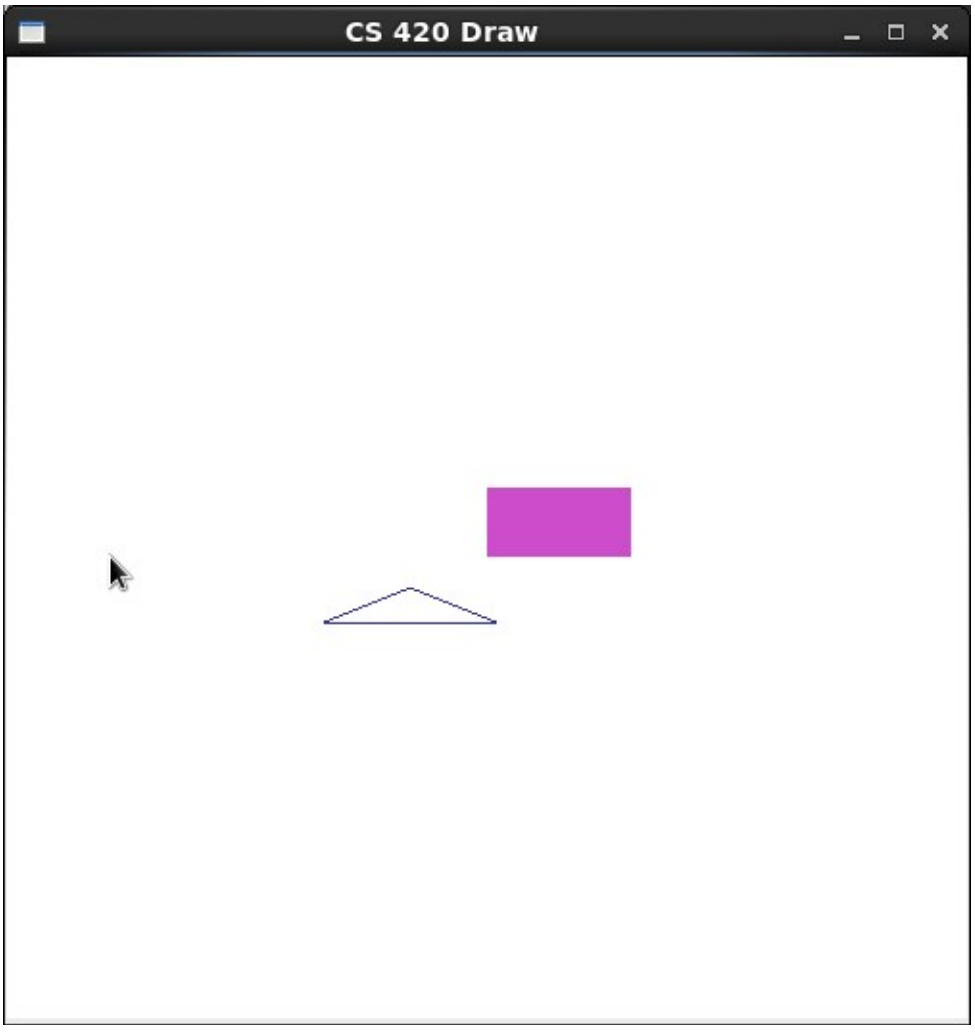
CSE420
Samuel Marrujo
Professor Yu
Lab 02

Draw02

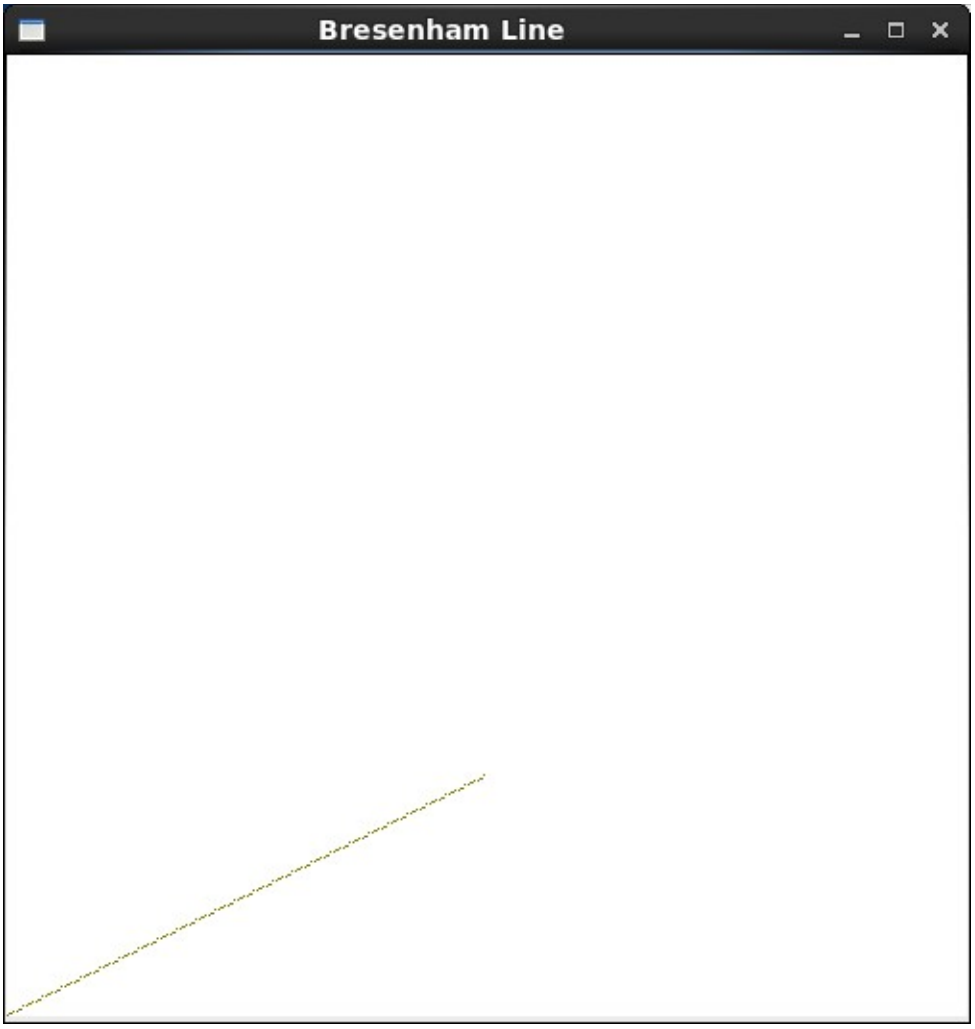
In this part of the lab, it was a modification of the first lab drawing of a rectangle and a triangle. Using the functions of glutInitWindowSize();, glutInitWindowPosition();, gluOrth2D();, and glViewport();, it was possible to modify the orientation and display window of the image. After changing some of the functions, the lab asked to restore the program to the original draw. I believe I was able to accomplish this task successfully, since there were no errors and the display window was consistently changing. Here are my results for this program:



```
//draw.cpp : demo program for drawing 3 dots, two lines, ploylines, rectangles
#include <GL/glut.h>

//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( 8.0 );                    //a dot is 4x4
  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();                      //replace current matrix with identity matrix
  gluOrtho2D( 0.0, 500.0, 0.0, 500.0 );
}

void display( void )
{
  glViewport( 150, 200, 250, 100);
  glClear( GL_COLOR_BUFFER_BIT );    //clear screen


  glColor3f ( 0.2, 0.2, 0.6 );
  glBegin( GL_LINE_STRIP );
    glVertex2i( 30, 30 );
    glVertex2i( 120, 120 );
    glVertex2i( 210, 30 );
    glVertex2i( 30, 30 );
  glEnd();

  glColor3f( 0.8, 0.3, 0.8 );            //bright grey
  glRecti( 200, 200, 350, 380 );
  glFlush();                             //send all output to screen
}
```

Bresenham

Lastly, in the bresenham programs, we are to use the line algorithms discuessed in class to form lines and circles. Due to the screen-shots given and the code supplied, I was able to successfully complete the Lab. So, the outcome of the programs led to the following screen shots in the program:

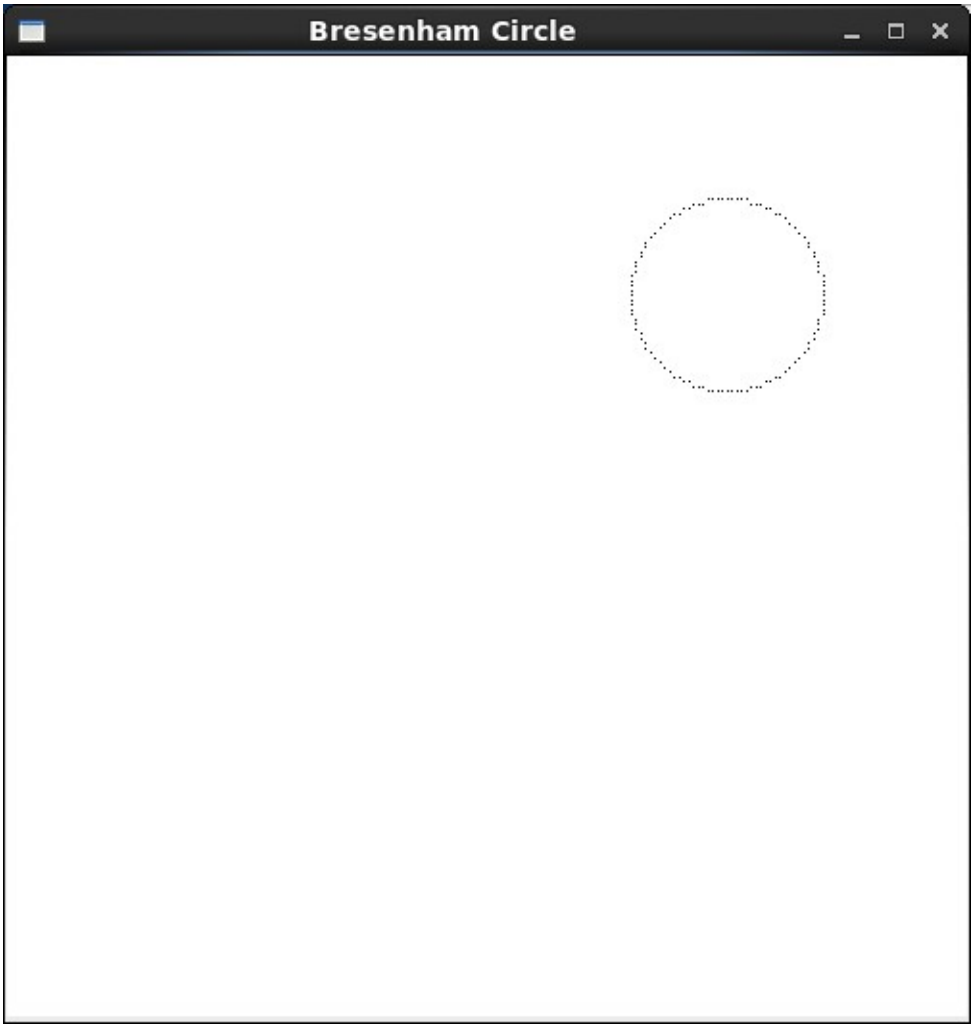


Bresenham Line Code:

```
//bline.cpp : Bresenham Line algorithm, works only for |m| < 1

#include <GL/glut.h>
#include <stdio.h>
#include <math.h>

void init(void)
{
  glClearColor(1.0,1.0,1.0,0.0);
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0.0,400.0,0.0,400.0);
}

void setPixel(GLint x,GLint y)
{
  glBegin(GL_POINTS);
     glVertex2i(x,y);
  glEnd();
}

void line()
{
  int x0 = 0, y0 = 0, xn = 200, yn = 100, x, y;
  int     dx, dy,          //deltas
          pk,              //decision parameter
          k;               //looping variable

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f( 0.5 , 0.5, 0);
  setPixel(x0, y0);       //plot first point

  // difference between starting and ending points
  dx = xn - x0;
  dy = yn - y0;
  pk = 2 * dy - dx;
  x = x0;          y = y0;

  for ( k = 0; k < dx-1; ++k ) {
    if ( pk < 0 ) {
      pk = pk + 2 * dy;                  //calculate next pk
                                         //next pixel: (x+1, y )
    } else {
                                         //next pixel: (x+1, y+1)
      pk = pk + 2*dy - 2*dx;             //calculate next pk
      ++y;
    }
    ++x;
```

```
    setPixel( x, y );
  }

  glFlush();
}

int main(int argc,char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Bresenham Line");
    init();
    glutDisplayFunc( line );
    glutMainLoop();
    return 0;
}
```

Here is the Bresenham circle with Radius 20:



Bresenham Circle Code:

```
//bcircle.cpp : Bresenham Circle algorithm

#include <GL/glut.h>
#include <stdio.h>
#include <math.h>

void init(void)
{
  glClearColor(1.0,1.0,1.0,0.0);
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0.0,200.0,0.0,200.0);
}

void setPixel(GLint x,GLint y)
{
  glBegin(GL_POINTS);
     glVertex2i(x,y);
  glEnd();
}

void Circle(){

  int xCenter=150,yCenter=150,r=20;
  int x=0,y=r;
  int d = 3/2 - r;                                  // = 1 - r
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f( 0 ,0, 0);
  while(x<=y){
    setPixel(xCenter+x,yCenter+y);
    setPixel(xCenter+y,yCenter+x);                  //find other points by symmetry
    setPixel(xCenter-x,yCenter+y);
    setPixel(xCenter+y,yCenter-x);
    setPixel(xCenter-x,yCenter-y);
    setPixel(xCenter-y,yCenter-x);
    setPixel(xCenter+x,yCenter-y);
    setPixel(xCenter-y,yCenter+x);

    if (d<0)
          d += (2*x)+3;
    else {
          d += (2*(x-y))+5;
          y -= 1;
    }
    x++;
  }

  glFlush();
```

```
}

int main(int argc,char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Bresenham Circle");
    init();
    glutDisplayFunc(Circle);
    glutMainLoop();
    return 0;
}
```