THE PROBLEMS OF MODELING AND RENDERING OF THE REALISTIC COMPLEX SCENES

Kovalov S., Korotin U.& Malcheva R. *(DonSTU, Donetsk, Ukraine)*

In this article, we discuss the problems of architecture's organization of the modelling and rendering graphics systems with the high-level parallelism for the scenes of various complexity.

Interactive graphics is a field whose time has come. In the last few years it has benefited from the steady and sometimes even spectacular reduction in the hardware price/performance ratio, and now it is finally ready to fulfil its promise to provide us with pictorial communication and man/machine interaction.

Perhaps the most important new movement in graphics is the increasing concern for modelling objects, not just for creating their pictures. Furthermore, interest is growing in describing the time-varying geometry and behaviour of 3D objects. Thus graphics is increasingly concerned with simulation, animation, and «back to physics» movement in both modelling and rendering in order to create objects that look and behave as realistically as possible.

Displaying large database at high frame rates clearly requires dramatic system performance, both in trams of computations and of memory of bandwidth. We have seen that the geometry portion of a graphics system can require more processing power than a single CPU can provide. Pipeline and parallel processors are the basic building blocks of virtually all current high-performance graphics systems.

Ray tracing is the modern and a powerful rendering method that can generate extremely realistic images. Unfortunately, it requires a great deal of computation: a typical image can require minutes or hours to compute on a typical workstation. Fortunately, ray-tracing algorithms can be parallelized in several ways [1]:

- ***Component parallelism***. Computations for a single ray can be parallelized. For example, reflection, refraction, and intersection calculations all require computing the *x*, *y*, and *z* components of vectors or points. These three components can be calculated in parallel, resulting in a speedup by a factor of 3.
- ***Image parallelism***. Ray-primitive intersections can be calculated in separate pixels, since the calculations for each ray are independent. To take advantage of this form of parallelism, however, pixels potentially need access to the entire database, since the ray tree for a particular ray may reach any portion of the database.

- ***Object parallelism***. Primitives in the database can be distributed spatially over multiply pixels. Each pixel, then, is responsible for all rays that pass through its region. It computes ray-object intersections if the ray hits an object, and forwards the ray to the next pixel otherwise.

The first proposed object-parallel ray-tracing architectures used uniform spatial subdivision to assign portions of the universe to pixels [2]. This resulted in poor efficiency for many scenes, since most of primitives were clustered in a few regions.

The image-parallel ray-tracing algorithm has been developed for Thinking Machines'SIMD Connection Machine [3], in which the database is repeatedly broadcast to all of the pixels, which perform ray-object intersections in parallel. Implementations have been reported on shared-memory multiprocessors, such as the BBN Butterfly [4]. Here, the database does not need to be stored at each pixel or broadcast repeatedly; instead, pixels request portions of the database from the memory system as needed. Unfortunately, contention for shared memory resources increases with the number of pixels, so only modest performance increases can be achieved in such systems.

Another level of parallelism is available in virtual-buffer systems that use rectangular regions and complete bucket sorting. In such a system, each buffer is initially assigned to a region. Parallel virtual-buffer systems do present two difficulties. First, transferring buckets to multiple rasterization buffer in parallel requires a high-performance data network between the front-end and rasterization subsystems, as well as sophisticated control and synchronization software [5]. Second, in some images, most of the primitives in the database can fall into a single region, making the extra layer of parallelism useless. The primitives in the overcrowded region could be allocated to more than one rasterizer, but then the multiple partial images would have to be combined. Although this complicates the rasterization process, it can be done by compositing the multiple images into one buffer at the end of rasterization.

The notion of combining images after rasterization can be used to build a second type of multilevel parallel architecture, ***image-composition*** or ***composite*** architectures. The central idea is to distribute primitives over a number of complete rendering system. The multiple renderers are synchronized so they identical transformation matrices and compute the same frame at the same time. Each renderer then computes its partial image independently and stores that partial image in its own frame buffer. A tree of pipeline of compositors combines the RGB and z-streams from each renderer using the special technique and algorithms. Figure 1 shows a composite system for displaying 4n triangles per frame built from four renderers, each of which can display n triangles per frame.

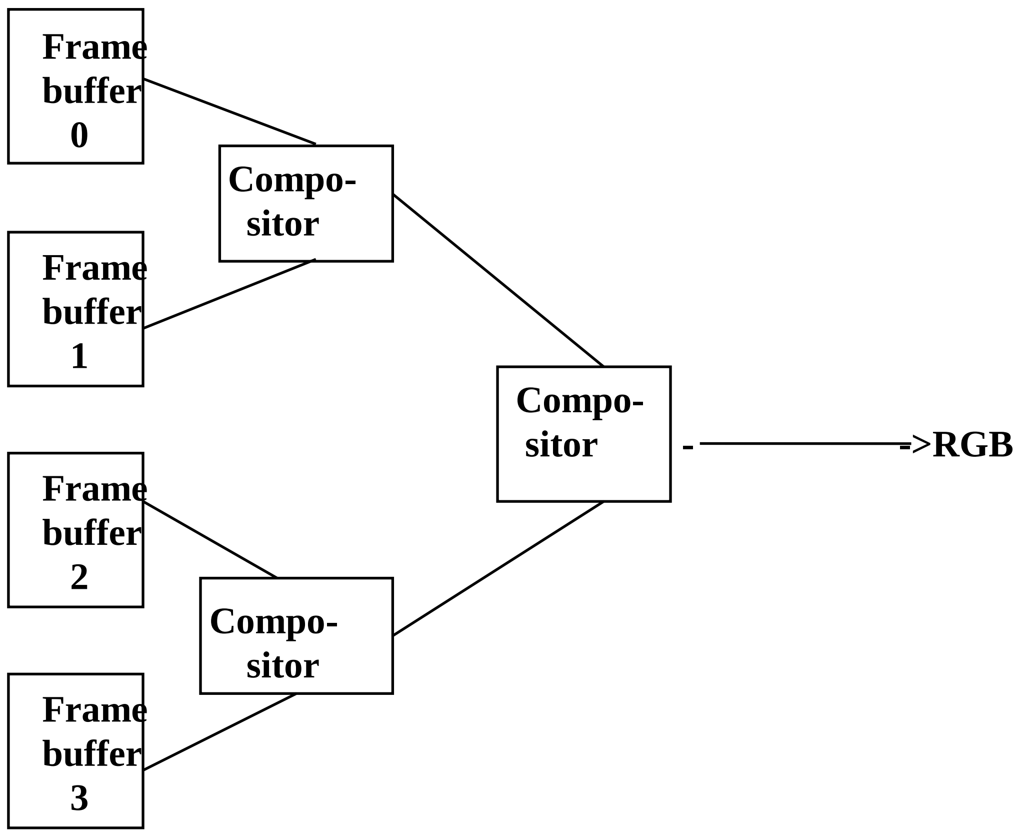

Fig.1. An image-composition system composed of 4 individual renderers

The main difficulties with this approach are the need to distribute the database over multiple processors, which incurs all difficulties of parallel front ends; aliasing or erroneous pixels caused by the image-composition operation; and a lack of flexibility, since image composition restricts the class of rastrezation algorithms that can be implemented on the machine. Nevertheless, this approach provides an important way to realize system of extremely high performance.

Bibliography: 1. Foley J., Dam A. Computer Graphics. Principles and practice. - 2nd ed. In C. - AWPC, 1997. - 1175p. 2. Cleary J., Wyvill G. «Design and Analysis of a Parallel Ray Tracing Computer» // Proceedings of Graphics Interface'83, May 1983, 33-34. 3. Delany H. «Ray Tracing on Connection Machine» // Proceedings of the 1988 International Conference on Supercomputing, July 4-8, 1988, 659-664. 4. Jenkins R. «New Approaches in Parallel Computing» // Computers in Physics, 3(1), May 1989, 8-15. 5. Ellsworth D. Pixel-Planes 5 Rendering Control. - 1998.