An Efficient Massively Parallel Rasterization Scheme For a High Performance Graphics System

S. Karpf, C. Chaillou

ABSTRACT We present in this paper the IMOGENE II system, a massively parallel Multi-SIMD graphics system. This architecture uses a new rasterization scheme combining Object Parallelism and Parallel Virtual Buffers. This scheme leads to a better efficiency than other massively parallel SIMD systems, and allows a cost-effective, powerful and easily expandable system to be designed. The system consists of several SIMD Scan-Conversion Pipelines each connected to a Multi-Level Virtual Buffer, a Shading Unit computing true Phong Shading, a Virtual Accumulation Frame Buffer for anti-aliasing, and a classical Frame Buffer.

1.1 Introduction

The most popular way for rendering interactive synthesis images is still to use Gouraud-interpolated [17], Z-buffered triangles. This method requires a huge amount of interpolations to compute the RGB and Z values of every object at every pixel, and a very high Z-buffer bandwidth (one read/modify cycle for each pixel of each object of the scene).

In order to display images at (near) interactive rates, all current graphics workstations use several concurrent specific hardware units making interpolations and hidden-part elimination. The architectures of these systems are tightly coupled to available VRAM components, and the level of parallelism is generally low. The Silicon Graphics Iris Workstation [1][2] uses 5 interpolators (Span Processors) and 20 hidden-part elimination units (Image Engines). The HP 835 SRX [23] benefits from the parallelism of the PRC chip [29], and uses a cache-memory in order to speed up hidden-part elimination. The Alliant visualization system [30] and the Titan graphics supercomputer [11] use the IMU (Image Memory Unit) designed by Raster Technologies [10]. When displaying true colors, this unit uses 16 SLP chips (Scan Line Processor) in charge of the interpolations, and a 5-pixel parallel access per clock cycle.The Stellar GS 1000 [3] uses 16 chips processing 4×4 pixels in SIMD mode. The ATT Pixel Machine [25] is composed of an array (up to 8×8) of DSP32, each one handling part of the frame buffer. The Apollo DN 10000 [21] uses 5 quadratic interpolators and a 6-pixel access to the frame buffer.

In addition to these commercial workstations, fundamental research works have been carried out in order to define massively parallel graphics system following two main axis: The pixel approach (Pixel-Planes 4 [12], Pixel-Planes 5 [14], SAGE [15]) and the object approach (a system proposed by Weinberg [31], GSP-NVS [9], PROOF [26][27], IMOGENE [5][6]). The Ray-Casting Machine [20] is also an Object-Oriented system, specifically designed for rendering CSG-defined objects. However none of these projects has led to a commercial system yet, contrary to massively parallel general purpose computers (Con-

nection Machine, MasPar...). In the next section we will try to analyse the efficiency of the scan-conversion process. This analysis highlights the very low efficiency of massively parallel SIMD systems. Then we present a new architecture for a massively parallel, high efficiency system combining Object parallelism and parallel virtual buffers in a way never investigated before. This novel rasterization scheme allows to design an expandable, cost-effective and dynamically reconfigurable system. Several configurations are presented and their performance analyzed.

1.2 Analysis

We will now briefly analyse the efficiency of massively parallel systems, with regards to scan-conversion and Z-buffer (a more complete analysis of rasterization techniques can be found in [13] and [16]).

We define the efficiency as the ratio of the number of useful pixels to the number of generated pixels. A useful pixel is a pixel belonging to an object. For instance, the graphics unit of the Stellar GS 1000 processes 4×4-pixel blocks in SIMD mode. When drawing lines, an average of 6 pixels belongs to the 4×4 block, thus the efficiency of the system (when drawing lines) is 6/16.

Before analyzing massively parallel systems, let us describe more precisely the Silicon Graphics IRIS, which will be our reference. This machine has a tree structure: a processor (Edge Processor) determines the ends of the various vertical spans of the triangle (or trapezoid), and broadcasts them to 5 Span Processors making interpolations. The Span Processors broadcast their results to 20 Image Engines making hidden-part elimination (Z-buffer algorithm). This system makes a very smart use of 1MBytes VRAM components. It is a MIMD machine with a 100% efficiency (only useful pixels are generated), thus allowing very high performances (the new VGX is rated at 1,000,000 small triangles (100 pixels) per second [28]). Here follows an analysis of the efficiency of recently proposed massively parallel graphics systems.

4 SAGE chips are used to build a pipeline of 1024 Pixel-Processors. This pipeline is a SIMD machine, and the objects pass from left to right in order to process a complete scan-line. The efficiency is the average number of pixels belonging to an object on an active scan-line divided by 1024. When displaying small facets, the average width of an object is taken as 10 pixels. The efficiency is thus about 1/100. Of course, the efficiency increases when the average size of objects increases.

GSP-NVS can be considered as the symmetrical object-oriented architecture of SAGE. It is composed of a pipeline of Object Processors (up to 1000), each one handling one triangle on the active scan-line. A system with 1000 Object Processors has an efficiency of 1/100 when displaying small triangles (assuming that all the processors are active).

Pixel-Planes 4 is an array of 512×512 pixel-processors. The efficiency is the average number of pixels belonging to an object (on the entire screen) divided by the number of pixels in the screen. When displaying 100-pixel triangles, Pixel-Planes 4 has thus a very low efficiency (about 1/3000). A 1024×1024 system (i.e. 1024×1024 pixel-processors) would have an even lower efficiency (1/10000). As the primitives are broadcasted to the Pixel-Processors through a tree structure, the performances of the system do not depend on the size of the objects.

Pixel-Planes 5 is the successor of Pixel-Planes 4, and tries to address the low efficiency issue. The basic unit of the system is an array of 128×128 pixel-processors called Renderer. When displaying small triangles, the efficiency of each Renderer is about 1/150.

Moreover several Renderers can operate concurrently on non-overlapping virtual patches, thus increasing the performance of the system.

We have been studying for two years the IMOGENE machine, which can be considered as the symmetrical object-oriented architecture of Pixel-Planes 4. The system is composed of a very large number of Object Processors, each one rasterizing one primitive in raster-scan order on the entire screen. The efficiency of such a system is very low when displaying small triangles (the same as Pixel-Planes 4).

This analysis leads us to the following statements:

- Massively parallel SIMD systems have a (very) low efficiency. The main consequence is that these systems must use a considerable amount of hardware to balance the low efficiency (typical examples are Pixel-Planes 4, PROOF and IMOGENE).
- Systems using virtual techniques (virtual processors like GSP-NVS, or parallel virtual buffers like Pixel-Planes 5) have a better efficiency, and thus are more cost-effective.
- Using a MIMD parallelism (Silicon Graphics Iris, Pixel-Planes 5) allows the performance to be increased without decreasing the overall efficiency.
- The efficiency of a SIMD system increases when the average size of the objects (mainly triangles) increases. This means that the performances of massively parallel systems are quite insensitive to large triangles, contrary to "classical systems" like the Silicon Graphics Iris.

This analysis has led us to modify the IMOGENE architecture in order to increase the efficiency of the system by at least an order of magnitude. As we did not want to modify the Object Processors designed for IMOGENE, we adopted a multi-SIMD architecture with parallel rectangular virtual buffers (like Pixel-Planes 5). Contrary to pixel-oriented systems, the object approach allows the system to operate on any rectangular patch enclosed by the hardware virtual buffer, and thus to dynamically increase the efficiency of the system.

1.3 The IMOGENE II concept

The key idea of the IMOGENE II system is to increase the efficiency of massively parallel systems by generating as few useless pixels as possible. Moreover a dynamic allocation of the processors allows the system to be quite insensitive to "bad" databases (many large facets hiding each other, most of the primitives falling into a single region of the screen...). At last, the system is easily expandable.

The screen is subdivided into several non-overlapping rectangular patches. Scan-conversion is achieved in parallel by several Scan-Conversion Pipelines (SCP) operating on the patches. Each SCP is composed of a large number of Object Processors, each one scan-converting one graphics primitive in raster-scan order. Hidden part elimination is achieved through the pipeline which outputs the visible object at the current pixel and stores its characteristics (depth, surface normal vector, basic color) in the Virtual Buffer. An external Zbuffer operator is available if multiple scan conversion passes have to be made.The pool of Scan-Conversion Pipelines is a MIMD system. When a SCP has scan-converted all its primitives, the Shading Unit extracts the pixels from the Virtual Buffer (via a high speed bus), shades them (Phong method) and stores them (RGB values) in

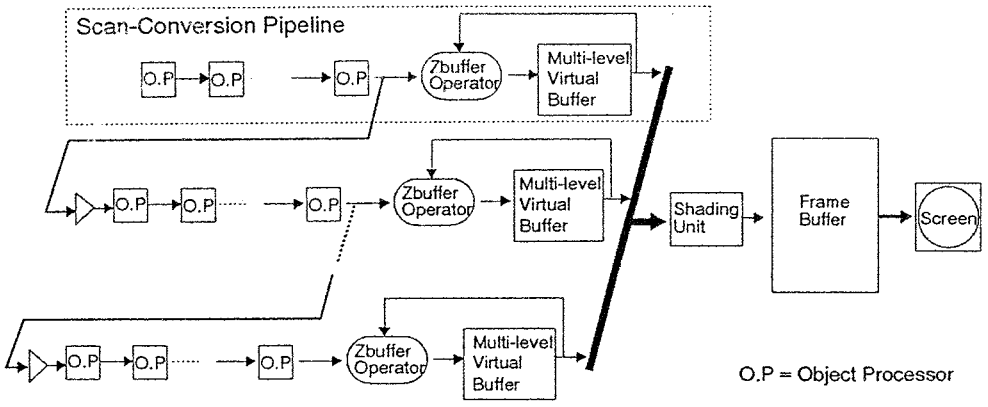

Fig. 1.1. The IMOGENE II Back-End Architecture

the Frame Buffer.The Shading Unit becomes available for the next SCP, and so on until the entire scene is completed. The pixels are then displayed on the screen. An operational system can be expanded without any dedicated hardware either by adding Object Processors to the SCPs (i.e. adding chips on a PCB), or by adding SCPs (i.e adding PCBs on a backplane). Here follows a description of the functional architecture of the Scan-Conversion Pipelines and the Shading processor.

1.3.1 The Scan-Conversion Pipeline (SCP)

The SCP is an object-parallel SIMD machine that rasterizes multiple primitives in raster-scan order into a virtual rectangular area representing a part of the screen. Each Object Processor (OP) in the pipeline handles one graphics primitive and outputs at least its depth, its surface normal vector and its basic color in raster-scan order (extra values like texture or anti-aliasing information can be output if high-quality images are to be generated). In order not to need any external Zbuffer chip to connect two processors, the Object Processor is also fitted with a Zbuffer operator: each OP receives the depth and the surface normal vector of its predecessor, compares the depth with its own one, and transmits the visible object to its successor. This requires that the Object Processors be not truly synchronous (they do not process the same pixel). Any geometric primitive can be used provided that its depth and surface normal vector can be computed at each pixel.

The Internal Object Generator receives the description of the objects from the host and stores them in its memory. This memory should be double-buffered, so that an Object Processor can receive and store new objects while scan-converting the previous ones, thus adding a coarse-grain pipeline effect during the generation of a frame. This requires that a separate datapath be available for the loading of the objects. GSP-NVS [9] had a single pipeline for loading and rasterizing. Although this reduces the chip pin-outs, it does not allow the system to benefit from the pipeline effect. We have designed an Object Processor that generates triangles. A complete description of this processor can be found in [22]. The chip operates at 16 Mhz and has been designed in semi-custom technology (ES2 SOLO 1400). It uses about 45,000 transistors and is housed in a 120-pin PGA package. An optimized design in full custom should greatly reduce the complexity of the Triangle

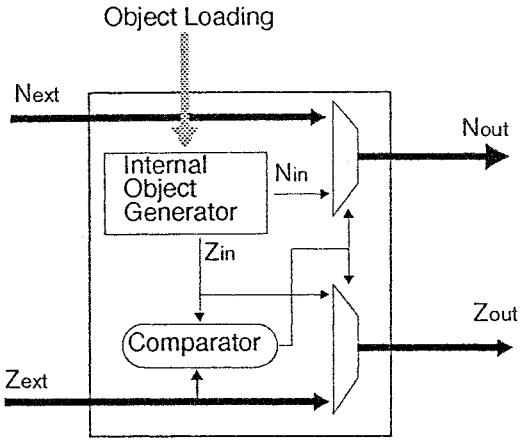

Fig. 1.2. An Object Processor

Processor and allow the clock rate to be increased (33 Mhz). Chips including ten Triangle Processors could then be designed without modifying the chip pin-out. A 100 OP scan-conversion pipeline will then require only ten chips.

The pipeline described above is similar to other Object Oriented Rendering Pipelines [31][9][27]. Two novel concepts makes the SCP more efficient than these systems by using a dynamic allocation of the resources.

The Multi-Level Virtual Buffer Concept

The output of the pipeline is connected to a Multi-Level Virtual Buffer via an external Zbuffer operator allowing multiple passes if the number of objects is greater than the number of Object Processors, and thus solving the overflow issue. The Virtual Buffer can be built with standard fast SRAM components, and represents a rectangular patch on the screen. The shape of this patch is not fixed, and can be dynamically configurated (either square or rectangular). For example, a 128×128 buffer must be considered as a 16K-pixel buffer, and can represent (assuming a 1280x1024 display) a 128×128 region, a 512×32 region, a 1280×12 region (i.e 12 scan-lines), and so on. This high flexibility is due to the scan-line processing of the object approach. Indeed pixel-oriented systems, like Pixel-Planes 5, have fixed virtual buffers whose size and shape depend on the number of pixel-processors. Moreover the rasterization area (square or rectangular) can be smaller than the actual size of the hardware buffer. For example a 128×128 patch can be dynamically subdivided into four 64×64 patches: the SCP scan-converts all the primitives falling into the first sub-patch, then all the primitives falling into the second sub-patch, and so on until the entire virtual buffer is completed. These subdivisions increase the efficiency of the SCP, but of course require more front-end computation power (examples are given in section 4).

Our rasterization scheme allows the system to operate on any rectangular region of the screen, and thus is very well suited for window systems (several SCPs can operate concurrently on several windows).

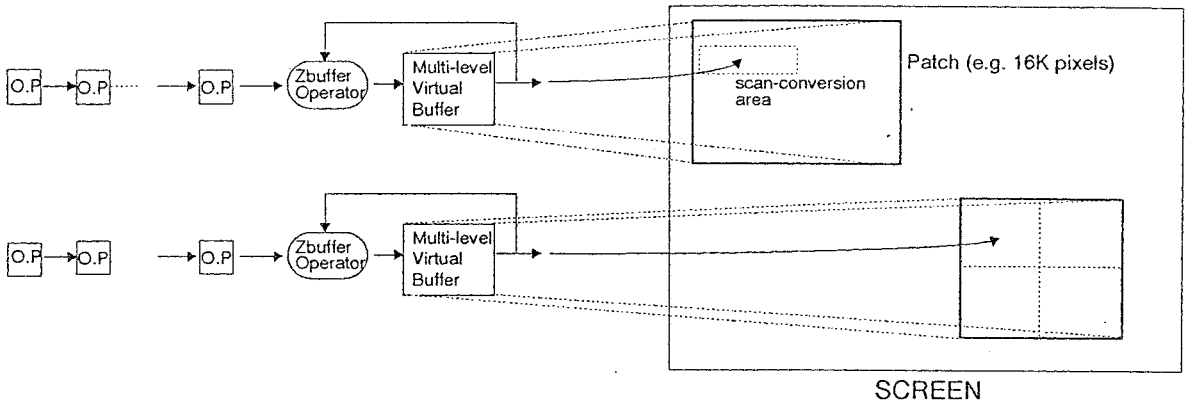

Fig. 1.3. The Multi-Level Virtual Buffer Concept

Dynamic allocation of the Object Processors

The first Object Processor of the pipeline can be dynamically connected to the last Object Processor of another pipeline if necessary (in that case, the virtual buffer of the first pipeline becomes unused). This allows the system to remain balanced even when most of the primitives fall into a single region by increasing the number of Object Processors allocated to it. For example, a system with 10 pipelines of 100 Object Processors each can be configurated as one pipeline of 300 O.Ps and 7 pipelines of 100 O.Ps each, or even as one pipeline of 1000 O.Ps.

1.3.2 The Shading Unit

This unit extracts the pixels (defined by their depth, surface normal vector and basic color) from the first available virtual buffer, shades them and stores the resulting RGB values in the frame buffer (see Figure 1).

The Shading Unit is composed of one Normalization Unit, and several Shading Processors. The surface normal vectors computed by the Object Processors are not normalized, which leads to inaccurate visual results when using Phong's method. Although the inaccuracy is acceptable for tessellated objects [27], it leads to low quality images when high level primitives (like quadrics) are used. The normalization stage is then compulsory.

The shading processors receives the normalized vectors, and computes the shading for their own light source (each shading processor handles one light source). The RGB values output by the shading processors are then added to produce the final RGB value to be stored in the frame buffer. The whole Shading Unit is pipelined, and thus can deliver one shaded pixel at each clock cycle. A description of the shading unit can be found in [22]

Let us note that a low cost system with no shading unit can be designed for triangle-based Gouraud-shaded images. The front-end makes shading computations at the vertices of the triangles (Gouraud or Phong lighting model), and the Triangle Processors directly interpolates these values (Gouraud interpolation) instead of interpolating the surface normal vector.

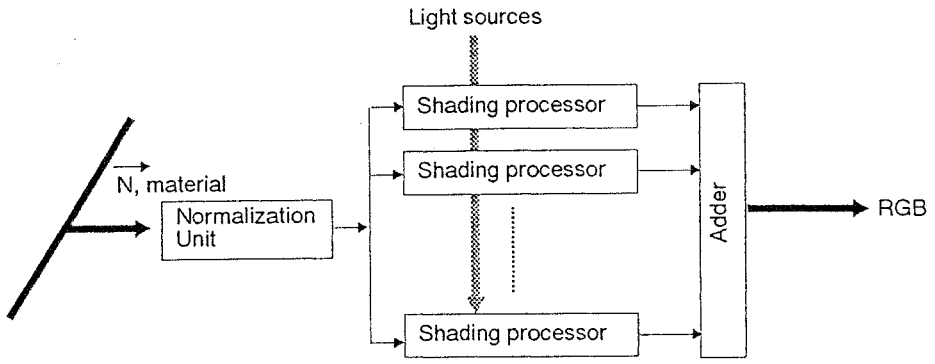

Fig. 1.4. Principle of The Shading Unit

1.3.3 Anti-aliasing

We are currently investigating several anti-aliasing methods for the IMOGENE II system. A possible solution is to add a virtual accumulation buffer between the shading unit and the frame buffer ([9] and [18] also use an accumulation buffer allowing multiple samples to be accumulated when displaying anti-aliased images). The virtual accumulation buffer is composed of a smal ALU for combining multiple samples, and a virtual buffer built with fast SRAM components. A classical true color frame buffer provides 8 bits per component (at least R, G and B values, and possibly an Alpha value). The accumulation buffer must provide extra memory in order to avoid overflows when combining several samples. As the accumulation buffer is a small virtual buffer (16K pixels for instance), adding extra memory is very cheap (much cheaper than a complete 1M-pixels accumulation buffer). For instance, 16 bits per component allow 256 samples to be accumulated without overflow.

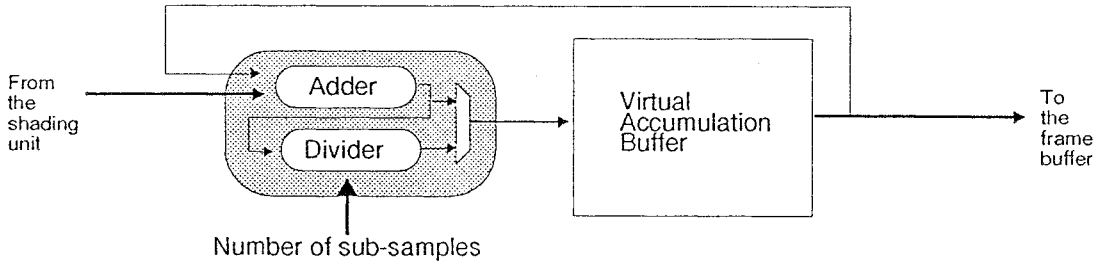

Fig. 1.5. Principle of The Virtual Accumulation Buffer

The ALU consists of one adder and one divider. The adder is used when accumulating two sub-samples. The divider is used to compute the final RGB value when the last sub-sample is taken. It can be replaced by a (low cost) variable shifter if the number of

sub-samples is always a power of two.

1.3.4 Front-End

The host computer computes all geometric transformations. Many projects [9], [14] have shown that systems capable of rendering more than 1M triangles/second must have very powerful floating-point front-end systems (at least 8 RISC or DSP). The specific architecture of our back-end system makes some computations easier than for other systems, but also requires complex sorting procedures. The host is in charge of the following computations:

- Classical geometric transformations.
- Shading computations: the host is not in charge of the shading computations. This is a substantial saving, for shading computations require a lot of computing power. For example, computing the Phong lighting model with two light sources at the vertices of a triangle require as many operations as all the other geometric transformations.
- Bucket sorting: the host must sort the primitives according to their position on the screen, and dispatch them to the corresponding SCPs. A primitive belonging to several patches must be rendered several times.
- Dynamic allocation of the resources: The host must choose the optimum division of the screen according to the application. The goal of this subdivision is to increase the efficiency of the system. A good subdivision should also optimize the load of the scan-conversion pipelines.

We are still working on the architecture of the front-end computer. A high-speed ring network like that developed for Pixel-Planes 5 [14] could be a good solution. Our SCPs could replace Pixel-Planes 5's renderers without modifying the host architecture. We are also investigating a Transputer-based solution with the new T9000.

1.4 Examples of configurations and expected performances

We try in this section to evaluate the performances of the back-end architecture. Of course we assume that the system has sufficient front-end computation power. We also assume that the whole system operates at 33 MHz and that the shading unit has enough shading processors.

Display rate depends on the number of pixels which must be transferred from the virtual buffers to the frame buffer. Assuming that at least one virtual buffer is always ready, the animation rate on a 1280×1024, 60Hz screen is about 20Hz. Of course this rate increases if a lower screen resolution is used (i.e. when fewer pixels have to be transferred).

- First configuration: one Scan-Conversion Pipeline with 100 Triangle Processors and a 16K-pixel Virtual Buffer.

(1) The 16K-pixel buffer represents a 128×128 region

The SCP can scan-convert 100 triangles in about 128*128*30ns = 0.5 ms. As the Oject Processors are double-buffered, the host can send new coefficients during scan-conversion. The SCP is always active if all the new coefficients can be loaded during

the previous scan-conversion, i.e in 0.5 ms. A triangle is defined by 64 bytes. Thus the necessary bandwidth of the loading bus is 13 MBytes/second. If the host can sustain this rate, then the system will be able to rasterize 100/0.0005 = 200,000 Phong-shaded triangles/second (Pixel-Planes 5's renderer is rated at 150,000 triangles/second). Let us note that the triangles can be up to 16K-pixel large, thus allowing the system to keep high performances even when displaying many large triangles.

These performances can be increased without modifying the architecture by using the multi-level virtual buffer concept. For instance, if more than 100 triangles fall into the 128×128 patch, the patch can be virtually subdivided into four 64×64 patches:

(2) The 16K-pixel buffer represents four 64×64 regions

The SCP can then scan-convert 100 triangles in about 64*64*30ns = 0.13ms, but he necessary bandwidth of the loading bus becomes 52 MBytes/second. Again, if the host can sustain that rate, the peak performance of the system is 800,000 Phong-shaded triangles/second.

Of course, these are theoretical peak performances, and actual performances will be reduced in part because of the triangles that fall into several patches. If we assume that 25% of the triangles fall into two patches, the SCP has an actual performance of 600,000 triangles/second. We believe that an optimized screen partitioning (made by the host) according to the application should allow the system to keep a very high efficiency, and thus high performance, even in a low-cost configuration.

In a single-SCP configuration, display rate depends on the load of the patches. When many primitives (far more than 100) fall into a patch, the shading unit must wait until all the primitives are processed.

- Second configuration: ten Scan-Conversion Pipeline with 100 Triangle Processors and a 16K-pixel Virtual Buffer each.

Each SCP has a peak performance of 600,000 triangles/second (assuming 64×64 patches). Ten SCPs operating concurrently have a theoretical peak performance of 6 millions triangles/second. Of course the system must have sufficient front-end computation power and a sufficient communication bandwidth between the front-end and the back-end.

Using several SCPs leads to an efficient load balancing. For instance, in a 10-SCP configuration, each SCP can make on an average 10 passes without reducing the display rate (heavy-loaded patches are balanced by the ones enclosing few primitives). Several SCPs can even be connected in order to increase the length of the SCP when a patch is overloaded. Such a system could thus update scenes containing 300,000 triangles (up to 4K-pixel) at 20 Hz on a 1280×1024 screen, and is thus very well suited for real-time applications (flight simulations, virtual reality...).

1.5 Conclusion

We have presented in this paper an efficient multi-SIMD massively parallel rasterization scheme. We believe that the main issue of massively parallel system is the poor utilization of the processors. Our proposal is a possible solution to this problem for object oriented graphics systems. The main innovation is the combination of SIMD object rendering pipelines with reconfigurable rectangular virtual buffers. The system is capable

of rendering high quality Phong-shaded, anti-aliased images. Other sophisticated effects (texture mapping and shadowing) are presently being studied. Of course much work is still to be done, especially for defining the matching front-end. Our scheme tries to increase the efficiency of an object-parallel graphics system, and thus leads to the following (yet unsolved) problems:

- the host must be a very powerful muti-processor system with a very efficient communication tool. Efficient sorting algorithms have to be developped for supporting the dynamic virtual rasterization scheme.
- Simulations must be made in order to choose the appropriate number of scan-conversion pipelines and the appropriate number of Object Processors per pipeline.

However, the main advantages of the back-end system are:

- easily expandable: the system can be upgraded either by adding Object Processors to each Scan-Conversion Pipeline, or by adding Scan-Conversion Pipelines.
- high performance: a 1000-processor system will be capable of rendering up to 6 millions triangles/second.

1.6 Acknowledgments

This work is supported by French Research Program on New Architectures (PRC/GDR ANM-CNRS). We wish to thank Professor Michel Meriaux for supervising this project. We also thank the IMOGENE team, especially Eric Nyiri for his work on the high level simulator, Vincent Lefevere for the design of the shading processor and Samuel Degrande for his technical support.

1.7 References

[1] Akeley, K. The Silicon Graphics 4D/240GTX Superworkstation. *IEEE Computer Graphics and Applications*, vol. 9 num. 4, july 89, pp 71-83

[2] Akeley, K. and Jermoluk, T. High-Performance Polygon Rendering. *ACM Computer Graphics*, vol. 22 num. 4, august 1988, pp 239-246

[3] Apgar, B. Bersack, B. and Mammen, A. A Display System for the Stellar Graphics Supercomputer Model GS1000. *ACM Computer Graphics*, vol. 22 num. 4, august 1988, pp 255-262

[4] Blinn, J. F. Computer Display of Curved Surfaces. *Ph.D. thesis*, University of Utah, Department of Computer Science, December 1978

[5] Chaillou, C. Etude d'un Processeur de Visualisation d'Images de Synthese en Temps Reel Exploitant un Parallelisme Massif Objet: le Projet I.M.O.G.E.N.E. *Ph.D. Thesis*, Universite de Lille, Janvier 1991

[6] Chaillou, C. Karpf, S. and Meriaux, M. I.M.O.G.E.N.E: A Solution to the Real Time Animation Problem *Advances in Computer Graphics Hardware V*, Springer Verlag, 1992

[7] Claussen, U. Real Time Phong Shading. *Advances in Computer Graphics Hardware V*, Springer Verlag, 1992

[8] Crow, F. Shadow Algorithms for Computer Graphics. *ACM Computer Graphics*, vol. 11 num. 3, july 1977, pp 242-248

[9] Deering, M. Winner, S. Schediwy, B. and al. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. *ACM Computer Graphics*, vol. 22 num. 4, august 1988, pp 21-30

[10] Denault, D. Ryherd, E. Torborg, J. and al. VLSI Drawing Processor Utilizing Multiple Parallel Scan-Line Processors. *Advances in Computer Graphics Hardware II*, Springer Verlag, 1988, pp 167-182

[11] Diede, T. Hagenmaier, C.F. Miranker, G.S. and al. The Titan Graphics Supercomputer Architecture. *IEEE Computer*, September 1988, pp. 13-30.

[12] Eyles, J. Austin, J. Fuchs, H. and al. Pixel-Planes 4: A Summary. *Advances in Computer Graphics Hardware II*, Springer Verlag, 1988, pp 183-208

[13] Foley, J. Van Dam, A. Feiner, S. and Hughes J. Computer Graphics: Principles and Practice (second edition) *The systems Programming Series*, Addison Wesley 12110, 1990

[14] Fuchs, H. Poulton, J. Eyles, J. and al. Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories. *ACM Computer Graphics*, vol. 23 num. 3, july 89, pp 79-88

[15] Gharachorloo, N. Gupta, S. Hokenek, E. and al. Subnanosecond Pixel Rendering with Million Transistor Chips. *ACM Computer Graphics*, vol. 22 num.4, august 1988, pp41-49

[16] Gharachorloo, N. Gupta, S. Sproull, R. and al. A Characterization of Ten Rasterization Techniques. *ACM Computer Graphics*, vol. 23 num. 3, july 89, pp 355-368

[17] Gouraud, H. Continuous Shading of Curved Surfaces. *IEEE Transaction on Computers*, vol. C-20 num. 6, june 1971, pp 623-629

[18] Haeberli, P. and Akeley, K. The Accumulation Buffer: Hardware Support for High-Quality Rendering. *ACM Computer Graphics*, vol. 24, num. 4, august 90, pp. 309-318

[19] Karpf, S. Chaillou, C. Nyiri, E. and Meriaux, M. Real-Time Display of Quadric Objects in the I.M.O.G.E.N.E Machine. *Proceedings ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, june 1991, pp 269-277

[20] Kedem, G. and Ellis, J.L. The Ray Casting Machine. *Parallel Processing for Computer Vision and Display*, Addison Wesley, pp 378-401

[21] Kirk, D. and Voorhies, D. The Rendering Architecture of the DN10000VS. *ACM Computer Graphics*, vol. 24 num. 4, august 90, pp 299-307

[22] Lefevere, V. Karpf, S. Chaillou, C. and Meriaux, M. The I.M.O.G.E.N.E Machine: Some Hardware Elements. *Sixth Eurographics Workshop on Graphics Hardware.* To be published in Advances in Computer Graphics Hardware VI, Springer Verlag.

[23] McLeod, J. HP delivers photo realism on an interactive system. *Electronics*, March, 17, 1988, pp. 95-97.

[24] Phong, B. T. Illumination for Computer Generated Pictures. *Communications ACM*, vol. 18 num. 18, june 1975, pp 311-317

[25] Potmesil, M. and Hoffert, E. The Pixel Machine: A Parallel Image Computer. *ACM Computer Graphics*, vol. 23 num. 3, july 89, pp 69-78

[26] Schneider, B.O. A Processor for an Object-Oriented Rendering System. *Computer Graphics Forum*, num. 7, 1988, pp301-310

[27] Schneider, B.O. and Claussen, U. PROOF: An Architecture for Rendering in Object Space. *Advances in Computer Graphics Hardware III*, Springer Verlag, pp 121-140.

[28] Silicon Graphics. *Power Series Technical Report.* 1990.

[29] Swanson, R. and Thayer, L. A Fast Shaded-Polygon Renderer. *ACM Computer Graphics*, vol. 20 num. 4, august 1986, pp 95-101

[30] Torborg, J. A Parallel Processor for Graphics Arithmetic Operations. *ACM Computer Graphics*, vol. 21 num. 4, july 87, pp 197-204

[31] Weinberg, R. Parallel Processing Image Synthesis and Anti-Aliasing. *ACM Computer Graphics*, vol. 15, num. 3, august 1981, pp. 55-62