NearestMark

Documentation

Contents

[Summary 3](#_Toc456949788)

[Running the Application 3](#_Toc456949789)

[NearestMark.exe TestFile.txt 3](#_Toc456949790)

[NearestMark.exe TestFile.txt InputFile.txt 3](#_Toc456949791)

[Developer Content 5](#_Toc456949792)

[NearestMark Projects 5](#_Toc456949793)

[NearestMark.Core 5](#_Toc456949794)

# Summary

NearestMark uses Euclidian distance to determine the distance between an Input Coordinate and a set of one or more “Test” Coordinates. Input Coordinates may be entered manually via the console or sequentially via text files. Test Coordinates are always loaded via text file.

# Running the Application

Start NearestMark.exe at the command-line, setting the first/required **TestFile.txt** argument, and optionally sending in a second **InputFile.txt** argument. TestFile.txt and InputFile.txt files contain Coordinates, and should always include at least one Coordinate, formatted as follows:

(-12,3)(13,4.5)(0,0)

*Note: Coordinate files must be located the* ***same folder*** *as* ***NearestMark.exe****.*

## NearestMark.exe TestFile.txt

This will load every Coordinate in TestFile.txt and wait for the user to enter an Input Coordinate. In Figure 1, we entered an Input Coordinate with 3 points: **12, -45, 0** and then pressed **ENTER**. The program displayed the results of the comparison, and then waited for another Input Coordinate to be entered:

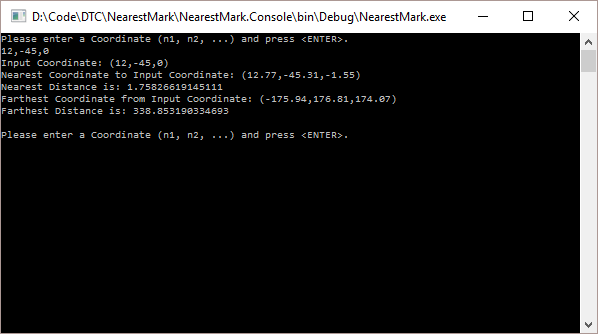


Figure 1Example of NearestMark.exe TestFile.txt

## NearestMark.exe TestFile.txt InputFile.txt

Setting both command-line file arguments will load every Test Coordinate in TestFile.txt and also load Input Coordinate in InputFile.txt. The *program* *will iterate through each Coordinate* in InputFile.txt, displaying the results of each Input Coordinate comparison with the set of Test Coordinates (Figure 2):

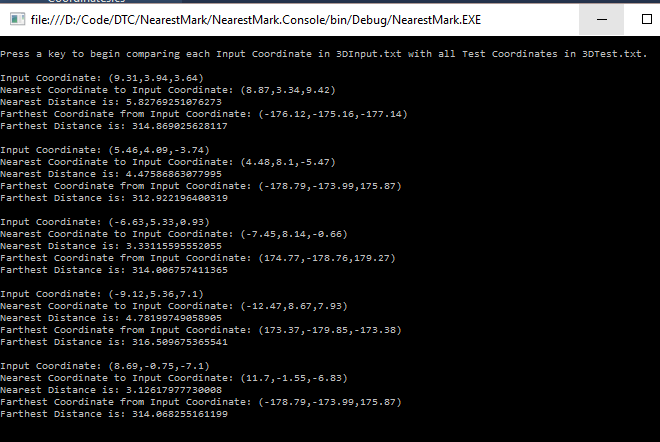


Figure 2: Displaying results of Input and Test Coordinate comparisons.

# Developer Content

NearestMark can be most easily built using Visual Studio 2015. Open NearestMark.sln and rebuild all.

A Test project is included with several unit tests. Sample input and test Coordinate files are also included in the Test project.

## NearestMark Projects

NearestMark.Core contains the string parsing, distance calculation, and distance comparison logic, implemented in a C# library. This library could be use in a variety of implementations, including service/API and mobile (Xamarin).

NearestMark.Console is a .NET/C# console application, and contains the file IO and user interface logic.

NearestMark.Tests is a unit test project.

## NearestMark.Core

NearestMark.Core uses the following algorithm to “efficiently” calculate the distance between two Coordinates (each Coordinate containing one or more Points):

Math.Sqrt(InputCoordinate.Points.Zip(TestCoordinate.Points, (a, b) => (a - b) \* (a - b)).Sum());