Programming Assignment 2: It’s All Greek to Me!

Contents

Assignment Description

Imagine you've built a time machine so you can travel back in time to meet the Greeks (specifically, Pythagoras and Hipparchus) who have helped give us amazing theorems (Pythagorean Theorem) and branches of mathematics (Trigonometry) that directly help us in our game development! Luckily, you're bringing your computer along with you (and, of course, a power source of some kind) so you can show off your programming skills.

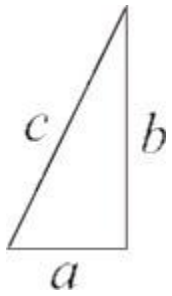
In this assignment, you'll calculate the distance between two points and the angle a game character would need to move in to go from the second point to the first point.

Why do we care?

There are lots of times in game development where we need to know the distance between two points. For example, an NPC might determine which target to go after based on which one is closest to the NPC. What about angle? For the same example, the NPC needs to figure out how to move toward the selected target; having the angle lets us build the appropriate velocity vector for the NPC.

The Pythagorean Theorem

The Pythagorean Theorem tells us how to calculate the length of the hypotenuse of a right triangle. A picture (from Wikipedia) will really help here:



We know from the theorem that $c^2 = a^2 + b^2$

so if we take the square root of $a^2 + b^2$, we'll get c, the length of the hypotenuse.

Skills Practiced

- Receiving input from user
- Printing output to console
- Using different data types and casting
- Using the Math class and referencing C# documentation

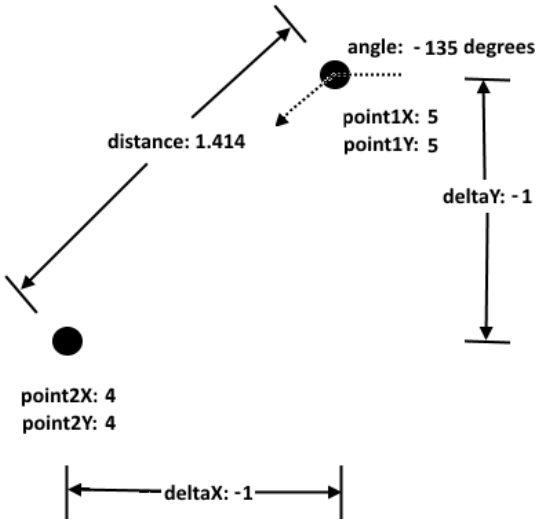
Getting Started

Clone your repository

1. Accept the assignment to create your repository for submitting your work: https://classroom.github.com/a/38TuXAIo
2. In GitHub Desktop, clone the repository to your desktop.
3. Open the project in Visual Studio.

Important: You MUST only add code as indicated by the comments in that file. If you don't, you're virtually guaranteed to fail all the test cases in the automated grader.

The picture below might help you with the following steps:



Requirements

The code I've provided in the **Program.cs** file reads in 4 floats for the x and y coordinates for two points and stores them in variables named **point1X**, **point1Y**, **point2X**, and **point2Y**. Your calculations (described below) must use those variables.

You must update the program to:

1. **Calculate** the delta x and delta y between the two points which are input; the term delta is often used in science and engineering for the difference or change between two values. s The subtraction order matters when you calculate the angle in a later step, so be sure to subtract the first point values from the second point values and not the other way around.
    - **deltaX** and **deltaY** represent the length of side ***a*** and ***b*** of the right triangle
2. **Calculate** the distance (as a float) between the two points using the Pythagorean Theorem. The distance we're calculating is the length of the hypotenuse (**c**) where we calculated **a** (delta x) and **b** (delta y) in the previous step. You might want to explore the documentation for the **Math** class to help with this calculation; you can find that documentation at https://msdn.microsoft.com/en-us/library/system.math(v=vs.110).aspx.

    Remember you can use type casting to convert a `double` to a `float`.

3. **Calculate** the angle (as a float) we'd have to move in to go from point 1 to point 2 (0 degrees is directly to the right). Be careful here; we're not trying to calculate any of the angles inside the triangle. Think of it this way: your character is at the first point, facing directly to the right. What angle would you need to rotate by to be facing the second point?
    - **HUGE HINT:** The **Atan2** method in the **Math** class can really help here, though you'll need to use type casting to store the result in the float for your angle
4. **Print** the distance (from step 2), a single space, and the angle (from step 3)
    - WARNING: Be sure to convert the angle from radians to degrees using Math.PI (and type casting again!) before printing it. Do NOT print anything else in your output; if you do, you'll confuse the automated grader and you'll get a bad grade!

- For example, if the point1 and point2 coordinates are 5, 5, 4, and 4 (as shown in the picture above), your output should be the following on a single line:

```
1.414214 –135
```

- We'd typically label the distance and angle to tell the user what those numbers are, but that will just confuse the automated grader. You must print ONLY the actual distance and angle in your output, with the distance and angle appearing on a single line with a single space between them.

5. In GitHub Desktop, commit your code with message: "Completed through step 4".

Running Your Code

When you run your program the command prompt window will open and it will sit there doing nothing. To make your code run, type in 4 floats with a single space between each one and press the <Enter> key; your code should then run so you can check your output. For example, your input could be:

```
5 5 4 4
```

You can actually run your code again if you want to by typing in 4 floats with a single space between each one and pressing the <Enter> key again. When you're ready to stop running your code, type q (for quit).

Here's what running the code multiple times with different inputs should look like. The first line is the float input (the coordinates for the two points), the second line is your output line, and so on:

```
5 5 4 4
1.414214 – 135
2 2 4 4
2.828427 45
q
```

If your output doesn't match the image above EXACTLY (no extra words, characters, spaces, or blank lines) you'll fail all the test cases in the automated grader.

Submit Your Work

Test your code using the

6. Test Case Inputs below.
7. Once it is working as expected, copy the output from the terminal window.
8. If you made any additional changes to your code, make sure you commit again. Finally push your updates to GitHub.
    - By committing and pushing your updates to GitHub you have submitted your assignment on GitHub Classroom.
9. Return to CodeHS. Paste your output into the code window to complete the assignment.

Test Case Inputs

The autograder uses the following set of test case inputs to test your code (1 test case per line):

```
5 5 4 4
2 2 4 4
0 0 1 0
0 0 0 1
0 0 -1 0
0 0 0 -1
2 2 -4 4
2 2 4 -4
6 5 4 3
3 4 5 6
```

You should figure out what the expected results are for each test case input and make sure your code is generating those expected results in the required format before submitting your code for grading.

The Math Looks Too Hard To Me

You should remember this math from your recent lesson with Mrs. Lemay. The math isn't nearly as bad as it looks at first glance. To calculate the delta x and delta y, all you need to do is subtraction. To calculate the distance between the two points, you need to square the delta x and delta y (multiply them by themselves), add the squares together, and take the square root of the result (use the appropriate method in the **Math** class to do that part). Finally, to calculate the angle between the two points, use the **Atan2** method in the **Math** class; you'll need to read the documentation for that method to call it appropriately, of course. That's all the math you need to do.

Taking Small Bites

Solving this problem is a big step compared to the exercises, but that's why you shouldn't try to chew it up all in one bite. Just do each step above, one at a time, in order, and you'll see that you can do it. Although there are a few things for which you'll need to read documentation or do Google searches, if you tackle the assignment one step at a time you should be fine.

Building a game is the same way, by the way. If we start thinking "How am I going to program this entire game?" it can be pretty intimidating! If we think in the much smaller steps we actually implement to eventually build a game, it becomes a much more achievable task.

Common Problems

If you're having trouble getting this to run, here are a couple things you should check:

1. You have to start with the Visual Studio project I gave you in the assignment materials and add your code in the section indicated. The code I gave you includes the appropriate structure for the automated grader to work. If you make changes outside of where the comments are, you'll almost definitely introduce errors.

2. I give you an example in this assignment (and all assignments) for what output you should get when you run the code multiple times. Be sure to try that example; if your code doesn't generate output exactly as shown in the example, your code isn't working properly
3. Remember that you'll need to use type casting to use the Math class with your **`float`** variables