PROCEEDINGS, VOLUME III

NCGA'S COMPUTER GRAPHICS '87
EIGHTH ANNUAL CONFERENCE AND EXPOSITION

PHILADELPHIA CIVIC CENTER
PHILADELPHIA, PA.
MARCH 22-26, 1987

TECHNICAL SESSIONS
VOLUME III

ISBN 0-941514-17-X (Volume III)
ISBN 0-941514-14-5 (Three-Volume Set)

Cover image courtesy of PDA Engineering

Accelerating Interactive Applications

Mary C. Whitton
Trancept Systems Inc.
521-F Uwharrie Court
Raleigh, NC 27606-1456

ABSTRACT

User interaction in an application requires significant processing after user input before a new image can be displayed. This processing consists of two parts: application processing and graphics processing. In real world problems application processing dominates graphics processing in terms of complexity and length. General purpose, function, and application accelerators are examined. The interactive application environment of the future requires an application accelerator which is a software reconfigurable multi-processor system with an architecture optimized for applications involving spatial and geometric data bases, graphics, and image processing.

INTRODUCTION

The goal of high performance graphics has always been to allow humans to better use computers by improving the man-machine interface. The human visual,system and computer graphics together make a highly efficient mechanism transferring information from computer to man. The target is real-time interaction of the user with the generation and manipulation of data from highly complex and diverse applications. Meeting this target requires evaluating the computational steps in the interactive process and developing systems which optimize performance at each step. The job will not be done until the human is the slow link in the system.

THE INTERACTIVE LOOP

The interactive process is a loop. First, the user enters data which causes processing and the generation of an image on the screen. Based on visual feedback from this image the user enters new data which causes a new image ... and so on (Figure 1). Two types of processing occur between user input and new image: graphics processing and application processing.

Graphics Processing

For the purpose of this paper graphics is restricted to raster graphics and the graphics task is defined as those processes needed to get from a graphical data set to pixels: transformation, clipping, visibility, and shading operations for vector, character, polygon, and pixel data. (Ray tracing and direct display of parametric data extend beyond the graphics task because they operate directly on applications data.)

Graphics at this level is becoming standardized. Applications have a fairly uniform set of graphics requirements. For instance, a 3-D

surface used in a seismic reservoir model can be displayed using the same routines as a 3-D surface in an architectural rendering if both are approximated with polygonal graphical data. Because a wide range of applications use the same graphics algorithms it is possible to develop special purpose hardware for these functions.

The effort to accelerate graphics processing began with the integration of a processor into the frame buffer to handle drawing. Later, application specific integrated circuits were developed for the viewing pipeline. More recently, custom silicon for drawing operations has become common. The microVAX-GPX, Hewlett-Packard 9000 Model 320SRX, and GE Graphicon all contain custom chips for painting pixels into the image memory.

With the promise of affordable 2000+ polygons per frame drawing engines in the future, the graphics process ceases to be the limiting factor in the interactive loop.

Application Processing

Today, interactive applications means more than changing a viewing angle and repainting the screen. For example, it means making changes in the application data base, either to geometry or to parameters, and viewing the effect of the changes in the screen image.

User inputs in real world applicatons are complex. For instance, a solid modeling command might be "add a negative cylinder to the current object" (to create a hole). The application processing involves computing the surfaces which bound the hole and modifying the data base to include the new surfaces. This task cannot be performed in real time today.

Increasingly, visual displays are essential for viewing results in applications such as thermal analysis, molecular modeling, and kinematic simulation. These applications are so compute intensive that they are run in batch or time shared mode today. Higher performance processors are required before these applications can begin to be interactive.

Application processing is truly application specific. The requirements of desktop publishing are nothing like those of medical imaging which in turn are nothing like those of electrical CAE. Requirements within industries also vary. There is little agreement among the users of mechanical CAD on what the "standard" primitives for CSG modeling should be and even less agreement on a standard parametric surface descriptor.

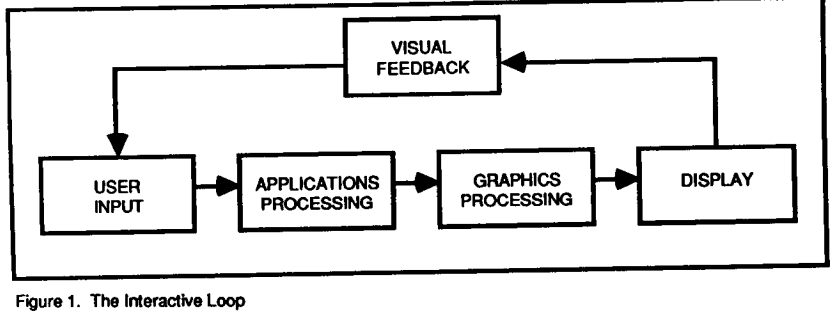

Figure 1. The Interactive Loop

Ratio of Application Processing to Graphics Processing

In simple applications such as viewing a static data base, the ratio of application to graphics processing is very small. In real world applications on real world data sets application processing overwhelmingly dominates graphics processing. To successfully implement high performance, interactive applications in the future we must address both processing steps in the interactive loop: the graphics requirements and the more complex application processing requirements. Traditional graphics subsystems meet only the first requirement.

THE EVOLUTION OF TODAY'S INTERACTIVE SOLUTIONS

The tasks in the interactive loop have historically been divided between the general purpose CPU and the graphics subsystem. The role of the subsystem expanded as new hardware made increasing functionality possible. Users concerned with high performance have taken advantage of the computational power available and moved an increasing amount of the interactive loop to the graphics subsystem. An examination of the evolution of graphics subsystems shows how the task division between CPU and subsystem has changed.

"Dumb" Frame Buffers

Raster frame buffers became available in the 1970's. Their only task was to display the pixels which were generated by the CPU. All application and graphics processing was performed by the CPU with pixel values passed from CPU to frame buffer. Pixel read back was an advanced feature. Grinell and Ramtek were early vendors.

"Smart" Frame Buffers

In the late seventies the first "smart" frame buffers came to market. The "smart" buffers could accept either pixels or commands such as "draw line" and "draw circle." This was the first off-loading of drawing computation from the CPU. The processors in the systems were not user programmable and functionality was limited to those primitives provided by the vendor. Commands were transferred as character strings. Redrawing the image involved transferring all of the commands to the graphics device again. Genisco and Lexidata had early systems in this category.

As processors become more capable and interfaces improved, display lists could be downloaded to the graphics subsystem. The role of the graphics processor expanded so that the picture could be redrawn locally and modified by changing the display list without transferring the entire command and data set from the host again. The Ramtek 9400 series is a good example of this class of machine.

Local 3-D transformations became available on raster devices in the early 1980's. Visibility calculations moved to the graphics processor with early implementations of the z-buffer in research labs on Genisco and Adage/Ikonas equipment and, commercially, on Lexidata hardware. With local transformations, shading, and visibility calculations now in the subsystem, the move of the graphics processing to the graphics subsystem is complete.

Firmware Controlled Graphics Subsystems

High performance graphics processors are computers intimately integrated with a display memory. Built of bit slice parts they have performance significantly higher than that of integrated

single chip processors. In graphics subsystems, the processor is typically controlled by firmware resident either in ROM or RAM.

As graphics processors became more generally programmable (by the vendor and the brave), more functionality became available in the libraries supplied by the vendors. The mid-eighties saw the emergence of application specific libraries, from Adage/Ikonas [Van Hook 84, Van Hook 86, England '86] and Lexidata. The libraries contain routines for drawing application specific primitives as well as routines for the local processing of the application data.

An example of an application specific primitive is the classic seismic "wiggle trace." An application specific routine allows these plots to be drawn directly from the source data with a single library call, avoiding a time consuming process of creating an enormous vector and polygon list which must then be drawn. The Hewlett Packard Series 9000 Model 320SRX library for the display of non-uniform rational B-spline surfaces is a good example of local processing of application data.

The programmability of the advanced graphics subsystem allows the application developer to move highly application specific blocks of software into the subsystem. The key to developing interactive applications is to move to the computationally intense applications processing from the CPU to a applicaton accelerator subsystem which is dedicated to the interactive task. See Figure 2.

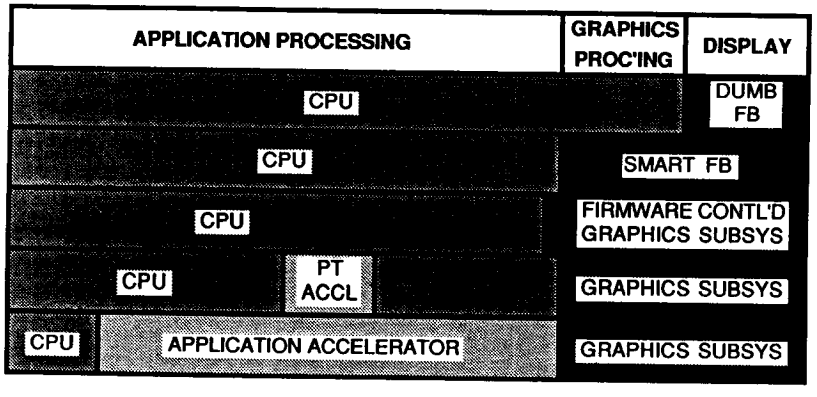

Figure 2. Division of tasks in the interactive loop.

STRATEGIES FOR ACCELERATING APPLICATION PROCESSING

If we take for granted that in the future displays will be drawn "instantaneously," the bottleneck in interactive applications is application processing to supply new data to the graphics subsystem. The task, taken across a range of applications, is both difficult and diverse. Significant processing power is required. Products which can accelerate this processing fall into three classes: general purpose computers used as accelerators, single function accelerators, and application accelerators. Figure 3 summarizes the sections which follow.

General Purpose Strategies

General purpose processors are the most general category of accelerators. Designed to be used in a wide range of problems,

they are not optimized for any one function or even a class of functions. While they accelerate everything a little bit, they accelerate nothing extremely well.

Network Servers. Several companies, including Convex, Alliant, and Culler, are producing mini-supercomputers which can be attached to a workstation network as computational servers. Advantages of these systems are their high performance and their standard language compilers. Existing software can be ported to the systems with some effort. Optimizing compilers allow the user to take advantage of the multi-processor architectures.

A major disadvantage of network servers is limited network bandwith which causes an I/O bottleneck. Only on very time consuming problems can the decrease in computation time offset the data transfer time. Network technology makes these machines unsuited for acceleration in a interactive environment.

The High-MIP Workstation. This is the year, 1987, that the "superworkstation" will come to the market. The market expects that these products will offer 10+ MIPS on the desktop with integrated real-time graphics. Most of the expectation surrounds the planned products of Stellar Computer and the Dana Group. These workstations will be appealing because some application software can be moved to this new environment. Applications may run faster, but, perhaps, still not interactively.

The downside of the systems based on a very fast CPU is the reality of the volume of processing that the CPU is expected to perform. In addition to running the operating system and usual windows based user interface, high end applications will increasingly require data base management, expert systems, and more sophisticated user interfaces such as voice recognition. If a central CPU is handling these tasks as well as interactive 3-D graphics, it seems that few cycles will remain for actual application processing.

Function Accelerators

Function accelerators accelerate one function within one application. For this reason they are sometimes called "point accelerators." They perform one specific task in a single application area, for instance the logic simulation function in the electrical CAE application. Function accelerators offer good performance for the task for which they are designed, but they are unable to do anything else well (if at all). They are seldom programmable and have little, if any, flexibility. Function accelerators can be highly specific integrated circuits or board or system level products.

Application software must be modified to take advantage of the accelerator. Such modifications are frequently difficult, particularly in existing application packages.

Because of the large investment required to develop a cost effective function accelerator, they are common only in large markets. Potentially, function accelerators offer the highest performance of any acceleration strategy. Overall application performance improvement may, however, be disappointing since only one task of the many is speeded up. System balance is critical.

Function accelerators available for interactive applications include the Weitek Z-buffer chip set for graphics, the Evans and Sutherland PS300 Energy Processor, and a variety of electrical CAE accelerators from companies such as Zycad and Silicon Solutions.

Several commercial vendors are considering developing function

accelerators for the analysis task. Work on products for solid modeling, as well as other applications, is hampered by the diversity of functions requiring acceleration. The market for each variation of solid modeler is not large enough by itself to support development of a function accelerator specific to that modeler. A more general acceleration strategy is needed.

Application Accelerators

Application accelerators speed up virtually all of the functions within an applicaton. For this reason they are sometimes called global accelerators. They offer the highest overall application acceleration of any of the three acceleration strategies. Even though their performance for any one function within the application may be slower than a function accelerator, their overall application throughput is higher.

An example of an application accelerator is a traditional image processing system. Such a system accelerates many of the functions required by the application: filtering, variable color maps, and image-image computations. The Mentor Compute Engine is an example of an application accelerator for electrical CAE. It addresses both logic and circuit simulation.

Functional flexibility in application accelerators is often achieved by designing with programmable parts. The devices are used by porting a portion or all of an application software package to the accelerator. The diversity of tasks addressed by these products means that they must have support for high level languages and tools to facilitate software porting.

Because an application accelerator is programmable, it can be used for purposes for which it was not designed. For instance, an integer product such as the PIXAR, which was primarily designed for image processing, can, with some difficulty, be programmed to generate graphical images.

Array processors are application accelerators which are well suited for signal processing. Their architecture is well matched to the data flow and processing required by this task. Only with great difficulty and the sacrifice of much of their performance can array processors be programmed for more general applications. They are not well suited to accelerate operations on geometric or spatially organized data as is required by many interactive applications.

The price of application accelerators varies widely based on what level of computational power is available, how much local memory is incorporated into the system, and the level of software support (compilers, debugging tools, libraries) the vendor offers.

Properly designed application accelerators offer performance very nearly equal to that of function accelerators. The product architecture can be optimized for the data type and size, data flow, and processing required by the application. Application accelerators have the flexibility to address the range of functions required by their target application, while incorporating features which make them significantly outperform general purpose computers. They have a place in the application enviroment of the future.

THE INTERACTIVE APPLICATION ENVIRONMENT OF THE FUTURE

Although "expert systems" and artificial intelligence will help in high end applications in the future, a human will remain in the interactive loop for the foreseeable future. This means that we will continue to need graphics for visual feedback and increasingly powerful accelerators for application processing.

TYPE	FUNCTIONALITY	PROS	CONS
GENERAL PURPOSE	• All functions of all applications	• Can do anything • Does all of task	• Requires software port • Limited I/O bandwidth • Modest performance improvement
APPLICATION ACCELERATOR	• All functions of application	• Max application through-put • Flexibility preserves software investment	• Requires some software porting
FUNCTION ACCELERATOR	• Single function of single application	• Low Cost • High function performance	• Requires software modification • Limited flexibility and functionality • Limited application performance improvement

Figure 3. Types of Application Accelerators

High Performance Graphics

Each generation of graphics products brings us closer to real time rendering of complex scenes. The future will bring us "enough" speed to support interaction and the quality of "enough" will improve. In the future interactive shaded polygons will give way to interactive ray tracing. The user will generate the highest quality images possible while still maintaining interactivity. Display size and resolution will increase to allow the user access to more information simultaneously.

Computational Power as a Personal Resource

The trend to personal computing resources will continue. High end application users have moved from batch/time sharing on a departmental computer to the engineering workstation in the last ten years. A personal resource with 2-4 MIPS computational power with integrated graphics and a window based user interface is the high-end norm today. Users in the future will expect even more power to be available to them, individually, dedicated to solving their application problem.

Parallel and Multi-Processor Systems

Very high performance architectures increasingly include multiple processors. Whether the processors are organized to operate on multiple independent tasks or on a single task in parallel, this move to multiple processors is a fundamental change in the way the world computes. Processors in this class today range from board level products with multiple copies of commercially available processors to highly specialized architectures such as the massively parallel and data flow machines.

The bus bandwidth between disk and memory and between processors in multi-processor systems will improve as newer, higher speed bus standards emerge. The serial port on video RAMs is excellent not only for output to displays, but for very high speed data input. Very high speed block transfer busses will be used in the future.

Previously, when there were few multi-processor computers, an elite group of programmers developed and optimized a very few programs for the machines. This is changing. In the future the average programmer will have to learn to think about problems in a multi-processor environment.

New compilers will allow code written for traditional single processor architectures to run efficiently in the new multi-processor environment. Optimizing compilers are important to allow

continued use of existing code, and existing programmers. Just as "good programming" today is judged by performance on single processor machines, "good programming" in the future must be defined as efficient execution on a multi-processor machine.

It is difficult to move portions of single threaded programs with many nested subroutine calls to any multi-processor environment. Functional blocks cannot be easily isolated. Even a conceptually clean partition such as moving rendering from the CPU to an attached graphics processor is difficult in large application programs. The problem is even more difficult when the target processor is an application accelerator, particularly the wrong accelerator for the job.

THE RIGHT ACCELERATOR FOR APPLICATION PROCESSING

The Right Strategy

Overall application performance and user productivity will be heavily affected by the choice of accelerator. To determine what architecture and acceleration strategy is most appropriate, the designer must look for common threads in the tasks to be accelerated. If the low level tasks are common, a function or point accelerator is appropriate. If there is no commonality at all, a general purpose accelerator should be used. An application accelerator is the best solution if there are common major characteristics and a diversity of low level functions.

The common threads in the high end applications which use the interactive loop are graphics, image processing, computational intensity, and data which is spatial and/or geometric in nature, whether 2-D or 3-D. This characteristic applies to all varieties of CAE, page layout, scene simulation, broadcast animation, geophysical interpretation, remote sensing, analysis, etc.

A general purpose processor is not appropriate for this class of applications because it is not optimized for the commonalities which exist. A single function accelerators is too narrowly focused to serve this class well. An application accelerator is the right strategy to improve performance in the interactive loop.

Accelerator Architecture to Preserve Software Investment

Porting or modifying application software for use with a given accelerator should only be done once. To insure that the software investment is preserved, the target accelerator must be based on a well though out architecture. The best architectures will allow higher levels of integration and the inclusion of higher performance processors in the future. Software developed for an implementation can be used only until the implementation changes; software developed for an architecture will have a lifetime covering many implementations of the architecture.

The Appropriate Architecture

An best architecture for an applicaton accelerator addressing the spatial/geometric/graphics/imaging class of problems is a superset of architectures optimized for the applications in the class. This means that the architecture must include data paths which allow it to access its local memory in 2-D or 3-D array modes, to look like an image processor with feedback data paths, to look like a pipeline for traditional graphics operations, or to look like a vectorized processor for analysis. Additionally, the accelerator must have very fast access to data stored on disk or in CPU memory.

The processing elements should be general purpose because of the diversity of low level functions the product must address. Bit slice based for maximum performance, the processors should include both 32 bit integer and single and double precision floating point. Lower precision is no longer acceptable in most high end applications.

The accelerator should be based on a wide instruction word computer (WIWC) for optimal performance. Figure 4 shows that this type of architecture has significant advantages over both CISC and RISC architectures in that multiple high level language instructions can be executed in each machine cycle. In comparison, multiple machine cycles are required to execute one high level instruction in both CISC and RISC machines. Discussions of efficiency which compare the percent of compute cycles relative to I/O cycles have no little bearing here; this is a comparison of the cycles which are actually performing the application processing. The wide instruction word architecture is the clear winner.

High level language tools are essential to allow the developer and user to easily move application software to the application accelerator.

MACHINE TYPE	# HLL INSTR	# MACHINE INSTR	# CYCLES/ MACH INSTR	HLL INSTR/ CYCLE
CISC	1	1	N	1/N
RISC	1	N	1	1/N
WIWC	N	1	1	N

Figure 4. Performance comparison of processor architectures.

THE FUTURE

Based on the requirements and trends outlined above, the application accelerator of the future will locally handle all parts of the application processing in the interactive loop except the small portion directly involved in transferring user input to the accelerator.

The accelerator will use very high speed external busses. These busses will allow it fast access to data on disk, data in the CPU, and data on other accelerators.

The accelerator itself will be a multiprocessor architecture for maximum performance. Internal busses between processors will be very fast and very wide. For super performance, multiple accelerator configurations can be formed using the external busses. High level language support and optimizing compilers will make all the power of the processors available to the programmer.

The combination of multiple processors and flexible, high speed busses make a system which is software reconfigurable into a processing structure optimized for a specific application. Performance levels rivaling function accelerators will be achieved through very high performance processing elements and the "superset" nature of the accelerator architecture. Many more applications will fall into the category of interactive.

REFERENCES

England, Nick, "A Graphics Systems Architecture for Interactive Application Specific Display Functions," IEEE Computer Graphics and Applications, 6(1), Jan.1986, pp.60-70.

Van Hook, Tim, "Advanced Techniques for Solid Modeling," Computer Graphics World, 7(11), Nov. 1984, pp.45-54.

Van Hook, Tim, "Real-Time Shaded NC Milling Display," Computer Graphics (Proceedings of SIGGRAPH '86), 20(4), Aug. 1986, pp. 15-20.

REFERENCES

England, Nick, "A Graphics Systems Architecture for Interactive Application Specific Display Functions," IEEE Computer Graphics and Applications, 6(1), Jan.1986, pp.60-70.

Van Hook, Tim, "Advanced Techniques for Solid Modeling," Computer Graphics World, 7(11), Nov. 1984, pp.45-54.

Van Hook, Tim, "Real-Time Shaded NC Milling Display," Computer Graphics (Proceedings of SIGGRAPH '86), 20(4), Aug. 1986, pp. 15-20.