Project A4

Due Date: April 14

Purpose

There are several objectives for this project. One is to gain experience adding code to an already existing codebase. Another is to write a program using an algorithm covered in class; that is, figuring out the details of an algorithm that has been described in more general terms. Finally, there are additional problems to work out not contained in the original algorithm.

Problem

Many applications require finding the convex hull of a set of points. These include applications from basic paint programs to pattern recognition, image processing, and statistical problems. We have looked at the problem and have found a solution using a brute-force algorithm that results in an efficiency of only $O(N^3)$. A second algorithm, Quickhull, results in an efficiency of $O(N \lg N)$ and possibly better ($O(Nh)$, where h is the number of points in the convex hull).[1] You will implement the Quickhull algorithm for an arbitrary set of planar points.

Input

The program should first prompt for a filename and then read that file. As in the last project, the file will contain highway data in the "collapsed" TMG format used on the METAL website. The program should then prompt for an output filename.

Output

The program should display the **minimal** set of points that make up the convex hull of the initial given longitude/latitude points. The points making up the convex hull should be printed in **clockwise** order starting from the **westernmost** value, one coordinate at a time. The coordinates should be displayed to 6 decimal places. Next, the program should display the number of iterations that were necessary for the *basic operation* to compute the result. Finally, an output file should be created that contains just the convex hull points in TMG format so that the convex hull can be displayed in METAL HDX.
For example, the output for the file `amsterdam5-area.tmg` would be:

```
The convex hull coordinates are:
42.956234, -74.272814
43.017502, -74.193850
42.967440, -74.113727
42.956542, -74.112954
42.908189, -74.121119
42.905037, -74.224620

Number of iterations: X
File myoutputfile.TMG created
```

[1] This could also be *worse* than $O(N \lg N)$!

where X is the number of iterations and *myoutputfile.TMG* is the filename that was input by the user.

Specifics

- Implement the Quickhull algorithm as described in class and in the text. Other implementations exist (such as Graham's scan), but part of this exercise is for you to code the given algorithm.
- As usual, follow good OOP practices. This time, there is some starting code that you **must** follow. This is on the course web page and is shown below:

```
class convexHull {      // can add to previous class(es) from A2
protected:
   int n;  // total number of points
   /* data structure to store points here */

public:
   convexHull ();       // read in the data points; can call previous A2 method
   void printHull ();   // display convex hull in clockwise order
   void printTMG ();    // create TMG file of convex hull points
};

class quickHull : public convexHull {
   int numOps;          // number of operations
   /* any needed data members here */
   void qHull (/* parameters here */); // Quickhull algorithm
   /* any needed helper method prototypes here */

public:
   quickHull () {numOps = 0;}
   void qHull ();       // Quickhull driver
   void printOps ();    // display number of operations
};

int main () {
   quickHull myHull;    // create Quickhull instance

   myHull.qHull();      // run Quickhull algorithm
   myHull.printHull();  // display the convex hull points
   myHull.printOps();   // display the number of operations
   myHull.printTMG();   // create file for use in METAL
   return 0;
}
```

You must use this code as written, but you can add any additional code as described in the comments. This code should work in conjunction with the code you wrote for A2.

- You should do an error check on the input filename. If the filename does not exist, the program should display an error message and quit gracefully.
- You do not have to do any error checking of the data contained in the file. You may also assume that there are at least three data points in the file.
- Remember to adequately comment your classes, methods, and all variables.

Notes

- Don't wait to start! Problems will probably arise.
- There are many solutions available online. Be aware that these are not always good solutions. Furthermore, is your goal to learn how to program or how to copy someone else's code?
- Reading and writing the data files should be done using your previous code.
- Start with small files to make sure your algorithm works properly. Test your output file in METAL.
- The number of operations should give you an indication if your solution is correct.
- Zip your code together using the same naming convention as before, as in `gousieA4.zip`. **Zip only your code! I do not want either of the `__MACOSX` or `cmake-build-debug` folders!** Figure out how to zip up only your `.cpp` and `.h` files, not the entire CLion (or other) folder! Turn in your zip file in Canvas before the due date.
- A printed version of your source code is due in class on April 15^{th}. This printout should contain text in black on a white background. Write/print and sign the Wheaton Honor Code Pledge on what you turn in: "I have abided by the Wheaton College Honor Code in this work."

C++ and Java, say, are presumably growing faster than plain C,
but I bet C will still be around.
– Dennis Ritchie, creator of C (1941-2011)