

BSP Visibility on S^2

John Fedorkiw, Sherif Ghali, and Todd Keeler
Department of Computing Science
University of Alberta

1 Introduction

Despite over four decades of published algorithms for the computation of visibility, a large gap between theory and practice remains. A significant objective is to design and implement a robust and efficient algorithm that makes it possible to generate the set of polygons visible from an observer in a piecewise linear scene. We describe here and in the accompanying manuscript an algorithm that partitions a sphere centered at the observer into regions corresponding to the visible polygons. The partition of the sphere of directions S^2 is stored in a doubly-connected edge list.

Many algorithms for computing vector-based visibility have been proposed over the years [2], but no algorithm has achieved simultaneously the objectives of efficiency and practicality. The first objective requires that the algorithm handle in sub-quadratic time large classes of useful inputs. The second objective requires that a robust implementation be feasible. A third objective, reusability, is also desirable. Reusability requires that the implementation be customizable for the various scenarios needing the use of vector visibility.

2 Algorithm

The algorithm is a variation on Thibault and Naylor’s fundamental work showing how boolean operations can be computed using BSP trees [4]. The two operations performed, set difference and union, have been frequently used for visibility computation (e.g. [5, 3]). The visible polygons are maintained incrementally as the input polygons are processed in near-to-far order. The difference and the union between a polygon and the view are simultaneously computed. The difference identifies the visible portions of the polygon and the union identifies the new “mask” to be used for the subsequent polygon. Subject only to the constraint that the observer not lie on the boundary (or in the interior) of any of the polyhedra defining the scene, we compute the partition of a sphere centered at the observer such that the faces of the partition correspond to the visible polygons. See Figure 1. Two

data structures are maintained: a topological map T partitions S^2 into faces that correspond to the visible (or the background) polygons and a BSP tree V acts as a search structure and helps identify the faces of T in which a polygon lies.

Figure 2 (a) and (b) show a sample scene and the spherical map and Figures 2 (c) and (d) show the experimental time and space needed to compute the visibility for various input sizes of the scene.

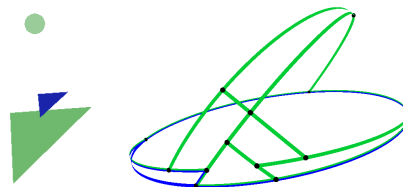


Figure 1: A scene consisting of two triangles and its spherical map as seen from the observer are shown.

3 Implementation

The algorithm was implemented in C++ using CGAL. Even though the implementation is generic to facilitate the comparison of different kernels, exactness (through rational arithmetic) and filtering are necessary for robustness and efficiency. It is worth noting that using BSP trees in combination with rational arithmetic makes it particularly difficult to process significantly larger scenes. Until the BSP leaf node of a polygon’s fragment is identified, the polygon is (potentially) divided by all partitioning planes along the path from the root to the leaf. Thus in turn the height of the expression tree of the coordinates of a polygon’s fragment will (also potentially) be as deep as the leaf node. We wanted to confirm whether the storage needed for the rational numbers we use grows exponentially with the depth of a node in the tree. Figure 3 (top) shows a visualization of the tree T resulting from processing a small scene. We found the number of bytes needed for the larger of the numerator and the denominator of each rational number (gmpq) at a particular depth

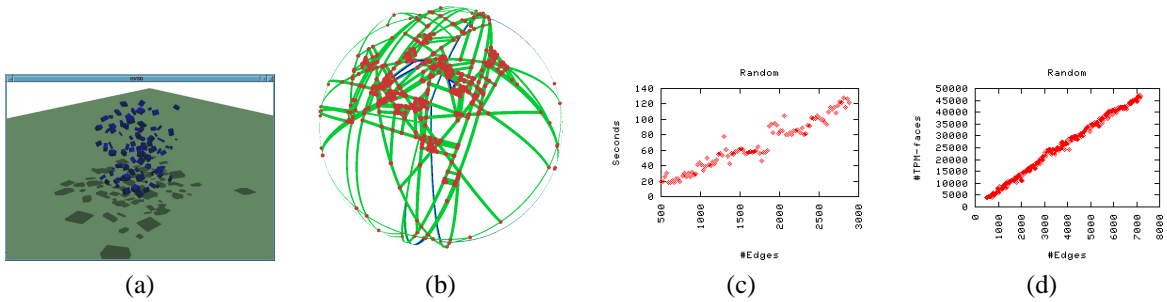


Figure 2: The relationships between the number of edges in a scene consisting of a set of randomly oriented and scaled cubes (a) and the time (c) and space (d) needed to process the scene are shown. The spherical map of a smaller scene is also shown (b).

in the tree, giving the sample points in Figure 3 (bottom). The curve shows the number of nodes used at a particular depth in the tree. The distribution of rational number sizes suggests that an exponential growth does not occur. Nevertheless, the storage needed for many large rational numbers precludes handling arbitrarily larger scenes.

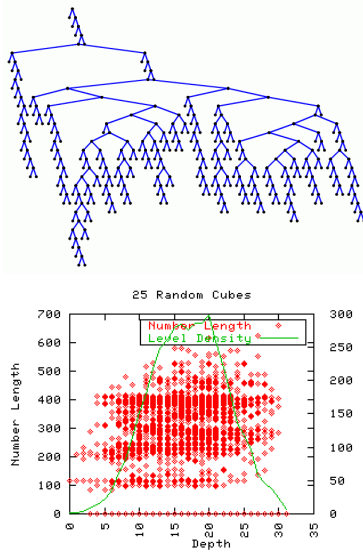


Figure 3: A visualization of the tree T (top) and statistics (bottom) illustrating the density of the tree as well as the growth of rational numbers are shown.

4 Future Work

Rewriting our code to comply with CGAL's design ideas of parameterizability in both the data structure and the algorithm dimensions may make tackling a large set of problems that rely on vector visibility possible. We briefly describe some of these problems.

View isomorphism Given a scene in 3D and two observer locations, how can one determine whether the two views are equal?

Critical points How can one determine efficiently the points along a line segment at which the view changes [1]?

Shadows How can the necessary and sufficient set of shadow boundaries be inserted into a scene lit by a point light source?

Non-Photorealistic Rendering Virtually all rendering algorithms in computer graphics use raster visibility, but most NPR algorithms are geared towards print media and so frequently require vector visibility computations.

Aspect graph How can one partition either 3d-space or the sphere of orthographic directions into iso-visibility regions?

References

- [1] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view. *Algorithmica*, 11:360–378, 1994.
- [2] S. E. Dorward. A survey of object-space hidden surface removal. *Internat. J. Comput. Geom. Appl.*, 4:325–362, 1994.
- [3] M. J. Katz, M. H. Overmars, and Micha Sharir. Efficient hidden surface removal for objects with small union size. *Comput. Geom. Theory Appl.*, 2:223–234, 1992.
- [4] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Computer Graphics*, 21(4):153–162, 1987.
- [5] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. *Computer Graphics*, 11(2):214–222, 1977. Proc. SIGGRAPH '77.