NPAC Projects on Parallel Terrain Rendering

Paul Coddington and Geoffrey Fox

Northeast Parallel Architectures Center
Syracuse University

September 1, 1994

This report provides information on the development of parallel rendering software allowing the display of realistic three dimensional images of terrain that could be used for a real-time flight simulator integrated with a Geographic Information System (GIS).

Project Status

We are currently writing a sequential terrain rendering program. We are using the basic Z-buffering algorithm to render texture-mapped polygons. Once this is working we will begin parallelizing the various modules of the program. We are also investigating the problems of running the rendering routine on a parallel computer and sending the results over a network to be displayed on a remote machine, which could be a workstation, PC or Mac.

Existing parallel algorithms

We have done a literature search on parallel rendering algorithms. This turned up surprisingly few papers, and many of them are not relevant since they use a different approach (e.g. raycasting) which is too slow for real-time animation, except on large supercomputers. Most of the programs are general purpose, whereas terrain rendering has certain specialized properties that could be used to simplify the program and improve the performance.

We know of only two existing terrain rendering programs that run on parallel computers. One is from NASA Jet Propulsion Laboratory, and uses raycasting. Onyx Sciences Corporation also have a parallel terrain renderer which runs in real time on a Maspar parallel computer, using a proprietary algorithm that is not polygon-based. We have contact with both of these groups.

Implementing the parallel program

We do not see any obstacles to implementing a parallel terrain rendering program. The approach and the algorithms to be used to create a scene from texture-mapped polygons seem fairly straightforward, and we expect the parallel program to be quite efficient. This application seems to be well-suited to moderate-sized MIMD parallel computers.

One issue is the message passing language to use. Originally we favored PVM, since this is portable and can also be used for passing results over a network from the parallel computer to a remote machine where they are displayed. However PVM has relatively large latencies for sending messages. Real-time animation at 30 frames per second would require that an entire frame be sent in about 30 milliseconds. This is roughly the same order as the software latency in setting up the message passing over a heterogeneous network using PVM, although of course it depends to some degree on the types of computers and networks that are used, and the details of how PVM is implemented. We may be forced to use native networking primitives to achieve the necessary performance in passing results from the parallel computer to the display machine, which will mean this part of the program may no longer be portable. If we abandon PVM for passing results from the parallel computer, then there is no longer any reason to use it as a message passing language for the parallel program, in which case a more efficient parallel language such as MPI would be the best option.

Implementing the parallel program

We do not see any obstacles to implementing a parallel terrain rendering program. The approach and the algorithms to be used to create a scene from texture-mapped polygons seem fairly straightforward, and we expect the parallel program to be quite efficient. This application seems to be well-suited to moderate-sized MIMD parallel computers.

One issue is the message passing language to use. Originally we favored PVM, since this is portable and can also be used for passing results over a network from the parallel computer to a remote machine where they are displayed. However PVM has relatively large latencies for sending messages. Real-time animation at 30 frames per second would require that an entire frame be sent in about 30 milliseconds. This is roughly the same order as the software latency in setting up the message passing over a heterogeneous network using PVM, although of course it depends to some degree on the types of computers and networks that are used, and the details of how PVM is implemented. We may be forced to use native networking primitives to achieve the necessary performance in passing results from the parallel computer to the display machine, which will mean this part of the program may no longer be portable. If we abandon PVM for passing results from the parallel computer, then there is no longer any reason to use it as a message passing language for the parallel program, in which case a more efficient parallel language such as MPI would be the best option.

Optimizations for real-time display

The data can be preprocessed to create texture-mapped polygons with appropriate shadowing using a standard lighting model. An important part of this procedure is the creation of two or more separate data sets of different resolution - high resolution for near-field and lower resolution for far-field. We are looking into mesh optimization algorithms for making a good approximation to the terrain using a given number of polygons that will in general be less than the total number of polygons from the digital elevation data.

This pre-processing phase will be dependent on the target architecture. The mesh optimization could be tuned to vary the number of polygons depending on the power of the computer used to do the real-time rendering. We will investigate other optimization techniques once we have a working prototype.

Compression

To achieve the required throughput on low-bandwidth networks may require compression. We are investigating compression techniques, especially JPEG, for which cheap special purpose chips may become available in the future. In the meantime, parallel programs for JPEG compression exist that a give performance on a modest number of nodes that is comparable to specialized compressions hardware.

References

Parallel Rendering

Polygon Rendering on a Dual-Paradigm Parallel Processor, Theoharis Theoharis, Computer and Graphics, Vol 13, No. 2, pp. 207-216, 1989.

A MIMD Rendering Algorithm for Distributed Memory Architectures, Thomas W. Crockett and Tobias Orloff, Proceedings of the ACM SIGGRAPH Symposium on Parallel Rendering, 1993, pp. 35-42.

A Multicomputer Polygon Rendering Algorithm for Interactive Applications, David Ellsworth, Proceedings of the ACM SIGGRAPH Symposium on Parallel Rendering, 1993, pp. 43-48.

Volume Reconstruction and Parallel Rendering Algorithms: A Comparative Analysis, Ulrich Neumann, Ph.D. Thesis, Univ. of North Carolina at Chapel Hill, 1993.

Parallel Terrain Rendering

Parallel Three Dimensional Perspective Rendering, P. Peggy Li and David W. Curkendall, Parallel Computing: From Theory to Sound Practice, Proceedings of EWPC '92, pp. 320-331, 1992.