Daniel Meyer

CSE 420

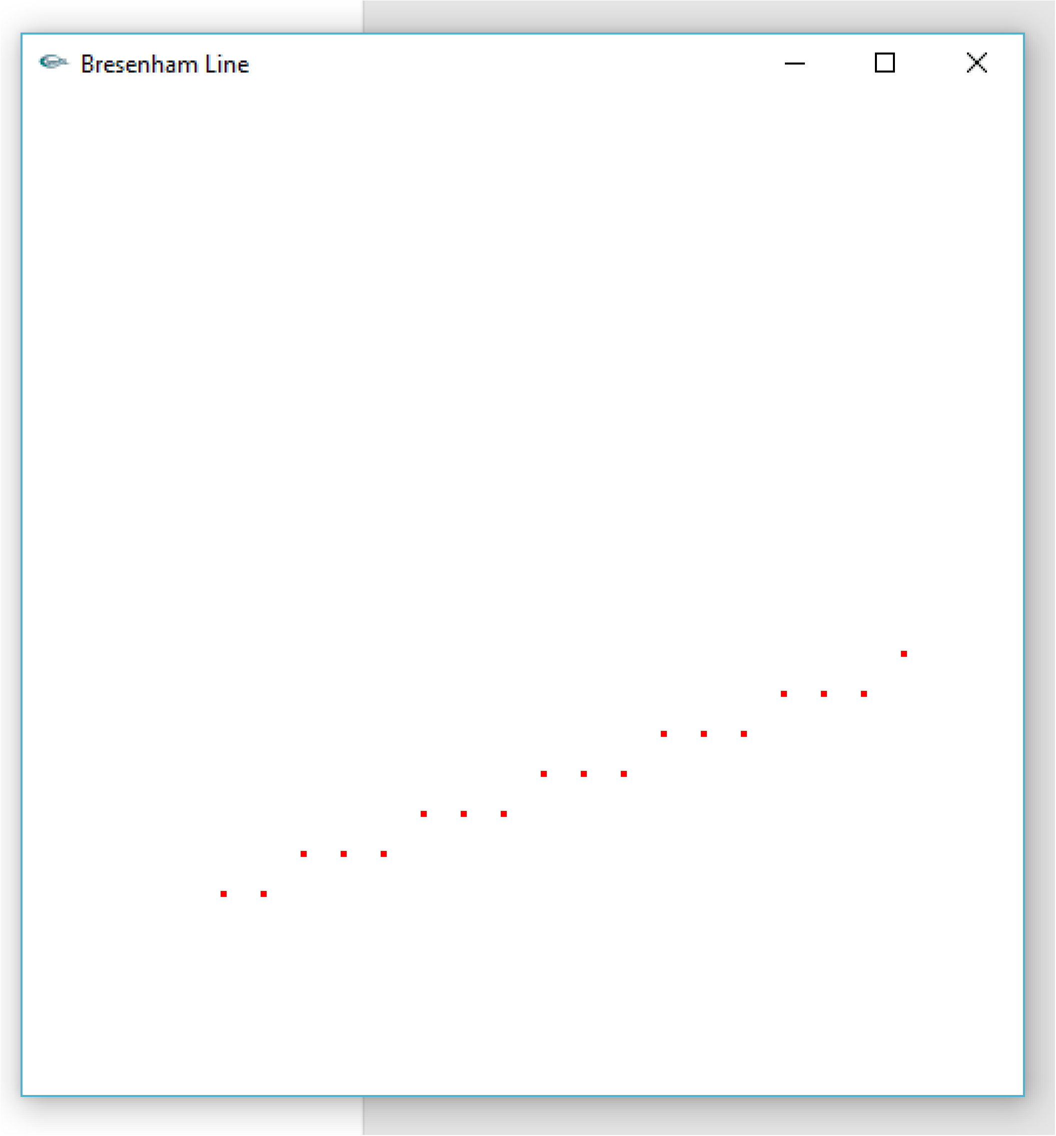
Fall 2018

Homework 1

Part 1: (Success)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <GL/glut.h>

void init(void)
{
     glClearColor(1.0, 1.0, 1.0, 0.0);
     glMatrixMode(GL_PROJECTION);
     gluOrtho2D(-5.0, 20.0, -5.0, 20.0);
     glPointSize(3.0);
}

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

void line()
{
     int x0 = 0, y0 = 0, xn = 18, yn = 6, x, y; //(0,0) to (18,6)
     int  dx, dy,          //deltas
          pk,         //decision parameter
          k;          //looping variable

     glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(1, 0, 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();
}

void myInit()
{
    glColor3f(0.0, 0.0, 1.0);
    glLineWidth(3.0);
}

int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(250, 250);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Bresenham Line");
    init();
    glutDisplayFunc(line);
    glutMainLoop();
    return 0;
}
```



1st 4 values of y:

x0 = 0, y0 = 0, xn = 18, yn = 6

dx = 18 – 0 = 18, dy = 6 – 0 = 6

Initial point = (x0, y0) = (0, 0)

p0 = 2dy – dx = 12 – 18 = -6 -> p0 < 0 so (x+1, y) -> (1,0)

p1 = p0 + 2dy = -6 + 12 = 6

p1 = 6 -> p1 > 0 so (x+1, y+1) -> (2,1)

p2 = p1 + 2dy – 2dx = 6 + 12 – 36 = -18

p2 = -18 -> p2 < 0 so (x+1, y) -> (3,1)

p3 = p2 + 2dy = -18 + 12 = -6

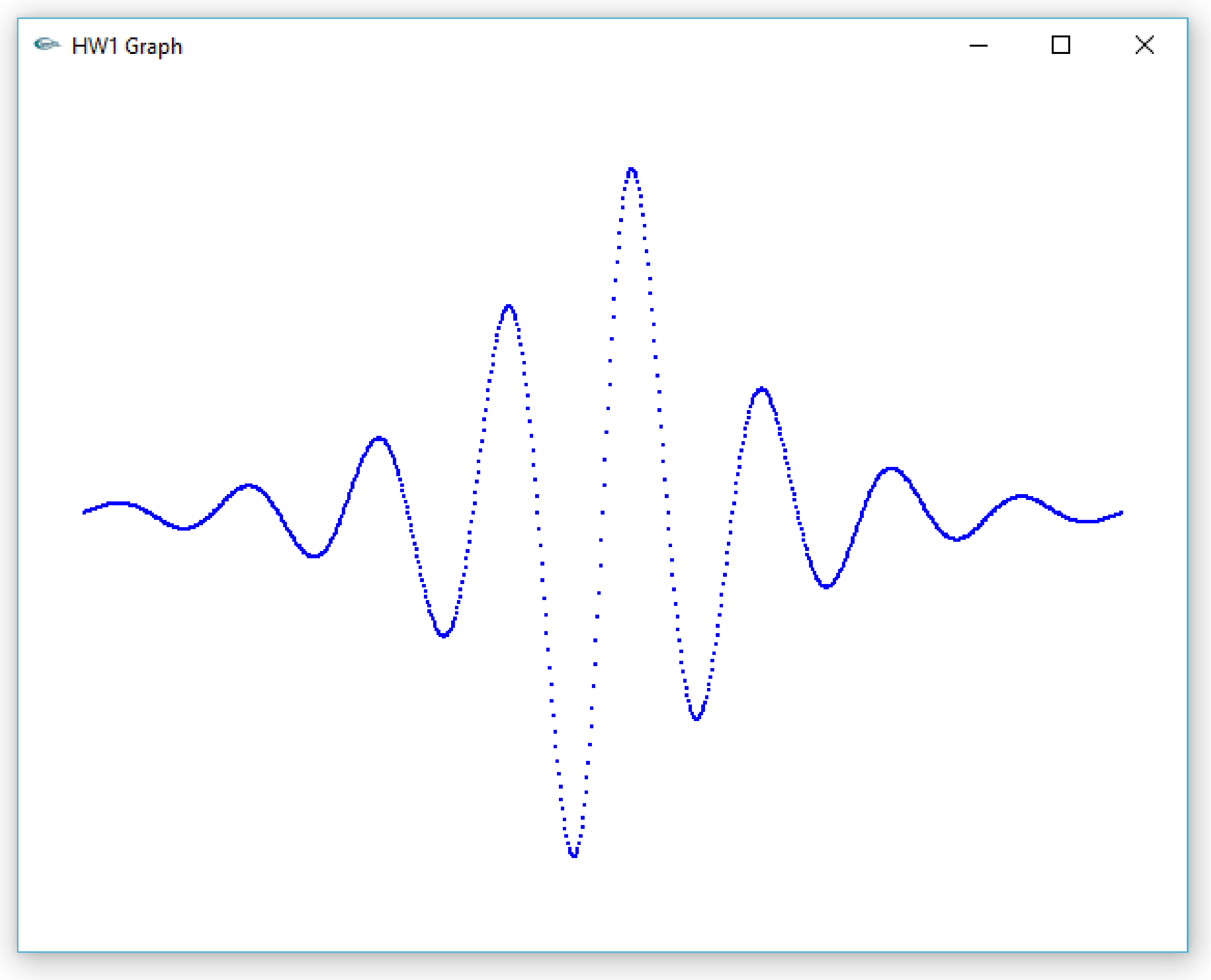
p3 = -6 -> p3 < 0 so (x+1, y) -> (4,1)

p4 = p3 + 2dy = -6 + 12 = 6

……

Part 2: (Success)

```
#include <Windows.h>
#include <iostream>
#include <math.h>
#include <GL/GL.h>
#include <GL/GLU.h>
#include <GL/GLUT.h>

const float pi = 3.14159265358979;
const float e = 2.7818;

void setWindow(GLdouble left, GLdouble right, GLdouble bottom,
GLdouble top)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(left, right, bottom, top);
}

void setViewport(GLint left, GLint right, GLint bottom, GLint top)
{
	glViewport(left, bottom, right - left, top - bottom);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();

	//glBegin(GL_LINE_STRIP);
	glBegin(GL_POINTS);

	for (float x = -4.0; x < 4.0; x += 0.01)
	{
		glVertex2f(x, (pow(e, -abs(x)) * sin(2 * pi * x)));
	}

	glEnd();
	glFlush();
}

void myInit(void)
{
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glColor3f(0.0f, 0.0f, 1.0f);
	glLineWidth(2.0);
	glPointSize(2.0);
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640, 480);
	glutInitWindowPosition(100, 150);
	glutCreateWindow("HW1 Graph");
	glutDisplayFunc(myDisplay);
	myInit();
	setWindow(-4.5, 4.5, -1.0, 1.0);
	setViewport(0, 640, 0, 480);
	glutMainLoop();
}
```



Part 3: (Success)

```
#include <Windows.h>
#include <iostream>
#include <math.h>
#include <GL/GL.h>
#include <GL/GLU.h>
#include <GL/GLUT.h>

class GLintPoint
{
public:
	GLint x, y;
};

class Point2
{
public:
	float x, y;
	void set(float dx, float dy) { x = dx; y = dy; }
	void set(Point2 &p) { x = p.x; y = p.y; }
	Point2(float xx, float yy) { x = xx; y = yy; }
	Point2() { x = y = 0; }
};

Point2 currPos;
Point2 CP;

const float pi = 3.14159265358979;

void moveTo(Point2 p)
{
	CP.set(p);
}

void moveTo(float x, float y)
{
	CP.set(x, y);
}

void lineTo(Point2 p)
{
	glBegin(GL_LINES);

	glVertex2f(CP.x, CP.y);
	glVertex2f(p.x, p.y);
```

```
    glEnd();
    glFlush();
    CP.set(p);
}

void lineTo(float x, float y)
{
    glBegin(GL_LINES);

    glVertex2f(CP.x, CP.y);
    glVertex2f(x, y);

    glEnd();
    glFlush();
    CP.set(x, y);
}

void myInit(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
}

void setWindow(float left, float right, float bottom, float top)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D((GLdouble)left, (GLdouble)right, (GLdouble)bottom,
(GLdouble)top);
}

void setViewport(int left, int right, int bottom, int top)
{
    glViewport(left, bottom, right - left, top - bottom);
}

//draw an n-sided regular polygon
void draw_polygon(int N, float cx, float cy, float radius, float
rotAngle)
{
    if (N < 3) return;                    //bad number of sides

    double angle = rotAngle * pi / 180;   //initial angle
    double theta = 2 * pi / N;       //angle increment
    moveTo(radius * cos(angle) + cx, radius * sin(angle) + cy);

    for (int k = 0; k < N; k++) //repeat n times
    {
        angle += theta;
        lineTo(radius * cos(angle) + cx, radius * sin(angle) + cy);
    }
} //draw_polygon

void rosette(int N, float radius)
{
    Point2 *pointlist = new Point2[N];
    GLfloat theta = (2.0f * pi) / N;

    for (int c = 0; c < N; c++)
    {
        pointlist[c].set(radius * sin(theta * c), radius * cos(theta
* c));
    }

    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            moveTo(pointlist[i]);
            lineTo(pointlist[j]);
        }
    }
}

void draw_circle(float cx, float cy, float radius)
{
    glColor3f(1.0, 0.0, 0.0);

    const int numVerts = 100;
    draw_polygon(numVerts, cx, cy, radius, 0);
    glPointSize(3);

    glFlush();
}

void draw_arc(float cx, float cy, float radius, float sAngle, float
sweep)
{
    glColor3f(0.0, 1.0, 0.0);
    const int n = 30;
    float angle = sAngle * pi / 180;
    float theta = sweep * pi / (180 * n);

    moveTo(cx + radius * cos(angle), cy + radius * sin(angle));

    for (int i = 1; i < n; i++)
    {
        lineTo(cx + radius * cos(angle), cy + radius * sin(angle));
        angle += theta;
    }
}

void draw_star(float cx, float cy, float radius, float rotAngle)
{
    float angle = rotAngle;
    moveTo(cx + radius * cos(angle), cy + radius * sin(angle));

    for (int i = 0; i <= 5; ++i)
    {
        lineTo(cx + radius * cos(0.017453393 * angle), cy + radius * 
sin(0.017453393 * angle));
        angle += 144;
    }
}

void render()
{
    glClear(GL_COLOR_BUFFER_BIT);
    setWindow(-12.0, 12.0, -12.0, 12.0);
    setViewport(0, 500, 0, 500);
    draw_star(8.0, -2.0, 3.0, 55.0);
    draw_polygon(5, 7.0, 5.0, 3.0, 18.0);
    rosette(25, 5.0);
    glFlush();
}

void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glutInitWindowSize(640, 480);
    glutCreateWindow("Turtle");
```

```
        glutDisplayFunc(render);

        myInit();
        glutMainLoop();
}
```