Due: Thursday, February 9, 2017, 11:59:59 Midnight

Introduction

Today's lab is designed to give you practice using some of the C++ skills we have discussed in class. It will also help strengthen your problems solving and/or critical thinking skills.

Lab Objectives

- Practice using c++ style input/output (cin/cout)
- Practice using structs
- Practice working with multiple files
- Practice passing pointers to functions
- Practice using io manipulators

Prior to Lab

- Review the notes we have gone over the last several days.
- Review zyBook Chapter 1, sections 3 & 4 (input/output)
- Review zyBook Chapter 9, sections 1 & 2 (input/output)
- Review C++ Structs Chapter 14, section 12 (structs)
- Review C++ File input/output Chapter 9, section 5 (file input/output)

Instructions

You are going to write a program that will determine if a given point on a 2D plane is within the parameters of a pre-defined triangle on the 2D plane. Since we are working with images this semester, think of the 2D plane as an image. You are **NOT** required to create a ppm image in this lab. You are going to write an algorithm to determine if a given point is within the parameters of the triangle.

Background

Suppose we are given a defined geometric plane of size n x m, and three points that define a triangle

T = p1(x,y), p2(x,y), p3(x,y)

within the plane, we can use the **Barycentric Coordinate System** to determine if a given point on the plane,

p(x,y)

is within the boundaries of the defined triangle.

Barycentric coordinate defines 3 scalars **a, b, c** in the following way:

a = ((y2 - y3) * (x - x3) + (x3 - x2) * (y - y3)) / ((y2 - y3) * (x1 - x3) + (x3 - x2) * (y1 - y3))

b = ((y3 - y1) * (x - x3) + (x1 - x3) * (y - y3)) / ((y2 - y3) * (x1 - x3) + (x3 - x2) * (y1 - y3))

c = 1 - a - b

point 'p' lies within T if and only if the following is true:

0 <= a <= 1 and **0 <= b <= 1** and **0 <= c <= 1**

Program Specifications

This program will be a multi-file project. In your lab directory, create a Lab4 folder. You will create three files:

functions.h

1. Define a struct (**point_t**) to represent a point on a plane. Each point represents the x and y position on the plane. These points should be of type double.
2. Define a function prototype called checkPoint - **int checkPoint(point_t *p, point_t *test)**

functions.cpp

1. Implement the checkPoint function. You will implement the Barycentric Coordinate System in this function (see above). This function has two parameters:
   a. An array of points_t that represent the points that define the triangle.
   b. A pointer that represents the point_t being tested to determine if it lies within the parameter of the triangle.
2. The return value will be 1 or 0.
   a. Return 1 if the point is within the boundaries of the triangle
   b. Return 0 if the point is not within the boundaries of the triangle

main.cpp

1. Main will be the programs driver. Your program should ask the user for the dimensions of the plane (width first then height)
2. Read the data entered by the user
3. Now that we have the dimensions of the plane, ask the user to enter the three points needed to define a triangle on the plane.
   Ex. Prompt the user to:
   "Enter point 1 (x then y)" read the values then prompt the user for the second point, etc.
4. You will need to loop through each point in the plane. Remember we are treating this plane as if it were an image and the origin (x = 0, y = 0) will be in the upper left corner.

   For each point:
   a. Call checkPoint passing in the initialized array of point_t's and the point you are testing.
   b. If checkPoint returns 1, this means the point entered is within the defined triangle. You will then need to print the x and y coordinates using the following format:

   X,Y example: 38.0, 18.0

   Following is an example of the output for a plane with a width of 50 and height of 50. The points of the triangle are defined as:
   p1(18.0, 23.0), p2(31.0, 35.0), p3(39.0, 18.0)

** The output below is shown in columns to save space. You do not need to print in columns. To get the decimal points to print, use the io manipulators discussed in class.

Sample output:

```
Enter the dimensions of the plane (width then height)
50 50
Enter point 1 x then y:
18.0 23.0
Enter point 2 x then y:
31.0 35.0
Enter point 3 x then y:
39.0 18.0

HIT POINTS:
39.0,18.0
35.0,19.0
36.0,19.0
37.0,19.0
38.0,19.0
31.0,20.0
32.0,20.0
33.0,20.0
34.0,20.0
35.0,20.0
36.0,20.0
37.0,20.0
38.0,20.0
27.0,21.0
28.0,21.0
29.0,21.0
30.0,21.0
31.0,21.0
32.0,21.0
33.0,21.0
34.0,21.0
35.0,21.0
36.0,21.0
37.0,21.0
23.0,22.0
24.0,22.0
25.0,22.0
26.0,22.0
27.0,22.0
28.0,22.0
29.0,22.0
30.0,22.0
31.0,22.0
32.0,22.0
33.0,22.0
34.0,22.0
35.0,22.0
36.0,22.0
37.0,22.0
18.0,23.0
19.0,23.0
20.0,23.0
21.0,23.0
22.0,23.0
23.0,23.0
24.0,23.0
25.0,23.0
26.0,23.0
27.0,23.0
28.0,23.0
29.0,23.0
30.0,23.0
31.0,23.0
32.0,23.0
33.0,23.0
34.0,23.0
35.0,23.0
36.0,23.0
20.0,24.0
21.0,24.0
22.0,24.0
23.0,24.0
24.0,24.0
25.0,24.0
26.0,24.0
27.0,24.0
28.0,24.0
29.0,24.0
30.0,24.0
31.0,24.0
32.0,24.0
33.0,24.0
34.0,24.0
35.0,24.0
36.0,24.0
21.0,25.0
22.0,25.0
23.0,25.0
24.0,25.0
25.0,25.0
26.0,25.0
27.0,25.0
28.0,25.0
29.0,25.0
30.0,25.0
31.0,25.0
32.0,25.0
33.0,25.0
34.0,25.0
35.0,25.0
22.0,26.0
23.0,26.0
24.0,26.0
25.0,26.0
26.0,26.0
27.0,26.0
28.0,26.0
29.0,26.0
30.0,26.0
31.0,26.0
32.0,26.0
33.0,26.0
34.0,26.0
35.0,26.0
23.0,27.0
24.0,27.0
25.0,27.0
26.0,27.0
27.0,27.0
28.0,27.0
29.0,27.0
30.0,27.0
31.0,27.0
32.0,27.0
33.0,27.0
34.0,27.0
24.0,28.0
25.0,28.0
26.0,28.0
27.0,28.0
28.0,28.0
29.0,28.0
30.0,28.0
31.0,28.0
32.0,28.0
33.0,28.0
34.0,28.0
25.0,29.0
26.0,29.0
27.0,29.0
28.0,29.0
29.0,29.0
30.0,29.0
31.0,29.0
32.0,29.0
33.0,29.0
26.0,30.0
27.0,30.0
28.0,30.0
29.0,30.0
30.0,30.0
31.0,30.0
32.0,30.0
33.0,30.0
27.0,31.0
28.0,31.0
29.0,31.0
30.0,31.0
31.0,31.0
32.0,31.0
28.0,32.0
29.0,32.0
30.0,32.0
31.0,32.0
32.0,32.0
29.0,33.0
30.0,33.0
31.0,33.0
30.0,34.0
31.0,34.0
31.0,35.0
```

TESTING

I have included a sample txt file that can be used to test your program. You can run this using redirection. Ex. ./a.out <input.txt

FORMATTING

1. Your program should be well documented
2. Each file should include a header (example below)
3. Your program should consist of proper and consistent indention Ex. You should choose a specific number of spaces to indent – 3, 4, or 5 – and be consistent
4. No lines of code should be more than 80 characters
5. Variable names should be meaningfull

5 – 10 points will be deducted for each of the above formatting infractions. Below is an example of a program that meets each of the above formatting requirements:

```
/*******************
  your name
  username
  Lab 4
  Lab Section:
  Name of TA
*******************/
```

Submission Instructions

Use handin (http://handin.cs.clemson.edu) to submit a tarred file called Lab4.tar.gz containing main.cpp, functions.cpp, functions.h, and input.txt