

## 6.4 Consistency Model

In their paper on the parallel API, Igehy, Stoll and Hanrahan defined the concept of sequential consistency for parallel graphics systems. The graphics system presented in their paper provides command-sequential consistency, which means that each OpenGL command is considered to be an atomic operation. WireGL provides a weaker form of consistency called fragment-sequential consistency. In this consistency model, only operations on the framebuffer are considered to be atomic. When considered in isolation, each tile in WireGL is command-sequentially consistent, but the final image is not. If two clients each draw a triangle without any explicit ordering or depth buffering, WireGL may show one or the other on top on a per-tile basis. Igehy notes that any graphics system that supports the parallel API should provide at least fragment-sequential consistency. Parallel applications that must always produce exactly the same final image can achieve this in one of two ways: they can use depth buffering, or they may express their ordering requirements through the use of the parallel API. WireGL provides both of these capabilities, and we have not found any application that both produces deterministic images and also relies on the stronger command-sequential consistency model.

## 6.5 Future Work

The main future direction of the WireGL project is to add additional flexibility. The current system is suitable for many applications, but some parallel rendering tasks require a more flexible configuration. When considering the visualization server configuration of WireGL, it is clear that each node in the cluster is acting as an OpenGL stream processor. The application is a stream source, generating multiple streams to a number of rendering tiles. The intermediate pipeservers accept an incoming stream of geometry and generate a new outgoing stream of imagery. The final compositing pipeserver accepts multiple imagery streams and generates a final image for display.

Because the system is closed (that is, each stream is in exactly the same format), it is easy to imagine that other useful stream processing configurations could be constructed. The next version of our cluster rendering software will allow the user to describe an arbitrary directed graph of graphics stream processing units. Stream processors will be written using a standardized interface so that new stream processors can easily be created and plugged into the cluster rendering framework. This will provide researchers with a framework for testing their own cluster rendering algorithms, be they sort-last, sort-first, retained-mode, or extremely specialized. We will be developing parallel applications for volume rendering, interactive exploration of unstructured grid data, terrain flythroughs, as well as parallelized versions of commonly used visualization packages such as VTK, all targeted at this new common cluster rendering framework.

Another promising application of this new technology is transparent support for CAVEs or arrays of casually aligned projectors. A version of WireGL has already been adapted to allow unmodified applications to run in a CAVE, and we have seen a demonstration of WireGL used for a tiled display consisting of off-axis projectors. In addition, we have nearly completed a Microsoft DirectShow backend for the Visualization Server to leverage the latest technology in streaming video codecs for rendering at a distance. Our latest results allow us to deliver the rendering power of our cluster at  $640 \times 480$  across a 100 megabit network, and recent codec advances will allow us to use even slower networks for scalable remote visualization. A version of the system described in this paper has already been developed to perform sort-last parallel rendering, using Lightning-2 to perform depth compositing on the pixel chain.

## 7 Conclusions

We have described WireGL, a scalable graphics system for clusters of workstations. By integrating parallel submission into our sort-first parallel renderer, we are able to achieve scalable rendering performance for a variety of application types. WireGL allows users to build a graphics system capable of handling demanding real-time, immediate mode tasks at a fraction of the cost of a traditional graphics supercomputer. Alternately, it is possible to realize much higher performance on a cluster of workstations for the same price.

WireGL is a more flexible graphics system than an internally parallel standalone graphics accelerator. By leveraging commodity parts, the building blocks of WireGL can be easily upgraded as technology improves. WireGL enjoys the economies of scale of off-the-shelf parts, providing excellent price performance. In addition, algorithm or system designers can use WireGL as a base for experimentation with parallel rendering. WireGL's flexibility and scalable performance make it an attractive system for real-time rendering on clusters.

## Acknowledgments

The authors would like to thank the entire Lightning-2 team for their efforts, without which much of this work would be impossible. John Gerth and Chris Niederauer were instrumental in procuring, constructing, and setting up our cluster. Randy Frank, Dino Pavlakos and Brian Wylie provided valuable guidance in designing the cluster and selecting equipment. Nick Triantos and Andrew Ritger provided timely updates to NVIDIA Linux drivers, often providing top-of-tree driver builds. Bob Felderman and Glenn Brown from Myricom were very helpful in tracking down our Myrinet bugs (performance and otherwise). Maureen Stone and François Guimbretière have been very patient users of WireGL as it was maturing. This work was sponsored by the DOE VIEWS program (contract B504665) and the DARPA DIS program (contract DABT63-95-C-0085-P00006).

## References

- [1] W. Blanke, C. Bajaj, D. Fussel, and X. Zhang. The Metabuffer: A Scalable Multiresolution Multidisplay 3-D Graphics System Using Commodity Rendering Engines. TR2000-16, University of Texas at Austin, February 2000.
- [2] N. Brden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet: A Gigabit-per-second Local Area Network. *IEEE Micro*, pages 29–36, February 1995.
- [3] I. Buck, G. Humphreys, and P. Hanrahan. Tracking Graphics State for Networked Rendering. *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware*, August 2000.
- [4] R. Cunniff. visualize fx Graphics Scalable Architecture. *Proceedings of Eurographics/SIGGRAPH Hot3D*, pages 29–33, August 2000.
- [5] Digital Visual Interface Specification. <http://www.ddwg.org>.
- [6] M. Eldridge, H. Igehy, and P. Hanrahan. Pomegranate: A Fully Scalable Graphics Architecture. *Proceedings of SIGGRAPH 2000*, pages 443–454, July 2000.
- [7] H. Fuchs, J. Poulton, J. Eyles, T. Greer, H. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel.

- Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories. *Proceedings of SIGGRAPH 89*, pages 79–88, July 1989.
- [8] P. D. Heermann. Production Visualization for the ASCI One TeraFLOPS Machine. *Proceedings of IEEE Visualization*, pages 459–462, October 1998.
- [9] A. Heirich and L. Moll. Scalable Distributed Visualization Using Off-the-Shelf Components. *IEEE Parallel Visualization and Graphics Symposium*, pages 55–59, October 1999.
- [10] G. Humphreys, I. Buck, M. Eldridge, and P. Hanrahan. Distributed Rendering for Scalable Displays. *IEEE Supercomputing 2000*, October 2000.
- [11] H. Igehy, M. Eldridge, and P. Hanrahan. Parallel Texture Caching. *Proceedings of SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 95–106, August 1999.
- [12] H. Igehy, G. Stoll, and P. Hanrahan. The Design of a Parallel Graphics Interface. *Proceedings of SIGGRAPH 98*, pages 141–150, July 1998.
- [13] M. Kilgard. GLR, an OpenGL Render Server Facility. *Proceedings of X Technical Conference*, February 1996.
- [14] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. *Proceedings of SIGGRAPH 2000*, pages 131–144, July 2000.
- [15] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Proceedings of SIGGRAPH 87*, pages 163–169, July 1987.
- [16] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A Sorting Classification of Parallel Rendering. *IEEE Computer Graphics and Algorithms*, pages 23–32, July 1994.
- [17] S. Molnar, J. Eyles, and J. Poulton. PixelFlow: High-Speed Rendering Using Image Composition. *Proceedings of SIGGRAPH 92*, pages 231–240, August 1992.
- [18] J. Montrym, D. Baum, D. Dignam, and C. Migdal. InfiniteReality: A Real-Time Graphics System. *Proceedings of SIGGRAPH 97*, pages 293–302, August 1997.
- [19] C. Mueller. The Sort-First Rendering Architecture for High-Performance Graphics. *1995 Symposium on Interactive 3D Graphics*, pages 75–84, April 1995.
- [20] OpenGL Specifications.  
<http://www.opengl.org/Documentation/Specs.html>.
- [21] W. C. Reynolds and M. Fatica. Stanford Center for Integrated Turbulence Simulations. *IEEE Computing in Science and Engineering*, pages 54–63, April 2000.
- [22] J. Rohlf and J. Helman. IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. In *Proceedings of SIGGRAPH 94*, pages 381–395, July 1994.
- [23] R. Samanta, T. Funkhouser, K. Li, and J. P. Singh. Sort-First Parallel Rendering with a Cluster of PCs. *SIGGRAPH 2000 Technical Sketch*, August 2000.
- [24] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. P. Singh. Load Balancing for Multi-Projector Rendering Systems. *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 107–116, August 1999.
- [25] SGI multipipe. <http://www.sgi.com/software/multipipe/>.
- [26] SGI vizserver. <http://www.sgi.com/software/vizserver/>.
- [27] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. Lightning-2: A High-Performance Display Subsystem for PC Clusters. *Proceedings of SIGGRAPH 2001*, August 2001.



P.B.5818 - Patentlaan 2  
 2280 HV Rijswijk (ZH)  
 ☎ (070) 3 40 20 40  
 FAX (070) 3 40 30 16

11K  
 01 JAN 2006

Europäisches  
 Patentamt

European  
 Patent Office

Office européen  
 des brevets

Generaldirektion 1

Directorate General 1

Direction générale 1

Howe, Steven  
 Lloyd Wise  
 Commonwealth House,  
 1-19 New Oxford Street  
 London WC1A 1LW  
 GRANDE BRETAGNE



EPO Customer Services

Tel.: +31 (0)70 340 45 00

Date	04.01.06
------	----------

Reference SH-46739	Application No./Patent No. 03257464.2 - 2218
Applicant/Proprietor ATI Technologies Inc.	

**COMMUNICATION**

The European Patent Office herewith transmits as an enclosure the European search report (under R. 44 or R. 45 EPC) for the above-mentioned European patent application.

If applicable, copies of the documents cited in the European search report are attached.

Additional set(s) of copies of the documents cited in the European search report is (are) enclosed as well.

The following specifications given by the applicant have been approved by the Search Division :

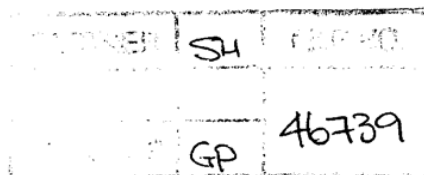
Abstract  Title

The abstract was modified by the Search Division and the definitive text is attached to this communication.

The following figure will be published together with the abstract : 2

**Refund of search fee**

If applicable under Article 10 Rules relating to fees, a separate communication from the Receiving Section on the refund of the search fee will be sent later.





DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	FOLEY JAMES ET AL: "Computer Graphics, Principles and Practice" COMPUTER GRAPHICS. PRINCIPLES AND PRACTICE, READING, ADDISON WESLEY, US, 1990, pages 873-899, XP002360077	1-7,9-24	G06T15/00
Y	* pages 874,876, *  * pages 887,888 *	1-3,8, 19-24	
X	CROCKETT T W: "An introduction to parallel rendering" PARALLEL COMPUTING ELSEVIER NETHERLANDS, vol. 23, no. 7, July 1997 (1997-07), pages 819-843, XP004729969 ISSN: 0167-8191	1-7, 9-18, 20-24	
Y	* pages 822-823 *  * page 828 *	1-3, 20-24	
Y	MONTRYM J S ET AL: "InfiniteReality: a real-time graphics system" COMPUTER GRAPHICS PROCEEDINGS, SIGGRAPH 97 ACM NEW YORK, NY, USA, 1997, pages 293-302, XP002360078 ISBN: 0-89791-896-7	1-3, 20-24	TECHNICAL FIELDS SEARCHED (IPC)
A	* pages 294-296; figures 1-3 *		G06T G09G
Y	US 6 424 345 B1 (SMITH WADE K ET AL) 23 July 2002 (2002-07-23) * column 3, line 31 - line 60; figure 7 * * column 6, line 20 - line 40 *  ----- -/--	1-3,8, 19-24	
The present search report has been drawn up for all claims			
Place of search <b>Munich</b>		Date of completion of the search <b>19 December 2005</b>	Examiner <b>Meinl, W</b>
CATEGORY OF CITED DOCUMENTS		T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document	
X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document			

EPO FORM 1503 03.82 (P04C01)



DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	HUMPHREYS G ET AL: "WireGL: a scalable graphics system for clusters" COMPUTER GRAPHICS PROCEEDINGS. SIGGRAPH 2001 ACM NEW YORK, NY, USA, 2001, pages 129-140, XP001049881 ISBN: 1-58113-374-X * abstract; figures 1,2,5 * * pages 129,130 * * page 135 *	1-3, 20-24	
			TECHNICAL FIELDS SEARCHED (IPC)
The present search report has been drawn up for all claims			
Place of search <b>Munich</b>		Date of completion of the search <b>19 December 2005</b>	Examiner <b>Meinl, W</b>
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document		T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document	

2  
EPO FORM 1503 03.82 (P04C01)

**ANNEX TO THE EUROPEAN SEARCH REPORT  
ON EUROPEAN PATENT APPLICATION NO.**

EP 03 25 7464

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on  
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

19-12-2005

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6424345	B1	23-07-2002	NONE

EPO FORM P0453

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

XP-002360077

SECOND EDITION IN C

# Computer Graphics

PRINCIPLES AND PRACTICE

**James D. Foley**

Georgia Institute of Technology

**Andries van Dam**

Brown University

**Steven K. Feiner**

Columbia University

**John F. Hughes**

Brown University



ADDISON-WESLEY PUBLISHING COMPANY

Reading, Massachusetts • Menlo Park, California • New York  
Don Mills, Ontario • Wokingham, England • Amsterdam • Bonn  
Sydney • Singapore • Tokyo • Madrid • San Juan • Milan • Paris

Summing the floating-point requirements for all of the geometry stages gives a total of 2,220,000 multiplications/divisions and 1,470,000 additions/subtractions per frame. Since a new frame is calculated every  $\frac{1}{10}$  second, a total of 22.2 million multiplications/divisions and 14.7 million additions/subtractions (36.9 million aggregate floating-point operations) as required per second—a very substantial number.

**Rasterization calculations and frame-buffer accesses.** Let us now estimate the number of pixel calculations and frame-buffer memory accesses required in each frame. We assume that  $z$  values and RGB triples each occupy one word (32 bits) of frame-buffer memory (typical in most current high-performance systems). For each pixel that is initially visible (i.e., results in an update to the frame buffer),  $z$ ,  $R$ ,  $G$ , and  $B$  values are calculated (4 additions per pixel if forward differences are used), a  $z$  value is read from the frame buffer (1 frame-buffer cycle), the  $z$  values are compared (1 subtraction) and new  $z$  values and colors are written (2 frame-buffer cycles). For each pixel that is initially not visible, only the  $z$  value needs to be calculated (1 addition), and a  $z$  value is read from the frame buffer (1 frame-buffer cycle), and the two  $z$  values are compared (1 subtraction). Note that initially visible pixels may get covered, but initially invisible pixels can never be exposed.

Since we assume that one-half of the pixels of each triangle are visible in the final scene, a reasonable guess is that three-quarters of the pixels are initially visible and one-quarter of the pixels are initially invisible. Each triangle covers 100 pixels, so  $\frac{3}{4} \cdot 100 \cdot 10,000 = 750,000$  pixels are initially visible and  $\frac{1}{4} \cdot 100 \cdot 10,000 = 250,000$  pixels are initially invisible. To display an entire frame, therefore, a total of  $(750,000 \cdot 5) + (250,000 \cdot 2) = 4.25$  million additions and  $(750,000 \cdot 3) + (250,000 \cdot 1) = 2.5$  million frame-buffer accesses is required. To initialize each frame, both color and  $z$ -buffers must be cleared, an additional  $1280 \cdot 1024 \cdot 2 = 2.6$  million frame-buffer accesses. The total number of frame-buffer accesses per frame, therefore, is  $2.5$  million +  $2.6$  million =  $5.1$  million. If 10 frames are generated per second, 42.5 million additions and 51 million frame-buffer accesses are required per second.

In 1989, the fastest floating-point processors available computed approximately 20 million floating-point operations per second, the fastest integer processors computed approximately 40 million integer operations per second, and DRAM memory systems had cycle times of approximately 100 nanoseconds. The floating-point and integer requirements of our sample application, therefore, are just at the limit of what can be achieved in a single CPU. The number of frame-buffer accesses, however, is much higher than is possible in a conventional memory system. As we mentioned earlier, this database is only modestly sized for systems available in 1989. In the following sections, we show how multiprocessing can be used to achieve the performance necessary to display databases that are this size and larger.

## 18.4 INTRODUCTION TO MULTIPROCESSING

Displaying large databases at high frame rates clearly requires dramatic system performance, both in terms 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. Likewise, rasterization can require more bandwidth into memory than a single memory system can provide. The only way to attain such performance levels is to



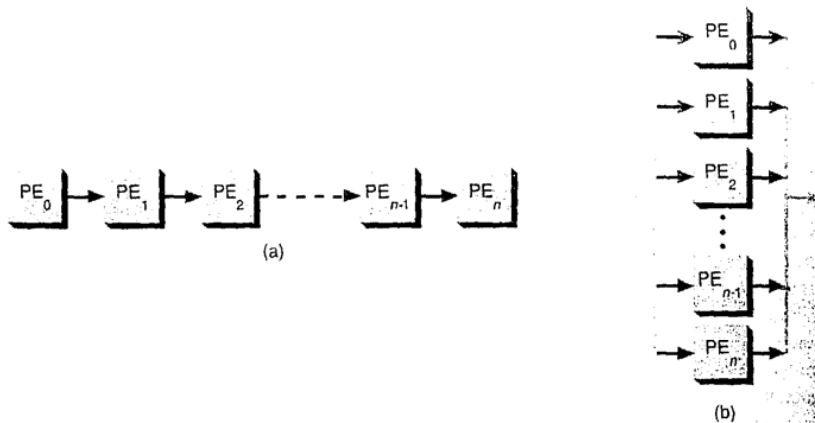


Fig. 18.9 Basic forms of multiprocessing: (a) pipelining, and (b) parallelism.

perform multiple operations concurrently and to perform multiple reads and writes to memory concurrently—we need concurrent processing.

Concurrent processing, or *multiprocessing*, is the basis of virtually all high-performance graphics architectures. Multiprocessing has two basic forms: *pipelining* and *parallelism* (we reserve the term *concurrency* for multiprocessing in general). A *pipeline processor* contains a number of *processing elements* (PEs) arranged such that the output of one becomes the input of the next, in pipeline fashion (Fig. 18.9a). The PEs of a *parallel processor* are arranged side by side and operate simultaneously on different portions of the data (Fig. 18.9b).

#### 18.4.1 Pipelining

To pipeline a computation, we partition it into stages that can be executed sequentially in separate PEs. Obviously, a pipeline can run only as fast as its slowest stage; so the processing load should be distributed evenly over the PEs. If this is not possible, PEs can be sized according to the jobs they must perform.

An important issue in pipeline systems is *throughput* versus *latency*. Throughput is the overall rate at which data are processed; latency is the time required for a single data element to pass from the beginning to the end of the pipeline. Some calculations can be pipelined using a large number of stages to achieve very high throughput. Pipeline latency increases with pipeline length, however, and certain computations can tolerate only a limited amount of latency. For example, real-time graphics systems, such as flight simulators, must respond quickly to changes in flight controls. If more than one or two frames are in the rendering pipeline at once, the system's interactivity may be impaired regardless of the frame rate.

### 18.4.2 Parallelism

To parallelize a computation, we partition the data into portions that can be processed independently by different PEs. Frequently, PEs can execute the same program. *Homogeneous* parallel processors contain PEs of the same type; *heterogeneous* parallel processors contain PEs of different types. In any parallel system, the overall computation speed is determined by the time required for the slowest PE to finish its task. It is important, therefore, to balance the processing load among the PEs.

A further distinction is useful for homogeneous parallel processors: whether the processors operate in *lock step* or independently. Processors that operate in lock step generally share a single code store and are called *single-instruction multiple-data* (SIMD) processors. Processors that operate independently must have a separate code store for each PE and are called *multiple-instruction multiple-data* (MIMD) processors.

**SIMD processors.** Because all the PEs in a SIMD processor share a single code store, SIMD processors are generally less expensive than MIMD processors. However, they do not perform well on algorithms that contain conditional branches or that access data using pointers or indirection. Since the path taken in a conditional branch depends on data specific to a PE, different PEs may follow different branch paths. Because all the PEs in a SIMD processor operate in lock step, they all must follow every possible branch path. To accommodate conditional branches, PEs generally contain an *enable register* to qualify write operations. Only PEs whose enable registers are set write the results of computations. By appropriately setting and clearing the enable register, PEs can execute conditional branches (see Fig. 18.10a).

Algorithms with few conditional branches execute efficiently on SIMD processors. Algorithms with many conditional branches can be extremely inefficient, however, since

<pre> statement 1; <b>if not condition then</b>   enable = FALSE; statement 2; <b>toggle enable;</b> statement 3; statement 4; enable = TRUE; statement 5; statement 6; </pre> <hr style="width: 100%;"/> <p>Total operations:  10 if condition evaluates TRUE,  10 if condition evaluates FALSE</p> <p style="text-align: center;">(a)</p>	<pre> statement 1; <b>if condition then</b>   statement 2; <b>else</b>   <b>begin</b>     statement 3;     statement 4;   <b>end;</b> statement 5; statement 6; </pre> <hr style="width: 100%;"/> <p>Total operations:  5 if condition evaluates TRUE,  6 if condition evaluates FALSE</p> <p style="text-align: center;">(b)</p>
---	---

**Fig. 18.10** (a) SIMD and (b) MIMD expressions of the same algorithm. In a SIMD program, conditional branches transform into operations on the enable register. When the enable register of a particular PE is FALSE, the PE executes the current instruction, but does not write the result.

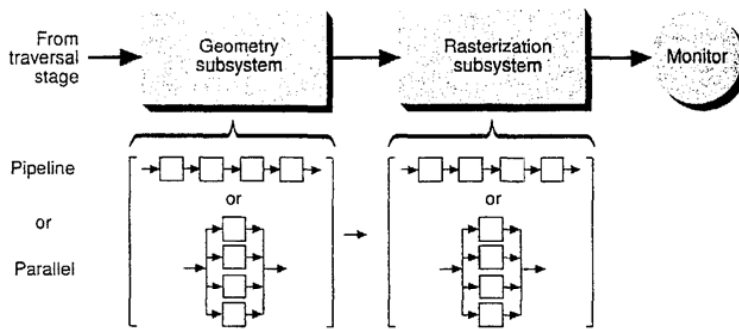
most PEs may be disabled at any given time. Data structures containing pointers (such as linked lists or trees) or indexed arrays cause similar problems. Since a pointer or array index may contain a different value at each PE, all possible values must be enumerated to ensure that each PE can make its required memory reference. For large arrays or pointers, this is an absurd waste of processing resources. A few SIMD processors provide separate address wires for each PE in order to avoid this problem, but this adds size and complexity to the system.

**MIMD processors.** MIMD processors are more expensive than SIMD processors, since each PE must have its own code store and controller. PEs in a MIMD processor often execute the same program. Unlike SIMD PEs, however, they are not constrained to operate in lock step. Because of this freedom, MIMD processors suffer no disadvantage when they encounter conditional branches; each PE makes an independent control-flow decision, skipping instructions that do not need to be executed (see Fig. 18.10b). As a result, MIMD processors achieve higher efficiency on general types of computations. However, since processors may start and end at different times and may process data at different rates, synchronization and load balancing are more difficult, frequently requiring FIFO buffers at the input or output of each PE.

### 18.4.3 Multiprocessor Graphics Systems

Pipeline and parallel processors are the basic building blocks of virtually all current high-performance graphics systems. Both techniques can be used to accelerate front-end and back-end subsystems of a graphics system, as shown in Fig. 18.11.

In the following sections, we examine each of these strategies. Sections 18.5 and 18.6 discuss pipeline and parallel front-end architectures. Sections 18.8 and 18.9 discuss pipeline and parallel back-end architectures. Section 18.10 discusses back-end architectures that use parallel techniques in combination.



**Fig. 18.11** Pipelining and parallelism can be used to accelerate both front-end and back-end portions of a graphics system.

## 18.5 PIPELINE FRONT-END ARCHITECTURES

Recall from Section 18.3 that the front end of a graphics display system has two major tasks: traversing the display model and transforming primitives into screen space. As we have seen, to achieve the rendering rates required in current applications, we must use concurrency to speed these computations. Both pipelining and parallelism have been used for decades to build front ends of high-performance graphics systems. Since the front end is intrinsically pipelined, its stages can be assigned to separate hardware units. Also, the large numbers of primitives in most graphics databases can be distributed over multiple processors and processed in parallel. In this section, we discuss pipeline front-end systems. We discuss parallel front-end systems in Section 18.6.

In introducing the standard graphics pipeline of Fig. 18.7, we mentioned that it provides a useful conceptual model of the rendering process. Because of its linear nature and fairly even allocation of processing effort, it also maps well onto a physical pipeline of processors. This has been a popular approach to building high-performance graphics systems since the 1960s, as described in [MYER68], a classic paper on the evolution of graphics architectures. Each stage of the pipeline can be implemented in several ways: as an individual general-purpose processor, as a custom hardware unit, or as a pipeline or parallel processor itself. We now discuss implementations for each stage in the front-end pipeline.

### 18.5.1 Application Program and Display Traversal

Some processor must execute the application program that drives the entire graphics system. In addition to feeding the graphics pipeline, this processor generally handles input devices, file I/O, and all interaction with the user. In systems using immediate-mode traversal, the display model is generally stored in the CPU's main memory. The CPU must therefore traverse the model as well as run the application. In systems using retained mode, the model is generally (but not always) stored in the display processor's memory, with the display processor performing traversal. Because such systems use two processors for these tasks, they are potentially faster, although they are less flexible and have other limitations, as discussed in Section 7.2.2.

Where very high performance is desired, a single processor may not be powerful enough to traverse the entire database with sufficient speed. The only remedy is to partition the database and to traverse it in parallel. This relatively new technique is discussed in Section 18.6.1.

### 18.5.2 Geometric Transformation

The geometric transformation stages (modeling transformation and viewing transformation) are highly compute-intensive. Fortunately, vector and matrix multiplications are simple calculations that require no branching or looping, and can readily be implemented in hardware.

The most common implementation of these stages is a single processor or functional unit that sequentially transforms a series of vertices. A pioneering processor of this type was the Matrix Multiplier [SUTH68], which could multiply a four-element vector by a

homogeneous transformation matrix in 20 microseconds. Other special-purpose geometry processors have been developed since then, most notably Clark's Geometry Engine, which can perform clipping as well (see Section 18.5.5). Recent geometry processors have exploited the power and programmability of commercial floating-point chips.

If pipelining does not provide enough performance, transformation computations can be parallelized in several ways:

- Individual components of a vertex may be calculated in parallel. Four parallel processors, each containing the current transformation matrix, can evaluate the expressions for  $x$ ,  $y$ ,  $z$ , and  $w$  in parallel.
- Multiple vertices can be transformed in parallel. If primitives are all of a uniform type—say, triangles—the three vertices of each triangle can be transformed simultaneously.
- Entire primitives can be transformed in parallel. If  $n$  transformation engines are available, each processor can transform every  $n$ th primitive. This technique has many of the advantages and disadvantages of parallel front-end systems, which we will discuss in Section 18.6.

### 18.5.3 Trivial Accept/Reject Classification

Trivial accept and reject tests are straightforward to implement, since they require at worst a dot product and at best a single floating-point comparison (or subtract) to determine on which side of each clipping plane each vertex lies. Because these tests require little computation, they are generally performed by the processor that transforms primitives.

### 18.5.4 Lighting

Like geometric transformation, lighting calculations are straightforward and are floating-point-intensive. A specialized hardware processor can calculate vertex colors based on a polygon's color and the light vector. More frequently, lighting calculations are performed using a programmable floating-point processor. In lower-performance systems, lighting calculations can be done in the same processor that transforms vertices. Note that if Phong shading is used, lighting calculations are deferred until the rasterization stage.

### 18.5.5 Clipping

Polygon clipping was once considered cumbersome, since the number of vertices can change during the clipping process and concave polygons can fragment into multiple polygons during clipping. Sutherland and Hodgman [SUTH74] showed that arbitrary convex or concave polygons can be clipped to a convex view volume by passing the polygon's vertices through a single processing unit multiple times. Each pass through the unit clips the polygon to a different plane. In 1980, Clark proposed unwrapping this processing loop into a simple pipeline of identical processors, each of which could be implemented in a single VLSI chip, which he named the *Geometry Engine* [CLAR82]. The Geometry Engine was general enough that it could transform primitives and perform perspective division as well.

Clipping using a Geometry Engine (or similar processor) can be performed either by a single processor that clips each polygon by as many planes as necessary, or by a pipeline of clipping processors, one for each clipping plane. The technique chosen affects the worst-case performance of the graphics system: Systems with only one clipping processor may bog down during frames in which large numbers of primitives need to be clipped, whereas systems with a clipping processor for each clipping plane can run at full speed. However, most of the clipping processors are idle for most databases and views in the latter approach.

General-purpose floating-point units recently have begun to replace custom VLSI transformation and clipping processors. For example, Silicon Graphics, which for many years employed custom front-end processors, in 1989 used the Weitek 3332 floating-point chip for transformations and clipping in their POWER IRIS system (described in detail in Section 18.8.2). The delicate balance between performance and cost now favors commodity processors. This balance may change again in the future if new graphics-specific functionality is needed and cannot be incorporated economically into general-purpose processors.

#### 18.5.6 Division by $w$ and Mapping to 3D Viewpoint

Like geometric transformation and lighting, the calculations in this stage are straightforward but require substantial floating-point resources. A floating-point divide is time consuming even for most floating-point processors (many processors use an iterative method to do division). Again, these stages can be implemented in custom functional units or in a commercial floating-point processor. In very high-performance systems, these calculations can be performed in separate, pipelined processors.

#### 18.5.7 Limitations of Front-End Pipelines

Even though pipelining is the predominant technique for building high-performance front-end systems, it has several limitations that are worth considering. First, a different algorithm is needed for each stage of the front-end pipeline. Thus, either a variety of hard-wired functional units must be designed or, if programmable processors are used, different programs must be written and loaded into each processor. In either case, processor or functional-unit capabilities must be carefully matched to their tasks, or bottlenecks will occur.

Second, since the rendering algorithm is committed to hardware (or at least to firmware, since few systems allow users to reprogram pipeline processors), it is difficult to add new features. Even if users have programming support for the pipeline processors, the distribution of hardware resources in the system may not adequately support new features such as complex primitives or collision detection between primitives.

A final shortcoming of pipelined front ends is that the approach breaks down when display traversal can no longer be performed by a single processor, and this inevitably occurs at some performance level. For example, if we assume that traversal is performed by a 20-MHz processor and memory system, that the description of each triangle in the database requires 40 words of data (for vertex coordinates, normal vectors, colors, etc.), and that each word sent to the pipeline requires two memory/processor cycles (one to read it

from memory, another to load it into the pipeline), then a maximum of  $20,000,000 / (2 - 40) = 250,000$  triangles per second can be displayed by the system, no matter how powerful the processors in the pipeline are. Current systems are rapidly approaching such limits.

What else can be done, then, to achieve higher performance? The alternative to pipelining front-end calculations is to parallelize them. The following section describes this second way to build high-performance front-end systems.

## 18.6 PARALLEL FRONT-END ARCHITECTURES

Since graphics databases are regular, typically consisting of a large number of primitives that receive nearly identical processing, an alternate way to add concurrency is to partition the data into separate streams and to process them independently. For most stages of the front-end subsystem, such partitioning is readily done, for example, the geometric transformation stages can use any of the parallel techniques described in Section 18.5.2. However, stages in which data streams diverge (display traversal) or converge (between the front end and back end) are problematic, since they must handle the full data bandwidth.

### 18.6.1 Display Traversal

Almost all application programs assume a single, contiguous display model or database. In a parallel front-end system, the simplest technique is to traverse the database in a single processor (serial traversal) and then to distribute primitives to the parallel processors. Unfortunately, this serial traversal can become the bottleneck in a parallel front-end system. Several techniques can be used to accelerate serial traversal:

- Traversal routines can be optimized or written in assembly code
- The database can be stored in faster memory (i.e., SRAM instead of DRAM)
- A faster traversal processor (or one optimized for the particular structure format) can be used.

If these optimizations are not enough, the only alternative is to traverse the database in parallel. The database either can be stored in a single memory system that allows parallel access by multiple processors (a shared-memory model), or can be distributed over multiple processors, each with its own memory system (a distributed-memory model).

The advantage of the shared-memory approach is that the database can remain in one place, although traversal must be divided among multiple processors. Presumably, each processor is assigned a certain portion of the database to traverse. Unfortunately, inherited attributes in a hierarchical database model mean that processors must contend for access to the same data. For example, each processor must have access to the current transformation matrix and to other viewing and lighting parameters. Since the data bandwidth to and from a shared-memory system may not be much higher than that of a conventional memory system, the shared-memory approach may not provide enough performance.

In the distributed-memory approach, each processor contains a portion of the database in its local memory. It traverses its portion of the database for each frame and may also perform other front-end computations. Distributing the database presents its own problems, however: Unless the system gives the application programmer the illusion of a contiguous database, it cannot support portable graphics libraries. Also, the load must be balanced

over the traversal processors if system resources are to be utilized fully. Hierarchical databases exacerbate both of these problems, since attributes in one level of a hierarchy affect primitives below them, and structures deep in a hierarchy may be referenced by multiple higher-level structure calls.

The following two sections examine two ways to distribute a hierarchical database over multiple processors: by structure, where each traversal processor is given a complete branch of the structure hierarchy; or by primitive, where each traversal processor is given a fraction of the primitives at each block in the hierarchy.

**Distributing by structure.** Distributing by structure is outwardly appealing, since state-changing elements in the structure apparently need to be stored only once. This can be an illusion, however, since multiple high-level structures may refer to the same lower-level substructure. For example, a database containing several cars, each described by a separate *car* structure, can be distributed by assigning each *car* structure to a separate processor. However, if each *car* structure refers to a number of *wheel* structures, *wheel* structures must also be replicated at every processor.

Load balancing among processors is also difficult. Since primitives in a structure are likely to be spatially coherent, changing the viewpoint or geometry within a scene may cause entire portions of the structure to be clipped or to reappear. Maintaining even loading among the multiple processors would require reassigning portions of the database dynamically.

**Distributing by primitive.** Distributing by primitive is costly, since the entire hierarchical structure of the database and any state-changing commands must be replicated at each processor. Structure editing is also expensive, since changes must be broadcast to every processor. Load balancing, however, is automatic. Since objects in a hierarchical database typically contain a large number of simple primitives (e.g., polygons forming a tiled surface), these primitives will be scattered over all the processors, and each processor will have a similar processing load.

Parallel display traversal is a relatively new technique. In 1989, the highest-performance architectures were just approaching the point where serial traversal becomes insufficient, and only a few systems had experimented with parallel traversal [FUCH89]. Neither of the distribution techniques for hierarchical databases that we have described is ideal. Compared to geometry processing, which easily partitions into parallel tasks, display traversal is much more difficult. Nevertheless, parallel traversal is likely to become increasingly important as system performance levels increase.

### 18.6.2 Recombining Parallel Streams

The transition between the front-end and back-end portions of the rendering pipeline is troublesome as well. In a parallel front-end system, the multiple streams of transformed and clipped primitives must be directed to the processor or processors doing rasterization. This can require sorting primitives based on spatial information if different processors are assigned to different screen regions.

A second difficulty in parallel front-end systems is that the ordering of data may change as those data pass through parallel processors. For example, one processor may transform two small primitives before another processor transforms a single, larger one. This does not



matter for many graphics primitives and rendering techniques. Certain global commands, however, such as commands to update one window instead of another or to switch between double buffers, require that data be synchronized before and after the command. If a large number of commands such as these occurs, some type of hardware support for synchronization may be necessary. A Raster Technologies system [TORB87] incorporates a special FIFO into each PE that stores tag codes for each command and allows commands to be resynchronized after they have been processed in separate PEs.

### **18.6.3 Pipelining versus Parallelism**

We have seen that both pipelining and parallelism can be used to build high-performance front-end subsystems. Although pipelining has been the predominant technique in systems of the last decade, parallelism offers several advantages, including reconfigurability for different algorithms, since a single processor handles all front-end calculations, and more modularity, since PEs in a parallel system can be made homogeneous more easily than in a pipeline system. Because the performance of a pipeline system is limited by the throughput of its slowest stage, pipelines do not scale up as readily as do parallel systems. Parallel systems, on the other hand, require more complicated synchronization and load balancing and cannot use specialized processors as well as can pipelined systems. Both designs are likely to be useful in the future; indeed, the highest-performance systems are likely to combine the two.

## **18.7 MULTIPROCESSOR RASTERIZATION ARCHITECTURES**

Recall that the output of the front-end subsystem is typically a set of primitives in screen coordinates. The rasterization (back-end) subsystem creates the final image by scan converting each of these primitives, determining which primitives are visible at each pixel, and shading the pixel accordingly. Section 18.2.4 identified two basic reasons why simple display-processor/frame-buffer systems are inadequate for high-performance rasterization subsystems:

1. A single display processor does not have enough processing power for all the pixel calculations.
2. Memory bandwidth into the frame buffer is insufficient to handle the pixel traffic—even if the display processor could compute pixels rapidly enough.

Much of the research in graphics architecture over the past decade has concerned ways to overcome these limitations. A great variety of techniques has been proposed, and many have been implemented in commercial and experimental systems. In this section, we consider low-cost, moderate-performance architectures that cast conventional algorithms into hardware. In Sections 18.8 and 18.9, we consider ways to improve performance by adding large amounts of parallelism to speed the calculation of the algorithm's "inner loop." In Section 18.10, we consider hybrid architectures that combine multiple techniques for improved efficiency or even higher performance. Figure 18.12 summarizes the concurrent approaches we shall discuss here.

Rasterization algorithm	Architectural technique		
	Serial pipeline	Highly parallel	Hybrid
Object order z-buffer, depth-sort, and BSP-tree algorithms	Pipelined object order Polygon/edge/span-processor pipeline	Image parallel Partitioned image memory Logic-enhanced memory	Virtual buffer/virtual processor  Parallel virtual buffer
Image order Scanline algorithms	Pipelined image order Scan-line pipeline	Object parallel Processor per primitive pipeline Tree-structured	Image composition

Fig. 18.12 Taxonomy of concurrent rasterization approaches.

### 18.7.1 Pipelined Object-Order Architectures

A direct way to add concurrency to rasterization calculations is to cast the various steps of a software algorithm into a hardware pipeline. This technique has been used to build a number of inexpensive, moderately high-performance systems. This approach can be used with either of the two main rasterization approaches: object order (z-buffer, depth-sort, and BSP-tree algorithms) and image order (scan-line algorithms). We consider object-order rasterization now and image-order rasterization in Section 18.7.2.

Object-order rasterization methods include the z-buffer, depth-sort, and BSP-tree algorithms (the z-buffer is by far the most common in 3D systems). The outer loop of these algorithms is an enumeration of primitives in the database, and the inner loop is an enumeration of pixels within each primitive. For polygon rendering, the heart of each of these algorithms is rasterizing a single polygon.

Figure 18.13 shows the most common rasterization algorithm for convex polygons. This algorithm is an extension of the 2D polygon scan-conversion algorithm presented in Section 3.6, using fixed-point arithmetic rather than integer arithmetic. Delta values are used to calculate the expressions for  $x$ ,  $z$ ,  $R$ ,  $G$ , and  $B$  incrementally from scan line to scan line, and from pixel to pixel. We shall describe each step of the algorithm.

**Polygon processing.** Computations performed only once per polygon are grouped into this stage. The first step is to determine the initial scan line intersected by the polygon (this is determined by the vertex with the smallest  $y$  value). In most cases, the polygon intersects this scan line at a single pixel, with two edges projecting upward, the *left* and *right* edges. Delta values are calculated for  $x$ ,  $z$ ,  $R$ ,  $G$ , and  $B$  for each edge. These delta values are sometimes called *slopes*.

**Edge processing.** Computations performed once for each scan line are grouped here. Scan lines within each primitive are processed one by one. The delta values computed previously are used to calculate  $x$ ,  $z$ ,  $R$ ,  $G$ , and  $B$  values at the intersection points of the left and right edges with the current scan line ( $P_{left}$  and  $P_{right}$  in the figure). A contiguous sequence of pixels on a scan line, such as those between  $P_{left}$  and  $P_{right}$ , is called a *span*. Delta

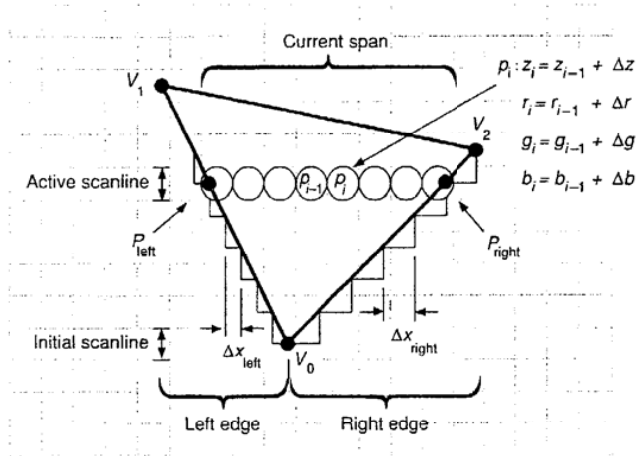


Fig. 18.13 Rasterizing a triangle. Each vertex ( $V_0$ ,  $V_1$ , and  $V_2$ ), span endpoint ( $P_{left}$  and  $P_{right}$ ), and pixel ( $p_0$ ,  $p_1$ , etc.) has  $z$ ,  $R$ ,  $G$ , and  $B$  components.

values for incrementing  $z$ ,  $R$ ,  $G$ , and  $B$  from pixel to pixel within the span are then calculated from the values at  $P_{left}$  and  $P_{right}$ .

**Span processing.** Operations that must be performed for each pixel within each span are performed here. For each pixel within the span,  $z$ ,  $R$ ,  $G$ , and  $B$  values are calculated by adding delta values to the values at the preceding pixel. The  $z$  value is compared with the previous  $z$  value stored at that location; if it is smaller, the new pixel value replaces the old one.

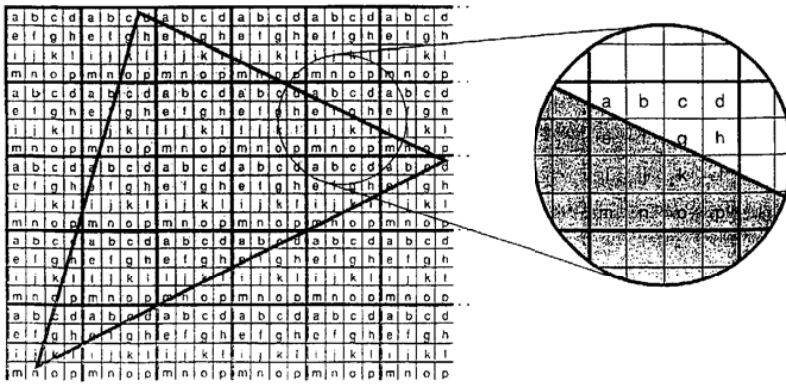


Fig. 18.14 A 4 × 4 interleaved memory organization. Each memory partition ("a" through "p") contains one pixel from each 4 × 4 block of pixels.

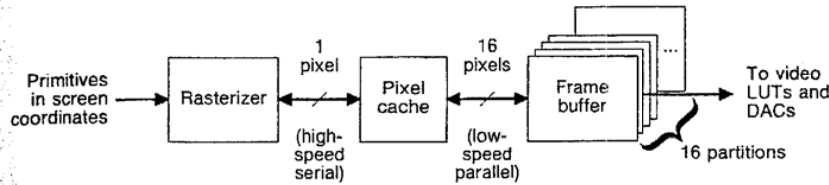


Fig. 18.15 A pixel cache matches bandwidth between a high-speed serial link to the rasterizer and a low-speed parallel link to the frame buffer.

A pipelined system containing one PE for each of the preceding three steps generates images dramatically faster than does a general-purpose display processor. In fact, it may generate pixels faster than a standard frame-buffer memory can handle. The Hewlett-Packard SRX [SWAN86], which uses this approach, generates up to 20 million pixels per second, approximately twice the speed of a typical VRAM memory system. In such systems, the rasterization bottleneck is access to frame-buffer memory.

**Pixel cache.** Pixels can be read and written faster if the frame buffer has some degree of parallel access. One way to accomplish this is to divide the memory into multiple—say, 16—partitions, each of which contains every fourth pixel of every fourth scan line, or perhaps every sixteenth pixel in every scan line (see Fig. 18.14). In this way, 16 pixels can be read or written in parallel. This technique, called *memory interleaving*, is also used in general-purpose CPU memory design.

A *pixel register* or *pixel cache* containing 16 pixels can be inserted between the rasterization pipeline and the interleaved image memory [GORI87; APGA88], as in Fig. 18.15. A cache allows the rasterizer to access individual pixels at high speed, assuming that the pixel is already in the cache. Multiple pixel values can be moved in parallel between the cache and the frame buffer at the slower speeds accommodated by the frame buffer.

As in any cache memory unit, performance depends on *locality of reference*, the principle that successive memory accesses are likely to occur in the same portion of memory. Erratic access patterns cause a high percentage of cache misses and degrade performance. For polygon rendering, the access pattern can be predicted precisely, since the extent of the polygon in screen space and the order in which pixels are generated are known before the pixels are accessed. Using this information, a cache controller can begin reading the next block of pixels from the frame buffer while the previous block of pixels is processed [APGA88].

Enhancing a rasterization subsystem with this kind of parallel-access path to frame-buffer memory may well increase the system throughput to the point where the bottleneck now becomes the single-pixel path between the rasterizer and the pixel cache. A logical next step is to enhance the rasterizer so that it can generate multiple pixel values in parallel. We consider such *image-parallel* architectures in Section 18.8.

### 18.7.2 Pipelined Image-Order Architectures

The alternative to object-order rasterization methods is *image-order* (or *scan-line*) rasterization, introduced in Section 15.4.4. Scan-line algorithms calculate the image pixel by pixel, rather than primitive by primitive. To avoid considering primitives that do not

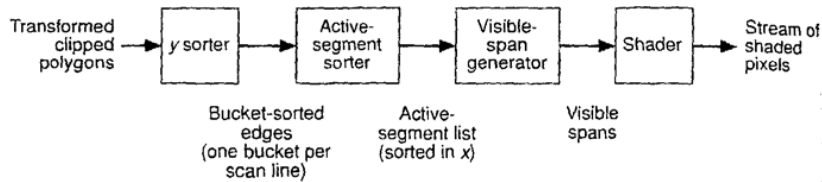


Fig. 18.16 Block diagram of a pipelined scan-line rasterizer.

contribute to the current scan line, most scan-line algorithms require primitives to be transformed into screen space and sorted into *buckets* according to the first scan line in which they each appear.

Scan-line algorithms can be implemented in hardware using the same approach as object-order algorithms: by casting the steps of the software algorithm into a pipelined series of hardware units. Much of the pioneering work on hardware scan-line systems was done at the University of Utah in the late 1960s [WYLI67; ROMN69; WATK70].

Figure 18.16 is a block diagram of a typical scan-line rasterizer. The *y sorter* places each edge of each polygon into the bucket corresponding to the scan line in which it first appears. The *active-segment generator* reads edges from these buckets, maintaining a table of active edges for the current scan line. From this table, it builds a list of active segments (a segment is a span within a single polygon), which is sorted by the *x* value of the left endpoint of each segment. The *visible-span generator* (called the *depth sorter* in the Utah system) traverses the active segment list, comparing *z* values where necessary, and outputs the sequence of visible spans on the current scan line. The *shader* performs Gouraud shading on these spans, producing a pixel stream that is displayed on the video screen.

Notice that no frame buffer is needed in this type of system, provided that the system can generate pixels at video rates. The original Utah scan-line system generated the video signal in real time for a modest number (approximately 1200) of polygons. However, since the rate at which pixels are generated depends on local scene complexity, a small amount of buffering—enough for one scan line, for example—averages the pixel rate within a single scan line. A double-buffered frame buffer allows complete independence of image-generation and image-display rates. This architecture was the basis of several generations of flight-simulator systems built by Evans & Sutherland Computer Corporation in the 1970s [SCHA83].

### 18.7.3 Limits of Pipeline Rasterization and the Need for Parallelism

Two factors limit the speedup possible in a pipeline approach. First, most rasterization algorithms break down easily into only a small number of sequential steps. Second, some of these steps are performed far more often than are others, particularly the steps in the inner loop of the rasterization algorithm. The processor assigned to these steps, therefore, becomes the bottleneck in the system.

The inner loop in an object-order (*z*-buffer) system is calculating pixels within spans; the inner loop in an image-order (scan-line) system is processing active edges on a scan line. For rasterization to be accelerated beyond the level possible by simple pipelining, these inner-loop calculations must be distributed over a number of processors. In *z*-buffer

systems, this produces image parallelism; in scan-line systems, this produces object parallelism. The following two sections discuss each of these approaches. Virtually all of today's high-performance graphics systems use some variation of them.

## 18.8 IMAGE-PARALLEL RASTERIZATION

Image parallelism has long been an attractive approach for high-speed rasterization architectures, since pixels can be generated in parallel in many ways. Two principal decisions in any such architecture are (1) how should the screen be partitioned? (into rows? into columns? in an interleaved pattern?), and (2) how many partitions are needed? In the following sections, we shall describe the most heavily investigated alternatives, discussing the advantages and disadvantages of each. Also, we shall identify which schemes are approaching fundamental limits in current architectures. Note that, because an image-parallel system rasterizes in object order, a frame buffer is required to store intermediate results.

### 18.8.1 Partitioned-Memory Architectures

Two obvious partitioning strategies are to divide pixels into contiguous blocks (Fig. 18.17a) [PARK80] and to divide them into an interleaved checkerboard pattern (Fig. 18.17b) [FUCH77a]. In either approach, a processor (or PE) is associated with each frame-buffer partition. Such organizations increase a graphics system's computation power by providing parallel processing, and its memory bandwidth by providing each PE with a separate channel into its portion of frame-buffer memory. During rasterization, polygons are transferred from the front end to the PEs in parallel, and each PE processes primitives in its portion of the frame buffer.

**Contiguous partitioning.** In the contiguous-region partitioning scheme, primitives need to be processed in only those regions in which they may be visible. These regions can be determined rapidly using geometric extents. If primitives are small compared to the region size, each primitive is likely to fall into a single region. Large primitives may fall into

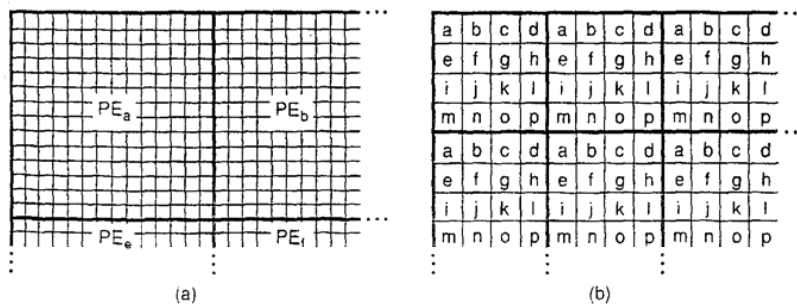


Fig. 18.17 Two schemes for frame-buffer partitioning. In (a), processors are assigned contiguous blocks of pixels; in (b), processors are assigned pixels in an interleaved pattern.

multiple regions. If the region size is chosen appropriately, the number of primitives handled by each processor will be approximately  $m/p$ , where  $m$  is the number of primitives in the database and  $p$  is the number of processors. Note, however, that if the viewpoint is chosen so unfortunately that all of the primitives fall into a single screen region, one processor must rasterize all of the primitives, and system performance decreases dramatically. In a contiguous region system, the frame rate is determined by the number of primitives in the busiest region.

**Interleaved partitioning.** Interleaved partitioning, on the other hand, achieves a better balance of workload, since all but the tiniest polygons lie in all partitions of the frame buffer. Since each processor handles every primitive (although only a fraction of its pixels), this scheme is less efficient in the best case than is the contiguous region approach. However, its worst-case performance is much improved, since it depends on the *total* number of primitives, rather than on the number in the busiest region. Because of this intrinsic load balancing, interleaved systems have become the dominant partitioned-memory architecture.

The polygon scan-conversion algorithm described in Section 18.7.1 requires set-up calculations to determine delta values and span endpoints before pixel computations can begin. These calculations need be performed only once per polygon or once per span, and can be shared among a number of PEs. The first proposed interleaved memory architectures [FUCH77a; FUCH79] contained no provision for factoring out these calculations from the PEs (see Fig. 18.18). Since each PE had to perform the entire rasterization algorithm for every polygon, many redundant calculations were performed.

Clark and Hannah [CLAR80] proposed an enhancement of this architecture to take advantage of calculations common to multiple PEs. In their approach, two additional levels of processors are added to perform polygon and edge processing. A single polygon processor receives raw transformed polygon data from the front-end subsystem and determines the polygon's initial scan line, slopes of edges, and so on. Eight edge processors (one per column in an  $8 \times 8$  grid of pixel processors) calculate  $x$ ,  $z$ ,  $R$ ,  $G$ , and  $B$  values at span endpoints. The edge processors send span information to the individual PEs (span processors), which interpolate pixel values along the span. The added levels of processing allow the PEs to perform only the calculations that are necessary for each pixel—a large improvement in efficiency. The rasterization portion of Silicon Graphics' recent high-performance systems uses this approach (see Section 18.8.2).

**SIMD versus MIMD.** A variation between systems of this type is whether PEs are SIMD or MIMD. Let us consider SIMD processors first. Figure 18.14 shows the mapping of processors to pixels in a  $4 \times 4$  interleaved scheme. With a SIMD processor, the 16 PEs work on a contiguous  $4 \times 4$  block of pixels at the same time. This arrangement is sometimes called a *footprint processor* because the  $4 \times 4$  array of processors (the footprint) marches across the polygon, stamping out 16 pixels at a time. Notice that, if any pixel of a  $4 \times 4$  block needs updating, the footprint must visit that block. For example, in the block of pixels shown in the inset of Fig. 18.14, processors  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $g$ , and  $h$  must disable themselves, while processors  $e$ ,  $f$ ,  $i$ ,  $j$ ,  $k$ ,  $l$ ,  $m$ ,  $n$ ,  $o$ , and  $p$  process their respective pixels.

A disadvantage of SIMD processors is that they do not utilize their PEs fully. This occurs for two reasons. First, many of the PEs map to pixels outside the current primitive if

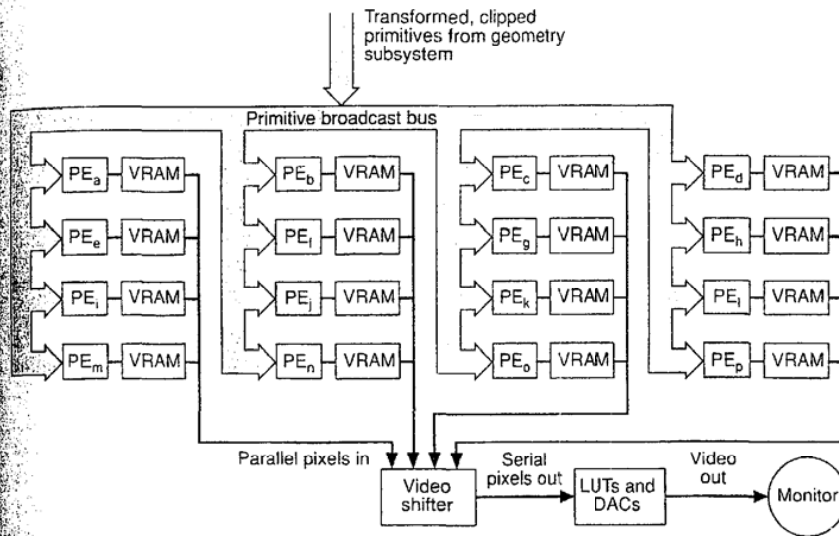


Fig. 18.18 Block diagram of a typical interleaved memory system. Each PE is responsible for one pixel in every  $4 \times 4$  block of pixels.

small primitives are being rendered. For example, PEs in a  $4 \times 4$  footprint processor rasterizing 100-pixel polygons map to pixels within the triangle as little as 45 percent of the time [APGA88]. Second, the choice of rasterization algorithm affects PE utilization. As remarked in Section 18.4.2, algorithms containing few conditional branches (including rasterizing convex polygons with Gouraud shading) can be implemented quite efficiently. Algorithms containing many conditional branches or using complicated data structures (such as rendering curved-surface primitives, texturing, shadowing, or antialiasing with a list of partially covering polygons) can be extremely difficult to make efficient. SIMD processors, however, can be built inexpensively and compactly, since a single code store and controller suffice for all the PEs. This offsets to some degree the poor PE utilization in a SIMD system. Several SIMD interleaved memory systems have been proposed and developed, including Gupta and Sproull's 8 by 8 Display [GUPT81b] and Stellar's GS2000 (see Section 18.11.3).

If we wish to support complex algorithms or to eliminate the idle processor cycles indicated in Fig. 18.14, we can add a control store to each PE, changing it from SIMD to MIMD. In a MIMD system, PEs do not need always to work on the same  $4 \times 4$  pixel block at the same time. If each PE has FIFO input buffering, PEs can even work on different primitives. The separate control stores, FIFO queues, and hardware required to synchronize PEs add size and complexity to the system. Examples of successful MIMD interleaved-memory systems include AT&T's Pixel Machine [MCM187] and Silicon Graphics' POWER IRIS (Section 18.8.2). Such systems compete well against SIMD systems on more complicated types of primitives or with more complicated rendering algorithms. For



example, AT&T's Pixel Machine Model PXM 924 ray traces simple scenes at interactive speeds—a feat unmatched by any SIMD system.

Since interleaved-memory architectures distribute the frame buffer over multiple processors, some provision must be made for scanning out pixels in the uninterrupted stream required by the video controller and display monitor. If the distributed frame buffer is constructed of VRAMs, this can be done in the manner shown in Fig. 18.18. Note that this is very similar to the technique described in Section 18.1.5 for implementing high-speed video scanout in a conventional frame-buffer memory.

**Further subdivision.** Suppose we wish to build a system with even higher performance. Since increasing the number of partitions in an interleaved-memory system increases the system's memory bandwidth and processing power, we might consider increasing the number of frame-buffer partitions—say, from 16 to 64, or even more. Unfortunately, the additional processors and datapaths required for each partition make such systems increasingly expensive.

An even more serious difficulty is supporting a larger number of partitions with the same number of frame-buffer memory chips. Each partition requires a minimum number of chips. For example, a partition with a 32-bit datapath between the PE and memory requires 8 4-bit wide chips, or 32 1-bit wide chips. Suppose we wish to build a 16-partition 1024-by-1024-pixel frame buffer with 128 bits per pixel. Using  $256K \times 4$  VRAM chips, each partition requires 8  $256K \times 4$  VRAM chips, so  $16 \cdot 8 = 128$  chips are needed to support all 16 memory partitions. This is the exact number required to store the pixel data.

Suppose, however, that we increase the number of partitions from 16 to 64 (an  $8 \times 8$  footprint). Although we still need only 128 memory chips to store the pixel data, we need  $64 \cdot 8 = 512$  memory chips to support the PE-memory bandwidth. The extra 384 memory chips are needed only to provide communication bandwidth—not for memory. This is an extra expense that continues to grow as we subdivide the frame buffer further.

Increasing the density of memory parts from 1 Mbit to 4 Mbit exacerbates this problem even further. For example, if  $1Mbit \times 4$  VRAM memory chips are used in the example mentioned above, 512 chips are still needed, even though each one contains sixteen times the memory actually required. Current systems such as the Silicon Graphics' POWER IRIS GTX (described in the next section), which uses 20 frame-buffer partitions, are already at the bandwidth limit. A way to ameliorate this problem would be for memory manufacturers to provide more data pins on high-density memory parts. Some 4-Mbit DRAM chips have eight data pins, rather than four, which helps somewhat, but only reduces the bandwidth problem by a factor of 2.

### 18.8.2 Silicon Graphics' POWER IRIS 4D/240GTX—An Interleaved Frame-Buffer Memory Architecture<sup>2,3</sup>

Silicon Graphics' POWER IRIS 4D/240GTX [AKEL88; AKEL89] uses many of the techniques described in this chapter. Like a number of its competitors, including the Ardent

<sup>2</sup>Material for this example is adapted from [AKEL88] and [AKEL89].

<sup>3</sup>In 1990, Silicon Graphics announced POWERVISION (similar to the GTX) that renders 1 million Gouraud-shaded triangles/sec and with 268 bits/pixel for antialiasing and texturing.

Titan [BORD89], the Megatek Sigma 70 [MEGA89], and the Stellar GS2000 (Section 18.11.3), the SGI POWER IRIS is a high-end graphics workstation, designed to combine general-purpose processing and high-speed 3D graphics for engineering and scientific applications.

The POWER IRIS has a powerful general-purpose CPU composed of four tightly coupled multiprocessors sharing a single memory bus. Its graphics subsystem can render over 100,000 full-colored, Gouraud-shaded,  $z$ -buffered quadrilaterals per second [AKEL89]. The POWER IRIS continues Silicon Graphics' tradition of immediate-mode display traversal, aggressive use of custom VLSI, hardware front-end pipeline, and interleaved-memory frame-buffer architecture. The POWER IRIS's architecture, diagrammed in Fig. 18.19, is composed of five major subsystems:

1. *CPU subsystem*—runs the application and traverses the display model
2. *Geometry subsystem*—transforms and clips graphical data to screen coordinates
3. *Scan-conversion subsystem*—breaks points, lines, and polygons into pixels
4. *Raster subsystem*—computes visibility and writes pixel data to frame buffer
5. *Display subsystem*—displays contents of frame buffer on color monitor.

**CPU subsystem.** The *CPU subsystem* runs the application and traverses the database. It is composed of four tightly coupled, symmetric, shared-memory multiprocessors. Hardware provides high-speed synchronization between processors, so parallelism can be achieved within a single process (although special programming constructs are required).

**Geometry subsystem.** The *geometry subsystem* transforms, clips, and lights primitives. It is composed of five floating-point processors arranged in a pipeline. Each of these processors, called a *geometry engine* (GE), contains an input FIFO, a controller, and a floating-point unit capable of 20 MFLOPS. Unlike Silicon Graphics' earlier Geometry Engine (see Section 18.5.4), the POWER IRIS' GEs are based on a commercial floating-point chip, the Weitek 3332.

The first GE transforms vertices and vertex normals. The second GE performs lighting calculations (supporting up to eight point light sources). The third GE performs trivial accept/reject clipping tests. The fourth GE performs exact clipping on primitives that cross clipping boundaries, and also does perspective division for all primitives. The fifth GE clips color components to maximum representable values, calculates depth-cued colors where necessary, and converts all coordinates to screen-space integers.

**Scan-conversion subsystem.** The *scan-conversion subsystem* rasterizes primitives using the pipeline approach described in Section 18.7.1, except that its spans are vertical columns of pixels, rather than the horizontal rows we have assumed so far (the only effect on the rasterization algorithm is that  $x$  and  $y$  coordinates are interchanged).

The single *polygon processor* sorts the vertices of each polygon from left to right in screen space. The sorted vertices are then used to decompose the polygon into vertically aligned trapezoids. The upper pair of vertices and the bottom pair of vertices of each trapezoid are used to calculate slopes for use by the edge processors.

The *edge processor* uses vertex and slope information to compute  $x$ ,  $y$ , and  $z$  coordinates and color values for each pixel that lies on the top or bottom edges of each

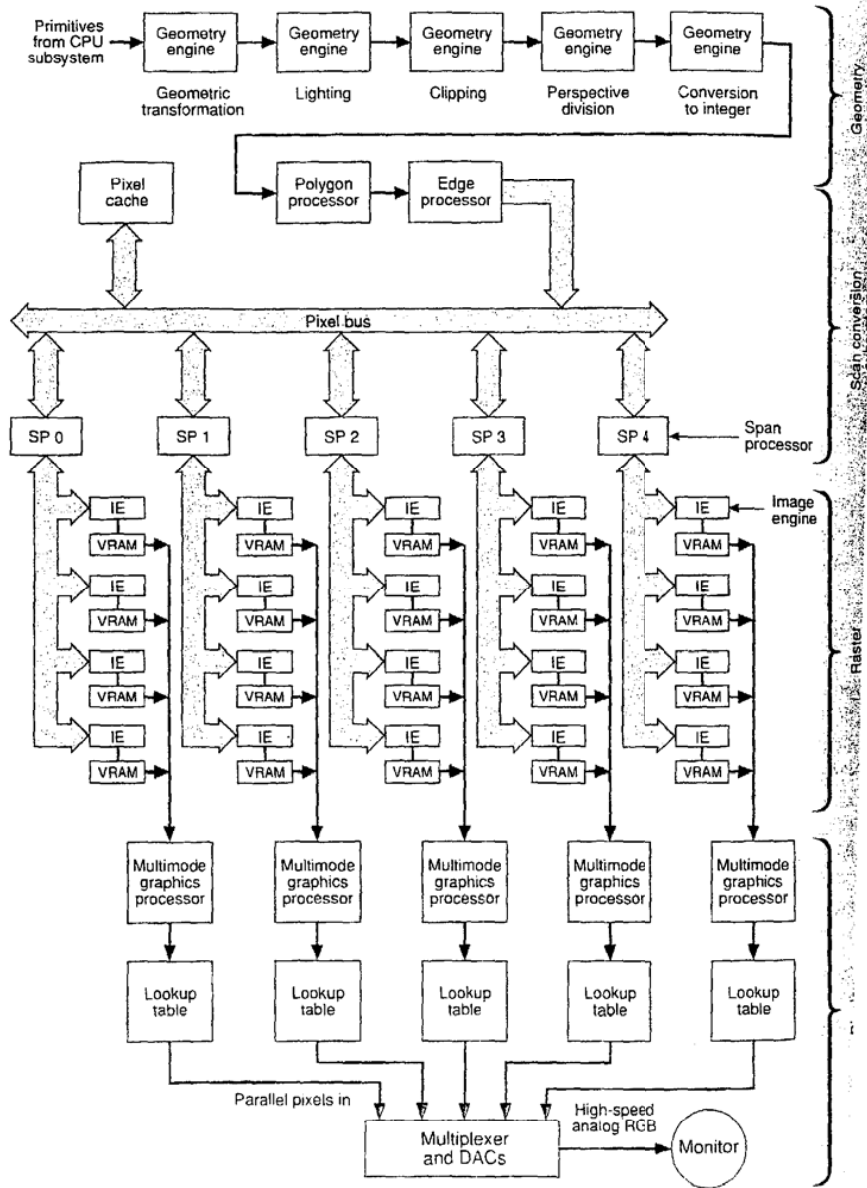


Fig. 18.19 Silicon Graphics' GTX system architecture (based on [AKEL88]).

trapezoid. In a given vertical column, a pair of pixels determines the top and bottom endpoints of a vertical span. These pixel pairs, together with slope information, are passed on to the span processors.

Each of the five parallel *span processors* is responsible for one-fifth of the columns of the display screen. For example, span processor 0 manages scan lines 0, 5, 10, and so on. Span processors calculate  $z$ ,  $R$ ,  $G$ ,  $B$ , and  $\alpha$  (for transparency and antialiasing) values for each pixel in the span. Because spans generated from a single polygon are adjacent, the processing load over span processors is approximately uniform.

The *pixel cache* buffers blocks of pixels during pixel copy operations so that the full bandwidth of the pixel bus can be used.

**Raster subsystem.** The *raster subsystem* takes pixel data generated by the span processors and selectively updates the image and  $z$  bitplanes of the frame buffer, using the results of a  $z$  comparison and  $\alpha$  blending. The raster subsystem is composed of 20 *image engines*, each responsible for one-twentieth of the screen's pixels, arranged in a  $4 \times 5$ -pixel interleaved fashion. 96 bits are associated with each pixel on the screen: two 32-bit image buffers ( $r$ ,  $G$ ,  $B$ , and  $\alpha$ ), a 24-bit  $z$ -buffer, four overlay/underlay bitplanes, and four window bitplanes.

The overlay/underlay bitplanes support applications that use pop-up menus or windowing backgrounds. The window bitplanes define the display mode (single- or double-buffered, etc.) and window-masking information.

The 20 image engines work on pixels in parallel. Each can blend values based on the pixel's  $\alpha$  value, allowing transparent or semitransparent objects to be displayed, and allowing supersampling antialiasing.

**Display subsystem.** The display subsystem contains five *multimode graphics processors* (MGPs), each assigned one-fifth of the columns in the display. The MGPs concurrently read image data from the frame buffer (together with window-display-mode bits), process them using the appropriate display mode (RGB or pseudocolor), and send them on to digital-to-analog converters for display.

The GTX's architecture is a good example of many of the techniques we have discussed so far. It provides high performance for polygon rendering at a reasonable cost in hardware. Because the GTX's rendering pipeline is highly specialized for graphics tasks, however, the system has difficulty with the advanced rendering techniques we shall discuss in Section 18.11, and its resources cannot be applied easily to nongraphics tasks. Section 18.11.3 discusses the architecture of Stellar's GS2000, which has complementary advantages and disadvantages.

### 18.8.3 Logic-Enhanced Memory

Since commercial memories may not support enough frame-buffer partitions, one might consider building custom memories with a large number of concurrently accessible partitions on a single chip. Since each (intrachip) partition must have its own connection to its associated (external) processor, extra pins must be added to each memory package to support these additional I/O requirements. Alternatively, multiple processors could be built

onto the chip itself. The first possibility—that of adding pins to memory chips—directly increases the memory bandwidth, but makes the chip package and associated circuit boards larger and more expensive, and also increases the power requirements. These packaging effects become progressively more severe as memory densities increase. (In the past two decades, the number of bits in a typical RAM has increased by a factor of 1000, while the size of the package and the number of pins have changed hardly at all.)

In this section, we shall concentrate on the second option—that of adding processing to multipartition memory chips. In the simplest schemes, only new addressing modes are provided, such as the ability to address an entire rectangle of memory pixels in parallel. At the other extreme, an entire microprocessor (including code store) could be provided for each internal partition of memory.

Before we describe specific logic-enhanced-memory approaches, let us consider the advantages and disadvantages of any logic-enhanced-memory scheme. First, adding logic or processing to memories has the potential to increase vastly the processing power within a system. By increasing the number of internal memory partitions and providing processing for each on the same chip, enormous processor/memory bandwidths can be achieved. Second, in custom VLSI chips, options become available that are impractical in board-level systems, since VLSI technology has an entirely different set of cost constraints for gates, wiring channels, and memory. Third, off-chip I/O bandwidth can potentially be reduced, since the only off-chip communication needed is to control the processor and to scan pixels out of the chip; this translates into fewer pins in the package and thus to a smaller package and less board space.

The principal disadvantages of an enhanced-memory approach are low memory densities and increased cost. With enormous production volumes, commercial DRAM manufacturers can afford to develop specialized, high-density fabrication capabilities and to incur large development costs to fine-tune their designs. Design and fabrication resources for custom memory chips, however, are generally more limited, resulting in densities lower than those of commercial RAMs. The price per chip is also high, since the costs for designing a custom VLSI chip are not offset by such large sales volumes. In spite of these disadvantages, at least one custom memory chip for graphics has become commercially successful—the VRAM. It remains to be seen whether other custom memory designs for graphics have sufficient market appeal to justify large-scale commercial development.

**Pixel-Planes.** An early and very general logic-enhanced-memory design is Pixel-Planes [FUCH81]. Pixel-Planes pushes frame-buffer subdivision to its extreme: It provides a separate processor for every pixel in the display. Each SIMD *pixel processor* is a 1-bit processor (ALU) with a small amount of memory. Figure 18.20 shows a block diagram of an enhanced-memory chip in Pixel-Planes 4, a prototype system completed in 1986 [EYLE88]. Its design is similar to that of the VRAM chip of Fig. 18.3, only here the 1-bit ALUs and associated circuitry replace the video shifter. Each enhanced-memory chip contains 128 pixels (columns in the memory array), and each pixel contains 72 bits of local memory (rows within the column).

Pixel-Planes' performance is not based simply on massive parallelism. If it was, each PE would have to perform all the operations for scan conversion independently, resulting in many redundant calculations and a grossly inefficient system. Rather, Pixel-Planes uses a

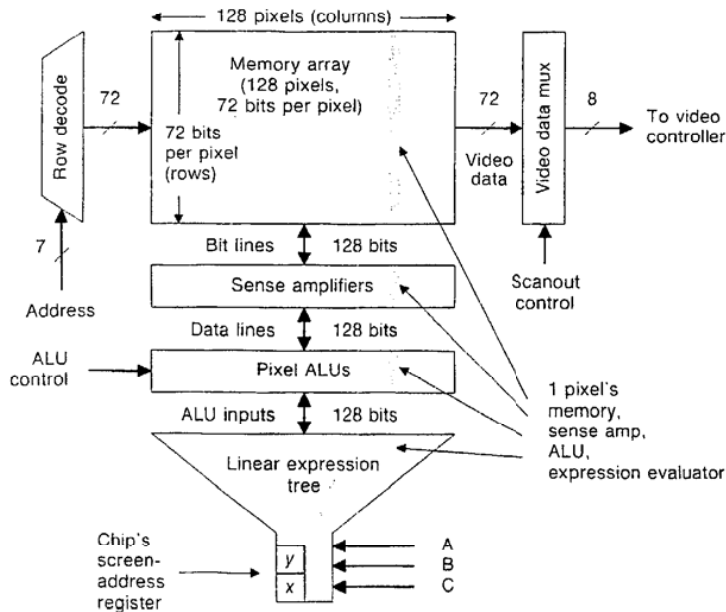


Fig. 18.20 Pixel-Planes 4 logic-enhanced-memory chip.

global computing structure called a *linear expression tree* that evaluates linear expressions of the form  $F(x, y) = Ax + By + C$  for every pixel  $(x, y)$  of the screen in parallel [FUCH85].  $A$ ,  $B$ , and  $C$  floating-point coefficients are input to the tree; each pixel receives its own value of  $F$  in its local memory, 1 bit per clock cycle (approximately 20–30 cycles are required for each linear expression). The linear expression tree is especially effective for accelerating rasterization calculations, since many of these can be cast as linear expressions.

For example,

- Each edge of a convex polygon can be described by a linear expression. All points  $(x, y)$  on one side of the edge have  $F(x, y) \geq 0$ ; all points on the other side of the edge have  $F(x, y) \leq 0$ .
- The  $z$  value of all points  $(x, y)$  within a triangle can be described as a linear expression.
- $R$ ,  $G$ , and  $B$  color components of pixels in a Gouraud-shaded triangle can be described as linear expressions.

Double-buffering is implemented within Pixel-Planes chips by providing a *video-data multiplexer* that reads pixel values from specific bits of pixel memory while the image is computed in the remaining bits. Video data are scanned out of the chip on eight video data pins.

Displaying images on Pixel-Planes requires modifying the algorithms we have assumed so far. In addition to transforming and clipping primitives in the usual manner, the

front-end subsystem must compute coefficient sets for the linear equations describing each primitive's edges,  $z$  values, and color values. Also, rasterization proceeds in parallel, since large areas of pixels can be affected at once using the linear expression tree. The following section describes a sample algorithm on Pixel-Planes.

**Rasterizing a triangle on Pixel-Planes.** Here, we briefly describe the algorithm to display Gouraud-shaded triangles on a Pixel-Planes system. Pixel-Planes can display more general polygons, although the algorithms are somewhat more complicated.

*Scan conversion.* Figure 18.21 shows the steps in scan converting a triangle. The first step is to enable all the pixel processors in the display. Edges are encoded as linear expressions  $F(x, y) = Ax + By + C = 0$ , as described previously. Each expression is then evaluated in parallel at every pixel in the screen, using the linear expression tree. Each pixel processor tests the sign of  $F$  to determine whether it lies on the proper side of the edge. If it lies outside the edge, the pixel processor disables itself by setting its enable bit to 0. After all the edges of a polygon have been tested, the only pixel processors still enabled are those lying within the polygon. These pixel processors alone participate in visibility and shading calculations.

*z-buffering.* After a polygon has been scan converted, Pixel-Planes evaluates the linear expression for  $z$  for all pixels in parallel. Each pixel processor compares this new  $z$  value with the one stored in its  $z$ -buffer. If the new  $z$  value is smaller, the current polygon is visible at the pixel; the pixel processor updates its  $z$ -buffer and remains enabled. If the new  $z$  value is larger, the pixel disables itself and does not participate in shading calculations.

*Gouraud shading.* The linear expressions for  $R$ ,  $G$ , and  $B$  components of the color are evaluated for each pixel in parallel by the linear expression tree. Pixel processors that are still enabled write the new color components into their pixels' color buffers.

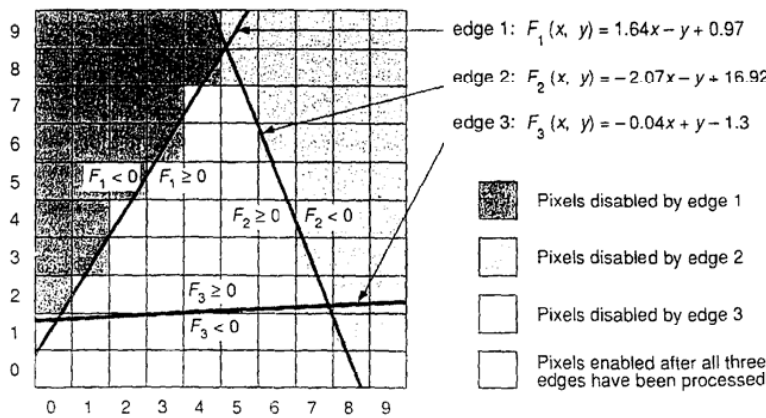


Fig. 18.21 Rasterizing a triangle on Pixel-Planes.

Note that scan conversion in Pixel-Planes is completely independent of a polygon's size, so a large polygon is rasterized as rapidly as a small one. Note, also, that operations that cannot be expressed as linear equations may take much longer to execute on Pixel-Planes than those that can (e.g., a quadratic expression can be calculated in pixel memory by multiplying values 1 bit at a time). Nevertheless, efficient algorithms for drawing spheres, casting shadows, antialiasing, and texture mapping have been developed for Pixel-Planes [FUCH85].

A full-scale system, Pixel-Planes 4, containing 262,144 processors forming a 512- by 512-pixel image, was completed in 1986. Although its performance was impressive for its time (40,000 Gouraud-shaded triangles of arbitrary size per second), it contained 2048 custom memory chips—a prohibitive expense for a commercial graphics system. This highlights the fundamental disadvantage of such a highly parallel SIMD approach: the very low utilization of pixel processors. Since all the PEs work in lock step, they cannot be retargeted to other tasks, even if only a few of them are still calculating useful results. This problem is especially severe when large numbers of small polygons are being drawn, because the first steps of the scan-conversion process disable almost all the screen's pixel processors.

Several logic-enhanced-memory graphics architectures have been developed that are more frugal in their use of silicon, at some sacrifice in either speed or generality. Although neither of the architectures described next directly supports the 3D rendering techniques assumed to this point, both provide very high performance in their respective domains (displaying 2D rectangles and generating 2D halftone images), and both provide insight into the potential of the logic-enhanced-memory approach.

**Rectangle area-filling memory chip.** Whelan proposed modifying the row and column addressing in the 2D memory-cell grid of a typical RAM to allow an entire rectangular region to be addressed at once [WHEL82]. Minimum and maximum row and column addresses specify the left, right, top, and bottom boundaries of the region. One write operation can store a single data value in every location within the region. This allows upright, constant-shaded rectangles to be rasterized in very few clock cycles—just enough to specify the four address values and the constant data.

**Scan Line Access Memory.** Demetrescu designed a more complicated chip called a Scan Line Access Memory (SLAM) for rasterizing more general 2D primitives [DEME85]. Like VRAMs and Pixel-Planes, SLAM takes advantage of the fact that, internally, a RAM reads or writes an entire row of its memory array in one cycle. Figure 18.22 shows a block diagram of a single SLAM chip. Each chip contains  $256 \times 64$  bits of frame-buffer memory. Each row of memory corresponds to 1 bit in a scan line of pixels. In a system with  $k$  bits per pixel, a SLAM chip can store up to  $64/k$  scan lines of 256 pixels each. In each memory cycle, a SLAM chip can read or write one row of memory from its memory array. By specifying appropriate  $x_{\min}$  and  $x_{\max}$  values, one can address any contiguous span of pixels on the current scan line, allowing fast polygon scan conversion. Video scanout is accomplished using a display shifter in exactly the same manner as a VRAM chip.

In a single clock cycle, either a row address, an  $x_{\min}$  value, an  $x_{\max}$  value, or a 16-bit repeating data pattern (for specifying halftone patterns) can be specified. Concurrent with



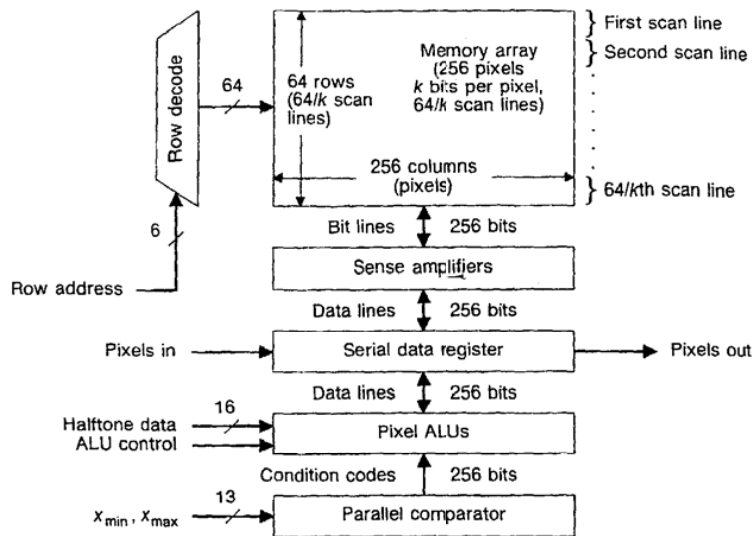


Fig. 18.22 Block diagram of a SLAM chip (for a system configured with  $k$  bits per pixel).

any one of these commands, SLAM can write the current scan-line segment into its memory array and can optionally increment the row address. Since, for many primitives, row addresses and halftone patterns need to be specified only once (for the initial scan line), succeeding scan lines can be processed rapidly. SLAM therefore can scan convert a convex polygon covering  $n$  scan lines in  $2n + 2$  cycles (four cycles to specify the row address,  $x_{\min}$ ,  $x_{\max}$ , and the halftone pattern for the initial scan line, and two cycles to specify  $x_{\min}$  and  $x_{\max}$  for each of the remaining  $n - 1$  scan lines).

A SLAM system is composed of a number of SLAM chips (the number depends on the dimensions of the display screen). Figure 18.23 shows a SLAM system for updating a 512 by 512 monochrome display screen. Systems with more bits per pixel require proportionately more SLAM chips. SLAM can also be extended to display Gouraud-shaded polygons by adding a Pixel-Planes-style linear-expression tree. However, this enhanced version of SLAM would require approximately the same amount of hardware as Pixel-Planes and would suffer the same low utilization of PEs, since the pixels in any given (small) primitive would likely be contained in just a few SLAM chips.

Although both Whelan's architecture and the original SLAM architecture use less hardware than does Pixel-Planes, they do not offer the generality necessary to render realistic 3D images. Pixel-Planes and the enhanced version of SLAM do offer this generality, but suffer poor PE utilization. It would be useful to gain the performance of these processor-per-pixel architectures, but with higher PE utilization for small primitives. Section 18.10.1 examines a way to accomplish this using enhanced-memory arrays smaller than the full screen size.

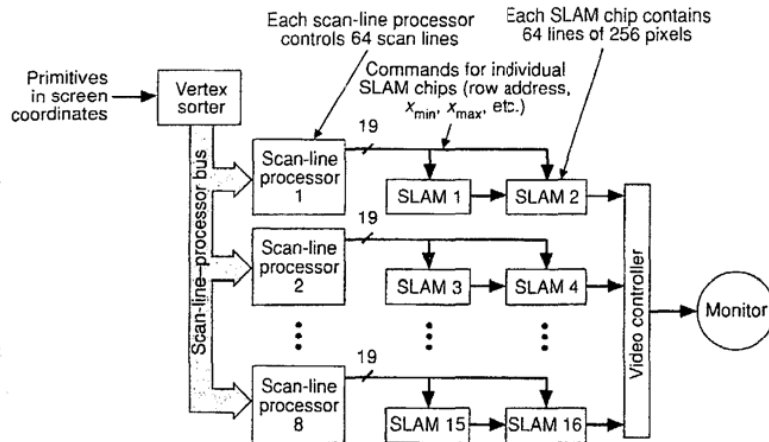


Fig. 18.23 SLAM system for updating a 512- by 512-pixel monochrome display.

## 18.9 OBJECT-PARALLEL RASTERIZATION

So far, we have focused on image-parallel architectures. The *object-parallel* family of parallel architectures parallelizes the inner loop of image-order (generally scan-line) algorithms. In an object-parallel architecture, multiple primitives (objects) are processed in parallel, so that final pixels may be generated more rapidly.

The usual object-parallel approach is to assign primitives (either statically or dynamically) to a number of homogeneous *object processors*, each of which can generate an entire image containing its primitive(s). During rasterization, each object processor enumerates the pixels of the display in some specific order (generally scan-line), generating color,  $z$ , and possibly partial-coverage values for its primitive(s). The pixel streams from each of the object processors are then combined to produce a single pixel stream for the final image. Although any number of primitives may be assigned to each object processor, most designs allocate a single primitive per processor. The advantage is that each processor can perform a well-defined task and thus can be reproduced inexpensively.

**General Electric's NASA II.** A pioneering real-time processor-per-primitive system was General Electric's NASA II flight simulator [BUNK89], delivered to NASA in 1967. The NASA II contained a number of hardware units called *face cards*, each of which rasterized a single polygon at video rates. At any given instant, each face card would process the same pixel.

The NASA II used a depth-sort visibility algorithm, rather than a  $z$ -buffer. The output of each face card included a bit indicating whether or not the pixel was covered by its polygon, the pixel color, and the polygon priority number. This information was fed into a priority multiplexer so that, at each pixel, the color of the highest-priority visible polygon was output. Since face cards were expensive, they were reassigned to new polygons when their polygons no longer intersected the current scan line. The NASA II system

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	1076773
<b>Application Number:</b>	10459797
<b>Confirmation Number:</b>	4148
<b>Title of Invention:</b>	Dividing work among multiple graphics pipelines using a super-tiling technique
<b>First Named Inventor:</b>	Mark M. Leather
<b>Customer Number:</b>	29153
<b>Filer:</b>	Christopher J. Reckamp/Christine Wright
<b>Filer Authorized By:</b>	Christopher J. Reckamp
<b>Attorney Docket Number:</b>	00100.02.0053
<b>Receipt Date:</b>	13-JUN-2006
<b>Filing Date:</b>	12-JUN-2003
<b>Time Stamp:</b>	14:27:17
<b>Application Type:</b>	Utility
<b>International Application Number:</b>	

### Payment information:

Submitted with Payment	no
------------------------	----

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part	Pages
1		10459797_Response.pdf	325832	yes	9

<b>Multipart Description</b>			
<b>Doc Desc</b>	<b>Start</b>	<b>End</b>	
Amendment After Final	1	1	
Claims	2	7	
Applicant Arguments/Remarks Made in an Amendment	8	9	

**Warnings:**

**Information:**

<b>Total Files Size (in bytes):</b>	325832
-------------------------------------	--------

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

**New Applications Under 35 U.S.C. 111**

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**REMARKS**

Claims 1-26 are now pending in the application. Applicants respectfully traverse and request reconsideration.

Claims 1-18, 20-26 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Furtner (U.S. Pat. No. 6,778,177).

With regard to claim 1, Furtner fails to show, teach, or suggest, inter alia, at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions.

Furtner is directed to a method for rasterizing a graphics basic component. The Examiner cites Figure 21b and Col. 1, lines 40-49, which is located in the “Background” section of Furtner, as disclosing at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions. However, this portion merely discloses a method of accelerating image-rendering of three-dimensional images. The method uses multi-processors or hardware pipelines in parallel. Each processor or pipeline may be responsible for rendering only a single contiguous block of pixels as illustrated in Figure 21a. As such, no processor or pipeline processes a repeated tile pattern as required by the claim. Figure 21b depicts a per pixel, not a repeating tile pattern, based processing scheme. For example, the processing of each individual pixel is effected in an interleaved manner by the processors or pipelines. Neither of these configurations process a set of tiles (or blocks of pixels) in a repeating pattern that includes a horizontally and vertically repeating pattern of square regions. Therefore, reconsideration and withdrawal of the rejection of claim 1 is respectfully requested.

Claims 20, 24, and 25 are allowable for at least similar reasons as claim 1. Thus, reconsideration and withdrawal of the rejections is respectfully requested.

Claims 2-19, 21-23, and 26 each ultimately depend on claims 1, 20, 24, and 25, respectively, and are therefore allowable for at least similar reasons and are believed to be allowable for having novel and non-obvious subject matter.


The Examiner states that claim 19 would be allowable if rewritten in independent form. Applicants have presently refrained from rewriting claim 19 in view of the discussion above. Applicants reserve the right to amend claim 19 into their originally allowable form at a later date if needed.

#### CONCLUSION

It is believed that all of the stated grounds of rejection have been properly traversed, accommodated, or rendered moot. Applicant therefore respectfully requests that the Examiner reconsider and withdraw all presently outstanding rejections. It is believed that a full and complete response has been made to the outstanding Office Action and the present application is in condition for allowance. Thus, prompt and favorable consideration of this response is respectfully requested. If the Examiner believes that personal communication will expedite prosecution of this application, the Examiner is invited to telephone the undersigned at (312) 609-7500.

Respectfully submitted,

Date: 6/13/06

By:   
Christopher J. Reckamp  
Registration No. 34,414

Vedder, Price, Kaufman & Kammholz, P.C.  
222 North LaSalle Street, Suite 2600  
Chicago, Illinois 60601  
phone: (312) 609-7599  
fax: (312) 609-5005

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicants: Mark M. Leather et al.  
Serial No.: 10/459,797  
Filing Date: June 12, 2003  
Confirmation No.: 4148

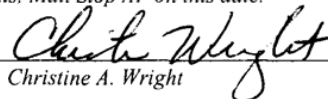
Examiner: Joni Hsu  
Art Unit: 2671  
Our File No.: 00100.02.0053

Title: **DIVIDING WORK AMONG MULTIPLE GRAPHICS PIPELINES USING  
A SUPER-TILING TECHNIQUE**

*Certificate of Electronic Submission*

Mail Stop AF  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

*I hereby certify that this Response is being forwarded via  
electronic submission to: Electronic Business Center,  
Commissioner for Patents, Mail Stop AF on this date.*

6-13-06        
Dated                      Christine A. Wright

**RESPONSE**

Dear Sir:

In response to the Final Office Action mailed March 13, 2006, Applicants submit the following response.

**Listing of the Claims** begins on page 2 of this paper.

**Remarks** begin on page 8 of this paper.

**LISTING OF THE CLAIMS**

1. (previously presented) A graphics processing circuit, comprising:  
  
at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile,  
  
wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions.
2. (original) The graphics processing circuit of claim 1, wherein the square regions comprise a two dimensional partitioning of memory.
3. (original) The graphics processing circuit of claim 2, wherein the memory is a frame buffer.
4. (original) The graphics processing circuit of claim 1, wherein each of the at least two graphics pipelines further includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process a portion of the pixel data.
5. (original) The graphics processing circuit of claim 4, wherein each of the at least two graphics pipelines further includes a scan converter, coupled to the back end circuitry, operative to determine the portion of the pixel data to be processed by the back end circuitry.
6. (original) The graphics processing circuit of claim 1, wherein each tile of the set of tiles further comprises a 16x16 pixel array.



7. (original) The graphics processing circuit of claim 4, wherein the at least two graphics pipelines separately receive the pixel data from the front end circuitry.

8. (original) The graphics processing circuit of claim 4, wherein the at least two graphics pipelines are on multiple chips.

9. (previously presented) The graphics processing circuit of claim 1, further including a memory controller coupled to the at least two graphics pipelines, operative to transfer pixel data between each of a first pipeline and a second pipeline and a memory.

10. (original) The graphics processing circuit of claim 4, wherein a first of the at least two graphics pipelines processes the pixel data only in a first set of tiles in the repeating tile pattern.

11. (original) The graphics processing circuit of claim 10, wherein the first of the at least two graphics pipelines further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the first set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the set of tiles is to be processed by the back end circuitry.

12. (previously presented) The graphics processing circuit of claim 1, wherein a second of the at least two graphics pipelines processes the data only in a second set of tiles in the repeating tile pattern.

13. (previously presented) The graphics processing circuit of claim 12, wherein the second of the at least two graphics pipelines further includes a scan converter, coupled to front

end circuitry and back end circuitry, operative to provide position coordinates of the pixels within the second set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the set of tiles is to be processed by the back end circuitry.

14. (original) The graphics processing circuit of claim 1 including a third graphics pipeline and a fourth graphics pipeline, wherein the third graphics pipeline includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process the pixel data in a third set of tiles in the repeating tile pattern, and wherein the fourth graphics pipeline includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process the pixel data in a fourth set of tiles in the repeating tile pattern.

15. (original) The graphics processing circuit of claim 14, wherein the third graphics pipeline further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the third set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the sets of tiles is to be processed by the back end circuitry.

16. (original) The graphics processing circuit of claim 14, wherein the fourth graphics pipeline further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the fourth set of tiles to

be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the sets of tiles is to be processed by the back end circuitry.

17. (original) The graphics processing circuit of claim 14, wherein the third and fourth graphics pipelines are on separate chips.

18. (original) The graphics processing circuit of claim 14, further including a bridge operative to transmit vertex data to each of the first, second, third and fourth graphics pipelines.

19. (original) The graphics processing circuit of claim 17 wherein the data includes a polygon and wherein each separate chip creates a bounding box around the polygon and wherein each corner of the bounding box is checked against a super tile that belongs to each separate chip and wherein if the bounding box does not overlap any of the super tiles associated with a separate chip, then the processing circuit rejects the whole polygon and processes a next one.

20. (previously presented) A graphics processing method, comprising:

receiving vertex data for a primitive to be rendered;

generating pixel data in response to the vertex data;

determining the pixels within a set of tiles of a repeating tile pattern corresponding to screen locations to be processed by a corresponding one of at least two graphics pipelines in response to the pixel data, the repeating tile pattern including a horizontally and vertically repeating pattern of square regions; and

performing pixel operations on the pixels within the determined set of tiles by the corresponding one of the at least two graphics pipelines.

21. (original) The graphics processing method of claim 20, wherein determining the pixels within a set of tiles of the repeating tile pattern to be processed further comprises determining the set of tiles that the corresponding graphics pipeline is responsible for.

22. (original) The graphics processing method of claim 20, wherein determining the pixels within a set of tiles of the repeating tile pattern to be processed further comprises providing position coordinates of the pixels within the determined set of tiles to be processed to the corresponding one of the at least two graphics pipelines.

23. (original) The graphics processing method of claim 20, further comprising transmitting the processed pixels to memory.

24. (previously presented) A graphics processing circuit, comprising:

front end circuitry operative to generate pixel data in response to primitive data for a primitive to be rendered;

first back end circuitry, coupled to the front end circuitry, operative to process a first portion of the pixel data in response to position coordinates;

a first scan converter, coupled between the front end circuitry and the first back end circuitry, operative to determine which set of tiles of a repeating tile pattern are to be processed by the first back end circuitry, the repeating tile pattern including a horizontally and vertically repeating pattern of square regions, and operative to provide the position coordinates to the first back end circuitry in response to the pixel data;

second back end circuitry, coupled to the front end circuitry, operative to process a second portion of the pixel data in response to position coordinates;

a second scan converter, coupled between the front end circuitry and the second back end circuitry, operative to determine which set of tiles of the repeating tile pattern are to be processed by the second back end circuitry, and operative to provide the position coordinates to the second back end circuitry in response to the pixel data; and

a memory controller, coupled to the first and second back end circuitry operative to transmit and receive the processed pixel data.

25. (previously presented) A graphics processing circuit, comprising:

at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile, wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of regions.

26. (previously presented) The graphics processing circuit of claim 25 wherein the horizontally and vertically repeating pattern of regions include NxM number of pixels.

# BEST AVAILABLE COPY

## PATENT APPLICATION FEE DETERMINATION RECORD

Effective January 1, 2003

Application or Docket Number

10459797

### CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	24	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	24 minus 20 =	4
INDEPENDENT CLAIMS	3 minus 3 =	0
MULTIPLE DEPENDENT CLAIM PRESENT	<input type="checkbox"/>	

SMALL ENTITY TYPE  OR OTHER THAN SMALL ENTITY

RATE	FEE		RATE	FEE
BASIC FEE	375.00	OR	BASIC FEE	750.00
X\$ 9=		OR	X\$18=	72
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL		OR	TOTAL	

\* If the difference in column 1 is less than zero, enter "0" in column 2

### CLAIMS AS AMENDED - PART II

1-506

	(Column 1)		(Column 2)		(Column 3)
AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR		PRESENT EXTRA
	Total	26	Minus	6	=
	Independent	4	Minus	9	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>					

SMALL ENTITY OR OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

6/13/00

	(Column 1)		(Column 2)		(Column 3)
AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR		PRESENT EXTRA
	Total	26	Minus	20	=
	Independent	4	Minus	4	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>					

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

	(Column 1)		(Column 2)		(Column 3)
AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR		PRESENT EXTRA
	Total		Minus		=
	Independent		Minus		=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>					

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

\* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.  
 \*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."  
 \*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."  
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.



UNITED STATES PATENT AND TRADEMARK OFFICE

*Handwritten mark*

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/459,797	06/12/2003	Mark M. Leather	00100.02.0053	4148
29153	7590	06/22/2006	EXAMINER	
ATI TECHNOLOGIES, INC. C/O VEDDER PRICE KAUFMAN & KAMMHOLZ, P.C. 222 N.LASALLE STREET CHICAGO, IL 60601			HSU, JONI	
			ART UNIT	PAPER NUMBER
			2628	

DATE MAILED: 06/22/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Advisory Action Before the Filing of an Appeal Brief</b>	<b>Application No.</b> 10459,797	<b>Applicant(s)</b> LEATHER ET AL.	
	<b>Examiner</b> Joni Hsu	<b>Art Unit</b> 2628	

**--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

THE REPLY FILED 13 June 2006 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1.  The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:

a)  The period for reply expires 3 months from the mailing date of the final rejection.

b)  The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SX MONTHS from the mailing date of the final rejection.

Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**NOTICE OF APPEAL**

2.  The Notice of Appeal was filed on \_\_\_\_\_. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

**AMENDMENTS**

3.  The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because

(a)  They raise new issues that would require further consideration and/or search (see NOTE below);

(b)  They raise the issue of new matter (see NOTE below);

(c)  They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or

(d)  They present additional claims without canceling a corresponding number of finally rejected claims.

NOTE: see attached sheet. (See 37 CFR 1.116 and 41.33(a)).

4.  The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).

5.  Applicant's reply has overcome the following rejection(s): \_\_\_\_\_.

6.  Newly proposed or amended claim(s) \_\_\_\_\_ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).

7.  For purposes of appeal, the proposed amendment(s): a)  will not be entered, or b)  will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.

The status of the claim(s) is (or will be) as follows:

Claim(s) allowed: \_\_\_\_\_.

Claim(s) objected to: 19.

Claim(s) rejected: 1-18 and 20-26.

Claim(s) withdrawn from consideration: \_\_\_\_\_.

**AFFIDAVIT OR OTHER EVIDENCE**

8.  The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).

9.  The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).

10.  The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

**REQUEST FOR RECONSIDERATION/OTHER**

11.  The request for reconsideration has been considered but does NOT place the application in condition for allowance because: see attached sheet.

12.  Note the attached Information Disclosure Statement(s). (PTO/SB/08 or PTO-1449) Paper No(s). 4/4/06

13.  Other: \_\_\_\_\_.



Applicant argues that Furtner (US006778177B1) does not teach at least two graphics pipeline operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions (page 8).

In reply, the Examiner disagrees. According to Applicant's disclosure, this repeating tile pattern for the two pipe configuration refers to a "checkerboard" pattern [0045] wherein each pipe is responsible for generating all of the pixels with its assigned tiles, and the tiles are distributed evenly across all pipes, in a checkerboard pattern [0047], and the patterns repeat across the whole screen, in both X & Y directions [0048], as shown in Figure 3. Furtner discloses subdividing a frame buffer into sub-sections which normally have the same size and associating each sub-section with a processor so that each of the processors is equally loaded (Col. 1, lines 23-32). Figure 21 shows two possibilities of partitioning the frame buffer with regard to the case of a graphics system operating with four graphics processing engines. Figure 21a shows that frame buffer 10 is subdivided into four equally sized blocks which are associated to the engines. Figure 21b shows that the processing of individual pixels 12 is effected in an interleaved manner by the four graphics processing engines of the graphics system (Col. 1, lines 37-49), and as can be seen in Figure 21, this results in a checkerboard pattern. Even though Figure 21b is described as partitioning the frame buffer into individual pixels, this is described as being only one possibility of partitioning the frame buffer (Col. 1, lines 37-39). Furtner describes subdividing a frame buffer into sub-sections which normally have the same size (Col. 1, lines 30-32) and does not specify the each sub-sections must contain only one pixel, and in fact describes that the sub-sections can be four equally sized blocks or tiles (Col. 1, lines 41-44).

Art Unit: 2628

Therefore, Furtner does disclose at least two graphics pipeline operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions.

  
ULKA CHAUHAN  
SUPERVISORY PATENT EXAMINER

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicants: Mark M. Leather et al.                      Examiner: Joni Hsu  
Serial No.: 10/459,797                                      Art Unit: 2671  
Filing Date: June 12, 2003                              Our File No.: 00100.02.0053  
Confirmation No.: 4148

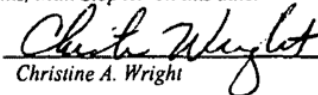
Title: **DIVIDING WORK AMONG MULTIPLE GRAPHICS PIPELINES USING  
A SUPER-TILING TECHNIQUE**

Certificate of Electronic Submission

Mail Stop AF  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

*I hereby certify that this Response is being forwarded via  
electronic submission to: Electronic Business Center,  
Commissioner for Patents, Mail Stop AF on this date.*

6-13-06  
Dated

  
Christine A. Wright

**RESPONSE**

Dear Sir:

In response to the Final Office Action mailed March 13, 2006, Applicants submit the following response.

**Listing of the Claims** begins on page 2 of this paper.

**Remarks** begin on page 8 of this paper.

Do not  
enter  
6/16/06  
JH

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO  <b>INFORMATION DISCLOSURE                  STATEMENT BY APPLICANT</b> <i>(Use as many sheets as necessary)</i>	<b>Complete if Known</b>	
	Application Number	10/459,797
	Filing Date	June 12, 2003
	First Named Inventor	Mark M. Leather
	Art Unit	2671
	Examiner Name	Joni Hsu
Sheet 1 of 2	Attorney Docket Number	00100.02.0053

U. S. PATENT DOCUMENTS					
Examiner Initials*	Cite No. <sup>1</sup>	Document Number	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code <sup>2</sup> (if known)			
JH		US- 6,424,345 B1	07-23-2002	Smith et al.	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS						
Examiner Initials*	Cite No. <sup>1</sup>	Foreign Patent Document	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T <sup>4</sup>
		Country Code <sup>3</sup> Number * Kind Code <sup>5</sup> (if known)				

Examiner Signature	/Joni Hsu/	Date Considered	06/14/2006
--------------------	------------	-----------------	------------

<sup>1</sup>EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 809. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. <sup>2</sup>Applicant's unique citation designation number (optional). <sup>3</sup>See Kinds Codes of USPTO Patent Documents at [www.uspto.gov](http://www.uspto.gov) or MPEP 901.04. <sup>4</sup>Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). <sup>5</sup>For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. <sup>6</sup>Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.18 if possible. <sup>7</sup>Applicant is to place a check mark here if English language translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Substitute for form 1449/PTO		<b>Complete if Known</b>	
		<b>Application Number</b>	10/459,797
<b>INFORMATION DISCLOSURE STATEMENT BY APPLICANT</b>		<b>Filing Date</b>	June 12, 2003
		<b>First Named Inventor</b>	Mark M. Leather
		<b>Art Unit</b>	2671
		<b>Examiner Name</b>	Joni Hsu
		<b>Attorney Docket Number</b>	00100.02.0053
Sheet	2	of	2
<i>(Use as many sheets as necessary)</i>			

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. <sup>1</sup>	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-Issue number(s), publisher, city and/or country where published.	T <sup>2</sup>
JH	—	European Search Report from European Patent Office; European Application No. 03257464.2; dated April 4, 2006	1
JH	—	FOLEY, James et al.; Computer Graphics, Principles and Practice; Addison-Wesley Publishing Company; 1990; pages 873-899	1
JH	—	CROCKETT, Thomas W.; An introduction to parallel rendering; Elsevier Science B.V.; 1997; pages 819-843	1
JH	—	MONTRYM, John S. et al.; InfiniteReality: A Real-Time Graphics System; Silicon Graphics Computer Systems; 1997; pages 293-302	1
JH	—	HUMPHREYS, Greg et al.; WireGL: A Scalable Graphics System for Clusters; ACM Siggraph; 2001; pages 129-140	1

<b>Examiner Signature</b>	/Joni Hsu/	<b>Date Considered</b>	06/14/2006
---------------------------	------------	------------------------	------------

\*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.  
 1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.  
 This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9198 (1-800-786-9198) and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

<b>NOTICE OF APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS AND INTERFERENCES</b>		Docket Number (Optional) 00100.02.0053	
I hereby certify that this correspondence is being forwarded via electronic submission to: Electronic Business Center, Commissioner for Patents, Mail Stop AF on <u>July 13, 2006</u> Signature <u><i>Christine Wright</i></u> Typed or printed name <u>Christine A. Wright</u>		In re Application of <b>Mark M. Leather et al.</b>	
		Application Number <b>10/459,797</b>	Filed <b>June 12, 2003</b>
		DIVIDING WORK AMONG MULTIPLE GRAPHICS PIPELINES FOR USING A SUPER-TILING TECHNIQUE	
		Art Unit <b>2628</b>	Examiner <b>Joni Hsu</b>
Applicant hereby <b>appeals</b> to the Board of Patent Appeals and Interferences from the last decision of the examiner.			
The fee for this Notice of Appeal is (37 CFR 41.20(b)(1))		\$ <u>500.00</u>	
<input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee shown above is reduced by half, and the resulting fee is:		\$ _____	
<input type="checkbox"/> A check in the amount of the fee is enclosed.			
<input type="checkbox"/> Payment by credit card. Form PTO-2038 is attached.			
<input type="checkbox"/> The Director has already been authorized to charge fees in this application to a Deposit Account. I have enclosed a duplicate copy of this sheet.			
<input checked="" type="checkbox"/> The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. <u>50-0441</u> . I have enclosed a duplicate copy of this sheet.			
<input checked="" type="checkbox"/> A petition for an extension of time under 37 CFR 1.136(a) (PTO/SB/22) is enclosed.			
<b>WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.</b>			
I am the			
<input type="checkbox"/> applicant/inventor.		<u><i>Christopher J. Reckamp</i></u> Signature	
<input type="checkbox"/> assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)		<b>Christopher J. Reckamp</b> Typed or printed name	
<input checked="" type="checkbox"/> attorney or agent of record. Registration number <u>34,414</u>		<u>312-609-7599</u> Telephone number	
<input type="checkbox"/> attorney or agent acting under 37 CFR 1.34. Registration number if acting under 37 CFR 1.34. _____		<u>July 13, 2006</u> Date	
NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.			
<input checked="" type="checkbox"/> *Total of <u>1</u> forms are submitted.			

This collection of information is required by 37 CFR 41.31. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11, 1.14 and 41.6. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	1112614
<b>Application Number:</b>	10459797
<b>Confirmation Number:</b>	4148
<b>Title of Invention:</b>	Dividing work among multiple graphics pipelines using a super-tiling technique
<b>First Named Inventor:</b>	Mark M. Leather
<b>Customer Number:</b>	29153
<b>Filer:</b>	Christopher J. Reckamp/Christine Wright
<b>Filer Authorized By:</b>	Christopher J. Reckamp
<b>Attorney Docket Number:</b>	00100.02.0053
<b>Receipt Date:</b>	13-JUL-2006
<b>Filing Date:</b>	12-JUN-2003
<b>Time Stamp:</b>	18:28:02
<b>Application Type:</b>	Utility
<b>International Application Number:</b>	

### Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	\$ 620
RAM confirmation Number	419
Deposit Account	500441

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:  
Charge any Additional Fees required under 37 C.F.R. Section 1.16 and 1.17

### File Listing:

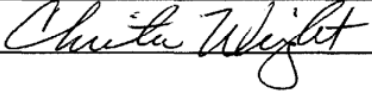

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part	Pages
1	Notice of Appeal Filed	10459797_NoticeofAppeal.pdf	87816	no	1
<b>Warnings:</b>					
<b>Information:</b>					
2	Miscellaneous Incoming Letter	10459797_PreAppealBriefRequest.pdf	72556	no	1
<b>Warnings:</b>					
<b>Information:</b>					
3	Miscellaneous Incoming Letter	10459797_Remarks.pdf	173743	no	4
<b>Warnings:</b>					
<b>Information:</b>					
4	Fee Worksheet (PTO-875)	fee-info.pdf	8337	no	2
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			342452		
<p>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b>  If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b>  If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p>					



Doc Code: AP.PRE.REQ

PTO/SB/33 (07-05)  
Approved for use through xx/xx/200x. OMB 0651-00xx  
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

<b>PRE-APPEAL BRIEF REQUEST FOR REVIEW</b>		Docket Number (Optional) 00100.02.0053	
I hereby certify that this correspondence is being forwarded via electronic submission to: Electronic Business Center, Commissioner for Patents, Mail Stop AF		Application Number 10/459,797	Filed June 12, 2003
on <u>July 13, 2006</u>		First Named Inventor Mark M. Leather	
Signature <u></u>		Art Unit 2628	Examiner Joni Hsu
Typed or printed name <u>Christine A. Wright</u>			
Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.			
This request is being filed with a notice of appeal.			
The review is requested for the reason(s) stated on the attached sheet(s). Note: No more than five (5) pages may be provided.			
I am the		<u></u> Signature	
<input type="checkbox"/>	applicant/inventor.	Christopher J. Reckamp Typed or printed name	
<input type="checkbox"/>	assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)		
<input checked="" type="checkbox"/>	attorney or agent of record.      34,414 Registration number _____	312-609-7599 Telephone number	
<input type="checkbox"/>	attorney or agent acting under 37 CFR 1.34. Registration number if acting under 37 CFR 1.34 _____	July 13, 2006 Date	
NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.			
<input checked="" type="checkbox"/>	*Total of <u>1</u> forms are submitted.		

This collection of information is required by 35 U.S.C. 132. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11, 1.14 and 41.6. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop AF, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicants: Mark M. Leather et al.  
Serial No.: 10/459,797  
Filing Date: June 12, 2003  
Confirmation No.: 4148

Examiner: Joni Hsu  
Art Unit: 2628  
Our File No.: 00100.02.0053

Title: **DIVIDING WORK AMONG MULTIPLE GRAPHICS PIPELINES USING  
A SUPER-TILING TECHNIQUE**

*Certificate of Electronic Submission*

Mail Stop AF  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

*I hereby certify that the this Remarks for Pre-Appeal Brief  
Request for Review is being forwarded via electronic submission  
to: Electronic Business Center, Commissioner of Patents, Mail  
Stop AF, on this date.*

7-13-06  
Date

  
Christine Wright

**REMARKS FOR PRE-APPEAL BRIEF REQUEST FOR REVIEW**

Dear Sir:

Applicants respectfully submit that the Examiner's rejections include clear errors because one or more limitations are not met by the cited reference and the reference does not teach what the Examiner alleges. Claims 1-18 and 20-26 stand rejected under 35 U.S.C. §102(e) as being anticipated by Furtner.

As to claim 1, Applicants claim a graphics processing circuit that includes at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile, wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions. The Furtner reference fails to teach the claimed subject matter.

Furtner instead describes a non-repeating tile pattern approach and alternatively a per pixel processing approach, neither of which anticipate the claimed subject matter. The cited

FIG. 21A shows a non-repeating tile based approach, and is the only tile based approach described by the cited portion of Furtner. This non-repeating tile based approach is different from the claimed approach in at least that the Furtner tile approach breaks down a frame into a non-repeating four tile configuration wherein each of one of four graphics engine processes a single tile. There is no repeating tile pattern utilized that includes horizontally and vertically repeating tile patterns wherein a respective graphics pipeline is operative to process data in repeating patterns of square regions. For example, only a single tile is processed by each engine for a frame as taught by Furtner. Furtner does not use a horizontally and vertically repeating tile pattern within a frame but only describes using a single tile per engine.

In contrast, as shown in Applicants' Specification and as claimed, Applicants' approach breaks down screen locations of a frame into a repeating tile pattern that includes horizontally and vertically repeating patterns of square regions where a respective graphics pipeline operates to process data in a dedicated tile of the repeating tile pattern. As such, one engine in Applicants' apparatus is configured to process multiple tiles in a repeating tile pattern to effect, among other things, improved loading. As admitted in the Advisory Action, FIG. 21A merely shows a frame buffer 10 subdivided into four equally sized blocks where each block is associated to one engine. There is no repeating pattern of horizontal and vertical tiles shown in FIG. 21A nor is there any description of an apparatus that operates or is configured as Applicants' claim requires.

The Advisory Action also appears to read information into the Furtner reference based on Applicants' own claimed invention. For example, when applying FIG. 21B of Furtner, which merely shows a non-tile based approach wherein each pixel is handled individually by a different graphics processing engine, the office action alleges that this is "only one possibility of

portioning the frame buffer”. Applicants respectfully note that the reference actually states that the two versions shown in FIG. 21A and FIG. 21B are actually “the above-described possibilities” (column 1, lines 37) meaning that these techniques are the ones described in the paragraphs above disclosed by Furtner. The reference cannot be cited for possibilities that are not disclosed in the reference.

In any event, the actual teaching in Furtner is as FIG. 21A and FIG. 21B show. FIG. 21B is a non-tile based approach whereas Applicants claim a repeating tile based approach and Furtner describes with respect to this figure that a separate engine handles a single pixel. As such, it is a per pixel approach and not a tile based approach.

In addition, the Advisory Action states Furtner describes “subdividing a frame buffer into subsections which normally have the same size (column 1, lines 30-32) and does not specify [that] each subsections must contain only one pixel, and in fact describes that the subsections can be four equally sized blocks or tiles (column 1, lines 41-44).” (Page 2 of Advisory Action). Applicants respectfully submit that the cited portion and, as admitted by the Advisory Action, requires a system that utilizes four equally sized blocks or tiles, each tile being handled by a different graphics processing engine and that only one tile per engine is (four blocks or tiles) described and shown. No repeating tile pattern per frame is employed. Although Furtner describes a tile based approach, he describes it as four tiles for a frame and each tile being processed by a different graphics processing engine. This is a non-repeating tile based approach.

In contrast, Applicants claim a repeating tile pattern approach wherein, among other things, respective graphics pipelines processed dedicated tiles of a repeating tile pattern. The only tile based approach taught in Furtner is a non-repeating tile approach. As such, the reference does not anticipate Applicants’ claimed invention. In addition, Applicants respectfully

note that the Advisory Action also appears to exclude claim language in an effort to render the claim unpatentable. As shown on page 3 of the Advisory Action, the Examiner's use of Applicants' claim language fails to include that a respective one of at least two graphics pipelines are operative to process data in a dedicated tile wherein there is a repeating tile pattern. As noted, there is no repeating tile pattern shown in the figures or described in the cited portion of Furtner.

Applicants respectfully reassert the relevant remarks made above with respect to other independent claims.

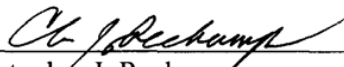
The dependent claims add additional novel and non-obvious subject matter.

As to claim 25, Applicants again respectfully submit that there is no repeating tile pattern that includes a horizontally and vertically repeating patterns of regions wherein graphics pipelines are operative to process data in corresponding sets of tiles of a repeating tile pattern corresponding to screen locations. As such, this claim is also believed to be in condition for allowance.

Reconsideration and withdrawal of the rejection of the claims is respectfully requested. A Notice of Allowance is also respectfully requested.

Respectfully submitted,

Date: 7/13/06

By:   
Christopher J. Reckamp  
Registration No. 34,414

Vedder, Price, Kaufman & Kammholz, P.C.  
222 North LaSalle Street, Suite 2600  
Chicago, Illinois 60601  
phone: (312) 609-7599  
fax: (312) 609-5005

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	10459797			
<b>Filing Date:</b>	12-Jun-2003			
<b>Title of Invention:</b>	Dividing work among multiple graphics pipelines using a super-tiling technique			
<b>First Named Inventor:</b>	Mark M. Leather			
<b>Filer:</b>	Christopher J. Reckamp/Christine Wright			
<b>Attorney Docket Number:</b>	00100.02.0053			
Filed as Large Entity				
<b>Utility Filing Fees</b>				
<b>Description</b>	<b>Fee Code</b>	<b>Quantity</b>	<b>Amount</b>	<b>Sub-Total in USD(\$)</b>
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
Notice of appeal	1401	1	500	500
<b>Post-Allowance-and-Post-Issuance:</b>				
<b>Extension-of-Time:</b>				

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Extension - 1 month with \$0 paid	1251	1	120	120
<b>Miscellaneous:</b>				
<b>Total in USD (\$)</b>				<b>620</b>



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/459,797	06/12/2003	Mark M. Leather	00100.02.0053	4148

29153                      7590                      01/16/2007  
ADVANCED MICRO DEVICES, INC.  
C/O VEDDER PRICE KAUFMAN & KAMM HOLZ, P.C.  
222 N.LASALLE STREET  
CHICAGO, IL 60601

EXAMINER
----------

HSU, JONI

ART UNIT	PAPER NUMBER
----------	--------------

2628


MAIL DATE	DELIVERY MODE
-----------	---------------

01/16/2007

PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.



<b>Application Number</b> 	<b>Application/Control No.</b> 10/459,797	<b>Applicant(s)/Patent under Reexamination</b> LEATHER ET AL.	
	Joni Hsu	<b>Art Unit</b> 2628	
<b>Document Code - AP.PRE.DEC</b>			

## Notice of Panel Decision from Pre-Appeal Brief Review



This is in response to the Pre-Appeal Brief Request for Review filed 7/13/06.

1.  **Improper Request** – The Request is improper and a conference will not be held for the following reason(s):

- The Notice of Appeal has not been filed concurrent with the Pre-Appeal Brief Request.
- The request does not include reasons why a review is appropriate.
- A proposed amendment is included with the Pre-Appeal Brief request.
- Other:

The time period for filing a response continues to run from the receipt date of the Notice of Appeal or from the mail date of the last Office communication, if no Notice of Appeal has been received.


2.  **Proceed to Board of Patent Appeals and Interferences** – A Pre-Appeal Brief conference has been held. The application remains under appeal because there is at least one actual issue for appeal. Applicant is required to submit an appeal brief in accordance with 37 CFR 41.37. The time period for filing an appeal brief will be reset to be one month from mailing this decision, or the balance of the two-month time period running from the receipt of the notice of appeal, whichever is greater. Further, the time period for filing of the appeal brief is extendible under 37 CFR 1.136 based upon the mail date of this decision or the receipt date of the notice of appeal, as applicable.


- The panel has determined the status of the claim(s) is as follows:
- Claim(s) allowed: \_\_\_\_\_.
  - Claim(s) objected to: \_\_\_\_\_.
  - Claim(s) rejected: \_\_\_\_\_.
  - Claim(s) withdrawn from consideration: \_\_\_\_\_.


3.  **Allowable application** – A conference has been held. The rejection is withdrawn and a Notice of Allowance will be mailed. Prosecution on the merits remains closed. No further action is required by applicant at this time.

4.  **Reopen Prosecution** – A conference has been held. The rejection is withdrawn and a new Office action will be mailed. No further action is required by applicant at this time.

All participants:

(1) Ulka Chauhan. 

(3) Joni Hsu. 

(2) Kee Tung. 

(4) \_\_\_\_\_.

## EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	402	345/506.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L2	319	345/530.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L3	319	345/505.ccls..	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L4	47	345/588.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L5	105	345/544.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L6	517	345/545.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03
L7	79	345/532.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:03

## EAST Search History

L8	805	345/501.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:04
L9	405	345/502.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:04
L10	664	345/531.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:04
L11	1652	til\$3 and repeat\$3 and pattern\$3 and pixel\$1 and pipelin\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:05
L12	1336	til\$3 and repeat\$3 and pattern\$3 and pixel\$1 and horizontal\$2 and vertical\$2 and pipelin\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:05
L13	147	til\$3 and repeat\$3 same (til\$3 pattern\$3) same pixel\$1 and pattern\$3 and horizontal\$2 and vertical\$2 and pipelin\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:06
L14	111	til\$3 and repeat\$3 with (til\$3 pattern\$3) same pixel\$1 and pattern\$3 and horizontal\$2 and vertical\$2 and pipelin\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:07

## EAST Search History

L16	258	scan adj conver\$5 and pixel\$1 and til\$3 and pipelin\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:11
L17	356	(multiple plurality) same pipelin\$3 and scan adj conver\$5	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:12
L18	106	(multiple plurality) near2 pipelin\$3 and scan adj conver\$5	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:12
L19	101	18 and pixel\$1	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:12
L20	95	non adj square same til\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:13
L21	39	repeat\$3 and til\$3 and 20	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:14
L22	22	parallel adj processor\$1 with graphic\$1 and pixel\$1 and (til\$3 block\$1) and repeat\$3 and pattern\$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:16

### EAST Search History

L23	78	parallel adj processor\$1 with graphic\$1 and pixel\$1 and (til\$3 block\$1)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/24 10:16
-----	----	--	--	----	----	------------------



UNITED STATES PATENT AND TRADEMARK OFFICE

*ml*

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.			
10/459,797	06/12/2003	Mark M. Leather	00100.02.0053	4148			
29153	7590	02/09/2007	<table border="1"> <tr><td>EXAMINER</td></tr> <tr><td>HSU, JONI</td></tr> </table>		EXAMINER	HSU, JONI	
EXAMINER							
HSU, JONI							
ADVANCED MICRO DEVICES, INC. C/O VEDDER PRICE KAUFMAN & KAMMHOLZ, P.C 222 N.LASALLE STREET CHICAGO, IL 60601			<table border="1"> <tr> <th>ART UNIT</th> <th>PAPER NUMBER</th> </tr> <tr> <td>2628</td> <td></td> </tr> </table>	ART UNIT	PAPER NUMBER	2628	
ART UNIT	PAPER NUMBER						
2628							

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/09/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

<b>Office Action Summary</b>	<b>Application No.</b> 10/459,797	<b>Applicant(s)</b> LEATHER ET AL.	
	<b>Examiner</b> Jcni Hsu	<b>Art Unit</b> 2628	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1)  Responsive to communication(s) filed on 13 July 2006.
- 2a)  This action is FINAL.
- 2b)  This action is non-final.
- 3)  Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4)  Claim(s) 1-26 is/are pending in the application.
  - 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5)  Claim(s) \_\_\_\_\_ is/are allowed.
- 6)  Claim(s) 1-26 is/are rejected.
- 7)  Claim(s) \_\_\_\_\_ is/are objected to.
- 8)  Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9)  The specification is objected to by the Examiner.
- 10)  The drawing(s) filed on \_\_\_\_\_ is/are: a)  accepted or b)  objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11)  The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12)  Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
  - a)  All   b)  Some \*   c)  None of:
    - 1.  Certified copies of the priority documents have been received.
    - 2.  Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
    - 3.  Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1)  Notice of References Cited (PTO-892)
- 2)  Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3)  Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 4)  Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_.
- 5)  Notice of Informal Patent Application
- 6)  Other: \_\_\_\_\_.

**DETAILED ACTION**

*Response to Amendment*

1. Applicant's arguments with respect to claims 1-18 and 20-26 have been considered but are moot in view of the new ground(s) of rejection.
  
2. Applicant's arguments, see pages 1-3, filed July 13, 2006, with respect to the rejection(s) of claim(s) 1-7 and 20-26 under 35 U.S.C. 102(e) and Claims 8-18 under 35 U.S.C. 103(a) have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of Kelleher (US005794016A).
  
3. Applicant argues that Furtner (US006778177B1) describes a non-repeating tile pattern approach and alternatively a per pixel processing approach, neither of which anticipate the claimed subject matter (page 1). The cited FIG. 21A shows a non-repeating tile based approach, and is the only tile based approach described by the cited portion of Furtner. Furtner describes that the per pixel processing approach is repeating. However, Furtner does not teach a repeating tile based approach. The Examiner attempted to cite the reference for possibilities that are not disclosed in the reference (pages 2-3).

In reply, the Examiner agrees. However, new grounds of rejection are made in view of Kelleher.



*Claim Objections*

4. Claim 25 is objected to because it is exactly the same as Claim 1, and therefore is a repeated claim. Appropriate correction is required.

*Claim Rejections - 35 USC § 101*

5. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 20-23 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claim 20 recites a graphics processing method, however it appears to be directed to an abstract idea rather than a practical application of the abstract idea. The claimed invention as a whole must accomplish a practical application. That is, it must produce a “useful, concrete and tangible result (*State Street*, 149 F.3d at 1373, 47 USPQ2d at 1601-02). The tangible requirement requires that the claim must set forth a practical application of the 101 judicial exception to produce a real-world result (*Benson*, 409 U.S. at 71-72, 175 USPQ at 676-77). See MPEP 2106 II A. Since there is no tangible result recited in these claims, these claims are directed to non-statutory subject matter.

Claims 21-23 are non-statutory for the same reasons discussed above.

*Claim Rejections - 35 USC § 102*

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1-5, 7, 9, 10, 12-16, 18, and 20-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Kelleher (US005794016A).

9. With regard to Claim 1, Kelleher discloses a graphics processing circuit (10C, Figure 3) comprising at least two graphics pipelines (20A, 20B; *graphics system 10C with N rendering processors 20A-20N*, Col. 3, lines 22-23; *rendering processor 20 provides a video pipeline*, Col. 4, lines 9-14) operative to process data in a corresponding set of tiles (group of pixel blocks 52, Figure 4) of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile, wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions, as shown in Figure 4 (*graphics memory 22 that has been partitioned into a plurality of pixel blocks 52 that are tiled in the x- and y-direction of the graphics memory 22, the graphics memory 22 renders a 1280x1024 screen display, pixel blocks 52 are organized into noncontiguous groups of blocks 52, groups of blocks 52 are then assigned to the rendering processors 20, each rendering processor 20 writes only those pixels that are located in the blocks 52 of the assigned groups*, Col. 4, line 60-Col. 5, line 19).

10. With regard to Claim 2, Kelleher discloses that the square regions (blocks 52) comprise a two dimensional partitioning of memory (22, Figure 4) (*graphics memory 22 that has been partitioned into a plurality of pixel blocks 52 that are tiled in the x- and y-direction of the graphics memory 22, Col. 4, lines 60-62*).

11. With regard to Claim 3, Kelleher discloses that the memory (22, Figure 4) is a frame buffer (*graphics memory 22, also known as a frame buffer, Col. 3, lines 38-41*).

12. With regard to Claim 4, Kelleher discloses that each of the at least two graphics pipelines (20A, 20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) further includes front end circuitry (80, 82, Figure 7) operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry (84, Figure 7), coupled to the front end circuitry, operative to receive and process a portion of the pixel data (*each of the rendering processors 20 independently scan-converts the geometric objects, the rendering processor 20 first reads the command packets, the rendering processors 20 then processes the command packets in a pipeline process comprising a dispatch stage 80, a setup stage 82, and an updated stage 84, in a dispatch stage 80, the dispatch circuit 64 reads the command packets from the command queue 62 and dispatches the vertex data in the command to the next stage in the pipeline, the setup stage 82, the setup stage 82 includes the geometric setup circuit 66 and the attribute setup circuit 70 which accept the triangle vertex data and setup the triangles for scan-conversion by the*

*update stage 84, the update stage 84 includes the interpolator circuit 72, which interpolates final attribute values for the pixels in each triangle, Col. 8, line 52-Col. 9, line 23).*

13. With regard to Claim 5, Kelleher discloses that each of the at least two graphics pipelines (20A, 20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) further includes a scan converter (84, Figure 7), coupled to the back end circuitry, operative to determine the portion of the pixel data to be processed by the back end circuitry (*scan-converts the geometric objects into the memory blocks 52 indicated by their block enable field 61, Col. 8, lines 52-61; scan-conversion by the update stage 84, Col. 9, lines 1-23; block enable field 61 determines which groups of blocks 52 within the graphics memory 22 are allocated to and controlled by the rendering processor 20, Col. 6, lines 26-28).*

14. With regard to Claim 7, Kelleher discloses that the at least two graphics pipelines 20A, (20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) separately receive the pixel data from the front end circuitry (Col. 8, lines 52-65).

15. With regard to Claim 9, Kelleher discloses a memory controller (68, Figures 7 and 11) coupled to the at least two graphics pipelines (20A, 20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14), operative to transfer pixel data between each of a first pipeline and a second pipeline and a memory (22) (*address generation circuit 68 accepts the initial geometry values from the geometric setup circuit 66, and generates physical memory addresses, Col. 9, lines 18-23; Col. 10, lines 40-47).*

16. With regard to Claim 10, Kelleher discloses that a first of the at least two graphics pipelines (20A, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) processes the pixel data only in a first set of tiles (group 0 of pixel blocks 52) in the repeating tile pattern (*each rendering processor 20 writes to only those pixels that are located in the blocks 52 of the assigned groups, blocks in group "0" may be assigned to rendering processor 0*, Col. 4, line 65-Col. 5, line 19).

17. With regard to Claim 12, Kelleher discloses that a second of the at least two graphics pipelines (20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) processes the pixel data only in a second set of tiles (group 1 of pixel blocks 52) in the repeating tile pattern (*each rendering processor 20 writes to only those pixels that are located in the blocks 52 of the assigned groups, blocks in group "1" may be assigned to rendering processor 1*, Col. 4, line 65-Col. 5, line 19).

18. With regard to Claim 13, Claim 13 is similar in scope to Claim 11, and therefore is rejected under the same rationale.

19. With regard to Claim 14, Claim 14 is similar to Claims 4 and 10, except that Claim 14 is for a third and fourth graphics pipeline. Kelleher discloses four graphics pipelines (20A-20N, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14). Therefore, Claim 14 is rejected under the same rationale as Claims 4 and 10.

20. With regard to Claims 15 and 16, these claims are each similar in scope to Claim 11, and therefore are rejected under the same rationale.

21. With regard to Claim 18, Kelleher discloses a bridge (38, Figure 3) operative to transmit vertex data to each of the first (20A), second (20B), third (20C) and fourth (20N) graphics pipelines (Col. 3, lines 22-23; Col. 4, lines 9-14; *rendering processor 20 first reads the command packets sent to it over the PCI bus 30*, Col. 8, lines 56-65; *rendering processors 20 are connected to a system PCI bus 30A through a PCI bridge 38*, Col. 3, lines 46-50).

22. With regard to Claim 20, Kelleher discloses a graphics processing method (Col. 2, lines 27-28), comprising receiving vertex data for a primitive to be rendered; generating pixel data in response to the vertex data; determining the pixels within a set of tiles (group of pixel blocks 52) of a repeating tile pattern corresponding to screen locations to be processed by a corresponding one of at least two graphics pipelines (20A, 20B, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) in response to the pixel data, the repeating tile pattern including a horizontally and vertically repeating pattern of square regions, as shown in Figure 4; and performing pixel operations on the pixels within the determined set of tiles by the corresponding one of the at least two graphics pipelines (Col. 4, line 60-Col. 5, line 19; Col. 8, lines 56-65).

23. With regard to Claim 21, Kelleher discloses that determining the pixels within a set of tiles (group of pixel blocks 52) of the repeating tile pattern to be processed further comprises

determining the set of tiles that the corresponding graphics pipeline (20, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) is responsible for (Col. 4, line 60-Col. 5, line 19).

24. With regard to Claim 22, Kelleher discloses that the scan converter determines which groups of blocks 52 within the graphics memory 22 are allocated to and controlled by the graphics pipelines 20 (Col. 8, lines 52-61). The graphics memory is partitioned into a plurality of pixel blocks that are tiled in the x-and y-direction of the graphics memory (Col. 4, lines 60-62). Kelleher discloses that determining the pixels within a set of tiles (group of pixel blocks 52) of the repeating tile pattern to be processed (Col. 5, lines 6-19) inherently further comprises providing position coordinates of the pixels within the determined set of tiles to be processed to the corresponding one of the at least two graphics pipelines.

25. With regard to Claim 23, Kelleher discloses transmitting the processed pixels to memory (22, Figure 4) (*rendering processor 20 scan-converts the object into the graphics memory 22, the graphics memory stores pixel data*, Col. 3, lines 36-41).

26. With regard to Claim 24, Kelleher discloses a graphics processing circuit, comprising front end circuitry (80, 82, Figure 7) operative to generate pixel data in response to primitive data for a primitive to be rendered; first back end circuitry (84), coupled to the front end circuitry, operative to process a first portion of the pixel data in response to position coordinates; a first scan converter, coupled between the front end circuitry and the first back end circuitry, operative to determine which set of tiles of a repeating tile pattern are to be processed by the first back end

circuitry (Col. 3, lines 22-23; Col. 8, line 59-Col. 9, line 23), the repeating tile pattern including a horizontally and vertically repeating pattern of square regions, as shown in Figure 4 (Col. 4, line 60-Col. 5, line 19), and operative to provide the position coordinates to the first back end circuitry in response to the pixel data (Col. 4, lines 60-62; Col. 8, lines 52-65; Col. 6, lines 36-38); second back end circuitry, coupled to the front end circuitry, operative to process a second portion of the pixel data in response to position coordinates; a second scan converter, coupled between the front end circuitry and the second back end circuitry, operative to determine which set of tiles of the repeating tile pattern are to be processed by the second back end circuitry, and operative to provide the position coordinates to the second back end circuitry in response to the pixel data (Col. 3, lines 22-23; Col. 8, line 59-Col. 9, line 23; Col. 4, lines 60-62; Col. 8, lines 52-65; Col. 6, lines 36-38); and a memory controller (68, Figures 7 and 11), coupled to the first and second back end circuitry operative to transmit and receive the processed pixel data (Col. 9, lines 18-23; Col. 10, lines 40-47).

27. With regard to Claim 25, Claim 25 is the same as Claim 1, and therefore is rejected under the same rationale.

28. With regard to Claim 26, Kelleher discloses a horizontally and vertically repeating pattern of regions (Col. 4, line 65-Col. 5, line 19) include NxM number of pixels (*each pixel block 52 is 128x128 pixels in size*, Col. 6, lines 2-4).



29. Thus, it reasonably appears that Kelleher describes or discloses every element of Claims 1-5, 7, 9, 10, 12-16, 18, and 20-26 and therefore anticipates the claims subject.

*Claim Rejections - 35 USC § 103*

30. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

31. The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

32. Claims 6, 8, and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kelleher (US005794016A) in view of Furtner (US006778177B1).

33. With regard to Claim 6, Kelleher is relied upon for the teachings as discussed above relative to Claim 1.

However, Kelleher does not explicitly teach that each tile of the set of tiles further comprises a 16x16 pixel array. However, Furtner describes that each tile of the set of tiles further comprises a 16x16 pixel array (Col. 11, lines 45-48, 64-65).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Kelleher so that each tile of the set of tiles further comprises a 16x16 pixel array as suggested by Furtner because Furtner suggests that depending on the number of parallel image-rendering pipelines and depending on the memory organization, the optimum tile size and shape can be selected (Col. 11, lines 45-48, 64-65), and therefore it would be obvious to modify the tile size to be 16x16 pixels if that would be the optimum tile size for a particular number of parallel image-rendering pipelines and particular memory organization.

34. With regard to Claim 8, Kelleher does not teach that the at least two graphics pipelines are on multiple chips. However, Furtner describes that the at least two graphics pipelines are on multiple chips (Col. 6, lines 47-51).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Kelleher so that the at least two graphics pipelines are on multiple chips as suggested by Furtner because Furtner suggests that this makes the system more configurable by being able to easily add more graphics pipelines to increase the performance (Col. 6, lines 29-30, 42-51).

35. With regard to Claim 17, Claim 17 is similar in scope to Claim 8, and therefore is rejected under the same rationale.

36. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kelleher (US005794016A) in view of Hamburg (US005905506A).

Kelleher is relied upon for the teachings as discussed above relative to Claim 10. Kelleher discloses that the first of the at least two graphics pipelines (20A, Figure 3; Col. 3, lines 22-23; Col. 4, lines 9-14) further includes a scan converter (84, Figure 7), coupled to the front end circuitry (80, 82) and the back end circuitry (Col. 8, line 52-Col. 9, line 23). The scan converter determines which groups of blocks 52 within the graphics memory 22 are allocated to and controlled by the graphics pipelines (Col. 8, lines 52-65; Col. 6, lines 26-28). The graphics memory is partitioned into a plurality of pixel blocks that are tiled in the x-and y-direction of the graphics memory (Col. 4, lines 60-62). Therefore, the scan converter is inherently operative to provide memory addresses or position coordinates of the pixels within the first set of tiles to be processed by the back end circuitry.

However, Kelleher does not explicitly teach using tile identification data to indicate which tiles are to be processed. However, Hamburg discloses a pixel identification line for receiving tile identification data indicating which tiles are to be processed (*during pixel modification, the system must write a pixel within at least one tile within image B, the system determines a tile ID at which to write the pixel value*, Col. 5, lines 35-52).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Kelleher to include using tile identification data to indicate which tiles are to be processed as suggested by Hamburg because Hamburg suggests the

advantage of using tile identification data to easily track the storage locations of the tile pixel data and being able to easily retrieve data for a particular image tile (Col. 1, lines 46-54).

37. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kelleher (US005794016A) and Furtner (US006778177B1) in view of Kent (US 20030164830A1).

Kelleher and Furtner are relied upon for the teachings as discussed above relative to Claim 17. Kelleher discloses that the data includes a polygon. Each pixel block 52 is 128x128 pixels in size. This block size has been determined to effectively divide the parallelism of the graphics pipelines 20. Since polygons tend to be particularly small, and 128x128 blocks 52 are relatively large, polygons will not commonly cross block boundaries (Col. 5, line 65-Col. 6, line 12). Furtner describes that the at least two graphics pipelines are on multiple chips (Col. 6, lines 47-51), as discussed in the rejection for Claim 8.

However, Kelleher and Further do not teach creating a bounding box around the polygon and wherein each corner of the bounding box is checked against a super tile that belongs to each separate chip and wherein if the bounding box does not overlap any of the super tiles associated with a separate chip, then the processing circuit rejects the whole polygon and processes a next one. However, Kent discloses that the graphics pipeline [0006] calculates the bounding box of the primitive and testing this against the VisRect. If the bounding box of the primitive is contained in the other P10's super tile the primitive is discarded at this stage [0129]. A primitive can be a polygon (*independent primitives (triangles or quads)*, [0088]). The method used is to calculate the distance from each subpixel sample point in the point's bounding box to the point's center and compare this to the point's radius. Subpixel sample points with a distance greater

than the radius do not contribute to a pixel's coverage. The cost of this is kept low by only allowing small radius points hence the distance calculation is a small multiply and by taking a cycle per subpixel sample per pixel within the bounding box [0144]. Since the method calculates the distance from each subpixel sample point in the point's bounding box, this must include all the corners of the bounding box. Therefore, Kent discloses that the data includes a polygon and that the graphics pipeline creates a bounding box around the polygon and wherein each corner of the bounding box is checked against a super tile that belongs to the graphics pipeline and wherein if the bounding box does not overlap any of the super tiles, then the processing circuit rejects the whole polygon and processes a next one.

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Kelleher and Furtner to include creating a bounding box around the polygon and wherein each corner of the bounding box is checked against a super tile that belongs to each separate chip and wherein if the bounding box does not overlap any of the super tiles associated with a separate chip, then the processing circuit rejects the whole polygon and processes a next one as suggested by Kent because Kent suggests the advantage of processing the super tiles one at a time in order to hide the page break costs [0129, 0051].

#### *Conclusion*


Any inquiry concerning this communication or earlier communications from the examiner should be directed to Joni Hsu whose telephone number is 571-272-7785. The examiner can normally be reached on M-F 8am-5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ulka Chauhan can be reached on 571-272-7782. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

JH


  
ULKA CHAUHAN  
SUPERVISORY PATENT EXAMINER

<b>Search Notes</b>  	<b>Application/Control No.</b>  10459797	<b>Applicant(s)/Patent Under Reexamination</b>  LEATHER ET AL.
	<b>Examiner</b>  Hsu, Joni	<b>Art Unit</b>  2628

SEARCHED			
Class	Subclass	Date	Examiner
345	506, 530, 505, 588, 544, 545, 532, 501, 502, 531	3/1/2006	JH
Above	UPDATED	1/24/07	JH

SEARCH NOTES		
Search Notes	Date	Examiner
EAST (US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB) -- See attached search history.	1/24/07	JH

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

<b>Index of Claims</b> 	<b>Application/Control No.</b> 10459797	<b>Applicant(s)/Patent Under Reexamination</b> LEATHER ET AL.
	<b>Examiner</b> Hsu, Joni	<b>Art Unit</b> 2628

✓	<b>Rejected</b>	-	<b>Cancelled</b>	N	<b>Non-Elected</b>	A	<b>Appeal</b>
=	<b>Allowed</b>	+	<b>Restricted</b>	I	<b>Interference</b>	O	<b>Objected</b>

Claims renumbered in the same order as presented by applicant
  CPA
  T.D.
  R.1.47

CLAIM		DATE									
Final	Original	01/29/2007									
	1	✓									
	2	✓									
	3	✓									
	4	✓									
	5	✓									
	6	✓									
	7	✓									
	8	✓									
	9	✓									
	10	✓									
	11	✓									
	12	✓									
	13	✓									
	14	✓									
	15	✓									
	16	✓									
	17	✓									
	18	✓									
	19	✓									
	20	✓									
	21	✓									
	22	✓									
	23	✓									
	24	✓									
	25	✓									
	26	✓									



<b>Notice of References Cited</b>	Application/Control No. 10/459,797	Applicant(s)/Patent Under Reexamination LEATHER ET AL.	
	Examiner Joni Hsu	Art Unit 2628	Page 1 of 1

**U.S. PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification	
*	A	US-5,794,016 A	08-1998	Kelleher, Brian Michael	345/505
*	B	US-5,905,506 A	05-1999	Hamburg, Mark	345/672
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

**FOREIGN PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N				
	O				
	P				
	Q				
	R				
	S				
	T				

**NON-PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	U	Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)			
	V				
	W				
	X				

\*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)  
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Mark M. Leather et al.  
Serial No.: 10/459,797  
Filing Date: June 12, 2003  
Confirmation No.: 4148

Examiner: Joni Hsu  
Art Unit: 2628  
Our File No.: 00100.02.0053

Title: **DIVIDING WORK AMONG MULTIPLE GRAPHICS PIPELINES USING  
A SUPER-TILING TECHNIQUE**

Mail Stop Amendment  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**AMENDMENT AND RESPONSE**

Dear Sir:

In response to the Office Action mailed February 9, 2007, Applicants petition for a one month extension of time and submit the following response.

**Amendments to the Claims** are reflected in the Listing of the Claims, which begins on page 2 of this paper.

**Remarks** begin on page 8 of this paper.

**Amendments to the Claims:**

Re-write the claims as set forth below. This listing of claims will replace all prior versions and listings, of claims in the application:

**Listing of Claims:**

1. (currently amended) A graphics processing circuit, comprising:  
  
at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile[.]; and  
  
a memory controller in communication with the at least two graphics pipelines, operative to transfer pixel data between each of a first pipeline and a second pipeline and a memory;  
  
wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of square regions.
2. (original) The graphics processing circuit of claim 1, wherein the square regions comprise a two dimensional partitioning of memory.
3. (original) The graphics processing circuit of claim 2, wherein the memory is a frame buffer.
4. (original) The graphics processing circuit of claim 1, wherein each of the at least two graphics pipelines further includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process a portion of the pixel data.

5. (original) The graphics processing circuit of claim 4, wherein each of the at least two graphics pipelines further includes a scan converter, coupled to the back end circuitry, operative to determine the portion of the pixel data to be processed by the back end circuitry.

6. (original) The graphics processing circuit of claim 1, wherein each tile of the set of tiles further comprises a 16x16 pixel array.

7. (original) The graphics processing circuit of claim 4, wherein the at least two graphics pipelines separately receive the pixel data from the front end circuitry.

8. (canceled)

9. (canceled)

10. (original) The graphics processing circuit of claim 4, wherein a first of the at least two graphics pipelines processes the pixel data only in a first set of tiles in the repeating tile pattern.

11. (original) The graphics processing circuit of claim 10, wherein the first of the at least two graphics pipelines further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the first set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the set of tiles is to be processed by the back end circuitry.

12. (previously presented) The graphics processing circuit of claim 1, wherein a second of the at least two graphics pipelines processes the data only in a second set of tiles in the repeating tile pattern.

13. (previously presented) The graphics processing circuit of claim 12, wherein the second of the at least two graphics pipelines further includes a scan converter, coupled to front end circuitry and back end circuitry, operative to provide position coordinates of the pixels within the second set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the set of tiles is to be processed by the back end circuitry.

14. (original) The graphics processing circuit of claim 1 including a third graphics pipeline and a fourth graphics pipeline, wherein the third graphics pipeline includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process the pixel data in a third set of tiles in the repeating tile pattern, and wherein the fourth graphics pipeline includes front end circuitry operative to receive vertex data and generate pixel data corresponding to a primitive to be rendered, and back end circuitry, coupled to the front end circuitry, operative to receive and process the pixel data in a fourth set of tiles in the repeating tile pattern.

15. (original) The graphics processing circuit of claim 14, wherein the third graphics pipeline further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the third set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for

receiving tile identification data indicating which of the sets of tiles is to be processed by the back end circuitry.

16. (original) The graphics processing circuit of claim 14, wherein the fourth graphics pipeline further includes a scan converter, coupled to the front end circuitry and the back end circuitry, operative to provide position coordinates of the pixels within the fourth set of tiles to be processed by the back end circuitry, the scan converter including a pixel identification line for receiving tile identification data indicating which of the sets of tiles is to be processed by the back end circuitry.

17. (original) The graphics processing circuit of claim 14, wherein the third and fourth graphics pipelines are on separate chips.

18. (original) The graphics processing circuit of claim 14, further including a bridge operative to transmit vertex data to each of the first, second, third and fourth graphics pipelines.

19. (original) The graphics processing circuit of claim 17 wherein the data includes a polygon and wherein each separate chip creates a bounding box around the polygon and wherein each corner of the bounding box is checked against a super tile that belongs to each separate chip and wherein if the bounding box does not overlap any of the super tiles associated with a separate chip, then the processing circuit rejects the whole polygon and processes a next one.

20. (currently amended) A graphics processing method, comprising:  
receiving vertex data for a primitive to be rendered;  
generating pixel data in response to the vertex data;

determining the pixels within a set of tiles of a repeating tile pattern corresponding to screen locations to be processed by a corresponding one of at least two graphics pipelines in response to the pixel data, the repeating tile pattern including a horizontally and vertically repeating pattern of square regions; [[and]]

performing pixel operations on the pixels within the determined set of tiles by the corresponding one of the at least two graphics pipelines; and

transmitting the processed pixels to a memory controller, wherein the at least two graphics pipelines share the memory controller.

21. (original) The graphics processing method of claim 20, wherein determining the pixels within a set of tiles of the repeating tile pattern to be processed further comprises determining the set of tiles that the corresponding graphics pipeline is responsible for.

22. (original) The graphics processing method of claim 20, wherein determining the pixels within a set of tiles of the repeating tile pattern to be processed further comprises providing position coordinates of the pixels within the determined set of tiles to be processed to the corresponding one of the at least two graphics pipelines.

23. (canceled)

24. (previously presented) A graphics processing circuit, comprising:

front end circuitry operative to generate pixel data in response to primitive data for a primitive to be rendered;

first back end circuitry, coupled to the front end circuitry, operative to process a first portion of the pixel data in response to position coordinates;

a first scan converter, coupled between the front end circuitry and the first back end circuitry, operative to determine which set of tiles of a repeating tile pattern are to be processed by the first back end circuitry, the repeating tile pattern including a horizontally and vertically repeating pattern of square regions, and operative to provide the position coordinates to the first back end circuitry in response to the pixel data;

second back end circuitry, coupled to the front end circuitry, operative to process a second portion of the pixel data in response to position coordinates;

a second scan converter, coupled between the front end circuitry and the second back end circuitry, operative to determine which set of tiles of the repeating tile pattern are to be processed by the second back end circuitry, and operative to provide the position coordinates to the second back end circuitry in response to the pixel data; and

a memory controller, coupled to the first and second back end circuitry operative to transmit and receive the processed pixel data.

25. (currently amended) A graphics processing circuit, comprising:

at least two graphics pipelines operative to process data in a corresponding set of tiles of a repeating tile pattern corresponding to screen locations, a respective one of the at least two graphics pipelines operative to process data in a dedicated tile, wherein the repeating tile pattern includes a horizontally and vertically repeating pattern of regions;

wherein the horizontally and vertically repeating pattern of regions include NxM number of pixels.

26. (canceled)



**REMARKS**

Claims 1-26 are now pending in the application. Applicants respectfully traverse and request reconsideration.

Claim 25 stands objected to because it is allegedly exactly the same as claim 1 and is therefore a repeated claim. Applicants respectfully point out that claim 1 includes a “repeating pattern of square regions,” (emphasis added), but claim 25 claims a “repeating pattern of regions” (without the limitation that those regions be square). Furthermore, Applicants have amended claim 25 to incorporate the limitation of claim 26 into claim 25, as noted below. Therefore, Applicants respectfully request that the objection be withdrawn.

Claims 20-23 stand rejected under 35 U.S.C. § 101 for allegedly being directed to non-statutory subject matter. As an initial matter, Applicants note that claim 23 has been canceled without prejudice, thereby rendering this rejection moot as to claim 23.

With regard to claim 20, the Examiner merely states that the graphics processing method does not yield a “useful, concrete and tangible result.” The Examiner merely provides a conclusive statement that the claims allegedly constitute non-statutory subject matter without an explanation as to why the claims allegedly constitute non-statutory subject matter.

Applicants object to this rejection for at least the reason that no explanation has been given as to why the aforementioned claims are non-statutory. Accordingly, a *prima facie* case has not been established. Applicants kindly remind the Office that MPEP §2106(IV)(B) states that “[t]he burden is on the [Examiner] to set forth a prima facie case of unpatentability.” “After the examiner identifies and explains in the record the basis for why a claim is for an abstract idea with no practical application, then the burden shifts to the applicant to either amend the claim or make a showing of why the claim is eligible for patent protection.” MPEP §2106(IV)(D).

In addition, the Examiner states that the claimed method fails to yield a useful, concrete, and tangible result. However, according the Federal Circuit, patentable subject matter includes a “process if the claimed invention as a whole is applied in a ‘useful’ manner.” *AT&T Corp. v. Excel Communications, Inc.*, 172 F.3d 1352, 1357 (Fed. Cir. 1999). A proper inquiry requires “an examination of the contested claims to see if the claimed subject matter as a whole is . . . a ‘law of nature’ or an ‘abstract idea,’ or if the . . . concept has been reduced to some practical application rendering it ‘useful’.” *Id.* In addition, MPEP §2106(IV)(C) states that “[I]n evaluating whether a claim meets the requirements of section 101, the claim must be considered as a whole to determine whether it is for a particular application of an abstract idea, natural phenomenon, or law of nature, rather than for an abstract idea, natural phenomenon, or law of nature.” Therefore, the claimed invention as a whole should yield a useful, concrete, and tangible result.

Applicants respectfully submit that at least independent claim 20 recites statutory subject matter. The claim includes, for example, processed pixels, which are concrete (e.g., receiving the same input data for the primitive to be rendered will always produce the same processed pixels), useful (e.g., has a practical utility), and has a real world, tangible result. For example, Applicants present a method for performing graphics processing that comprises, among other things, transmitting processed pixels to a memory controller.

For such subject matter to be statutory, the claimed process must be limited to a practical application of the abstract idea or mathematical algorithm in the technological arts. *In re Alappat*, 33 F.3d 1526, 1543 (Fed. Cir. 1994). A claim is limited to a practical application when the method, as claimed, produces a concrete, tangible, and useful result. *AT&T Corp.*, 172 F.3d at 1357. In *AT&T* the claims in question were directed to generating a message record for an

interexchange call between an originating subscriber and a terminating subscriber and adding a primary interexchange carrier (PIC) indicator to the message record. *Id.* at 1354. The court held that “[t]he PIC indicator represents information about the call recipient’s PIC, a useful, non-abstract result that facilitates differential billing of long-distance calls made by a . . . subscriber.” *Id.* at 1358) Likewise, a machine claim is statutory when the machine, as claimed, produces a concrete, tangible and useful result (as in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*, 149 F.3d 1368, 1373 (Fed. Cir. 1998)). However, the analysis is the same regardless of whether the claim is directed to a machine or a process.

In *State Street*, the Federal Circuit reviewed a claim directed to a “data processing system for managing a financial services configuration of a portfolio established as a partnership, each partner being one of a plurality of funds, comprising” a variety of structural components including a “fifth means for processing data regarding aggregate year-end income, expenses, and capital gain or loss for the portfolio and each of the funds.” *Id.* at 1371-72. The Federal Circuit held that:

[t]he transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces ‘a useful, concrete and tangible result’ – a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades. *Id.* at 1373.

Notably, the claim did not require that the tangible result, which was the share price, be claimed or produced. The mere fact that the *State Street* claim required “processing data” regarding expenses etc. was enough to be considered statutory subject matter. Thus, at least according to the Federal Circuit, a claim need not expressly recite a result that is useful, concrete, and tangible in a last step or anywhere else in order to be considered statutory subject matter. The fact that

the claimed process led to or allowed for a recorded, accepted, and/or relied upon result was sufficient to establish compliance with 35 U.S.C. § 101.

While method claims take a different form than a machine claim, the fact that Applicants claim methods and not machines, as presented in *State Street*, has no impact on the present analysis. See *AT&T Corp.*, 172 F.3d at 1357. Consequently, the fact that the *State Street* opinion notes that the claim is directed to a machine has no weight on the instant analysis.

With regard to claim 20, the method for graphics processing requires, among other things, generating pixel data in response to the vertex data and performing pixel operations on the pixels within the determined set of tiles by the corresponding one of the at least two graphics pipelines. Similar to the claimed invention in *State Street*, Applicants' claimed subject matter is directed to a method that processes data not just once but twice: generating pixel data in response to the vertex data and performing pixel operations on the pixels. For at least this reason, claim 20 appears to be directed to a practical application of the section 101 judicial exception. Therefore, reconsideration and withdrawal of the rejection of claim 20 is respectfully requested.

Dependent claims 21-22 are believed to also be directed to statutory subject matter for the same or similar reasons as provided above. Therefore, reconsideration and withdrawal of the rejection of claims 21-22 is respectfully requested.

Claims 1-5, 7, 9, 10, 12-16, 18, and 20-26 stand rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,794,016 ("Kelleher"). Kelleher is generally directed to a parallel-processor graphics architecture appropriate for multimedia graphics workstations that is scalable to the needs of a user. (Abstract.) As shown in FIG. 3, for example, N separate rendering processors 20 each implement graphics and multimedia algorithms and interface with