The Hidden Face Determination Tree

A. James and A.M. Day

School of Information Systems, University of East Anglia,
University Plain, Norwich, Norfolk, NR4 7TJ. England.
aj@sys.uea.ac.uk, amd@sys.uea.ac.uk

Abstract

Hidden surface removal can be achieved using the Priority Face Determination (PFD) tree to create a priority ordering of n polygons in $\log_{4/3} n$ *time. In this paper, we describe the Hidden Face Determination (HFD) tree which is based upon the PFD tree. The HFD tree is constructed in a similar way as the PFD tree, but an addition means that polygons completely hidden by others in the scene can be determined at each node. Thus, not only does the HFD tree calculate the priority order of polygons in* $\log_{4/3} n$ *time, but determines which polygons are invisible from a given location.*

Keywords: area-to-area visibility, binary-space partitioning, hidden-surface removal, scene clustering

1. Introduction

The BSP tree developed by Fuchs, Kedem and Naylor [1] finds the visual priority order of a set of polygons. However, due to BSP splitting, the number of polygons in the scene (each of which requires testing at run-time) is increased. The number of tests can be reduced using the Priority Face Determination (PFD) tree algorithm [2] which is built using its associated BSP tree. The PFD tree reduces the number of tests to $\log_{4/3} n$. Two observations make this possible, the first is the use of a *height plane* to reduce the freedom of the observer to movements at a set height; this restriction occurs naturally in applications such as walk-throughs. The second observation is the existence of polygon dependencies in the scene; if we know our location relative to the front or back of a polygon, then in the expected worst-case scene, our position relative to one-quarter of the remaining polygons can be determined.

The PFD tree comprises a two-element node structure: a test-line (analogous to the sole element in a two-dimensional BSP tree's node) and a priority list of polygons. Priority ordering is achieved by a single root to leaf traversal of the tree. Each node is tested to see on which side the observer is located: for a front (back) result, the front (back) side of the tree is traversed. The polygons in each priority list are displayed in order when encountered.

2. Hidden Face Determination

Although priority ordering of polygons can be achieved in $\log_{4/3} n$ time, the ordering of polygons is not a significant bottleneck of the rendering pipeline thus no visual speed-up is apparent. A visual speed-up can be gained if the number of polygons drawn to the screen is minimized by determining which polygons are not visible from the observer's current position. Not only does this reduce the need to draw hidden polygons, but also their underlying shading, contained shadows and texture mapping etc.

The front-to-back display of BSP trees [3] is one way in which to draw only those polygons that are visible. However, such a method requires image space analysis thus increases the amount of run-time processing. Other work in this field such as portal sequences and portal textures have been successful [4, 5, 6, 7] but the Hidden Face Determination (HFD) tree, presented in the next section, combines priority ordering and hidden polygon rejection into a simple binary tree requiring only a logarithmic number of tests to determine the order of visible polygons needed to produce the final image.

3. The HFD Tree Algorithm

Based on the algorithm for PFD tree building [2], the HFD tree adds a visibility 'plug-in' which shifts most of hidden surface removal to the pre-processing stage. HFD tree visibility is introduced at the priority list promotion stage of the algorithm. In the PFD tree, a polygon was promoted to the priority list when processing on all higher priority polygons had been completed. In the HFD tree, this promotion step is intervened by an area-to-area visibility test. The visibility test is necessary from the current view area (corresponding to our position in the tree) to a polygon under consideration for addition to the tree, using the list of unprocessed polygons as shields. Any polygon not visible from the current view area can be rejected and removed from further consideration.

Given a scene comprising even a modest number of polygons, the amount of time spent on visibility calculations is dramatic when compared to the PFD tree's insertion. Fortunately, we only need to know whether a polygon is visible or invisible from a given area and not its *proportion* of visibility. We find acceptable visual results can be gained with point sampling using just a few points per polygon.

4. Example

Figure 1 shows a simple nine-facet scene, labels are placed upon the front side of each facet but no back face culling is used in this example. The underlying grey-scale intensities illustrate the fraction of polygons drawn in each tree, namely $\frac{HFD}{BSP}$. Figure 2 shows the BSP tree for the scene. Figure 3 shows the front side of the HFD tree and bracketed letters are used to indicate hidden polygons.

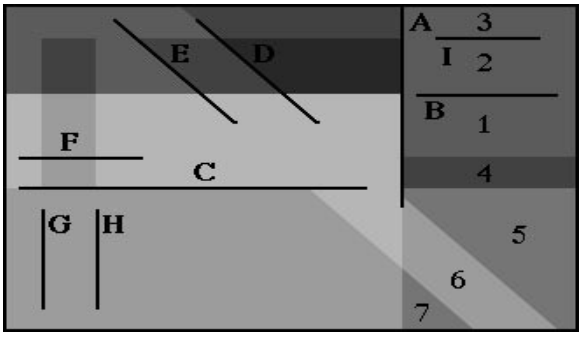

Figure 1: *Nine facet scene, underlying grey-scales illustrates run-time efficiency*

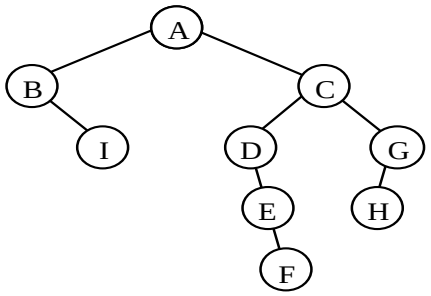

Figure 2: *BSP tree for the scene*

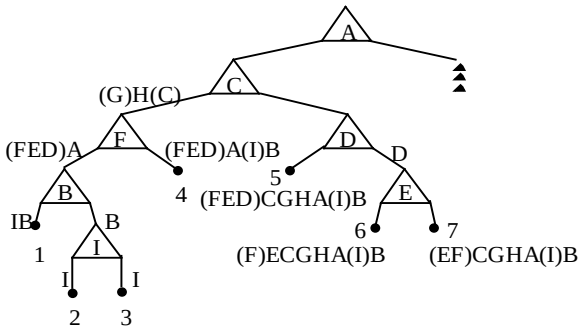

Figure 3: *Front side of the HFD tree*

References

1. H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. *ACM Computer Graphics*, 14(3):124 – 133, July 1980.

2. A. James and A. M. Day. The priority face determination tree for hidden surface removal. *Accepted for publication in Computer Graphics Forum (to appear)*, 1998.

3. D. Gordon and S. Chen. Front-to-back display of BSP trees. *IEEE Computer Graphics and Applications*, 11(5):79 – 85, 1991.

4. S. J. Teller and C. H. Sequin. Visibility preprocessing for interactive walkthroughs. *ACM Computer Graphics*, 25(4):61 – 69, July 1991.

5. S. J. Teller and P. Hanrahan. Global visibility for illumination computations. *Computer Graphics Proceedings*, pages 239 – 246, 1993.

6. N. Greene. Hierarchical polygon tiling with coverage masks. *ACM Computer Graphics Proceedings*, pages 65 – 74, 1996.

7. G. A. Aliaga and A. A. Lastra. Architectural walkthroughs using portal textures. *IEEE Visualisation Proceedings*, pages 355 – 362, October 1997.