STATISTICALLY OPTIMIZED TRAVERSAL ALGORITHM FOR BSP TREES

V. Havran

CTU, Fac. of Electrical Eng., Dept. of Computer Science & Engineering
Karlovo nám. 13, 121 35 Praha 2

Key words: computer graphics, BSP tree, ray-tracing

A BSP *(binary space partitioning)* tree is a commonly used spatial subdivision data structure in many graphics application. It is the analogue to binary search trees [4], but it process n-dimensional data. The BSP tree is used to accelerate the search queries for these data. A BSP tree hierarchically subdivides a volume containing a collection of objects. The tree is formed recursively subdividing the volume in two volumes, usually halves. The resulting data structure is a binary search tree in which each interior node represents a partitioning hyperplane and its children represent convex volumes determined by the partitioning. The leaf nodes of the tree are thus convex are non-overlapping, occupied by objects or vacant.

There are important applications of BSP tree in the field of image synthesis; to determine the *visibility of two points* in space, that can be occluded by some objects and a *ray-casting* problem [2]. Ray-casting is defined as follows: for a given ray find the closest object which is intersected by the ray if such an object exists. In this paper we describe the properties of common traversal algorithm for ray-casting problem and outline briefly a new statistically optimized algorithm. We refer to commonly used algorithm originally published in [3] and described in more detail in [1] as traversal algorithm TA_A.

The traversal of a ray through BSP tree is recursive operation. Each inner node represents one decision step during traversal. This decision is based on mutual geometrical position of a ray and the isothetic bounding box of the inner node. At each traversal step the child node closer to the origin of a ray must be determined. It is followed by computing the mutual position of the ray and the splitting plane located inside the bounding box of the inner node. Three examples of traversal cases are depicted in Fig. 1, there are 26 different positions of a ray and a inner node of BSP tree in general.

Figure 1: BSP tree traversal examples

The algorithm TA_A is based on computation of the signed distance from the origin of a ray to a splitting plane. It is computed as follows:

$$t = \frac{splitVal - ray.loc.axis}{ray.dir.axis}$$

Subsequently, near and far children are determined. In Fig. 1 (A) near child is below splitting plane and far child is above. If signed distance is smaller than zero or greater than the signed distance of the *exit* point, then only the near child is selected. Otherwise, if the signed distance to the splitting plane is smaller than the signed distance to the *entry* point, only far child is selected. Otherwise, first near child and then far child must be visited during traversal.

The result of a traversal step of a ray through an inner node is one of four cases:

- visit the left child only
- visit the left child first, the right child afterwards
- visit the right child only
- visit the right child first, the left child afterwards

Our analysis have shown, that the traversal algorithm TA_A based on the computation of the signed distance can fail if the origin of a ray is embedded into the splitting plane. The consequence of this failure is the incorrect image synthesis, which come to light in the form of disturbing pixels or even the missing parts in the resulting image.

The statistical analysis based on *surface area heuristics* and experimental analysis have shown, that computation of signed distance is necessary only for 26.1 % cases in the worst case, if a ray passes through both child nodes. This part of computation is the most time consuming.

Further, we discuss the function and the properties of our new traversal algorithm, that eliminates the shortcomings of algorithm TA_A. We refer to the new algorithm as algorithm TA_B in this paper.

We outline the principle of decision for example given in Fig. 1 (A) for algorithm TA_B: If the projection of entry point corresponding to signed distance a to current axis is less than or equal to the position of splitting plane ($x_a \leq x_{sp}$) and the projection of exit point corresponding to signed distance b to current axis is less than or equal to position of splitting plane ($x_b \leq x_{sp}$), then only left child is selected.

The detailed description and the statistical analysis of algorithm TA_B is beyond the scope of this paper. The algorithm TA_B is optimal, because the decision between four possible results is performed using only two comparisons ($\log_2 4 = 2$). We implemented a new traversal algorithm TA_B and performed the benchmark tests. Our measurements have shown, that speedup of the algorithm TA_B reaches from 1.54 to 2.10 in comparison with the algorithm TA_A. Moreover, it handles all the singular cases correctly, for which algorithm TA_A fails.

References:

[1] Sung, K. – Shirley, P.: *Ray Tracing with BSP Tree* pp. 271–274. ACM-PRESS, New-York 1992.

[2] Glassner, A. S.: *An Introduction to Ray Tracing* Academic Press, London 1991

[3] Jansen, F. W.: *Data Structures for Ray Tracing* in book *Data Structures for Raster Graphics*, Springer Verlag, June 24-28, 1985.

[4] Cormen, T. H. – Leiserson, C. H. – Rivest, R. L.: *Introduction to Algorithms* The MIT Press, Cambridge, Massachusetts, 1990.