Computational Geometry

Exercises (20 hours)

General requirements and recommendations

1. Each task must be performed as a completed and operable SW application that uses 2D graphic tools for demonstrating the computational results.
2. Any operating system and any programming language may be used, but for better efficiency C++ is highly recommended. Qt can be used as a UI and graphical platform.
3. User interface must provide entering algorithm parameters manually or by means of graphical interaction (for example, a rectangle window may be specified using the mouse).
4. The dimension of the problem (the number of points or segments) must be large enough to demonstrate the efficiency of the algorithm, 10^5-10^6 or greater. The graphical primitives should be as simple as possible in order not to take too many resources for drawing: use just a pixel to draw one point and default thickness for drawing segments. The application should be compiled in the **Release** configuration. The output streams `std::cout` and `std::cerr` should be used for debugging purposes only.

Tasks

1. Given a set of 10^6 points on the plane, report the points that are inside the specified orthogonal rectangle using one of the below algorithms
    a. Regular grid
    b. Quadtree
    c. 2-d-tree
2. Determine whether a point is inside a specified simple polygon. The test must run for 10^6 points generated randomly. The polygon should be specified interactively, using the mouse.
3. Implement a $O(n \log n)$ algorithm for searching intersections of orthogonal straight-line segments. The test must run for 10^5 segments or more.
4. Implement the Bentley-Ottmann algorithm for searching intersections of arbitrary straight-line segments. The test must run for 10^5 segments or more.
5. Implement one of the below algorithms for constructing a convex hull, the number of initial points must be 10^5 or more
    1. Jarvis, 2. QuickHull, 3. Graham’s scan