Using CSG Representations of Polygons for Practical Point-in-Polygon Tests

(Extended Abstract)

Robert Walker Jack Snoeyink

Department of Computer Science

University of British Columbia

2366 Main Mall

Vancouver, BC, Canada V6T 1Z4

Telephone: +1 604 822 3061

Fax: +1 604 822 5485

Email: {walker|snoeyink}@cs.ubc.ca

Summary

A CSG representation for polygons is used for performing point-in-polygon tests, and is compared to existing methods. It is far less memory-intensive than the grid method and faster than basic methods.

Keywords: Geometric Modeling, Rendering Algorithms

Using CSG Representations of Polygons for Practical Point-in-Polygon Tests (Extended Abstract)

There have been many point-in-polygon test algorithms presented; Haines [1] gives a thorough comparative treatment of existing point-in-polygon algorithms, and the terminology that we use is based upon that work. The best of the current methods utilizing a preprocessing phase (the grid method [1]) requires *a lot* of extra storage in addition to the basic polygon information. We present a method based on constructive solid geometry (CSG) representations of polygons [2] requiring only two integers and three reals per edge plus an integer and two pointers per polygon (the original representation can be discarded). The only drawback to this method is that it does not support self-intersecting polygons.

A polygon may be represented via CSG as a monotone Boolean formula on the half-planes bounded by the edges of the polygon [2]. If we orient the edges of the polygon in a counterclockwise fashion, the half-planes of interest lie to the left of the edges. These formulae may be represented as directed trees in which each edge of the polygon appears only once.

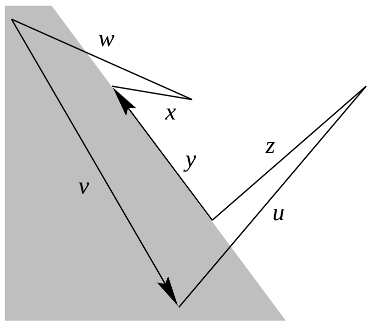

Figure 1. An example polygon yielding the formula $u \wedge v \wedge (w \wedge (x \vee y) \vee z)$. The half-plane defined by the edge y is also shown.

The point of interest is then tested against the leaves (edges) of the directed tree in left-to-right order to determine if it falls inside (TRUE) or outside (FALSE) the associated half-planes. Since an AND-node will not be satisfied if any of its children are FALSE, as soon as we determine that one of its children is FALSE, the remaining children need not be tested; the situation is analogous with OR-nodes. The idea is then to minimize the number of tests required for points on average for a given polygon; this may be done by altering the embedding of the directed tree by swapping any two children of a given node. As a simple first step, we chose to rearrange children beneath each node by size of subtree, and secondarily by length of edge. We are developing more sophisticated heuristics based upon coverage of the target plane (or bounding box) by the defined half-planes.

We confine the construction of the tree and its rearrangement to a preprocessing phase. This reduces the actual test algorithm to a series of point-in-half-plane tests, each resulting in a predetermined decision as to which edge to test next.

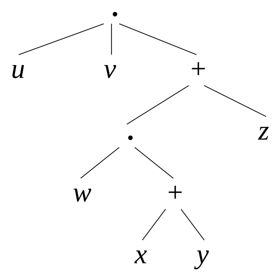

Figure 2. CSG representation for the polygon of Figure 1.

Haines' implementation of existing point-in-polygon tests [1] was augmented with our CSG method both with and without the sorting heuristics described above. Tests were performed as described by Haines, save that, to ensure that the polygons were non-self-intersecting, an edge-swap algorithm was used to untangle them. The tests were run on a Silicon Graphics Indigo workstation.

The simply-sorted CSG representation method is faster than the basic methods for all polygons save triangles. It uses significantly less memory than the grid method, and is faster than the grid method for small polygons. This method may be extended to 3D ray intersection, via the use of Pleucker coordinates, without having first to resort to determining a plane for the polygon and intersecting the ray with this plane.

	3	4	10	20	50	100
Crossings	6.8	7.1	11.4	19.3	41.2	77.1
Half-plane	2.4	3.8	11.3	23.7	59.0	117.8
Spackman	3.3	4.9	13.6	27.7	74.3	137.2
Grid 20×20	5.1	5.1	5.3	5.5	6.2	7.0
Grid 100×100	4.8	4.8	4.9	4.8	5.0	5.1
CSG w/sorting	3.0	3.9	7.6	11.3	20.5	32.4
CSG w/o sorting	3.4	4.3	7.8	12.7	24.4	41.5

Table. Average times for point-in-polygon tests (in microseconds), number of vertices per polygon vs. method. For each of 50 random polygons, 50 random points were tested. Some tests were repeated to reduce inaccuracy; see [1] for discussion.

References

[1] Eric Haines. Point in polygon strategies. In Paul S. Heckbert, editor, *Graphics Gems IV*, chapter 1.4, pages 24–46. Academic Press, Boston, MA, 1994.

[2] David Dobkin, Leonidas Guibas, John Hershberger, and Jack Snoeyink. An efficient algorithm for finding the CSG representation of a simple polygon. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 31–40, August 1988.