A FAST LINE-SWEEP ALGORITHM
FOR HIDDEN LINE ELIMINATION

OTTO NURMI
Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe,
Postfach 6380, D-7500 Karlsruhe, West Germany

Abstract.
Fast hidden line elimination algorithms can be obtained by minor modifications to algorithms
developed for reporting intersections of polygons. We show how the same modifications which have
been applied to segment trees can be applied to the data structure of Swart and Ladner as well,
leading to an $O((n+k)\log n)$ time hidden line elimination algorithm ($n$ is the number of boundary
edges of the input polygons and $k$ is the number of intersections of the edges on the projection plane).
The algorithm improves the fastest previous line-sweep algorithm for the problem by a factor
$O(\log n)$.

1. Introduction.
Given a set of planar polygonal faces in three-dimensional space and a
projection plane we wish to determine which parts of the (boundary) edges of
the faces are visible along the direction of the parallel projection so that we can
display only the visible parts.

One way to compute the visible parts of the edges is to sweep a horizontal
line from top to bottom through the projection plane and to store the faces
whose images are currently cut by the line. The faces are stored in an order
determined by their distances from the projection plane. If the polygons are
nonconvex one polygon can correspond to several intervals on the sweep line.
When the sweep line meets a point where the visibility status of an edge can
change, the data structure storing the faces is consulted. The data structure must
be able to reply with the face having minimum distance to the projection plane
among the faces which cover the edge. The data structure is updated when an
old interval ends or when a new one begins.

Ottmann and Widmayer [7] have shown how “segment trees” of Bentley and
Wood [3] and “tile trees” of McCreight [6] (or “interval trees” of Edelsbrunner
[4]) can be augmented to keep the distance information necessary for visibility
tests. The first structure needs $O(\log^2 n)$ time for each update operation but only
$O(\log n)$ time for a visibility test. The second needs at most $O(\log n)$ time for an

---

1 This work was supported by the grant Ot 64/4-2 from the Deutsche Forschungsgemeinschaft.
2 On leave from the Department of Computer Science, University of Helsinki, Finland.
Received July 1984. Revised February 1985.
update operation but $O(\log^2 n)$ for a visibility test ($n$ is the number of edges of the polygons). Both data structures lead to an $O((n+k)\log^2 n)$ time hidden line elimination algorithm, where $k$ is the number of intersections of the edges on the projection plane.

We show that the method by which segment trees could be modified in order to support visibility tests can be used in connection with the data structure of Swart and Ladner [9]. The augmented structure needs only $O(\log n)$ time for an update operation and for a single visibility test. The result is a hidden line elimination algorithm with $O((n+k)\log n)$ time bound ($n$ is the number of edges of polygons and $k$ the number of their intersections on the projection plane). It is interesting to observe that the time bound of the well-known algorithm of [2] for reporting intersections of line segments in a plane and the one of our algorithm differ only by a constant factor.

2. The algorithm.

We are given a set of two-dimensional polygonal faces in three-dimensional space. The faces may be concave and may have holes. The faces are projected to a plane. Each original face corresponds to one polygon of the picture on the projection plane. We assume that the coordinate system of the three-dimensional space has been chosen so that the projection plane has $z$-coordinate zero and that the faces have only positive $z$-coordinates. Now an edge is visible at a point $(x, y)$ if the corresponding face has the smallest $z$-value at $(x, y)$ among the given faces. If an edge is not visible it is hidden. The hidden parts of the edges are not displayed in the picture on the projection plane.

For simplicity, we assume first that each edge corresponds to one face only, and that there are no points on the picture that belong to more than two edges, and that the intersection points of the edges have pairwise different $y$-values. We will show later how these assumptions can be removed.

The faces are given by their edges with associated information on which side of the edge the interior of the face lies.

Visible parts of the edges are buffered so that always maximal visible parts are output, see e.g. [7]. The buffers tell us also whether an edge was visible at the preceding halting point of the sweep line.

The halting points of the sweep line are kept in a priority queue. Initially, the end points of the edges are sorted in decreasing $y$-order and stored in that order into the queue. The intersection points of the edges are later inserted into the structure. We assume without further notice that at most one intersection point for each edge is stored at a time in the priority queue. The priority queue can be organized so that each operation directed to it needs at most $O(\log n)$ time (see [1,2] for details). The main data structure of the algorithm stores the edges currently cut by the sweep line (active edges) into a balanced binary search tree from left to right along the sweep line, like the active segments are stored in the