Assignment 1: Robustness of the orientation predicate [6 points]

Computational Geometry course at DCGI FEE CTU, winter 2014

This is the first exercise for the Computational Geometry class. Its goal is to get some experience with a non-robust and robust version of geometric predicates. You can find links to additional interesting material about floating point arithmetic here.

IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic

Read the third chapter about numerical precision of the "IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic" [1]. It can be found in the IEEE digital library [2] and it is accessible from within the CTU network. There is also a "light" version which can be found on Wikipedia [3].

For deeper understanding of the representation of floating-point numbers, you may find interesting to play with the Float Converter applet [4], or to read the paper by David Goldberg [5]. For details about floating-point calculation support on various platforms see [7].

Shewchuk predicates

Jonathan Richard Shewchuk implemented fast robust orientation predicates [6] and released his code for public use. The predicates assume the input float/double parameters are exact numbers and use adaptive precision floating-point arithmetic to compute precise results. The adaptive approach for sign of determinant computation ("do only as much work as necessary to guarantee a correct result") and an unusual approach to exact arithmetic (splitting the operand into non-overlapping chunks of bits with increasing precision, pioneered by Douglas Priest) are key reasons why are his algorithms faster than traditional libraries of arbitrary precision numbers.

Compilation & Testing

Download the robustness.zip package for the first exercise and unpack it. Then open the robustness.sln and compile it. The program creates a set of *.tga images generated for different data types by means of different predicates and different values of exponent of delta. View them with your favorite image viewer (IrfanView, ACDSee, Picasa, etc…).

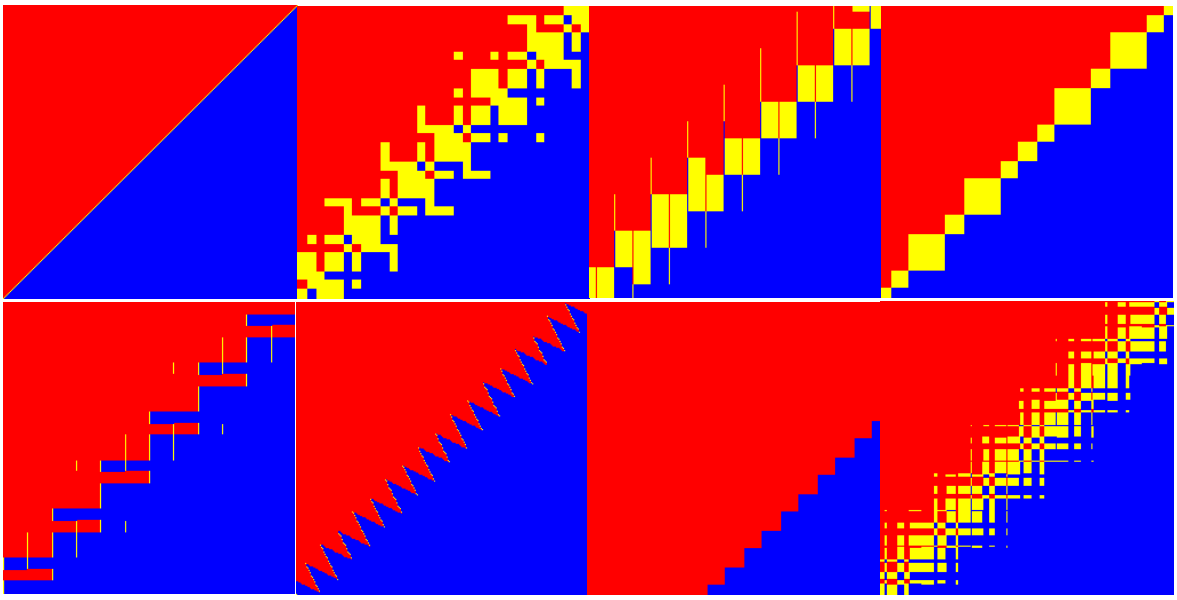
The true computation precision depends on the compiler setting. To force the compiler to use 24 bit precision for floats and 53bit precision for double we added `setFPURoundingTo24Bits()` and `setFPURoundingTo53Bits()`.

The tasks

1. Write two versions of code for naïve floating-point orientation predicate computation:
   a) by means of complete evaluation of the 3D determinant
   $$\text{orientation}\,(a,b,c) = \text{sign}\big(\,a_x b_y \;+\; b_x c_y \;+\; c_x a_y - a_x c_y - \;b_x a_y - c_x b_y\big)$$
   b) by means of 2D determinant after subtraction of a pivot – point c as Shewchuk
   $$\text{orientation}\,(a,b,c) = \text{sign}\Big((a_x - c_x)\big(b_y - c_y\big) - \big(a_y - c_y\big)(b_x - c_x)\Big)$$
2. Compare the results of both naïve implementation 1a) and 1b) tested on IEEE 754-2008 *float* and *double* data types

3. Compare the results of your *float*/*double* versions of naïve implementation from task 1 with exact and adaptive methods using Shewchuk predicates already implemented in the code. Discuss the differences.
4. Explain, why increment `delta` with exponent decreasing below -53 returns larger and larger yellow regions even for robust Shewchuk predicates for `pointsets 0, 2,` and `3`.
5. Explain, why for `pointset 1` the robust predicates return less and less or even no yellow points.
6. Order the naïve, exact and adaptive predicates by expected execution time.
7. [bonus task] Discuss which of the computation approaches 1a) or 1b) is more suitable for exact and for adaptive version of orientation predicate computations
8. [bonus task] Try your compiler and comment the calls of `setFPURoundingTo24Bits()` and `setFPURoundingTo53Bits()`.

Some examples of generated images



Links

[1] 754-2008 - IEEE Standard for Floating-Point Arithmetic, DOI: 10.1109/IEEESTD.2008.4610935

[2] http://ieeexplore.ieee.org/

[3] IEEE floating point. (2014, October 1). In *Wikipedia, The Free Encyclopedia*. Retrieved 11:17, October 2, 2014, http://en.wikipedia.org/wiki/IEEE_754-2008, or in Czech http://cs.wikipedia.org/wiki/IEEE_754

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

[5] David Goldberg: What Every Computer Scientist Should Know About Floating-Point Arithmetic, Computing Surveys, March, 1991, https://www.fer.unizg.hr/_download/repository/paper%5B1%5D.pdf

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

[7] Deterministic cross-platform floating point arithmetics. http://christian-seiler.de/projekte/fpmath/