Evaluation of BSP trees for ray-tracing

Vlastimil Havran
Czech Technical University, Faculty of Electrical Engineering,
Department of Computer Science and Engineering,
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
e-mail: havran@cs.felk.cvut.cz

Keywords: computer graphics, rendering, ray tracing, BSP tree.

Ray tracing is method commonly used for photo realistic rendering nowdays. It is based on simulation of light behaviour by means of geometrical optics, light beams used are usually called rays. In real world the light is emmitted from light sources and is reflected or refracted on surfaces of objects before it reaches the human eye. The ray-tracing is usually based on reverse process: the rays are cast from eye to the world and refracted and reflected on the objects. The intersection with closest object for each ray is calculated. Moreover, additional rays are cast from these intersection points to the light sources to find out if the intersection point is illuminated by the light or if it lies in shadow. These rays are called shadow rays. The result colour for a pixel is calculated by a special lighting model.

The most time consuming part of ray-tracing process is finding out of the closest objects. The naive method is to test a ray with each objects and select the closest intersection. Since the number of objects is usually very big, calculation takes up many. hours. Therefore space partitioning technique were designed. One of them is *Binary Space Partitioning Technique* (abbr. BSP). Let us explain how does it work.

The objects in the scene are bounded by rectangular parallepiped, it is actually the root node of the tree. The space is recursively subdivided by a plane perpendicular to one of coordinate axes. The splitting plane is positioned in a mid-point of the current axis; the axis is changed regularly in order x, y, and z. The objects laying in the left node are assigned to this node, similarly the objects in the right node, and the objects straddling the splitting plane are added to both halfspaces. The splitting is stopped, when the number of objects in current node is smaller than a constant, or when the depth of the node reaches maximal depth allowed. There is a very quick algorithm for traversing the BSP tree by a ray [3]. The intersection calculation are performed only with objects placed in leaves of the tree, it accelerates the computation process hundred times, but total time of computation is far from real-time rendering.

The scientists designed some sophisticated methods [2] improving the efficiency of BSP tree. The splitting plane is not placed in mid-point and its

orientation is not regularly changed in cyclic order. The trees are smaller and the execution time can be decreased up to by 80%.

The comparison between scientific papers is normaly given by time consumed by computation. It disables to compare the solution designed by scientists because of different implementation and measurement conditions given. The testing is performed on standard scenes proposed in [1]. The total computation time is composed of two parts, the time devoted to traversing and the time devoted to intersection testing.

We propose new method for comparing the quality of BSP tree, which is based on two n-tuples. First one, septet Δ, includes the characteristics of binary tree independent on implementation:

$$\Delta =< N_{LS}, R_{ETNLS}, N_{ADL}, N_{AOIFL}, R_{FVWV}, N_{RPRT}, N_{AT} > \qquad (1)$$

Δ covers following parameters: the number of leaves, the ratio of empty leaves to all leaves, the average number of objects duplication, average number of objects in all leaves, the average number of objects in full leaves, the ratio of volumes taken by non-empty leaves to the volume of the whole scene, the ratio of all intersection tests performed for to minimal number of intersection, and the average number of traversed nodes per one ray.

The second n-tuple, triplet Λ, enables to compare the quality of ray-tracer implementation, the architecture performance, and the quality of compiler:

$$\Lambda =< T_{CB}, T_{TR}, T_{TT}/T_{TR} >, \qquad (2)$$

Λ expresses: the time devoted to construction of BSP tree, total rendering time itself, and the ratio of time consumed by traversing to the total rendering time.

References

[1] E. A. Haines *A proposal for standard graphics environments.* IEEE CG & A, 7(11):3–5, Nov. 1987.

[2] J. D. MacDonald and K. S. Booth. *Heuristics for ray tracing using space subdivision. In* Proceedings of Graphics Interface '89*, pages 152–163, Toronto, Ontario, June 1989.*

[3] K. Sung and P. Shirley. Ray tracing with the BSP tree. *In Graphics Gems III, pages 271–274. Academic Press, San Diego, 1992.*