Fast Ray Tracing of Scenes With Unstructured Motion

Pankaj Khanna Jesper Mortensen Insu Yu Mel Slater

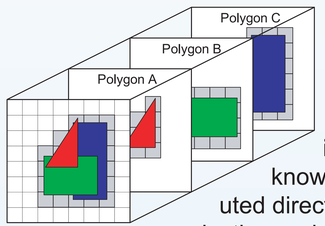
Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK.

vlfgroup@cs.ucl.ac.uk www.cs.ucl.ac.uk/vlf

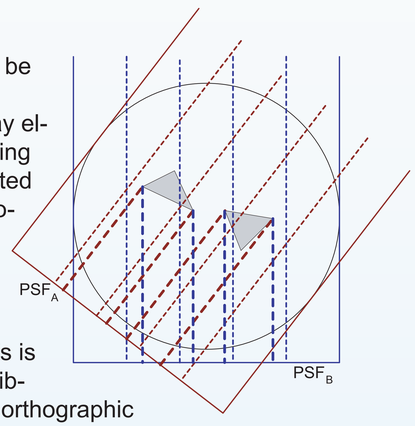
Ray tracing dynamically changing scenes with unstructured motion has long been a problem for ray-traversal acceleration schemes. When polygons are transformed arbitrarily, the cost of updating traditional spatial data-structures can be quite high [TL03][IW03]. We propose a ray traversal scheme that is well suited to scenes with dynamically changing objects during rendering with ray tracing, using a data structure that exploits coherence in view space. The main operation for handling arbitrary transformations of objects reduces to low resolution 2D polygon rasterisation.

Data Structure

Find the smallest bounding cuboid that encloses the scene, and divide the bottom face (parallel to the XY-plane) into NxN 2D square tiles. Each tile can be considered as one end of a 3D beam parallel to the Z-axis that intersects a number of polygons. The tiling is represented as a 2D array, where each array element stores a list of identifiers of polygons intersected by that beam. This tiling is easy and fast to compute – since the scene can be orthographically projected onto the cuboid base, and a 2D scan-line algorithm for each polygon will identify the tiles in which it falls. The whole set of beams is called a 'parallel subfield' (PSF) and ultimately represents the set of all possible rays in the beam direction, parallel to the z-axis (this is known as the canonical PSF). Other uniformly distributed directions are similarly represented by PSFs by an orthographic projection and rasterization of polygons in the corresponding direction. Each discretised direction thus has a set of tiles containing identifiers of overlapping polygons.

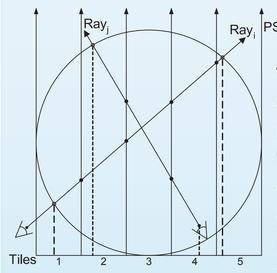


PSF Tiles

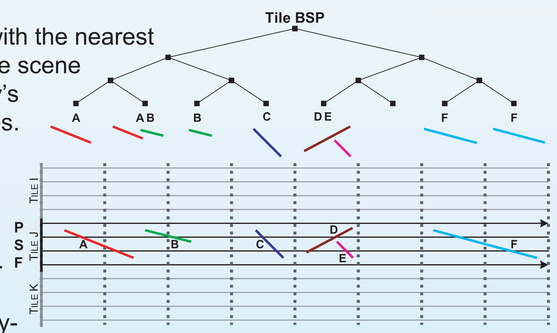


Ray Tracing Algorithm

The fundamental operation in ray tracing is to find the object with the nearest intersection along a given ray. Given an arbitrary ray through the scene we look up the closest represented direction and project the ray's end-points onto the base of its cuboid to find the intersected tiles.



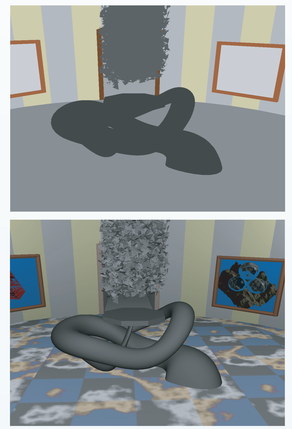
If the projected end-points suggest that the ray spans multiple tiles, these are traversed in a near to far order. These look-up operations immediately identify a very small proportion of the original scene polygons as candidates for intersection. These polygons can then be searched in logarithmic time using a 1D-BSP tree along the tile. During intersection-testing, depth values at the ray-tile crossings are used to efficiently process the polygons in the tile's 1D-BSP tree.



Acceleration Structure Update

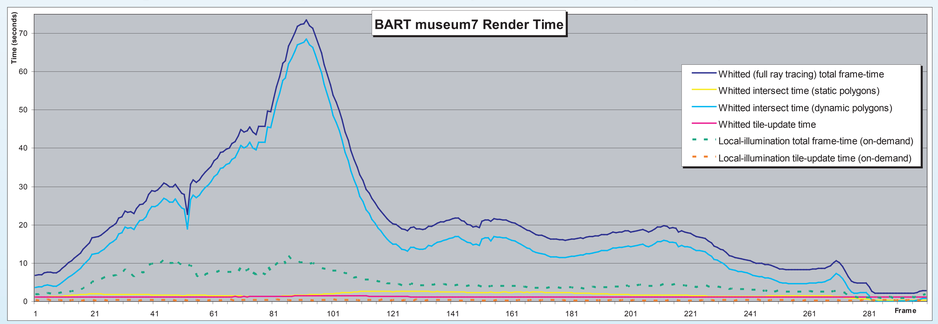
Along with static polygon identifiers written to tiles at initialisation, identifiers for moveable objects are also appended to the tiles during the animation. Following a change to the dynamic objects, a 2D rasterisation in all directions is performed and polygon identifiers are added to overlapping tiles. When a polygon is rasterised to a tile, that tile is marked as valid for the current frame. During rendering, memory corresponding to tiles invalidated for the current frame can be reclaimed. On the generation of a ray intersection task, ray intersection is first carried out with the static polygons in logarithmic search time, and then with the dynamic polygons with a linear search only if the tile is valid for the current frame, the closest intersection is reported.

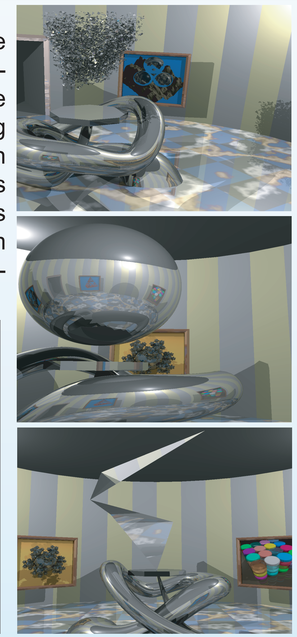
An alternative scheme for rasterising dynamic objects is to update required directions on-demand. When a ray intersection is required along a direction not updated for the current frame, that PSF is locked and rasterisation is performed. Locking delays are minimised during multi-threaded rendering by ray tracing alternate screen rows on different threads. This approach has significant speedup for simpler shading models (see right) but reduced (or negative) speedup on complex shading methods due to thread locking.



Results and Discussion

The graph below shows rendering time for the animated ray tracing benchmark scene "BART museum7" [JL01]. The scene comprises of 16K dynamic triangles undergoing unstructured motion and 10K static triangles. This requires the acceleration structure to be rebuilt for each frame; as a consequence intersection kernels requiring pre-processing cannot be efficiently used. Peaks on the graph correspond to portions of the animation where a large portion of the screen is filled by the dynamic polygons where ray tracing is slower. The graph also shows that the time to update the acceleration structure stays nearly constant, and is only a fraction of the rendering time. The images are taken from the animation and show how the triangles undergo unstructured animation, morphing between a triangle cloud into pyramid, then a sphere and finally into a planar shape.





Contribution and Future Work

This method breaks down the problem of generating an acceleration structure into very simple independent operations that can be efficiently implemented on hardware. The current implementation runs on a general purpose CPU and performs quite well. However, due to the high cost of intersecting dynamic polygons it may be beneficial to partition them into a coarse 1D-BSP tree on tile-insertion. Our main contribution is to present a novel and alternative way to think about acceleration structures that will be able to fully exploit growth in GPU performance and memory capacities in the near future. The approach works well for view-dependent global illumination applications and can be added easily to a Direction-and-Point Parameterisation [EC01] based light field. As the data structure is currently uniformly distributed, the approach is sensitive to "teapot in the stadium" type scenes. This can be solved by maintaining separate acceleration structures for complex objects in a hierarchical structure.

References

[EC01] Emilio Camahort, "4D Light-Field Modeling and Rendering", PhD Thesis, The University of Texas at Austin, May 2001.

[IW03] Ingo Wald, Carsten Benthin, and Philipp Slusallek, "Distributed Interactive Ray Tracing of Dynamic Scenes", Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003.

[JL01] Lext, Jonas, Assarsson, Ulf, and Moeller, Thomas, "BART: A Benchmark for Animated Ray Tracing", Technical Report 00-14, Chalmers University of Technology, May 2001.

[MS04] Mel Slater, Jesper Mortensen, Pankaj Khanna, and Insu Yu, "A Virtual Light Field Approach to Global Illumination", Proceedings of Computer Graphics International 2004, Greece, June 12-19, 2004.

[TL03] Larsson, Thomas, "Strategies for Bounding Volume Hierarchy Updates for Ray Tracing of Deformable Models", MRTC Report, Maalardalen University, February 2003.