Assignment 2: Implementation of two convex hull algorithms with robustness testing [16 points]

Computational Geometry course at DCGI FEE CTU, winter 2017
Petr Felkel, Vojtěch Bubník

This is the second exercise for the Computational Geometry class. Its first goal is to implement two convex hull algorithms: Graham scan and Jarvis march. Its second goal is to test robustness of these algorithms implemented by means of using standard floating point arithmetic and by means of Shewchuk's robust orientation predicates.

1) Implement two convex hull algorithms

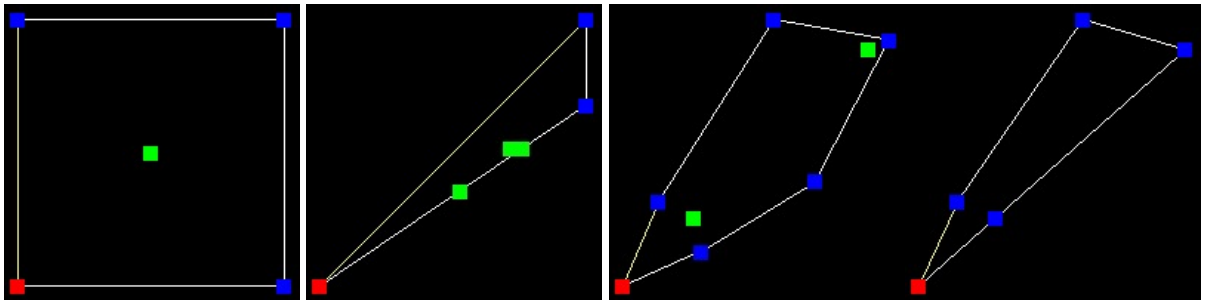
Implement Graham scan [1] (modification by Andrew [2]) and Jarvis march (gift wrapping) algorithms according to the lecture 4. Submit the resulting modified chull.cpp file.

Instead of searching for maximal angle in Jarvis march, use the orientation predicate for finding the rightmost point. To correctly handle degenerate input sets with some input points positioned exactly on the segments of the convex hull, a rightmost point is only accepted if it either forms a right turn towards all other input points (use the `Orient2d` predicate), or it is the furthest of the collinear points (use the new `Extended2dNaive / Extended2dExact` predicates).

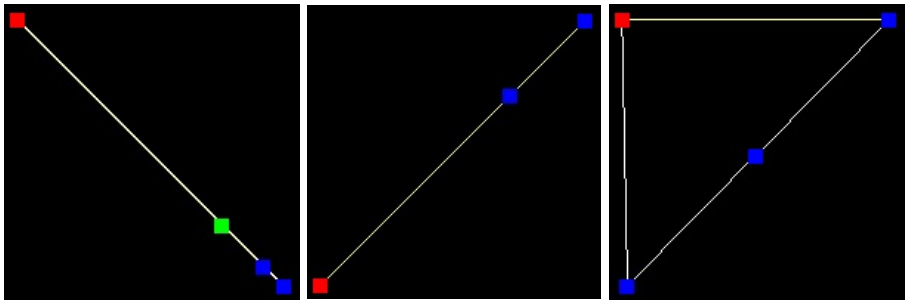
2) Test your implementation on provided sets of points

Run the program on provided sets of points from points1.txt to point7.txt, testing float, double and adaptive double predicates. Correct results are on the following images.

1) 2) 3) 4)



5) 6) 7)



- Point on the convex hull
- Input point not on convex hull
- First point of the convex hull

3) Explain of the wrong results

Remember examples of problematic points for orientation predicate from Assignment 1, resulting in wrong result of the predicate, such as LEFT_TURN answer instead of RIGHT_TURN. Having this experience in mind, explain the type of error for datasets 4 to 7.

4) Search for problematic sets of points

Inspire yourself with the paper Kettner et. Al [3]. The authors classify four different cases when the orientation predicate causes an incremental convex hull algorithm to fail:

- **Failure A1: a point outside sees no edge**
  A point *outside* the current hull, but the orientation predicate answers LEFT_TURN for all edges of the current hull (instead of RIGHT_TURN for one or more edges).
- **Failure A2: a point inside sees an edge**
  A point *inside* the current hull, but the orientation predicate answers RIGHT_TURN for one or more edges of the current hull (instead of LETF_TURN for all edges).
- **Failure B1: a point outside sees all edges**
  A point *outside* the current hull, but the orientation predicate answers RIGHT_TURN for all edges of the current hull (instead of LEFT_TURN for far side edges).
- **Failure B2: a point outside sees a non-contiguous set of edges**
  A point *outside* the current hull, but the orientation predicate answers RIGHT_TURN for a non-contiguous set of edges.

Propose a similar situation of failure for Graham and Jarvis algorithms and find a dataset demonstrating such situation. Results of Assignment one should help here too. As a reference, use the result of the algorithms using Shewchuk's robust predicates [4]. Consult the float converter applet, if you need [5].

Note: Keep in mind, that images in Kettner paper are flipped according to the main diagonal, resulting in exchange of blue and red colors…

5) Explain the new `Extended2dNaive` are `Extended2dExact` predicates

Why are two different predicates provided?

What would happen, if `Extended2dExact` was used with naïve Orient predicate and `Extended2dNaive` with the exact Orient predicate?

Hint: The naïve Orient predicate is not always correct and the `Extended2dExact` / `Extended2dNaive` predicates assume, that the three input points may be collinear.

Compilation & Testing

Download the chull.zip package and unpack it. Then open the chull.vcxproj, compile it and run. The program creates a set of *.tga images in three directories, generated by standard floating point operator (float, double), and for Shewchuk's robust predicates using adaptive precision floating-point arithmetic [4] (adaptive). View them with your favorite image viewer (IrfanView, ACDSee, Picasa, etc…).

The true computation precision depends on the compiler setting. To force the compiler to use 24 bit precision for floats and 53bit precision for double we added `setFPURoundingTo24Bits()` and `setFPURoundingTo53Bits()`.

Submit the document with your explanations.

Links

[1] De Berg, van Kreveld, Overmars, Schwarzkopf. Computational Geometry: Algorithms and Applications. 3rd edition, Springer-Verlag. ISBN 978-3-540-77973-5, Chapter 1

[2] Andrew's monotone chain convex hull algorithm. http://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain.

[3] Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, Chee Yap. Classroom examples of robustness problems in geometric computations, Computational Geometry, Volume 40, Issue 1, May 2008, Pages 61-78, ISSN 0925-7721, http://dx.doi.org/10.1016/j.comgeo.2007.06.003 or http://people.mpi-inf.mpg.de/~mehlhorn/ftp/classroomExamplesNonrobustness.pdf.

[4] Jonathan Richard Shewchuk: Adaptive Precision Floating-Point Arithmetic and Fast Robust Predicates for Computational Geometry, 1967, http://www.cs.cmu.edu/~quake/robust.html.

[5] Harald Schmidt: Float Converter applet, http://www.h-schmidt.net/FloatConverter/IEEE754.html.