Using Graphics Processor Units to Accelerate OneSAF: A Case Study in Technology Transition

Marlo Verdesca,
Jaeson Munro, Michael Hoffman
Science Applications International Corporation
Orlando, FL
Marlo.K.Verdesca@saic.com, Jaeson.Munro@saic.com, Michael.R.Hoffman@saic.com

Maria Bauer
RDECOM
Orlando, FL
Maria.Bauer@us.army.mil

Dinesh Manocha
University of North Carolina at Chapel Hill
Chapel Hill, NC
dm@cs.unc.edu

ABSTRACT

On-going research aims to accelerate the runtime processing speed of the One Semi-Automated Forces (OneSAF) Computer Generated Forces (CGF) simulation by converting and migrating some of the core algorithms from the host Central Processing Unit (CPU) to an on-board auxiliary Graphics Processor Unit (GPU). In this research the GPU chip is regarded as a surrogate stream processor and appropriate algorithms are designed to map to the GPU architecture. Processing speed gains are realized both through computational capabilities of the GPU as well as through offloading of the host CPU. Technology transfer of this research into the OneSAF user baseline is a key requirement of this research.

The OneSAF development program focuses on the same issues of scalability and runtime performance that will be directly affected by use of GPUs. As program architects are marshalling conventional approaches for resolving these challenges, the introduction of GPU-based solutions is being realized. This paper examines the challenges, planned approaches and benchmarked results for using GPUs to accelerate OneSAF simulation.

ABOUT THE AUTHORS

Marlo Verdesca is program manager for the OneSAF Objective System (OOS) GPU research project. She has led development efforts for OneSAF Testbed Baseline (OTB) and its predecessors, Modular Semi-Automated Forces (ModSAF). Marlo is a software engineer at Science Applications International Corporation (SAIC) and holds a Bachelor of Science in Management Information Systems at the University of Central Florida.

Jaeson Munro is lead developer for the OOS GPU research project. He has worked on development on the Environment Runtime Component (ERC) for OOS, which included leading the development of the Ultra-High Resolution Building Editor for OOS. Jaeson is a software engineer at SAIC and holds a Bachelor of Science in Computer Science at the University of Central Florida.

Michael Hoffman is a developer for the OOS GPU research project. Michael is a junior software engineer at SAIC and holds a Bachelor of Science in Computer Science at the University of Central Florida.

Maria Bauer (US Army RDECOM-STTC) is the Principal Investigator for the Defense Advanced Research Projects Agency (DARPA) sponsored GPU research project and the Computer Generated Forces Scalability Advanced Technology Office (ATO). Experience includes fifteen years of software engineering and five years of program management working with Department of Defense (DOD) military acquisition systems and simulation technology in Army programs. She holds a BSEE from the University of Miami, a MS in Engineering Management from the University of Central Florida and a PhD in Industrial Engineering.

Dinesh Manocha is currently a professor of computer science at the University of North Carolina at Chapel Hill. He received his B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi in 1987; M.S. and Ph.D. in computer science at the University of California at Berkeley in 1990 and 1992, respectively. He is currently leading a large research group on use of GPUs for simulation, database computations and geometric algorithms.

Using Graphics Processor Units to Accelerate OneSAF: A Case Study in Technology Transition

Marlo Verdesca, Jaeson Munro, Michael Hoffman
Science Applications International Corporation
Orlando, FL
Marlo.K.Verdesca@saic.com, Jaeson.Munro@saic.com
Michael.R.Hoffman@saic.com

Maria Bauer
RDECOM
Orlando, FL
Maria.Bauer@us.army.mil

Dinesh Manocha
University of North Carolina at Chapel Hill
Chapel Hill, NC
dm@cs.unc.edu

INTRODUCTION

Recent advancements in Graphics Processing Unit (GPU) technology have spurred an interest within gaming environments and personal computing systems. Compared to the Central Processing Unit (CPU), GPU performance has been increasing at a much faster rate. But, offloading extensive CPU based algorithms onto the GPU within the One Semi-Automated Forces (OneSAF) simulation is fairly new and experimental.

OneSAF is a composable simulation that is capable of modeling a range of entities from individual combatants (IC) to platforms. It allows operators, through Graphical User Interfaces (GUI), to compose entities, units, sophisticated behaviors and scenarios at various levels of fidelity. OneSAF provides the capability to effectively and accurately represent warfare, communications, combat support and combat service support, currently focused on land warfare. (OneSAF, 2005)

During a large scale simulation training exercise it is possible to have multiple OneSAF machines networked together simulating thousands of entities on a single battlespace. Within the exercise, entities are constantly performing complex and expensive line of sight (LOS) queries consuming significant amounts of process time. OneSAF studies have indicated that more than 55% of available CPU usage is allocated to three key functions: terrain placement, collision detection and LOS computations, leaving just 45% for cognitive models and other functions. As training environments create military-realistic exercises, more and more entities need to be simulated and as number of entities increase, so do the number of LOS queries, thus creating an $O(N^2)$ problem (Salomon et al, 2004).

Route planning has similar computational issues within OneSAF. As entities plan routes from a starting point to destination point, the expensive A-Star (A*) routing algorithm evaluates possible route segments based on their cost. A cost is assessed on specified criteria such as terrain feature intersections, trafficability, and shortest distance. A lower cost indicates a more favorable route (Condon, 2002). However, as OneSAF matures and higher fidelity IC models traverse through complex urban environments that contain Ultra High Resolution Buildings (UHRB), this can lead to runtime impacts on the system.

New GPU-based LOS and route planning algorithms have been created to take advantage of the GPU technology. The new technology has been able to offload some of this process time from the CPU and allow OneSAF to raise entity count, use higher fidelity models and behaviors, or intensify terrains to use complex urban environments that handle UHRBs.

The goal of this research project is to accelerate overall system performance by focusing on major process restrictions within OneSAF, such as LOS and route planning algorithms; while taking advantage of commercial off the shelf (COTS) hardware such as GPUs. This paper will examine the current OneSAF LOS and route planning algorithms and the newly created GPU-based algorithms. It will discuss the benchmarked findings that were performed while looking at future efforts to make the GPU technology an integral part of OneSAF.

GPU TECHNOLOGY

In recent months, video game enthusiasts around the world were privy to details about the latest upcoming consoles such as Sony's Playstation 3 and Microsoft's

Xbox 360. Both systems utilize high performance GPUs and CPUs. Graphics processing units have become an essential part of every game console or PC system today. Currently, the GPU is mainly utilized for rasterization of 3D primitives, which are often strenuous on the CPU. However, as more complex computations are evaluated on the graphics hardware, a significant increase in computer performance can be achieved by developing GPU-based algorithms. With the development of programmable GPUs came new languages and compilers, both of which permit potentially new applications to be executed. Furthermore, as hardware performance continues to increase, GPUs have the potential to alleviate already overwhelmed CPUs by performing common tasks, such as scientific computations, simulation and visibility problems, thereby allowing the CPU to be used for other tasks.

Processing Speed Gains

With the barrier between the CPU and GPU diminishing, a co-dependency is emerging between the two, and the processing speed gains are becoming apparent. Any sort of computation on a collection of data, such as a military simulation, where elements are continuously interacting with each other and their environment, can be extremely efficient on a GPU. Scientific computations such as linear algebra, Fast Fourier Transform, and partial differential equations can also benefit greatly from graphics hardware computations. Any and all computations that take advantage of the parallelism and pipelining on the GPU will see a significant increase in CPU resource availability. For example, the high-computational output of the GPU enables parallel sorts and searches to occur 4-5x faster[1] than the Pentium 4 CPU, while other estimates on highly complex simulations show that the GPU can be from 10-100x faster (Stam, 2003).

CPU / GPU Comparison

GPU research indicates that graphics hardware is affordable and offered within virtually every computer system available today. However, an even more important issue is the comparison in performance results. Figure 1 illustrates how graphics hardware has been evolving faster than the CPU, at least doubling every six months, a rate faster than Moore's law.

[1] http://gamma.cs.unc.edu/GPUSORT

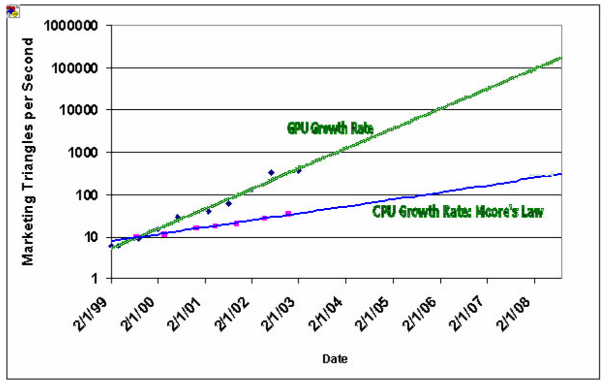

Figure 1: GPU / CPU Growth Rate

This trend is expected to continue for the next five years and has contributed to the operating conditions mentioned in the table below (Manocha et al, 2004). Table 1 examines both memory bandwidth and computational performance between the CPU and GPU.

Table 1: Example of CPU / GPU Comparison

	CPU	GPU
Memory Bandwidth	6.4 GB/s main	35.2 GB/s *
Peak Computational Performance	6 GFLOPS **	48 GFLOPS ***

* Comparable to the CPU L2 cache bandwidth
** 3.2 GHz Pentium 4 SSE Theoretical
*** GeForce FX 6800: Equivalent to a 24 GHz Pentium 4

Memory bandwidth affects the performance of the algorithm as it fetches new data from the memory and having a higher memory bandwidth results in faster performance. The relatively slow memory bandwidth within the CPU-memory interface is the classic Von Neumann bottleneck. The Computational performance shows that the peak performance of GPUs can be higher than CPUs for certain applications (Manocha et al, 2005).

While memory bandwidth and computational performance are obviously larger on a GPU, it does have its drawbacks. For example, program architects must consider how their application will be supported in future GPU research and implementations. There are no set standards that can be followed when designing advanced graphics related algorithms, which can leave a lack of certainty that the research will be accepted by a technologically knowledgeable audience. The goal is to maintain compatibility with not only the ever-changing technology but also in understanding the compatibility between CPU and GPU.

GPU and Beyond

The current state of developing GPU-based algorithms is an active area of research. Over the last few years GPU related research initiatives have exploded onto technical conferences and universities across the world. Topics like high-precision computations, utilizing the full capabilities of parallelism and high memory bandwidth of GPUs, and trying to speed up the data port between the GPU and CPU, are all becoming synonymous with the simulation and graphics communities (Manocha et al, 2005). In addition to those research areas, new performance modifications, including improved precision, programmability, rasterization performance, occlusion queries and the overall architecture of the GPU, will give GPUs an enormous push in the near future. PCs with multiple GPUs or networked GPU clusters will expand the potential of these enhancements.

To further the gap between what a single CPU can handle and how offloading computations onto an onboard graphics related hardware unit can unleash CPU resources, AGEIA Technologies Incorporated introduced the Physics Processing Unit (PPU). The PPU was developed to maintain fluid dynamics, universal collision detection, rigid-body dynamics, and smart particle systems to name a few. It is meant to be a physics accelerator chip and is an initial push into hardware-accelerated physics (Cross, 2005). Together the GPU and PPU will push the limits of performance related issues and the capabilities of simulations as known today.

By exploiting the computational abilities of GPUs, simulations such as OneSAF are able to increase complexity while maintaining real-time performance.

LINE OF SIGHT

As thousands of entities are simulated within OneSAF training exercises, complicated LOS algorithms are constantly being performed with resultant slowdowns in runtime performance. One goal of the GPU project is to integrate a GPU/CPU algorithm to effectively accelerate the overall system speedup of OneSAF while simulating 5,000 + entities.

OneSAF LOS

For terrain surface queries, the geometric line of sight algorithm traverses terrain triangles along a line of sight segment. At each triangle, this algorithm checks for intersections with the LOS segment. For each triangle where line of sight remains unblocked, the algorithm continues to the next triangle. This traversal stops when either the end of the segment has been reached, or LOS is blocked and the intersecting triangle is returned (See Figure 2).

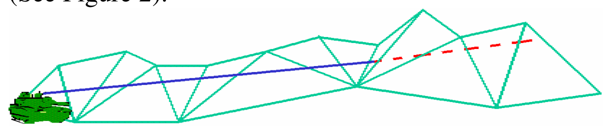

Figure 2: LOS (solid indicates visible; dashed, blocked)

The *get_first_triangle* method finds the first triangle that a segment traverses. The *get_next_triangle* method finds the triangle that shares a given segment with a given triangle. These *get_first_triangle* and *get_next_triangle* routines are the fundamental steps in retrieving and traversing the triangles during a LOS query. At each iteration, a bounds check is performed against the elevation of the triangle. If the segment is within the bounds at this triangle location, then a full ray-triangle intersection test is performed (Polygon Traversal, 2005). The full ray-triangle pseudo-code is listed as reference in Figure 3.

```
not_done = true
los_exists = true

get_first_triangle(segment, triangle)

while (los_exists and not_done)
   if (areal_feature or linear_feature) then raise_triangle_vertices
   if (elevations of triangle vertices < elevations of segment endpoints)
      // bounds check for efficient pruning
      los_exists = true
   else
      los_exists = not intersects(ray, triangle)
   while (triangle_point_feature_list not empty and los_exists)
      los_exists = not intersects (ray, point_feature_bounding_volume)
   not_done = get_next_triangle(segment, triangle)

return los_exists
```

Figure 3: Full Ray-Triangle Pseudo-Code

Clearly, the best case for performance here would be a segment that is intersected by the first triangle that it traverses, because it needs to only check one triangle. Conversely, a segment that is never intersected is the worst case for performance.

GPU LOS

The University of North Carolina (UNC) provided a hybrid GPU/CPU algorithm which performs conservative culling in the GPU portion of the algorithm. LOS queries whose segments are definitely unblocked are quickly culled away by the GPU, thereby reducing the number of segments that must be tested by traversing the terrain triangles while performing this intersection check with the CPU. As stated before, queries with unblocked line of sight are most expensive for the CPU. This actually becomes the best performance case for the hybrid algorithm, as these calls are likely to be returned after the culling step (Salomon et al, 2004).

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 a line segment is rendered 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) (Salomon et al, 2004).

Several optimizations have been made to the hybrid ray-casting algorithm. While performing exact tests, rays are traversed through a 2D grid representation of the terrain. The maximum height of the terrain for each cell is stored in the grid, so that a ray-triangle intersection check only becomes necessary for cells in which the ray is below the maximum height. The algorithm also incorporates a mailboxing system, which avoids testing a ray against the same triangle multiple times when it intersects multiple grid cells. When working with a large number of queries, the GPU and CPU can be performing LOS computations simultaneously. While culling one batch of queries with the GPU, the CPU is processing the non-culled queries from the previous batch (See Figure 4) (Salomon et al, 2004).

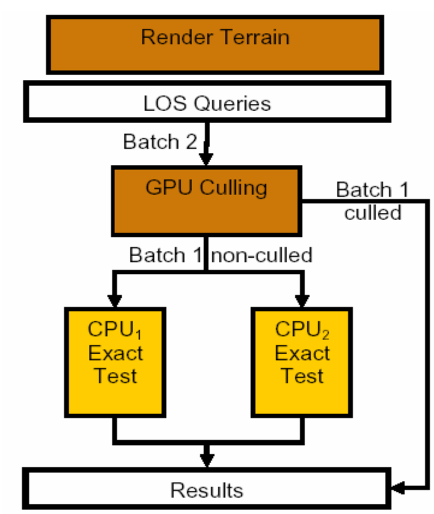

Figure 4: Batch 1 Produces Culled and Non-Culled Queries

LOS Results

The OneSAF LOS scenario that is shown in Figure 5 demonstrates real-time GPU-based algorithms being performed within OneSAF. This figure illustrates separate engagement areas consisting of 4 medium-resolution rotary winged aircrafts (RWA) performing complex LOS queries on approximately 5,000 low-resolution tanks on the OneSAF plan view display (PVD).

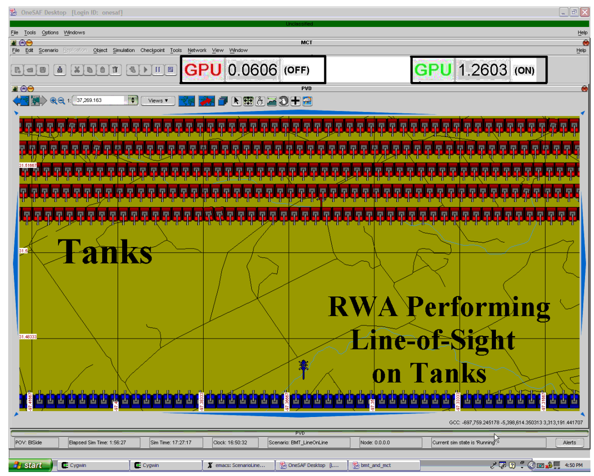

Figure 5: LOS Scenario with 5,000 Entities
(Build 24 of Block D)

As the RWAs are flying toward enemy tanks the GPU/CPU hybrid algorithm is being used to perform LOS queries. The GPU ratios [appearing on the Tool Bar of the PVD] represent *simulation time / real time*, while real time is constant and simulation time is based on the computational performance of OneSAF. The GPU 'OFF' button represents the ratio in which original OneSAF LOS calls are being performed. The GPU 'ON' represents the ratio in which the GPU/CPU hybrid algorithm is being performed. With respect to the magnitude of these values, higher numbers represent the scenario executing at a quicker rate.

After executing the scenario approximately 30+ times, the GPU ratios were compared and conclude that using the GPU algorithms within OneSAF produced an average overall system performance increase of 20x (See Figure 6).

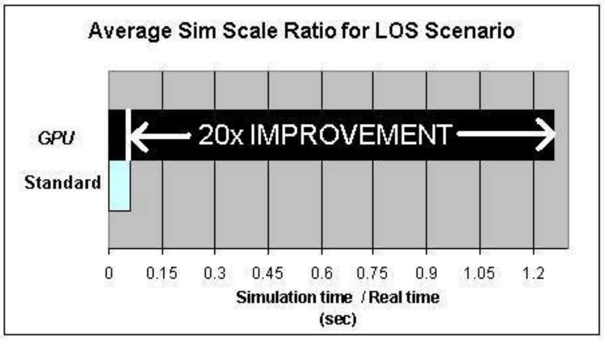

Figure 6: LOS Benchmark Data

The LOS calls alone improved from an average of 1000 microseconds without the GPU functionality to 12 microseconds with the GPU, a 100-200x improvement. The improved performance relative to our current proof of concept scenarios presented in November 2004 and May of 2005 are shown in Table 2. This improvement demonstrates that the GPU-based algorithms have the potential to steadily improve performance as the terrain becomes more complex and the amount of entities is increased.

Table 2: History of Performance Increase

Date	Number of Entities	Terrain	Overall System Performance Increase
Nov '04	400 Low Resolution M1A1 tanks	JRTC	**2x**
May '05	2934 Low Resolution M1A1 tanks 66 Medium Resolution AH-64 RWAs	JRTC	**10x**
Aug '05	4996 Low Resolution M1A1 tanks 4 Medium Resolution AH-64 RWAs	Ft. Hood	**20x**

ROUTE PLANNING

There are three basic types of routes within OneSAF: Direct, Networked and Cross Country. Direct routes follow waypoints exactly as entered by the operator and are faster than any other route type since a cost function is never called. Networked routes follow linear features such as rivers and roads and cross country routes utilize a grid of routing cells that form an implicit network for the A* algorithm to search (Condon, 2002). As units and entities route plan and traverse over dense urban terrains they execute the expensive A* algorithm which performs multiple feature checking. This computationally intensive algorithm has been shown to consume a great deal of OneSAF processing time.

Future terrain environments expected for OneSAF will contain large areas and high building densities. This can cause route planning to be a challenge. Overall system performance will be impacted, and entity-level route planning will deal with increased amounts of intersection checking against buildings and their interiors. The second goal of the GPU project is to integrate a GPU-based algorithm to effectively accelerate both feature intersection checking and overall system speedup of OneSAF.

OneSAF Route Planning

To determine a route of least cost, the Environmental Runtime Component (ERC) first creates a network of route nodes. An A* algorithm is implemented to traverse through the nodes, and determine a cost for each segment visited, which ultimately finds the route of least cost from the starting point to the end point.

The cost of a particular segment is computed by a cost function that has been selected by the user. These cost functions need to know which terrain features are intersecting a segment. Checking a segment for intersecting terrain features is the performance bottleneck for most route planning scenarios. This process is broken down into two routines: "Feature Read," which retrieves a list of features in the surrounding area of a node, and "Feature Analysis," which determines which of these features intersects a given segment associated with the node (See Figure 7).

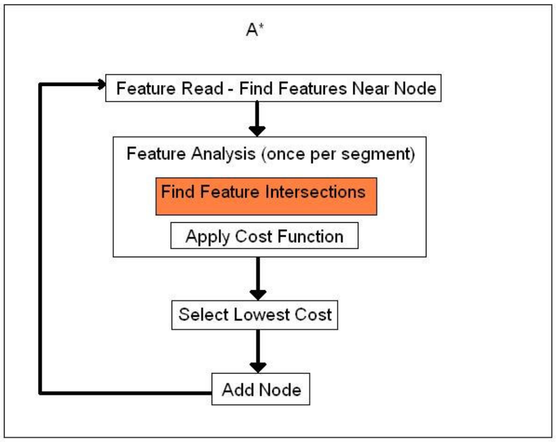

Figure 7: Operations Performed by A* within OneSAF

The 'Find Feature Intersection" box highlighted above, represents the portion of the routing algorithms that were replaced with GPU computations.

Feature Read

Each node in a route network has a slice associated with it. A slice is any specified area between minimum and maximum latitude, and a minimum and maximum longitude. No bounds are given to the elevation of a slice. The bounds of this slice are determined by expanding the node coordinate by the specified grid spacing in the positive and negative lat/long directions. The same node slice is used to evaluate all possible route segments leaving from this node (See Figure 8).

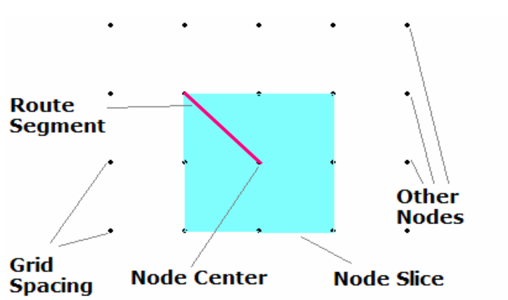

Figure 8: Size of a Node Slice Relative to Grid Spacing

Grid spacing is passed into routing calls by the user. The features that are considered to be within the slice of a node are determined during the "Feature Read" portion of the routing. This process involves first determining which "pages" overlap the node. A page is a fraction of a geotile. A Geotile is defined as a region 1 degree by 1 degree - approximately 10,000 square kilometers, and in our current case, our page is 1/400 of one geotile. Pages are defined at compile time, and their boundaries are static. A 2-D containment check on all the features in the overlapped pages is performed to determine which features are near this node. A feature is included if it passes the containment check for the node slice.

Feature Analysis

For a potential route segment, features are classified by traversing the list of features in the node slice and finding all features in the list that actually intersect the segment. This intersection checking is the "Feature Analysis" portion of the routing.

During intersection checking, each feature is classified as 'circle,' 'linear,' or 'areal.' For a circle, both endpoints of the segment are checked to determine if the circle contains them. If the circle contains neither, an intersection occurs if the line creates a chord in the circle. For a linear feature, a segment intersection check is performed with the center lines of the line segments that make up this feature. For an areal feature, it is first determined if the polygon for this feature contains the first point of the segment If it does not, then the segment intersection check is performed for every edge that defines the polygon for this feature.

GPU Route Planning

As stated earlier, the performance bottleneck in ERC route planning lies within the process of checking potential route segments for intersections with terrain features. GPU-based algorithms were developed not to replace the route planning routines as a whole, but rather to only replace the routines that check segments for intersections.

The GPU-based algorithms are given a list of features and a segment to check for intersections. Similarly to the way in which the GPU-CPU hybrid works for LOS, the GPU uses conservative culling to eliminate many of the features in this list, leaving a much smaller list of features to check with the CPU. Figure 9 displays how the GPU-culling proceeds in three phases: (Lin et al, 2005)

- The number of segments is reduced by culling them against the full feature set.
- The number of features is reduced by culling them against the reduced number of segments
- The reduced feature set is culled against each individual segment in the reduced segment set.

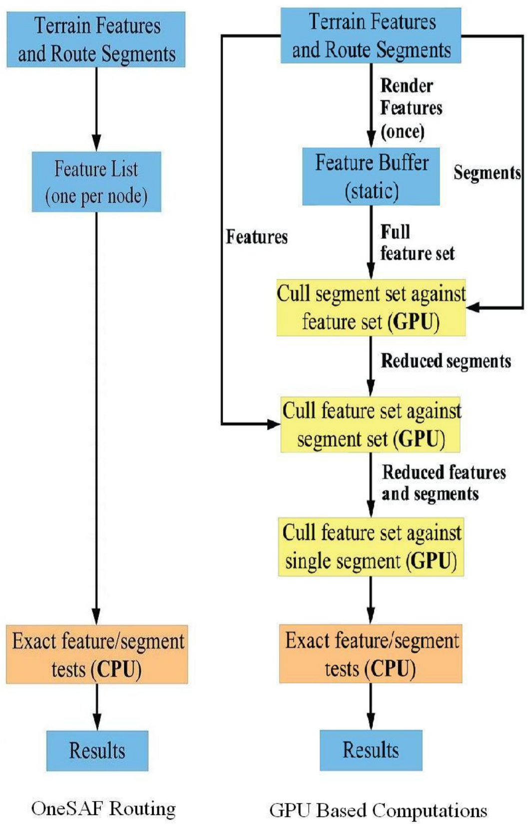

Figure 9: Comparison between OneSAF and GPU Route Planning

The figure also presents the difference between OneSAF and GPU-based route planning flow.

The new algorithms for feature intersection checking were also able to check multiple segments in parallel. Through some modifications to the existing A* algorithm in ERC, the GPU-CPU hybrid routine was run on all segments stemming from a single node at once.

Route Planning Results

The OneSAF route planning scenario that is shown in Figure 10 demonstrates real-time the GPU-based algorithms being used within OneSAF. The figure illustrates a medium resolution M1A1 tank platoon and multiple IC's tactically traveling through the dense urban environment of Ft. Hood, Texas. The units must first perform necessary complex and time intensive route planning algorithms to determine their route.

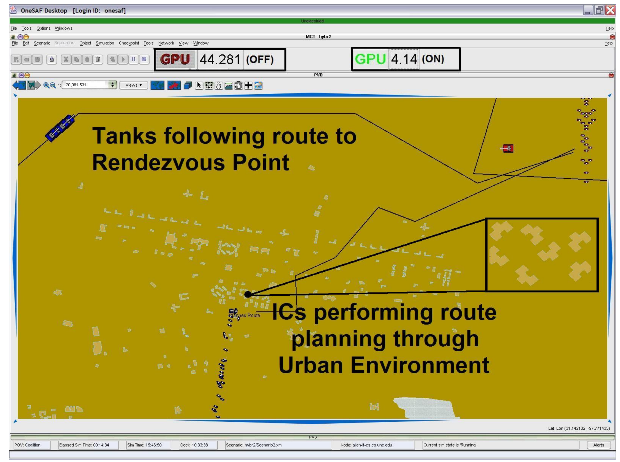

Figure 10: Route Planning Scenario with Urban Environment Zoom ***(Build 24 of Block D)***

The GPU time [appearing on the Tool Bar of the PVD] represents the cumulative time in seconds it takes for both units to calculate their routes. With respect to the magnitude of this value, lower numbers represent the route being calculated at a quicker rate.

After executing the scenario approximately 10+ times, the GPU time was compared and it was concluded that using the GPU algorithms within OneSAF produced a feature intersection checking improvement of 30x which produced an overall system increase of 10x. Feature intersection checking alone improved from an average of 68,000 milliseconds without GPU functionality to 2,200 milliseconds with the GPU; a 30x improvement. The cumulative route planning time for the scenario went from 45 seconds without GPU functionality to 4.5 seconds with the GPU; a 10x improvement (See Figure 11).

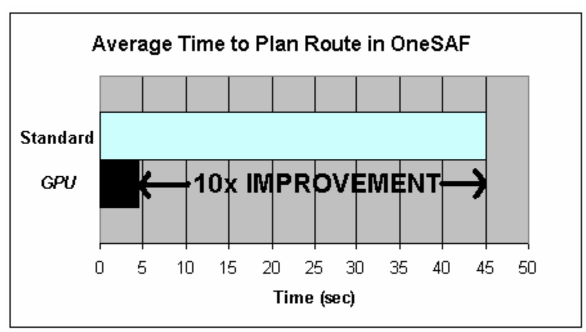

Figure 11: Route Planning Benchmark Data

Although there was only a slight overall system improvement with the GPU functionality, more improvements could be made. As terrains become more complex and ICs route through UHRBs, these numbers could show a significant increase. The GPU-based route planning work is only the first step. Continuing research will look into new algorithms to compute a new route in dynamic environments where buildings can be destroyed.

FUTURE WORK

The GPU project has proven itself to be a successful experiment for the OneSAF program. Preliminary LOS and route planning results have shown overall system improvements up to 20x; however, as on-going development within OneSAF continues and exacting requirements come into play, GPU research must continuously advance to address database complexities.

Feature Dense Terrain Databases

For both the LOS and route planning demos, the terrain used was relatively flat with low feature density (Ft. Hood, Texas). As more complex terrains, such as the Caspian Sea region, become available to OOS, routing will become an even greater problem for the current CPU based algorithms. Improvements that result from implementation of the GPU-based intersection checks will become even more significant.

Paging Terrain Data

ERC makes use of a paging system for handling data from the terrain database. A database is split up into pages, and one page remains in memory at a given time. Such a system is necessary due to the fact that current OOS databases contain far too much data to be held in memory, and future OOS databases will only be larger. The GPU-based algorithms that were integrated into OOS did not use such a paging system, and initially caused the system to run out of memory. The workaround for this problem was to trim the database down, until it was small enough that all necessary data for the GPU could be held in memory. GPU-based algorithms will need to be compatible with a paging system.

Other Uses

There are still other ways in which GPU power can be exploited for simulation performance gains in the near future. For example the idea of using multiple GPUs could provide an even greater advantage over the implementation of a single GPU. Better overall performance is expected by doing more of these GPU-based computations in parallel. Also expected, is the demand and the capability for entity counts to grow in the near future. Being an $O(N^2)$ problem, LOS becomes more of a strain on the simulation as entity counts become higher, and speeding up the LOS calls ultimately becomes more important.

CONCLUSION

Complex computational algorithms such as LOS and route planning challenge the capabilities of simulations such as OneSAF. As increased dense urban environments are introduced and high fidelity models incorporated, these expensive algorithms will be costly when additional entities are added to large exercises.

The GPU technology has proven that it is possible to use COTS hardware to make significant progress in order to accelerate runtime processing speed within OneSAF. By using GPU-based algorithms, increases to the overall OneSAF simulation speedup have been witnessed; LOS by 20x and route planning by a preliminary 10x. Our ability to utilize the GPU technology was essential in making this project successful.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Butler from SAIC for the initial abstract development and submittal. The GPU-based LOS algorithms were developed by Brian Salomon and Naga Govindaraju and the route planning algorithms were designed by David Knott, Ming Lin and Russ Gayle at the University of North Carolina at Chapel Hill. Michael Macedonia at PEO STRI, Angel Rodriguez at RDECOM-STTC and Bob Graybill at DARPA have been significant supporters of this project.

REFERENCES

Condon, Patty. (2002). *Routing Design Notes V0.1 For the OneSAF ERC Program.* Science Applications International Corporation.

Cross, Jason. *AGEIA Physics Processor at E3.* Retrieved June 6, 2005, from http://www.extremetech.com/article2/0,1558,1817922,00/asp?kc=ETRSS02129TX1K0

Lin, Ming, et al. (2005). *GPU-Accelerated Route Planning* [Brochure], Department of Computer Science: UNC.

Manocha, Dinesh, et al. (2005). *Accelerating Computer Generated Forces using GPUs* [Presentation], Department of Computer Science: UNC.

Manocha, Dinesh, et al. (2004). *Accelerating LOS Computations using GPUs*. [Brochure], Department of Computer Science: UNC.

OneSAF Objective System History. Retrieved June 15, 2005 from http://www.onesaf.org/public1saf.html

Polygon Traversal (LOS, Polygon RouteCrossing). Retrieved June 14, 2005, from https://www.onesaf.net/ERC/design_notes/poly_retrieval_et_al.doc

Salomon, Brian, et al. (2004). *Accelerating Line Of Sight Computation Using Graphics Processing Units*. UNC, SAIC, RDECOM, PEO/STRI.

Stam, Nick, (2003). *The Future of 3D Graphics*. Retrieved June 6, 2005 from http://www.extremetech.com/article2/0,1,1558,1091392,00.asp.