Deferred Ray Tracing Using Intersection Binning

Apollo I. Ellis, Eric G. Shaffer and John C. Hart
Department of Computer Science

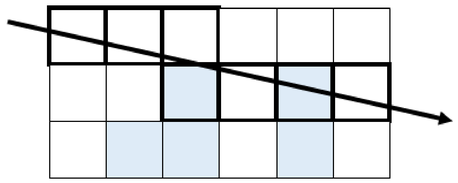
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

INTRODUCTION

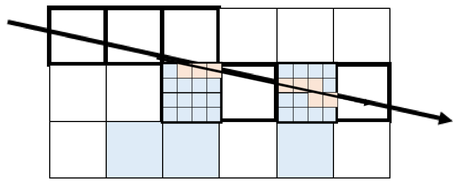
We present an effective approach for dealing with incoherent memory accesses and thread divergence in GPU ray tracing of diffuse rays, *deferred ray tracing*. We separate traversal from intersection, working under the guiding principle that the coherence we regain will more than make up for any extra work required. Our method achieves this coherence with minimal overhead by employing a ray binning scheme. Our approach can outperform a traditional “forward” ray tracer, by over 20X, in apples to apples comparisons. For preliminary evaluation of deferred ray tracing as a general approach, we have implemented it with a two level world space grid structure, such as that of Kalojanov et al. ‘11.

DEFERRED TRAVERSAL

Step 1: Traverse coarse grid cells recording grid cells that may contain intersections.



Step 2: Traverse fine grid cells inside each coarse grid cell to find any intersections.



Our traversal is akin to that of Kalojanov et al., except we defer all second level grid traversal until each ray traverses the entire first level grid. We then traverse the second level cells, one first level cell per thread. Deferring in this way increases thread coherence, yielding better performance than in-order traversal even though we are over traversing at both levels.

We also defer all intersection calculations. We store each ray-cell encounter as a pair, and we arrange these pairs in a buffer, such that pairs which reference the same grid cell are adjacent. This encourages adjacent threads to access the same memory locations.

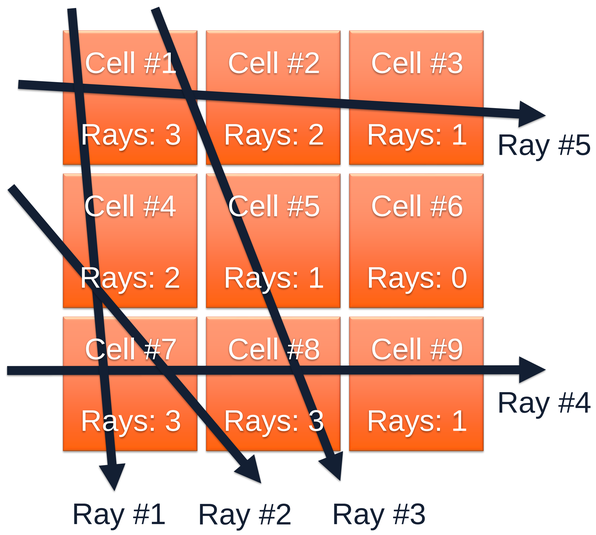
REFERENCES

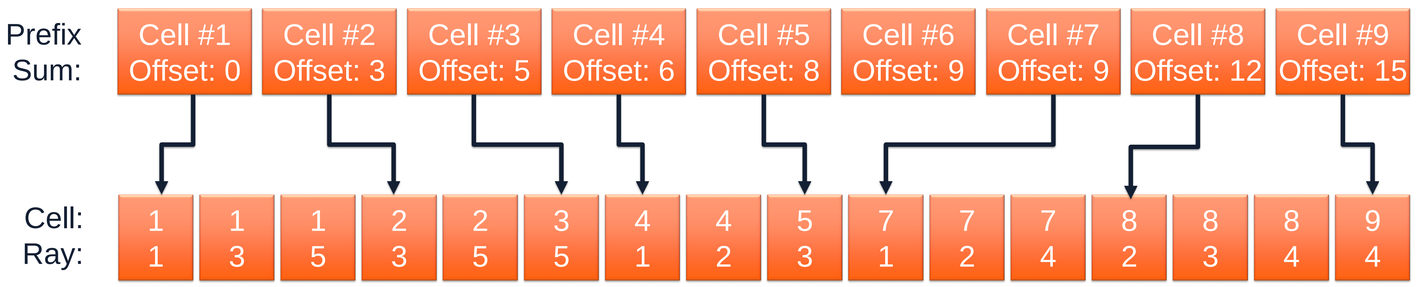
1. GUNTURY, S., AND NARAYANAN, P. J. 2012. Raytracing dynamic scenes on the gpu using grids. *IEEE Transactions on Visualization and Computer Graphics*.
2. KALOJANOV, J., BILLETER, M., AND SLUSALLEK, P. 2011. Two-level grids for ray tracing on gpus. *Computer Graphics Forum*.
3. PÉRARD-GAYOT, A., KALOJANOV, J., AND SLUSALLEK, P. 2017. GPU ray tracing using irregular grids. *Computer Graphics Forum*.
4. RESHETOV, A., SOUPIKOV, A., AND HURLEY, J. 2005. Multi-level ray tracing algorithm. In *ACMSIGGRAPH 2005 Papers*.

BINNING INTERSECTIONS

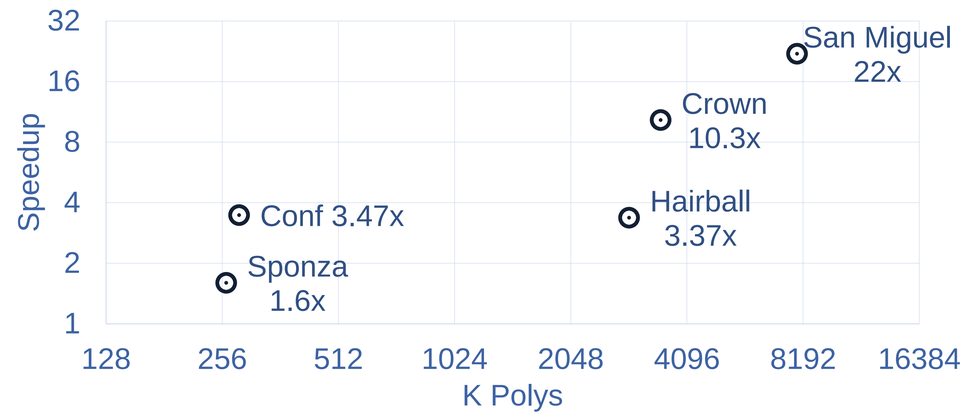
Arranging the cells in the buffer is a sort-like operation, and in fact sort is used in previous work by Guntury and Narayanan ‘12. Unfortunately, sorting the number of pairs encountered in complex scenes is prohibitive, on the order of 1 millisecond per 1M pairs. So we seek a rapid alternative to sorting.

We have implemented a two-pass binning traversal scheme. The first traversal pass counts ray visits to each cell, and the second pass packs ray-cell pairs at array offsets according to the prefix sum of the counts. This packs pairs with a common cell adjacent to each other. In this manner we attain the same effective layout as with sorting by cell, except that this approach is extremely rapid, incurring minimal overhead since traversal is reduced to an increment and a store.





RESULTS



For an intersection rate comparison with a forward system we compare against synthetic work optimal intersection. For each ray, we record how far into the grid we traversed to the final hit point. Then we run a kernel which spins over the recorded cells, in order, reports the first intersection, and halts. Here, a ray intersects a cell if and only if it would have on the path to the hit point in a forward ray tracer as well. This intersector is thus fair, barring obscure caching effects, and the scheduling may be more optimal than having to interleave traversal operations.

With respect to tracing diffuse rays in a grid, intersection calculations dominate over traversal. The chart above showcases our speed up over forward intersection relative to the scene size, it is up to 22X. This speedup persists even if we reduce the forward ray tracer’s hypothetical traversal cost to zero.

CONCLUSIONS

Uniform grid structures are not necessarily optimal for GPU ray tracing in comparison with recent work Pérard-Gayot et al. ‘17, thus, we need to evaluate deferral with hierarchical data structures. In addition, we believe deferred ray tracing could be significantly accelerated by empty occluders Reshetov et al. 05, which would yield less extra traversal steps and intersection tests. Overall, we believe deferred ray tracing has the potential to improve the state of the art in GPU ray tracing.