Czech Technical University in Prague

WORKSHOP 2011

Project carried out within the framework of the CTU Student grant competition in 2010

This research has been supported by SGS grant No. SGS10/289/OHK3/3T/13

When it is More Efficient to Use Uniform Grids for Ray Tracing

M. Hapala, O. Karlík, V. Havran (supervisor)

hapalmic@fel.cvut.cz

Department of Computer Graphics and Interaction, Faculty of Electrical Engineering
Czech Technical University in Prague

Abstract

Commonly used hierarchical data structures such as bounding volume hierarchies and kd-trees have rather high build times, which can be a bottleneck for applications rebuilding or updating the acceleration structure required by data changes. On the other hand uniform grids can be built almost instantly in linear time, however, they can suffer from severe performance penalty, in particular in scenes with non uniformly populated geometry. We improve on performance using a two-step approach that selects a more efficient data structure given a particular implementation of the algorithms which yields with high probability an overall smaller computational time.

Ray tracing is a technique that can be used for generating images by shooting rays into a 3D scene and finding closest intersections among rays and the scene objects. This basic visibility computation is used in a core of many rendering algorithms. Different data structures have been proposed to speed up ray tracing computation, among them a kd-tree [1] and a uniform grid [2]. Kd-tree ranks among hierarchical data structures which can adapt well to a scene with a non-uniform distribution of primitives. However, their build algorithm has a super-linear time complexity which can create a bottleneck. On the other hand the uniform grid is non-adaptive but can be created only in a linear time.

In this contribution we propose and study an algorithm that uses the estimate of performance properties choosing acceleration data structures on the fly, an issue previously studied by Havran et al. [3] and also by Müller and Fellner [4]. The algorithm uses a calibration phase

Project carried out within the framework of the CTU Student grant competition in 2010

which requires a set of scenes of different properties such as number of geometric primitives and their spatial distribution. The calibration phase is executed once before the algorithm is used for an application on a particular hardware. During the calibration phase for each scene we build both a uniform grid and a kd-tree and measure the time it takes to build them and for computing a single ray. This gives us implementation and hardware constants for building up data structures and also a practical efficiency of shooting rays.

Then for each run of the algorithm the data from the calibration phase are used to estimate efficiency of the data structures on the actual scene. Given an unknown scene we first build a uniform grid in a short time and test its performance by sampling a small set of representative rays to get the time to compute a single ray. Then we compute the build time and the time to compute shooting a single ray with a hierarchical data structure by estimating it from the number of primitives included in the scene and the data gathered in the calibration phase. From these we can easily compute what is the minimal workload (number of rays) we need to gain performance by building the hierarchical structure. Naturally, to make a correct decision we need to know at least roughly the number of rays to be computed.

The algorithm can have three outcomes. First, if the time to shoot a ray in the grid is lower than the time to shoot a ray in the hierarchical structure it makes no sense to build the latter. Second, if the number of rays to be shot is lower than the minimal workload it does not pay off to build the hierarchical structure. This is because the time to build a hierarchical data structure is relatively high even if it provides faster processing of a single ray. Third, for the number of rays larger than the minimal workload it will always be more efficient to discard the uniform grid, build the hierarchical data structure and use it to shoot the rays.

The comparison between the algorithm that combines the use of both structures with the one that will use only one of the two is as follows. When we use only a uniform grid the proposed algorithm is more efficient as it is always of the same performance or provides the speedup in cases when we detect that the grids are inefficient. We also compare the proposed algorithm to the use of only the hierarchical data structure as we need the additional time to build the uniform grid. Favorably, the time complexity $O(N)$ is asymptotically smaller than the time complexity needed to build the hierarchical data structure $O(N \cdot \log_2 N)$. Theoretically, the time complexity needed to build the uniform grid, which is possibly later discarded, gives the time complexity increase from $O(N \cdot \log_2 N)$ to $O((N(1+\log_2 N))$ that presents the slowdown of

building of only a hierarchical data structure $1+1/\log_2 N$. In practice when we take into account the particular implementation, the constants behind the time complexities for building them are even higher for hierarchical data structure when compared to uniform grids. Therefore such slowdown is negligible.

To test the algorithm in practice we have implemented a path tracer in C++ and tested it on the set of 28 scenes where always a certain number was used for calibration while the rest was estimated. From the measurements we have computed exactly the minimal workload where rendering using the hierarchical structure starts to be faster. This was compared to the estimated minimal workloads computed for each scene. The results have shown that the estimate on the tested scenes is about 25 percent more optimistic about the quality of the grid and is quite stable in this prediction except for the extremes where there are either not enough calibration scenes or not enough estimated scenes. The results have also shown that the range of rays where its does not pay off to build the hierarchical data structure can be significant in particular for scenes with a higher number of geometric primitives and that without computing the estimate the selection cannot be made in general.

We have proposed an algorithm that combines a uniform grid and a hierarchical data structure for ray tracing so that it takes advantages of both types. Based on the scene properties and a small number of rays computed using the grid we decide either to continue ray tracing with the grid or to build the hierarchical data structure such as kd-trees. To our best knowledge we present the first algorithm that decides when it is advantageous to use uniform grids in dependence on the number of rays to be shot.

References:

[1] J.L. BENTLEY: *Multidimensional binary search trees used for associative searching* Commun. ACM, 1975, pp. 509–517.

[2] A. FUJIMOTO, T. TANAKA, K. IWATA: *ARTS: Accelerated ray tracing system* IEEE Computer Graphics and Applications, 1986, pp. 16–26.

[3] V. HAVRAN, J. PRIKRYL, W. PURGATHOFER: *Statistical comparison of ray-shooting efficiency schemes* Technical Report TR-186-2-00-14, Vienna University of Technology, 2000.

[4] G. MÜLLER, D.W. FELLNER: *Hybrid Scene Structuring with Application to Ray Tracing* Proceedings. of Intl. Conf. on Visual Computing ICVC ’99, 1999.