A Simple and Efficient Haloed Line Algorithm for Hidden Line Elimination

W.R. Franklin' and V. Akman†

Abstract

An efficient algorithm, HALO, is given to compute haloed line drawings of wire frame objects. (Haloed line drawings are described by Appel et al. 1

HALO has two parts: CUT and DRAW. CUT uses an adaptive grid to find all edge intersections. It overlays a square grid, whose fineness is a function of the number and length of the edges, on the scene. It determines the cells that each edge passes through, sorts these by cell to obtain the edges in each cell, and then, in each cell, tests each pair of edges in that cell for intersection. For broad classes of input this takes time linear in the number of edges plus the number of intersections. CUT writes a file containing all the locations where each edge is crossed in front by another. Given a halo width, DRAW reads this file edge by edge. For each edge, it subtracts and adds the halo width to each intersection to get the locations where the edge becomes invisible and visible. It sorts these along the edge, and then traverses the edge, plotting those portions where the number of "visible" transitions is equal to the number of "invisible" transitions. DRAW takes time linear in the number of edge segments. Dividing HALO into two parts means that redrawing a plot with a different halo width is fast, since only DRAW need to be rerun.

CR Categories and Subject Descriptions: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - geometric algorithms, languages, and system; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems - geometrical problems and computations

General Terms: Algorithms, design.

Additional Key Words and Phrases: Hidden line elimination, haloed line effect, wire frame.

*Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180
USA

†Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

North-Holland
Computer Graphics Forum 6 (1987) 103-110

1. Introduction

As computer aided design (CAD) deals with more complicated databases, it becomes crucial to display the data effectively so that people can comprehend it. A suitable method must be efficient since users will be interactively manipulating and displaying their data. With special purpose hardware becoming less expensive, and even custom VLSI design becoming as easy as writing software (for those with the appropriate facilities), a suitable algorithm should lend itself to parallelism and implementation in silicon. Since a CAD database may contain wire frame models without any surface information, the algorithm should be able to handle them.

This paper offers an efficient algorithm called HALO to solve this problem via the technique of haloed line elimination. Haloed lines are introduced in Appel et al.1 which cites many reasons for using them and gives good examples. Briefly, we assume that each line has a narrow region, or halo, that runs along it on both sides. If another more distant line intersects this first line, then that part of the farther line that passes through the first line's halo is blotted out. For example, see figure 1 which shows four drawings of a pair of cubes. Figure 1(a) gives all the edges. Figure 1(b) draws the visible edges solidly, but removes the hidden edges. Figure 1(c) draws the visible edges solidly, but dashes the hidden edges. Finally, figure 1(d) gives all the edges, but adds the haloed line effect. It should be clear that the haloed drawing shows more 3-dimensional relationships than the other three. Figure, 1(b) and 1(c) do not give the 3-dimensional relationship between two edges that are both hidden, since either both will be omitted or both will be drawn dashed. Further, if we dash the hidden edges, we must be able to tell which edges are hidden, so we must know what the faces of the objects are. In contrast, with haloed lines we produce a gap on an edge where it passes behind another edge, so we need only the edge data and not the faces.

To be fair, Markowsky and Wesley2 show how to calculate the faces from just the edges, but the process is slow and subject to ambiguities. To observe that distinct objects can have the same wire frame, refer to figure 2 which shows an object with 9 vertices (the corners and the center of a rectangular block) and 20 edges (the edges of the block and those connecting each block corner to the center). This represents a closed

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org