ACCELERATING LINE OF SIGHT COMPUTATION USING GRAPHICS PROCESSING UNITS

Brian Salomon[1], Naga Govindaraju, Avneesh Sud, Russell Gayle, Ming Lin, and Dinesh Manocha
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599

Brett Butler
SAIC
Orlando, FL 32826

Maria Bauer*, Angel Rodriguez, Latika Eifert, and Audrey Rubel
RDECOM
Orlando, FL 32826

Michael Macedonia
PEO/STRI
Orlando, FL 32826

ABSTRACT

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.

1 INTRODUCTION

Computer Generated Forces (CGFs) are computer systems that emulate the battlefield entities and units whose tactical behaviors and decisions are either made in part by human operators (Semi-Automated Forces) or automated decision algorithms (Automated Forces). A number of products have been developed to support Army applications in three modeling and simulation domains: Training, Exercise, Military Operations (TEMO); Advanced Concepts and Requirements (ACR); and Research, Development and Acquisition (RDA). Over the last few years, some of the major efforts have been directed towards Semi-Automated Forces (OneSAF) operational requirements. The OneSAF refers to a composable, next generation CGF that can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level, with a variable level of fidelity that supports all models and simulation (M&S) domains.

Any entry level simulation requires interaction between the entities. To represent the effects of terrain, it requires some method for computing an unbroken geometric line of sight (LOS) between any two given entities or locations. Consider a simulation of a battle between a force of x entities and a force of y entities. Let n be the total number of entities simulated. To a large extent the events in the simulation will be driven by detections, which will be predicated on line of sight. This requires a poll of each entity for its current line of sight status to every other entity at each time step or event in the simulation. As the number of entities increases, this problem grows with the square of the number of entities (i.e. $O(n^2)$). Hence, for practical purposes current simulations use filters and heuristics to decrease the number of instances in which the full LOS calculation must be invoked. In other words, the simulation developer must reduce, by some reasonable means, the number of entities that require resolution of the line of sight question for a given situation or time step. Otherwise, the simulation becomes overburdened with solving the line of sight question between every sensor-target pair. The simplest heuristics either assume a maximum detection range or eliminate sensor-target pairs whose range exceeds this maximum and they are

[1] Contact: salomon@cs.unc.edu

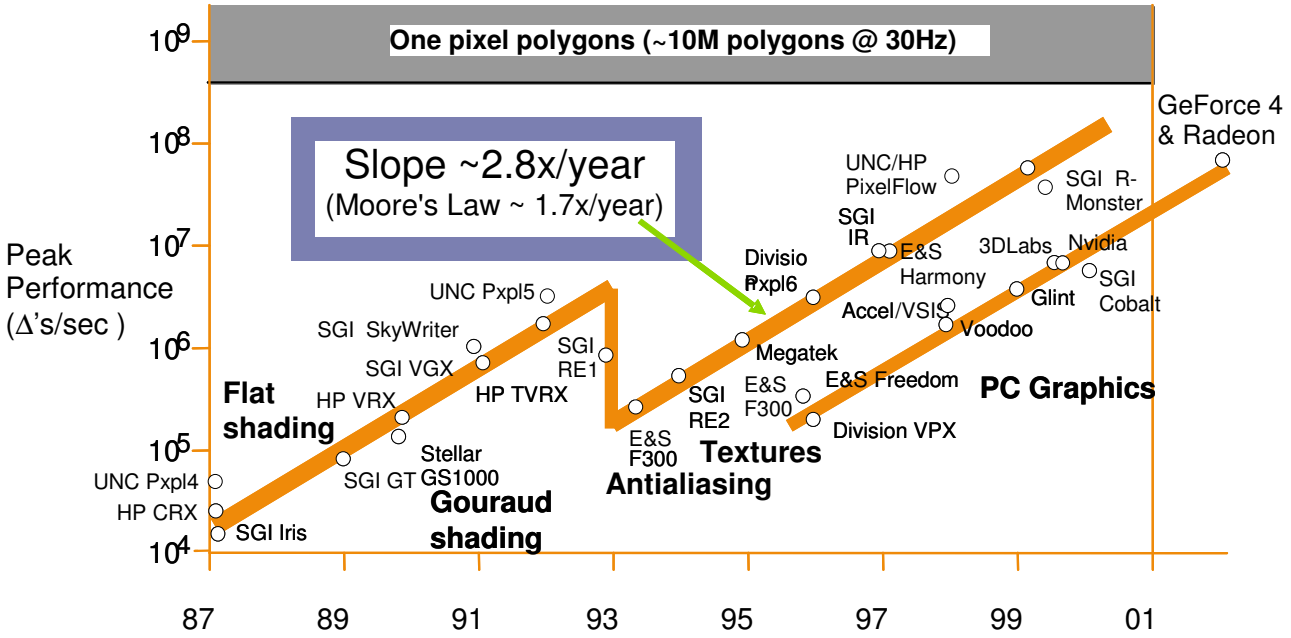

Fig. 1: *This graph highlights the growth of GPUs over the last 14 years. Note that the performance of GPUs is going up a factor of almost 3, whereas according to Moore's law, the CPU doubles in speed every 18 months. The growth rate of GPUs is expected to continue for the next 5 years. Source: John Poulton.*

used in DYNATACS, ModSAF, JANUS, and CASTFOREM (Henderson, 1999). Moreover, it is hard to predict the error introduced by these heuristics on the resulting simulation.

In practice the LOS queries can take a significant fraction of overall simulation time. In particular, terrain reasoning services consume a significant portion of computing resources in Modeling and Simulation (M&S) applications. For example, in the OneSAF Objective System (OOS), 41.95% of the available CPU can be allocated to the dynamics agent with 40% of this sub-allocated to collision detection and an additional 40% to terrain placement. Sensor agents could be allocated 45.0% of the CPU with 60% sub-allocated to LOS or 27% of the total CPU. In other words, more than 55% of the available CPU could be allocated to three key components, including terrain placement, collision detection and line-of-sight computation, leaving just over 45% for cognitive models. This constraint can severely limit the entity count sustainable by the simulation system. Reducing LOS computation will allow an increased number of entities or allow the use of more complex cognitive models.

2 GRAPHICS PROCESSORS (GPUS)

Fast graphics hardware including 3D rasterization, texturing, and dedicated vertex and pixel processing has become as ubiquitous as floating-point hardware. It is nearly impossible to buy a PC without dedicated 3D rasterization and texturing hardware. Moore's Law, the highly parallel nature of graphics rendering algorithms, and the computation demands for simulating visual reality have converged to drive the development of faster and more capable graphics hardware. The ubiquity and performance of this hardware leads us to consider the extent to which this hardware can be harnessed to solve scientific, simulation and visibility problems beyond the conventional domain of image synthesis for the sake of pretty animation.

The graphics processing units (GPUs) have been progressing at a rate faster than Moore's Law. The progress over the last decade years has been shown in Fig. 1. Notice that the performance of GPUs has an average slope of 2.8X, whereas the CPUs have improved in performance by 1.7X over the same time period. Moreover, the same growth rate is expected to continue for the next 3-5 years, giving us the capability to perform GFlops of computation on a $350 COTS GPU. This makes the GPU an excellent candidate for performing scientific, geometric and compute-intensive computations. Moreover, the GPUs have become programmable over the last few years. The new languages and compilers for programming the GPUs make it much easier to use them for different application.

One of the first GPUs was the Geometry Engine (GE) proposed by Clark in the early 1980's. It was

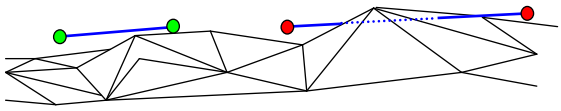

Fig 2: *Line of sight is a simple query that determines whether there is an unobstructed view of one entity from another. In this figure the entities on the left do have line of sight while those on the right do not.*

fabricated using a 3µm feature size and housed in a 40-pin package. Things have progressed steadily over the last two decades and different features like smooth shading, anti-aliasing, textures and programmable shading have been improved. Details of different GPUs are shown in Fig. 1 over the last two decades. Compared to the first GE, a recent GPU like NVIDIA's GeForce4 was manufactured using a 0.13µm process with a 550-pin package. Its peak performance, in terms of number of triangles per second, is more than four orders of magnitude higher as compared to the first GPU released about 15 years ago. Furthermore, it has greatly increased capabilities. The current GPUs are optimized for rasterization of 3D geometric primitives. They can also be regarded as an efficient processor of images. Moreover, the vertex and pixel shaders provide the application programmer a great deal of flexibility and power. Because of these capabilities, an incredible array of new algorithms and real-time implementations of old algorithms have been made possible. Moreover, as graphics hardware becomes more programmable, the barrier between the CPU and the GPU is being redefined. The new languages and compilers for programming the GPUs make it much easier to use them for different applications. Finally, the underlying precision of the GPUs is increasing as well. Current GPUs can support 32-bit floating point frame-buffers.

3 LINE OF SIGHT

A line of sight query determines whether two entities are mutually visible (Fig. 2). To answer an LOS query the ray between the entities positions must be tested for intersection with the terrain and any obstacles such as buildings or vegetation. Although it is a relatively simple query, it can be expensive to support line of sight queries between many entities. The computation to resolve all pairwise entities can grow as $O(n^2)$ where n is the number of entities. A 50,000 entity simulation would require over a billion LOS queries to determine LOS between all entity pairs. Furthermore, acquisition technologies keep improving leading to increasingly detailed terrain representations (Fig. 3).

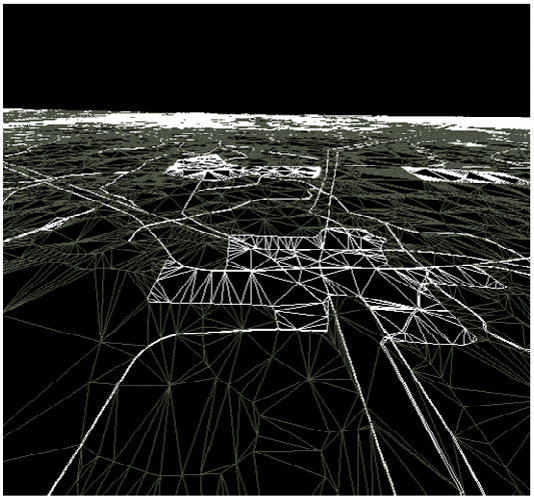

Fig 3: *The JRTC terrain. This terrain contains approximately 1M triangles and represents an area about 100km by 100km.*

4 HYBRID GPU-CPU LOS ALGORITHM

We use a hybrid CPU-GPU algorithm which uses the GPU to perform a conservative culling step. We use the GPU to quickly cull away LOS queries with a definite 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 more 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 (GL_GREATER). 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).

4.1 Conservative Rasterization

It is essential that our culling step is conservative and does not falsely cull queries because of sampling or precision errors. As in (Govindaraju, 2004), we use a Minkowski sum when rendering the terrain to ensure that our culling step is conservative (Fig. 4). Similarly, we use a Minkowski sum when rendering queries to ensure that the rendering of each query covers all pixels that the ray passes through. These sums are performed with a box that has dimensions equal to a pixel's dimensions in the world space. This insures that

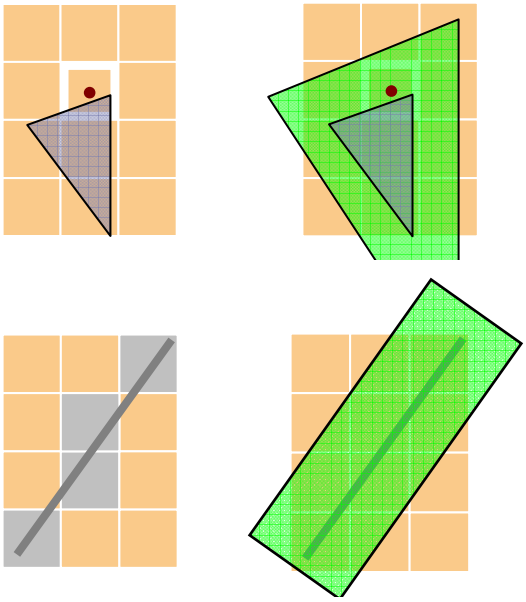

Fig 4: *Conservative rasterization of terrain triangles and LOS rays. We ensure that each pixel intersected by a terrain triangle or LOS ray generates a fragment with a conservative depth value.*

all pixels partially overlapped by a terrain triangle or an LOS line segment are rasterized. The box used in the sum has a depth equal to the depth buffer resolution. This ensures that the depth value generated for each pixel bounds the highest portion of the terrain triangle under that pixel. When rasterizing LOS rays we guarantee that the generated depth value bounds the lowest point along the ray under the corresponding pixel.

4.2 Implementation

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 intersects 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 (Fig. 5).

5 RESULTS

We have tested our initial implementation on an approximately 1M triangle terrain. The terrain region is nearly 400km on each side. The point pairs in our

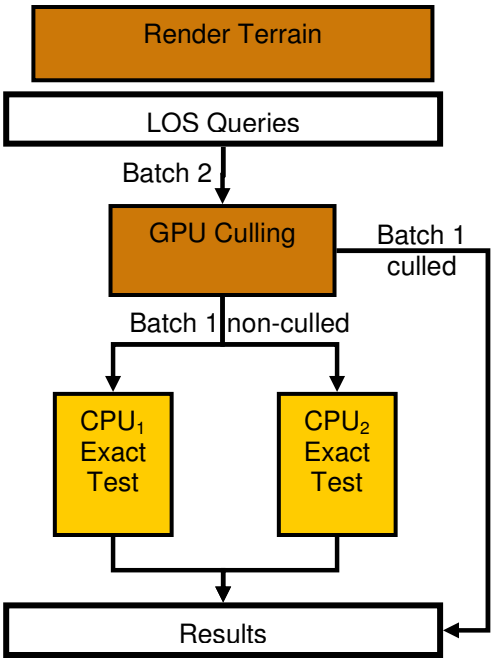

Fig 5: To utilize both the GPU and CPU(s) simultaneously we batch LOS queries. Culling is performed on one batch of queries. Then while exact tests are performed on the non-culled queries, the second batch is culled.

benchmark queries have a maximum separation of 4km and are offset 3m from the surface.

Our hybrid CPU-GPU algorithm averages 4µs per query using an Nvidia 6800 Ultra GPU and a Pentium 4 3.4GHz CPU.

6 FUTURE WORK

Our current algorithm provides a constant time speedup in LOS computation. For complex scenarios with tens of thousands of entities performing LOS queries for each entity pair is prohibitively expensive. We are currently developing GPU-based algorithms to perform hierarchical culling on entity clusters to reduce the $O(n^2)$ complexity of this problem. Furthermore, we propose to use a GPU-cluster to accelerate these computations, similar to use of multiple CPUs (Messina, 1999).

7 CONCLUSIONS

We believe that exploiting the computational power of GPUs is essential to increase the complexity of simulations that can be performed in real-time for computer generated forces using systems such as OneSAF. Our algorithm for accelerating LOS using the

GPU is one example. In many simulations, LOS computation is often the single most expensive simulation computation.

Our algorithm for LOS acceleration uses the GPU to reduce the number of queries processed by the CPU. We use a conservative rasterization based on a Minkowski sum and hardware occlusion queries to perform culling. Queries passing the GPU-based LOS test have a definite LOS and the remaining queries are tested for LOS by a CPU-based ray casting algorithm. In a batched mode we can perform culling while the CPU is used to perform ray-casting tests on non-culled queries.

Our algorithm is currently integrated into OneSAF which will reduce the time of LOS computation allowing more complex scenarios to be run. Warfighters rely on decisions based on the outcomes of such scenarios. Thus, they directly benefit from more accurate and realistic simulation scenarios.

ACKNOWLEDGEMENTS

We would like to acknowledge AMSO and DARPA Contract N61339-04-C-0043.

REFERENCES

Henderson, D. L., 1999: Modterrain: A proposed standard for terrain representation in entity level simulation, MS thesis, Naval PostGraduate School.

Messina, P., et al., 1999: Synthetic Force Express: A New Initiative in Scalable Computing for Military Simulation.

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

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