Accelerating Line of Sight Computation Using GPUs

Department of Computer Science University of North Carolina at Chapel Hill August 2005

The Challenge

We present a method to accelerate line-of-sight computation for computer generated forces (CGF) using graphics processing units (GPUs). GPUs have become commodity processors and they are part of every game console or PC system. Moreover, their performance has been increasing at a rate faster than CPUs and the trend is expected to continue in the foreseeable future. We present a hybrid algorithm that exploits the computational power of GPUs to perform visibility culling and combine it with exact visibility computations on the CPU. Our approach is directly applicable to dynamic terrains. It has been applied to complex terrain environments and our hybrid algorithm is able to perform line of sight computations in a few microseconds on a commodity PC.

Approach

Line of sight (LOS) computation is essential for military simulations. These simulations can contain tens of thousands of moving entities for which LOS computations must be performed. One such system is the OneSAF war simulator. In its current implementation, LOS computation may account for 40% of the CPU time. In order to improve the performance of OneSAF it is essential to accelerate LOS computation. We use a hybrid CPU-GPU algorithm that uses the GPU to perform a conservative culling step.

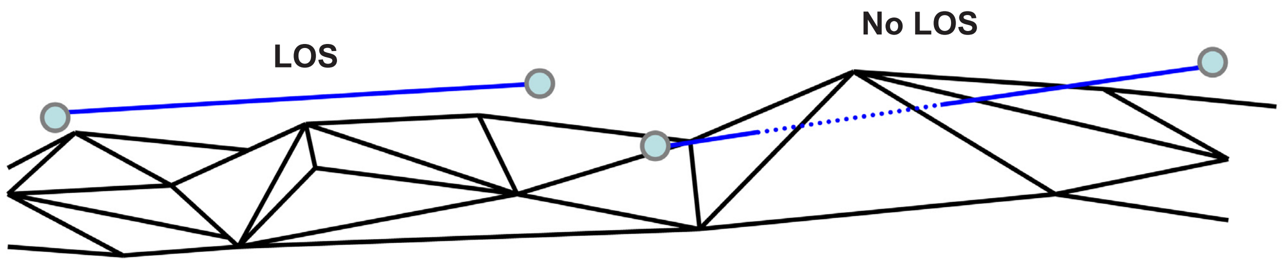
We use the GPU to quickly cull away LOS queries with a defi nite line of sight. This reduces the number of rays that must be traversed and intersected with terrain triangles on the CPU. Moreover, queries with line of sight are the most expensive for the CPU to evaluate as the full line segment between the query points must be traversed.

The algorithm works by first rendering the terrain from above orthographically. This initial rendering must be performed only once for a static terrain. Then, for each query we render a line segment between the two query points with a reversed depth test. With the depth test reversed only pixels for which the line is below the terrain will pass the depth test. Therefore, a query has LOS if no pixels pass the depth test as determined by an occlusion query (GL ARB occlusion query).

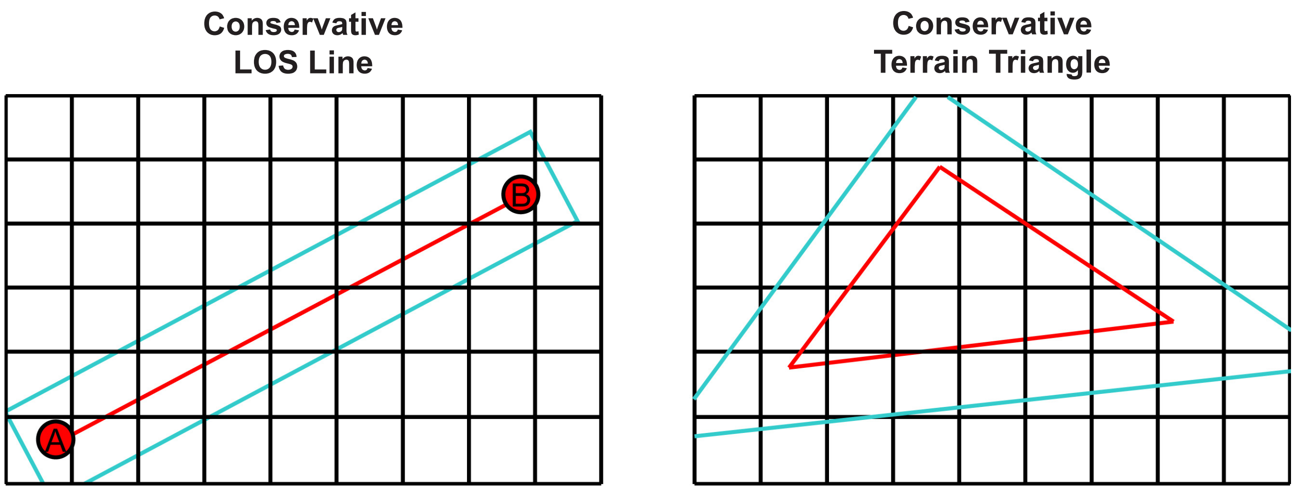
It is essential that our culling step is conservative and does not falsely cull queries because of sampling or precision errors. As in [Govindaraju et al. 2004], we use a Minkowski sum when rendering the terrain to ensure that depth values generated conservatively bound the maximum height of the terrain. Similarly, we use a Minkowski sum when rendering queries to

Highlights

- **Line of sight (LOS) is a visibility query between two entities with respect to a terrain and possibly other entities**
- **Simulations with many entities may be LOSbound; LOS can account for 40% of simulation time**
- **Improving simulation performance necessitates accelerating LOS queries**
- **We use the GPU to conservatively cull queries with definite LOS**
- **Demonstrated 200x speed up of LOS calls in OneSAF on a single CPU/GPU machine**
- **Demonstrated 15-20x overall speedup improvement in OneSAF system performance**
- **GPU LOS code transitioning into Block D Build 24 of OneSAF**
- **Will be distributed to every battlion in the Army (650 sites), every laboratory, and simulation center (150 sites)**
- **Potential for inclusion into FCS embedded training system and into UK FRES system**



A line of sight (LOS) query is simply a point-to-point visibility query with respect to a terrain. We present an algorithm to quickly perform many LOS queries for simulations that are LOS-bound.



To ensure that our culling step is conservative we compute the Minkowski sum of terrain triangles and LOS lines with a pixel-sized rectangle. This ensures that fragments are generated for all pixels through which a terrain triangle or LOS line partially covers.

ensure that the rendering of each query covers all pixels that the ray passes through.

Our hybrid GPU-CPU ray-casting algorithm has several optimizations. To perform exact tests, rays are traversed through a 2D grid imposed on the terrain. We store the maximum height of the terrain within each grid cell and only perform ray-triangle intersections for cells in which the ray falls below this maximum height. A mailboxing system is used to avoid testing a ray against the same triangle multiple times when it appears in multiple grid cells. When presented with a large query workload we attempt to utilize the GPU and CPU simultaneously. While one batch of queries is culled the non-culled queries from the previous batch are processed by the CPUs.

Project Leaders
Dinesh Manocha, professor
Ming Lin, professor

Team Member
Naga Govindaraju, research assistant professor

Graduate Research Assistants
Brian Salomon, Russell Gayle, Sung-Eui Yoon, Avneesh Sud

Other Investigators
Maria Bauer and Angel Rodriguez, RDECOM
Marlo Verdesca, Eric Root and Jaeson Munro, SAIC
Michael Macedonia, PEO STRI

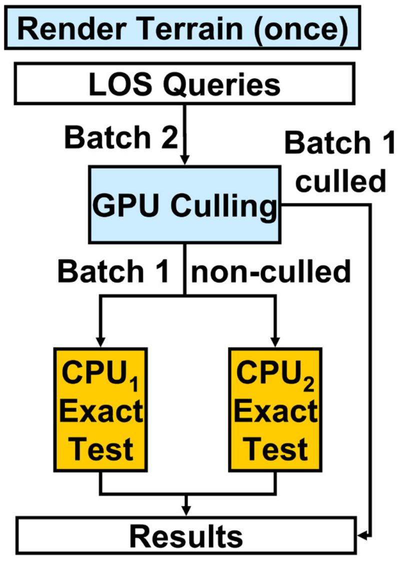
Research Sponsors
U.S. Army Research Office
Defense Advanced Research Projects Agency
Army Model and Simulation Office

Selected Publications
Govindaraju, N., S. Redon, M. Lin, and D. Manocha. CULLIDE: Interactive collision detection in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2003, 25-32.

Govindaraju, N., M. Lin, and D. Manocha. Fast and Reliable Collision Culling using Graphics Processors, *Proc. of ACM VRST*, 2004.

Key Words
GPU algorithms, line of sight, raycasting

For More Information
http://gamma.cs.unc.edu



By batching LOS queries we are able to hide the latency of GPU occlusion queries and utilize the CPU and GPU simultaneously. While one batch is culled, the non-culled queries from the previous batch are processed by the CPU.