To the Graduate Council:

I am submitting herewith a thesis written by Pramod Chandra entitled "Interactive Computer Graphics Systems." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

R. C. Gonzalez

Rafael C. Gonzalez, Major Professor

We have read this thesis
and recommend its acceptance:

J M Bailey

Robert E Bodenheimer

Accepted for the Council:

Lew Minkel

Vice Provost
and Dean of The Graduate School

INTERACTIVE COMPUTER GRAPHICS SYSTEMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Pramod Chandra

March 1987

To my parents,

Mrs. Raj Kumari and Late Dr. Ramchandra,

and my wife,

Cynthia

ACKNOWLEDGMENTS

In the course of my graduate studies and research, there have been several people who have assisted me considerably. I particularly wish to thank Professors Rafael C. Gonzalez, Robert E. Bodenheimer, and John M. Bailey, who served as my thesis advisor and committee respectively.

ABSTRACT

The complexity of computer graphics systems today has reached a point where it is becoming increasingly difficult to effectively design, manufacture, and market a graphics system in a highly competitive and rapidly changing computer graphics marketplace. Therefore, it is imperative that the graphics system designer find ways to minimize costly development time and reduce the risk of early obsolescence by the proper design and selection of graphics system components.

The design of an interactive computer graphics system involves a large number of considerations. The functional requirements and design considerations for various system components are drawn from current research and literature in the field of computer graphics and related areas. Various approaches to the design of different graphics system components, namely hardware architecture, software architecture, and human factors and ergonomics, are formulated, with emphasis on recent trends in the graphics industry towards three-dimensional graphics applications.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	HARDWARE DESIGN	5
	2.0 Functional Model of a Computer Graphics System	6
	2.1 Classification of Graphics Processors . .	10
	2.1.1 Special-Purpose Graphics Processors	10
	2.1.2 Limited-Purpose Graphics Processors	11
	2.1.3 General-Purpose Graphics Processors	14
	2.2 Bit-Slice Architectures	14
	2.3 The Graphics Pipeline Concept	17
	2.4 Display Memory Design for Graphics Systems	19
	2.4.1 Double-Buffer Systems	20
	2.4.2 Variable Resolution Memories . . .	21
	2.4.3 Cache Memories	21
	2.4.4 Dual-Ported RAM	22

2.5 Graphics Processors and Graphics Controllers 24
2.6 Requirements for a Graphics Processor . . 25
2.6.1 Programmability 25
2.6.2 Thirty-Two-Bit Internal Data and Address Paths 26
2.6.3 Instruction Set and Built-In Hardware Functions 27
2.6.4 Interfacing to Other Devices . . . 27
2.6.5 Other Characteristics 28
2.7 Limitations of Hardwired Graphics Controller ICs and General-Purpose Graphics Processors 29
2.8 Comparison of Various Graphics Processors and Graphics Controllers 31
2.8.1 Simultaneous Support of Text and Graphics 32
2.8.2 Memory Management: Parallel Pixel Access 35
2.8.3 Packed Pixel Architecture 38
2.8.4 Use of Barrel Shifters 39
2.8.5 Function Implementation in Hardware Versus Software 41
2.8.6 Reliance on Host CPU 42
2.9 Conclusion 43

III. SOFTWARE DESIGN 45

3.0 Types of User Interface Designs 46

3.0.1 Command-Driven Interfaces 46

3.0.2 Menu-Driven Interfaces 47

3.0.3 Graphics-Driven Interfaces 47

3.1 Device-Independence 48

3.2 Graphics Standards 50

3.2.1 Low-Level Graphics Standards . . . 51

3.2.1.1 The North American Presentation Level Protocol Syntax 51

3.2.1.2 The Computer Graphics Interface 51

3.2.1.3 The Computer Graphics Metafile 52

3.2.2 High-Level Graphics Standards . . 53

3.2.2.1 The Graphics Kernel System 53

3.2.2.2 The Programmer's Hierarchical Interactive Graphics Systems 53

3.2.2.3 Comparison of PHIGS and GKS 54

3.3 Graphics System Software Design Considerations 55

3.4 Hardware Implementation of Graphics Standards 60
3.5 Graphics Software Development Tools . . . 61
3.6 Conclusion 65

IV. OTHER DESIGN CONSIDERATIONS 67

4.0 Characteristics of Color CRT Displays . . 68
4.0.1 Size 70
4.0.2 Resolution 70
4.0.3 Phosphor 71
4.0.4 CRT Faceplate 71
4.0.5 Bandwidth 72
4.0.6 Refresh Rate 72
4.0.7 Horizontal and Vertical Synchronization 73
4.0.8 Shadow Mask 73
4.0.9 Misconvergence 74
4.0.10 Auto-Tracking 75
4.1 Human Factors and Ergonomic Considerations 75
4.1.1 Radiation Hazards 76
4.1.2 Flicker 77
4.1.3 Character Fonts and Sizes 77

4.1.4 Contrast Ratio 78
4.1.5 Glare 78
4.1.6 Labeling and Placement of Controls 79
4.1.7 Mechanical Features and Requirements 79
4.2 Standards and Agency Approvals 80
4.3 Conclusion 81

V. TRENDS IN COMPUTER GRAPHICS 83

5.0 Modeling Techniques for 2D and 3D Objects 84
5.0.1 Two-Dimensional Models 85
5.0.2 Three-Dimensional Models 86
5.0.2.1 Wire-Frame Model 88
5.0.2.2 Polygon Mesh Model . . . 89
5.0.2.3 Curve and Surface Specification Methods . . 90
5.0.2.4 Procedural Modeling Technique 92
5.0.2.5 Solid Modeling-- Constructive Solid Geometry Approach 93
5.1 Modeling of Natural Forms 93
5.1.1 Fractal Geometry 95

5.1.2 Polygon Subdivision 96
5.1.3 Growth Algorithm 97
5.1.4 Particle Systems 98
5.2 Considerations for Displaying Three-Dimensional Models 99
5.2.1 Hidden-Line and Hidden-Surface Removal 101
5.2.2 Shading 101
5.2.2.1 Gouraud Shading 102
5.2.2.2 Phong Shading 102
5.2.3 Illumination 103
5.2.4 Texturing 104
5.2.5 Transparency 104
5.2.6 Aliasing 106
5.2.6.1 Spatial Aliasing 106
5.2.6.2 Temporal Aliasing 107
5.2.6.3 Anti-Aliasing 107
5.3 The Z-Buffer Concept 108
5.4 Input Methods for 3D Model Information 110
5.5 Computer Graphics Applications 111
5.5.1 Industrial Applications 111
5.5.2 Computer Graphics in Film Making and Animation 113
5.5.3 Computer Graphics in Fine Art . . 114

5.5.4 Graphics Design and Advertising . 115
5.5.5 Business Applications 116
5.5.6 Computer Graphics Applications in Science and Simulation 117
5.5.7 Computer Graphics in Medicine . . 117
5.6 Conclusion 118

BIBLIOGRAPHY 119

VITA 126

LIST OF FIGURES

FIGURE		PAGE
1.	Hierarchy of graphics functions	8
2.	Hardware model representation of a graphics system	9
3.	Special-purpose processor architecture for a flight-simulation system	10
4.	Special-purpose processor architecture for an image processing system	11
5.	Limited-purpose graphics processor architecture . .	12
6.	General-purpose graphics processor architecture . .	15
7.	Cache memory implementation	23
8.	Dual-port memory organization	23

CHAPTER I

INTRODUCTION

The discipline of Computer Graphics began formally with the work of Sutherland in 1963. In his Ph.D. dissertation it was shown how a computer could be utilized as an interactive design machine system and used for man-machine interactive picture generation. Today, computer graphics is one of the fastest growing disciplines in the computer industry. Recent advances in this area have generated a large number of inter-disciplinary applications. Computer graphics has become an integral part of computer aided design, computer aided manufacturing, image processing, pattern recognition, robotics, education, medicine, entertainment electronics, film making and animation, to name a few areas of application, and the number of new applications is growing at a very rapid pace.

This thesis focuses on the complexity and the issues involved in the design of an interactive computer graphics system. A survey of current literature in the field of computer graphics and related areas reveals recent developments in hardware and software architecture design and human interface standards with development efforts aimed

particularly at applications involving three-dimensional graphics.

The second chapter "Hardware Design" reviews various technologies available to computer graphics system designers. The chapter discusses the various issues involved in the design of an interactive graphics system. Design considerations including different architectures for dedicated graphics systems, single chip VLSI graphics processors and controllers and memory design relating to graphics systems are discussed with examples from the industry. Several unique architectures, specialized graphics chips and software techniques have been developed. A new generation of computer displays based on bit maps are being developed. The hardware to support both raster operations and bit-mapped block transfer (BitBlt) techniques for image display has been developed. There have also been rapid developments in new memory architectures and graphics IC design, with an effort towards establishing graphics chip standards taking place in the industry.

The third chapter "Software Design" reviews the software aspects of designing a computer graphics system. Interactive software techniques, portability and device-independence are discussed. Graphics standards and the use of software development tools are also discussed.

The fourth chapter "Other Design Considerations" focuses on various other aspects of the design of an

interactive graphics system. Human-factors and ergonomics in designing such a system are discussed. Graphics displays represent the most important device interface between the user and the graphics system. Characteristics of graphics displays, such as display size, resolution, phosphor type, video bandwidth, refresh rates, radiation hazards and flicker should also be considered when designing an interactive computer graphics system.

The fifth chapter "Trends in Computer Graphics" focuses attention on the recent developments in the area of computer graphics. Issues involving hardware and software design of three-dimensional graphics systems are discussed. Various hardware design techniques to improve performance of three-dimensional systems are reviewed and software techniques pertaining to three-dimensional graphics including modeling techniques are discussed. Applications of three-dimensional graphics in solid modeling for industrial applications, business applications, advertising, simulation, medicine, computer art, film making and animation are also discussed.

Computer graphics systems consist of many components whose characteristics must be considered carefully when designing a graphics system for a specific set of applications. Different design criteria must be used when designing systems for general-purpose, limited-purpose or special-purpose applications where the advantages and

disadvantages of various component characteristics must be evaluated closely to determine the overall performance and effectiveness of the computer graphics system in different application environments.

CHAPTER II

HARDWARE DESIGN

This chapter discusses the hardware aspects of interactive computer graphics system design. A functional model description of an interactive graphics system is defined. Recent advances in computer graphics such as new hardware concepts and architectures have resulted in the development of new technologies to meet the computational power, speed and interactive requirements of graphics applications. Architectures for various types of graphics systems are also reviewed in this section. Several new concepts such as bit-slice architectures for graphics and the graphics display pipeline concept are discussed. Examples with reference to present-day graphics systems are reviewed. One of the major bottlenecks in the architecture design of graphics systems is the optimum design and efficient use of display memory. In this context a section is devoted to memory design pertaining to graphics systems. There is a trend in the industry towards customizing the graphics chip set for specific graphics applications. In addition, the industry is moving towards standardizing graphics processors and controllers. This chapter characterizes the desirable properties of graphics

processors. Various industry standard graphics processors and graphics controllers are discussed and compared to bring out the internal architectural features being used in these devices. The design of an interactive graphics system depends largely on its application. Each system component should be matched to the capabilities of the other components. Software design must also be matched to the system and its applications.

2.0 Functional Model of a Computer Graphics System

The functional model of a computer graphics system can be described as consisting of a pipeline of logical processors operating on representations of objects. A logical processor in the functional model corresponds to one or more physical processors, and two logical processors in the model may share a physical processor. The functional elements are classified as follows:

(a) The application model contains a description of the graphical and non-graphical properties of an object in a format determined by the application program or the modeling software.

(b) The display file compiler (a logical processor) maps the application model to the structured display file.

(c) The structured display file contains a hierarchical description of the graphical representation of the object.

(d) The display processing unit maps the structured display file to the linear display file.

(e) The (segmented) linear display file contains graphics primitives and mode settings describing the object produced after modeling and viewing.

(f) The display controller maps the linear display file to the image on the display screen.

(g) The screen displays a vector or raster image.

The graphics functions form a hierarchy, with the central processing unit (CPU) and system memory management functions at the top and the actual drawing techniques at the bottom, as shown in Figure 1. At the highest level is application software processing, which separates graphics commands from other software tasks. Next follows the drawing preprocessing level, which sorts the commands into a series of executable tasks, usually called a display list. The circuitry that creates the drawing is called a graphics engine.

One of the primary tasks performed by the graphics engine is the management of the flow of data between the system memory and the bit map memory within the engine. Another important task is the conversion of the bit map into a series of raster addresses--a sequence of x-y coordinates

that correspond to locations on a screen. A cathode ray tube (CRT) controller then synchronizes the transfer of the bit mapped memory to CRT scanning patterns.

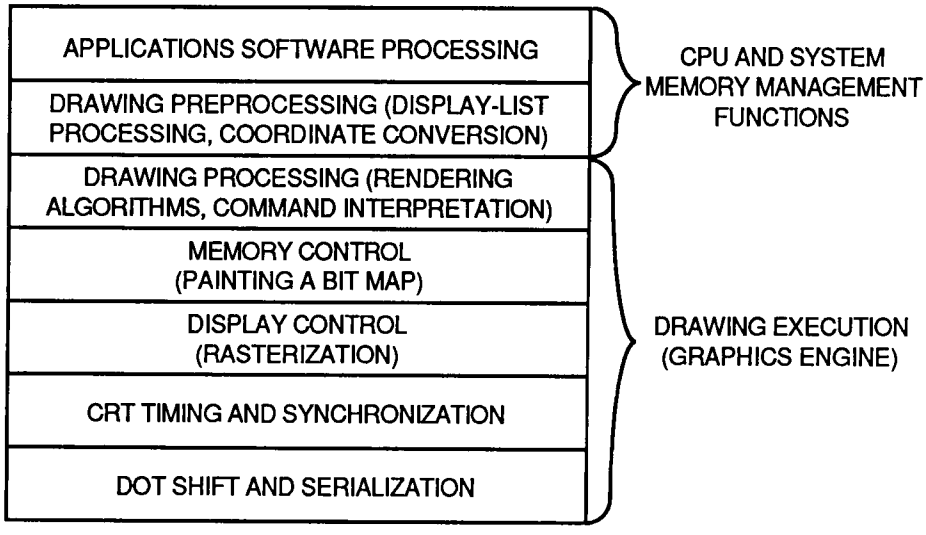


Figure 1. Hierarchy of graphics functions.

At the bottom of the functional hierarchy, a fast non-intelligent mechanism consisting of a high speed video-shift register, color palette (look-up table), and video digital-to-analog converter (DAC) moves the pixel data serially out of the bit map. The data is then converted to an analog signal and sent as separate red, green and blue (RGB) signals or as a composite signal to the CRT. A hardware model representation of a graphics system is shown in Figure 2.

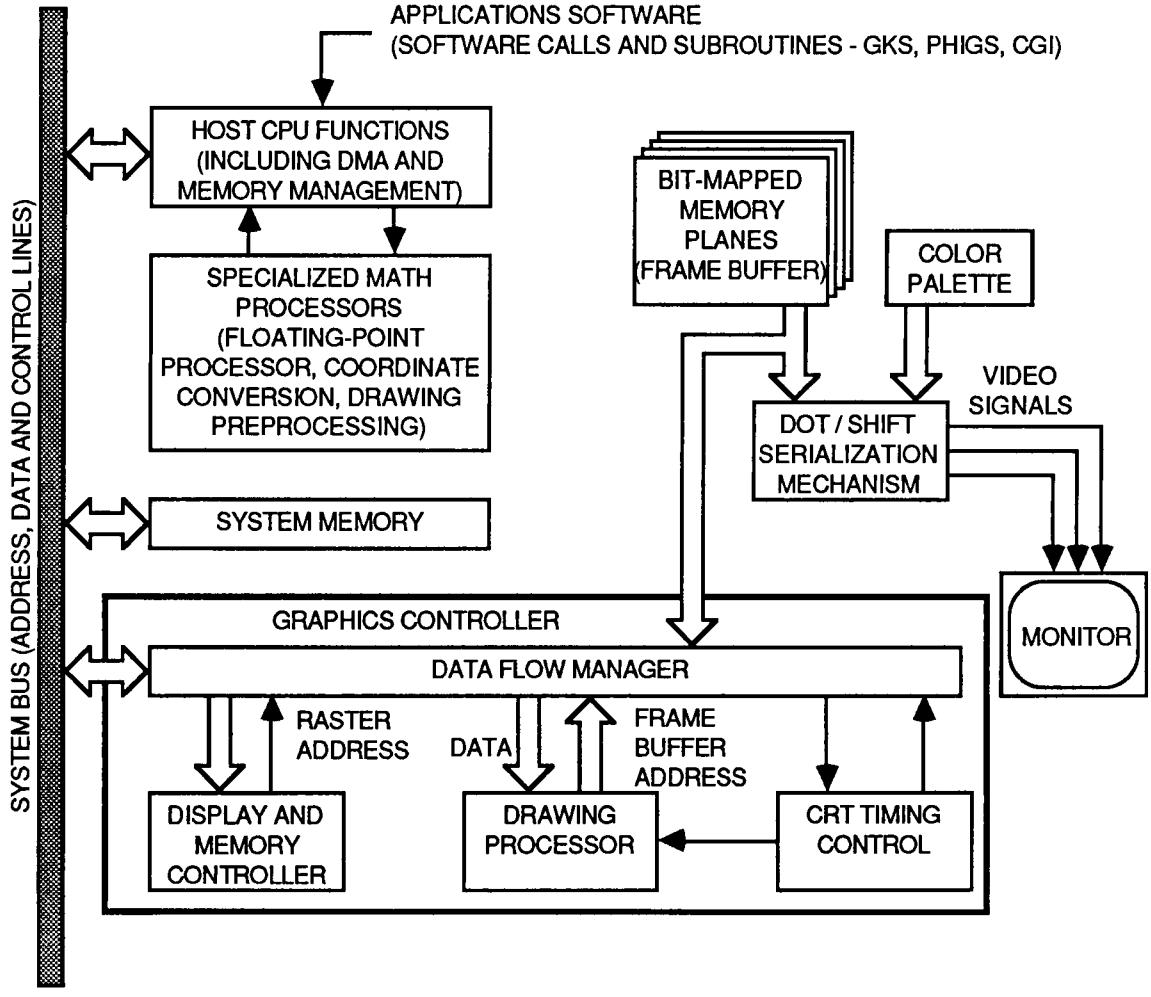


Figure 2. Hardware model representation of a graphics system.

2.1 Classification of Graphics Processors

2.1.1 Special-Purpose Graphics Processors

The architectures of special-purpose graphics processors are designed to meet the requirements of very specific applications. These types of architectures aim to achieve the highest level of performance for a particular task, but only by sacrificing flexibility. One example is in image processing. These architectures, though embedded with hardware for interactive image addition, subtraction, or convolution, still must rely heavily on the host processor to compute the address of every pixel of a graphic map overlay. Figures 3 and 4 show examples of special-purpose processor architectures for a computer generated imaging system such as that used in a flight-simulator and an image processing system, respectively.

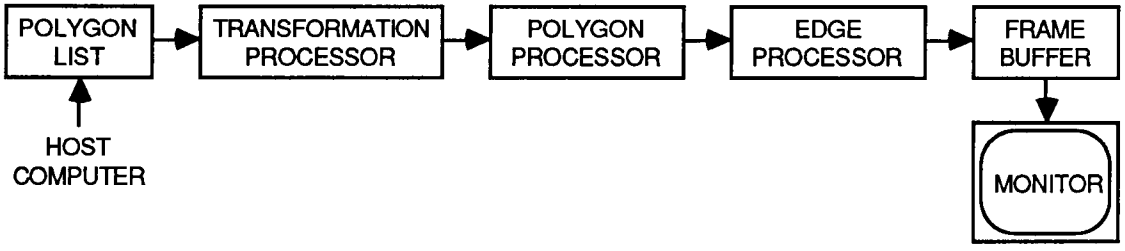


Figure 3. Special-purpose processor architecture for a flight-simulation system.

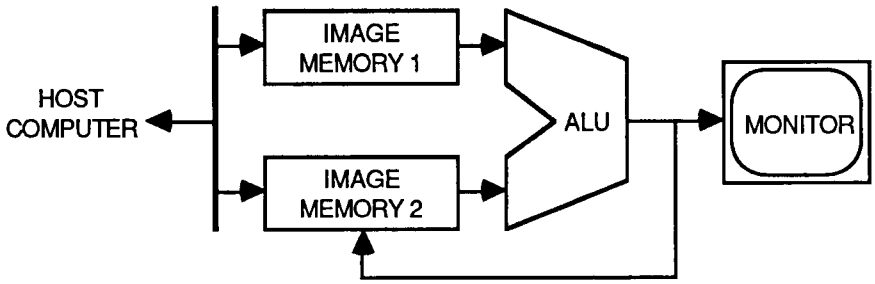


Figure 4. Special-purpose processor architecture for an image processing system.

2.1.2 Limited-Purpose Graphics Processors

Limited purpose graphics processors are designed to meet more than one specific application. The architecture of this class of processors typically has a small set of display functions which are optimized to provide a high level of performance for the set of applications supported. Examples of these functions are line drawing, character drawing, polygon drawing and transformations.

This type of architecture typically consists of multiple processing elements communicating in a fixed sequence. The block diagram of a limited-purpose graphics processor is shown in Figure 5. The communication processor is a general-purpose 16/32-bit microprocessor which handles communication to and from the application processor. An integer or floating-point transformation processor

(bit-slice or custom VLSI) provides transformation of three-dimensional data in a pipeline stage between the display list memory and the following graphics processor. On some systems this transformation task can be handled by the graphics processor and there would be no separate transformation processor.

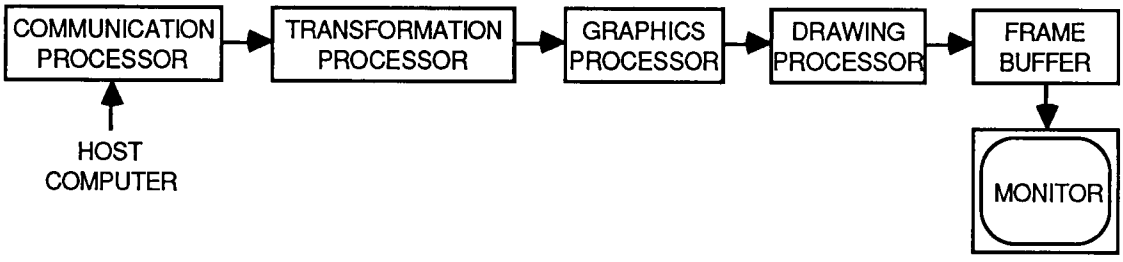


Figure 5. Limited-purpose graphics processor architecture.

The graphics processor is typically a 16/32-bit bit-slice microprogrammed device that performs an operation directly on the data it receives or preprocesses the data before sending it to the drawing processor.

The drawing processor is designed to operate at high speeds in the range of 3-30 million instructions per second (MIPS). The drawing processor can be a custom designed gate array or a bit-slice device which is tightly coupled to the raster frame buffer memory. The drawing processor is

capable of executing quickly, provided that they are simple functions like drawing vectors and characters.

Limited-purpose architectures are well-suited for simple applications where no bottlenecks in processing would occur. In other more complex applications, however, there are areas which cause problems. With a limited-purpose architecture, all frame buffer access must go through the drawing processor. For example, if the application requires that image data be displayed, a time consuming process of loading the color image data into the display list and then having the graphics processor draw a series of one-pixel-long vectors of the desired colors may have to be performed. In some systems the drawing processor can only load the data into the frame buffer memory but cannot read back the data from it. Consequently, image manipulation is impossible.

Another potential bottleneck lies in the limited set of functions and data structures provided to the application program. This can be overcome to an extent by additional microcode for the bit-slice graphics processor. In addition, microcode for this processor is usually kept at a minimum because of limited microcode memory and the difficulty of code development. The bottleneck is related more to the conceptual design of the system than to the hardware implementation. Such limited-purpose graphics systems are optimized for tasks that were well-defined and

well-specified at the time of design. They perform in an efficient manner only when restricted to providing a basic set of display functions. Present day workstations like the IRIS 2400 from Silicon Graphics are examples of this type of architecture.

2.1.3 General-Purpose Graphics Processors

General-purpose graphics processor architectures are designed to yield a relatively high level of performance in many different application areas. This is accomplished by providing large memory size, using a high performance 16/32-bit general-purpose processor with high memory-processor bandwidth, maintaining flexibility through modularity in hardware and software, and, finally, developing application-specific software. Such a system also uses a flexible display controller for ease of interface to display devices with varying resolution. An example of such a system is shown in Figure 6.

2.2 Bit-Slice Architectures

A major influence on the speed of graphics systems has been the implementation of bit-slice architectures. The ability to efficiently microcode wide-word architectures outweighs the problems encountered in programming unusual and untested code. Programming time is much longer with

microcode than with conventional languages having good software development tools.

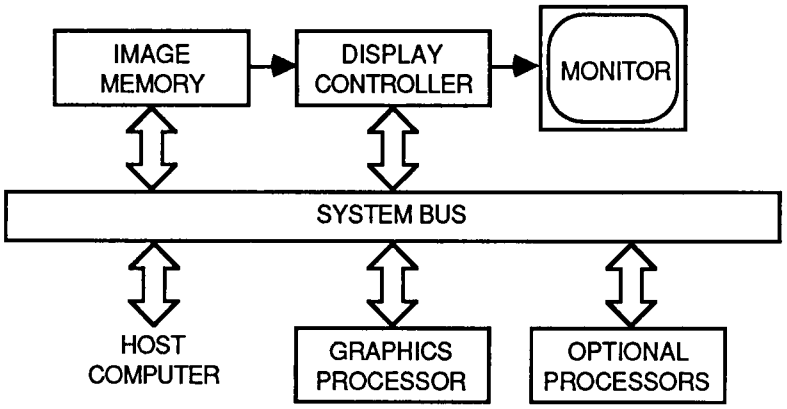


Figure 6. General-purpose graphics processor architecture.

An example of such an approach is a system developed by Qubix of San Jose, CA. In their computerized publishing system consisting of a distributed function architecture, multiple Motorola 68010 general-purpose microprocessors are used for their 2240 x 1680 x 1 pixel display. Using dual-ported random access memory (RAM), the 64K x 64 bit bit mapped system runs at a clock rate of 155 MHz. By scanning in 64-bit raster words, the system operates on only 1/64 of the clock rate, which is about 2.2 MHz.

A different approach is used by Parallax of Sunnyvale, CA. The bit-slice architecture is implemented by use of

discrete TTL devices instead of commercially available bit-slice CPUs. The versatility gained by this architecture allows drawing speed of 88 million pixels per second at a resolution of 1024 x 768 pixels, and a refresh rate of 60 Hz, non-interlaced. In this approach the simulation of the design of the system is possible prior to construction by using Programmable Logic Arrays (PLAs).

Several companies are now adopting the use of Advanced Micro Devices' newer 29116 series of bit-slice devices. The 29116 combined with the AMD 29516/517 16 x 16-bit multiplier makes possible an intensive rotation algorithm in 65 ns per multiply.

Many graphics processor boards available today make use of the bit-slice technology, while others make use of VLSI graphics integrated circuits (ICs). Both the VLSI graphics ICs and the bit-slice graphics processor boards offer several advantages. They both offer high resolution and fast drawing speeds. The graphics controller ICs are relatively easy to design with, they are generic to all system bus architectures and their cost decreases with increasing volumes. However, the bit-slice processors are faster, yielding vector drawing speeds of 250 ns/pixel, area fills less than 50 ns/pixel and shorter transform and clipping times. The main difference between the use of graphic processor boards and the use of VLSI graphics ICs is that of programmability. The bit-slice approach has the

advantage in that the graphics interface can be customized to a particular application's requirements. The bit-slice-based controller can be easily updated to comply with emerging graphics standards or it can be programmed to emulate an existing graphics device. Also, specialized graphics operations can be off-loaded from the host CPU to be executed by the bit-slice processor. Computer Graphics Interface (CGI) standard routines can also be microcoded to run on the bit-slice processor to supply a CGI-compliant graphics interface.

2.3 The Graphics Pipeline Concept

For high-end graphics systems, greater computational capability and speed are required than medium and low-end systems. High-end systems are typically used in applications such as aircraft flight simulation, three-dimensional (3D) modeling, computer imaging, and computer generated images for video and film-making. Such dedicated graphics systems are supported by a group of computational circuits that calculate the coordinates of the various two-dimensional bit maps that the master graphics engine finally combines and displays on the screen. These computational circuits include fast graphics accelerators and floating-point processors and are used to perform tasks such as shading an image, transforming a 3D image into a 2D

coordinate space, and generating display lists for a graphics engine.

In high-performance systems, raster graphics engines are made of fast bit-slice components where all instructions are hard-wired for speed rather than using VLSI parts. The pipeline concept transforms a model of some object into representations that become successively more machine-dependent and finally results in an image upon a particular screen. A number of operations using this concept may be performed: orthographic or two-dimensional modeling; three-dimensional modeling; solid modeling; translation or transformation of coordinates; clipping (to window or viewport limitations); dragging or rubberbanding; rotation; scaling; panning; zooming; representing depth by perspective or parallel techniques; hidden-line or surface removal; filling gray shades or color into surfaces; figuring shadows, reflections, highlights, or light transmission through a surface from scene lighting; scan conversion (vector to raster); and anti-aliasing. Some unit along the graphics display pipeline must be capable of executing the algorithms implemented in a particular system. Some manufacturers have started to apply custom VLSI to sections of the pipeline and make use of specialized processors to perform some of the operations which allow the user to have interactive control. These operations include

transformation (changing from one set of coordinates to another), translation, clipping, scaling, and shading.

Some manufacturers have also attempted to update parts of the graphics database offline. The intent here is to optimize the amount of concurrency and parallelism in the graphics system. Such techniques allow data to be handled efficiently so the system need not wait for data in operations that must be performed serially, such as pixel generation.

2.4 Display Memory Design for Graphics Systems

The architecture of the display/image memory also known as the frame buffer, particularly affects the performance of an interactive graphics system. Even though the display processor's access to the display memory for image update is a major factor in the design of the display memory, the architecture is significantly constrained by strict video timing standards as well as the limitations of semiconductor memories. The problem of contention for memory cycles impacts the image update performance, and is a significant factor in the design of the architecture of the graphics system. Processor design and access to memory are both influenced by the operations the processor is to perform. If the processor is to perform scan-line algorithms, sequential access to the display memory is desirable. If

area-oriented algorithms are to be performed, area access is preferred. If video rate input to the display memory is required simultaneously with the video output, particularly in image processing systems, an input write data path similar to the video refresh controller read data path is required. Some of the different display design techniques are discussed below.

2.4.1 Double-Buffer Systems

Double-buffer systems are used where one memory area is accessed by the display processor as it draws in a new image and the other memory area is accessed by the refresh controller to feed data to the video output. Double-buffering is expensive in that it requires twice as much memory; however, it is essential to ensure a smooth transition between successive images in applications in which an entirely new screen of data cannot be computed and/or written into the memories in the vertical retrace interval. Even if a new image can be written to the memories in one frame time, the display must be erased between frames, and this causes flicker. Therefore displays of moving, smooth-shaded solids and complex vector objects almost always require double-buffering.

2.4.2 Variable Resolution Memories

Some systems offer users the ability to trade pixel depth for x-y resolution. For example, a 1024 x 1024 x 8 memory can be alternately addressed as a 512 x 512 x 32 memory. Double-buffering can be implemented in these systems by using the higher resolution format.

Variable resolution is accomplished by manipulation of the video buffer. A buffer that can support one bit per pixel at 10 nanoseconds (ns)/pixel can also support four bits at 40 ns/pixel and eight bits at 80 ns/pixel. The video refresh controller formats the video buffer bits into pixels of a specified depth and passes them on to the video output circuitry at the rate required by the specified display resolution and refresh rate.

2.4.3 Cache Memories

Another way to increase the overall writing rate to the memory is to write to all memory chips during a cycle just as all chips are read at one time during a video refresh read cycle. Data from this parallel write is computed and stored in a cache buffer. Typically, the number of cache locations matches the number of memory chips in the system; i.e., a 1280 x 1024 system built with 20 64K dynamic random access memories (DRAMs) has a 20-pixel cache.

The cache consists of with high-speed components which enable it to accept data at the maximum rate of the display

processor. Since all access to the image memory from the display processor is through the cache, the cache has a read-write port to both the display memory and the display processor. An alternative solution is to have separate read and write caches. Figure 7 shows the representation of this technique.

2.4.4 Dual-Ported RAM

With dual-ported RAM, such as the Texas Instruments TMS4161, one port operates as a standard 64K x 4 memory and the other as a 256 x 1 shift register. The chip can be accessed simultaneously form the standard port, including page mode accesses, and from the shift register port. The memory cannot be accessed externally from either port during a move of data between the memory and the shift register. Also available is the TMS4416 as a 64K X 4 memory. The organization of the dual-port memory is shown in Figure 8.

The internal shift register functions as a large video buffer. This allows the total parts count to be reduced by eliminating the external video buffer, and display processor memory access is improved because of the buffer size.

The TMS4161 and the TMS4416 can be used in single-buffer systems and systems that double-buffer in a manner which shares memory chips. In a full double-buffered system, the saving occurs in the size of the shift register, since the display processor already has total access to one

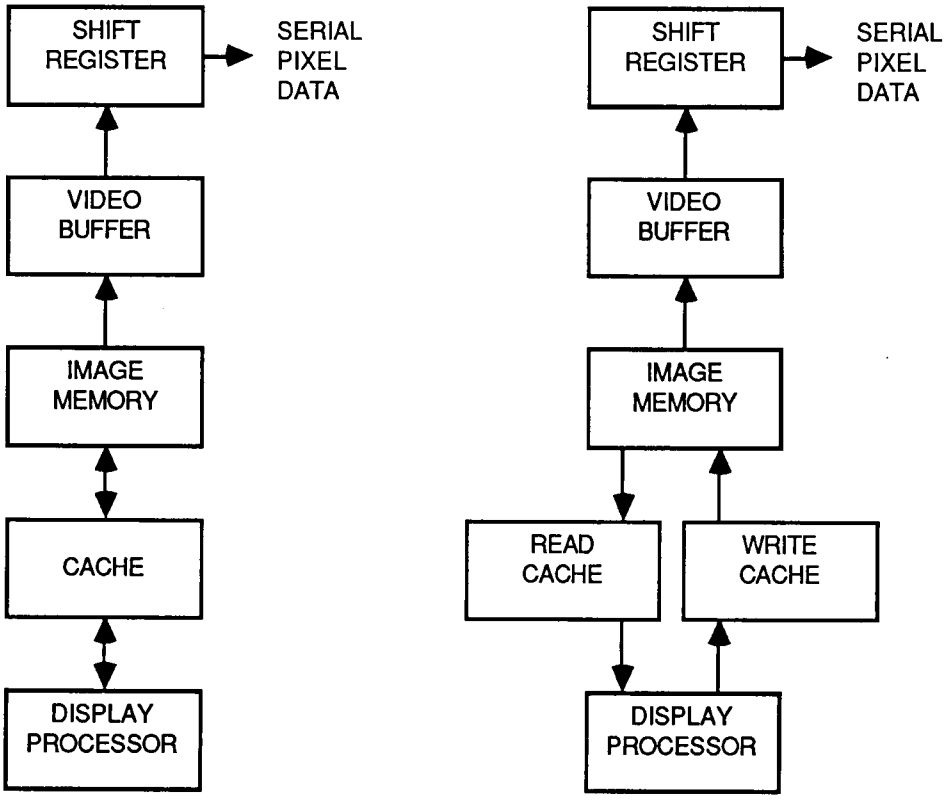


Figure 7. Cache memory implementation.

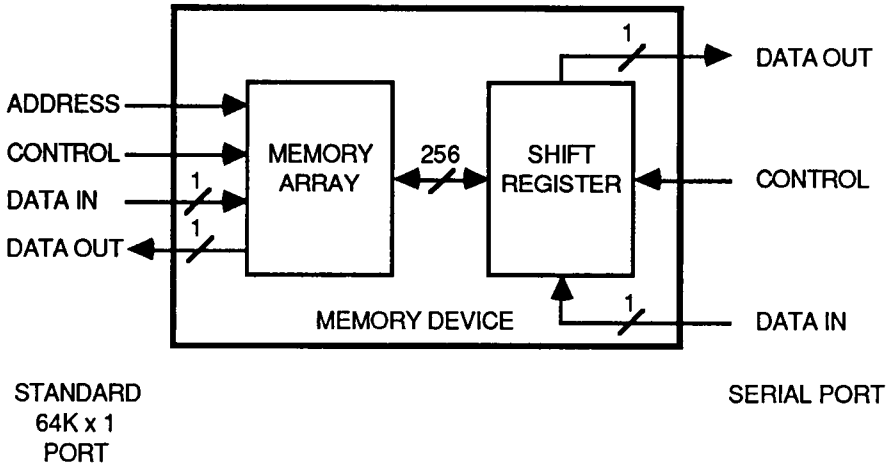


Figure 8. Dual-port memory organization.

memory while the video refresh controller has total access to the other.

2.5 Graphics Processors and Graphics Controllers

A graphics processor performs general-purpose processing tasks and is capable of efficiently manipulating pixels in the frame buffer. The graphics processor can also be programmed to perform virtually any graphics operation, unlike graphics controllers whose command sets are encoded in hardware. These controllers require constant supervision from a host CPU but a graphics processor can operate independently, thereby relieving the host of many of its graphics management responsibilities.

The graphics processor creates the image data, whereas the graphics controller is used as a special memory-management unit to read or write data from or to the frame buffer and calculate addresses needed for drawing operations. The controller is also responsible for implementing screen refresh tasks, including transferring pixel data from the frame buffer to the display. The graphics controller can also perform repetitive tasks easily, such as an area fill, upon a single command from the processor. The display controller also reads data out of the frame buffer and synchronizes memory reads and writes with timing of the display.

Graphics controller ICs which are available today are targeted mainly for mid-range systems which use two-dimensional graphics, support multiple windows with integrated text and graphics, have a resolution between 640 x 450 and 1024 x 1024, and act as a user interface to many applications and databases. Other features such as windowing and clipping are also being integrated into graphics controller ICs. Windowing and clipping involve scaling the image to the size of the window and/or clipping the image to the boundary in such a way that lines and edges terminate without looking uneven.

The major limiting factor in the performance of a graphics processor is speed. The speed of a system is dependent upon several factors: the raw clock speed, the efficiency of the processor design, the balance between hardware and software implementation of certain graphics functions, and the overall architectural design of the display system, particularly the organization and management of the display memory.

2.6 Requirements for a Graphics Processor

2.6.1 Programmability

A graphics processor must be programmable in much the same way that stored-program computers are able to fetch instructions from memory. It must be able to execute a

complete set of general-purpose instructions, as well as perform graphics functions. The instruction set should include such functions as arithmetic and Boolean operations, data moves, conditional jumps, subroutine calls, and returns. The graphics processor should also accept interrupt requests from external devices. With these capabilities, both non-graphics programs can be executed on the processor.

Program storage should be external to the graphics processor chip and consist of an arbitrary combination of RAM and read only memory (ROM). The graphics processor must also be able to execute programs written in high-level languages. Rather than provide a set of hardwired, high-level graphics functions, the instruction set should include a set of graphics primitives from which the programmer can construct efficient high-level functions. This allows the graphics processor to be able to adapt to existing graphics environments and to accommodate emerging graphics standards. The architecture must be designed to support a variety of pixel sizes, frame buffer sizes, and screen sizes efficiently.

2.6.2 Thirty-Two-Bit Internal Data and Address Paths

The internal data paths and general-purpose registers of the graphics processor should be 32 bits wide to support efficient manipulation of 32-bit addresses, which are

necessary for graphics applications, as well floating-point and 32-bit integer arithmetic. The large frame buffers and tables used in graphics applications demand a large, linear address space. The segmented architectures used to extend some 16-bit machines are therefore inadequate for graphics.

2.6.3 Instruction Set and Built-In Hardware Functions

The graphics processor must provide built-in functions for rapidly drawing lines, circles, and other curves using efficient incremental drawing algorithms. Pixel-array operations used to draw bit mapped text into the frame buffer must be fast enough to achieve response times necessary for interactive text applications. Hardware support for window functions must also be provided. The processor must provide support for clipping lines, circles, and other curves.

2.6.4 Interfacing to Other Devices

The graphics processor's interfaces to other integrated circuits in the subsystem must be designed to reduce the total parts count. Integrating certain non-processor functions such as CRT timing circuitry, display refresh control, and DRAM refresh control, with the graphics processor will reduce the parts count and improve performance. Of particular importance is integrating the DRAM control and multiplexed address generation into the

processor's memory interface. The graphics processor should interface directly to both the frame buffer and other local memory it needs. A single memory bus can be shared by both the video RAMs used for the frame buffer and the conventional DRAMs needed for program and data storage.

A communications interface is also required between the host processor and graphics processor. This interface facilitates the transfer of display lists from the host to the graphics processor and also provides a means for the host to download programs into the graphics processor's memory. The graphics processor should interface easily to a variety of host processors. The total parts count can therefore be significantly reduced if the host-graphics interface is part of the graphics processor chip.

In addition, the graphics processor chip should not incorporate any functions that fix the number of bits per pixel, frame buffer dimensions, or screen size since this limits the graphics processor to specific screen resolutions and sizes.

2.6.5 Other Characteristics

The graphics processor must support both color and black-and-white displays. For pixel operations, a large amount of special processing hardware is required. This hardware includes barrel shifters, support for x-y coordinate translation, and window comparators. Special

hardware is also needed to determine when a pixel has the attribute of transparency, allowing the background behind it to show through. The graphics processor must be able to efficiently move groups of bits within the bit map and perform logical operations on them.

A barrel shifter is used to position bits in particular locations across word boundaries. The barrel shifter can take two words containing a bit pattern which crosses their boundary and output a single word containing the partial but properly positioned bit pattern for the destination word. To complete the pattern transfer, additional shifts are performed by inputting a single additional source word per shift and always shifting by the same amount.

2.7 Limitations of Hardwired Graphics Controller ICs and General-Purpose Graphics Processors

Graphics controllers can perform certain graphics operations more rapidly than the host processor can by itself but these controllers have limitations. They cannot be easily enhanced to perform functions other than those for which they were specifically designed. New commands cannot be added to the controller except by designing a new version of the chip. Adapting such a controller to certain established graphics environments or graphics standards may not always be possible. The command set which the

controller executes usually consists of algorithms for drawing lines and other figures. Emulation of any command set other than that embedded in the controller requires an external microprocessor to translate commands from the applications software into the commands executed by the controller.

Graphics controllers today have other limitations as well. Some are able to handle few or only one pixel size effectively. Also, while some newer controllers perform pixel block transfer (PixBlt) or bit boundary block transfer (BitBlt) operations using Boolean operations, they may not include the arithmetic operations needed for manipulating color pixels. To simplify the on-chip hardware needed to perform PixBlt operations, some controllers process only a single pixel at a time, making them too slow for many text applications.

Although general-purpose processors have the programmability for graphics processing tasks, they do not have the special capabilities and hardware interfaces which are needed. Some of the latest 32-bit processors do have capabilities for extracting and inserting fields in memory which are useful for pixel manipulation but the processing is very slow. PixBlt operations also pose certain problems for the general-purpose processors. The fastest processors are incapable of supporting arbitrary pixel alignments in memory, arithmetic operations on groups of pixels, x-y

coordinate translation, window clipping, color-plane masking, and transparency control.

Existing general-purpose processors usually provide for a relatively small number of internal registers. The inner loops of graphics algorithms frequently contain more than 16 variables and when the number of variables exceeds the number of registers, a great deal of overhead can be incurred in moving variables in and out of the register file. This dictates the need for a large register file in a graphics processor since sixteen registers may not be enough for many graphics algorithms.

The hardware interfaces of current general-purpose processors are not well-suited to reducing the parts count for the graphics hardware. A number of support chips may be required to interface to video RAMs and DRAMs. External CRT-timing control and screen-refresh control must also be added.

2.8 Comparison of Various Graphics Processors and Graphics Controllers

In this section various industry-standard graphics processors and controllers are compared. The special architecture techniques and other unique features of these

chips are explained. The following processors and controllers are discussed in this section:

(a) Intel 82716 graphics controller

(b) Nippon Electric Corporation NEC PD7220A graphics display controller

(c) Texas Instruments TMS34061 graphics system processor (GSP)

(d) Motorola MC68490 raster graphics display processor

(e) Advanced Micro Devices Am95C60 quad pixel data flow manager (QPDM)

(f) National Semiconductor DP8500 raster graphics processor (RGP) and DP8510 BitBlt processing unit (BPU)

(h) Hitachi HD63484 advanced CRT controller

(i) Texas Instruments TMS34061 video system controller

2.8.1 Simultaneous Support of Text and Graphics

Text and graphics can be integrated by implementing by dedicating a separate window or scrolling area on the screen for text. This allows text to be stored in memory separate from the graphics data. The graphics data is stored as pixel data and the text is stored as ASCII. The ASCII code is sent to a character generator and then to a special shift register which outputs pixel data to the display at the

video rate. The Intel 82716 graphics controller supports text and graphics in this manner by using separate memory areas which serve as windows, but it can also superimpose text on graphics. The NEC PD7220A separates memory into text and graphics partitions and maintains up to four independently scrollable areas which can be used for either text or graphics.

Another scheme of supporting text and graphics simultaneously is BitBlt. This scheme avoids redrawing the entire display each time the screen must be updated by moving blocks of pixel data to various locations. BitBlt requires that processors operate on bit boundaries as well as word boundaries. In addition, these processors must be able to efficiently move blocks of display memory of various sizes from one place to another within the display memory. BitBlt allows an N x N array of pixels in the frame buffer to be defined and passed through various processing techniques before ultimately being transferred to another area in the frame buffer. Since operations are performed on a pixel-by-pixel basis, transfers do not need to be aligned on word boundaries in the frame buffer, but the processor handling the BitBlts must be able to work across word boundaries.

Two ICs which use BitBlt techniques to integrate text with graphics are the Hitachi HD63484 and Texas Instruments TMS34061 video system controller. These ICs use a frame

buffer which is larger than the screen coordinate space. They can then store various fonts in the off-screen memory and copy a given character as many times as necessary to the display area of the frame buffer. The character can also be scaled, rotated, or colored in the copying process. The Hitachi HD63484 also uses separate character memory and a character generator ROM to handle text.

The Advanced Micro Devices Am95C60 quad pixel data flow manager also supports bit-mapped text and graphics. It uses two areas of off-screen memory to handle text where one area stores the font and the other area contains a table of parameters and attributes that apply to each character. Using BitBlt, the Am95C60 can transfer up to 60,000 16 x 16 pixel characters to the screen per second.

National Semiconductor has introduced a two-chip set which consists of a single 8500 raster graphics processor and a slave 8510 BitBlt processing unit. With these two ICs, virtually any bit in any number of bit planes can be addressed by assigning the BitBlt processing unit to each bit plane. To address a bit within a pixel, the raster graphics processor generates an address in the plane and enables only those BitBlt processing units which must respond to change the addressed bit. A BitBlt processing unit can be controlled by a state machine, a general-purpose microprocessor or the raster graphics processor. By breaking out the bit addressing into two chips, it gives the

system designer greater flexibility in achieving desired cost/performance trade-offs.

2.8.2 Memory Management: Parallel Pixel Access

One major characteristic of display processors which particularly distinguishes one processor from another is the manner in which the processor manages display memory. Display memory, or more accurately, displayable memory, refers to the entire range of memory which the display processor is capable of addressing. Since the address ranges of display processors are typically as large as 4 Mbytes, all of the display memory cannot be displayed at once. The frame buffer, or bit map, is the portion of display memory which is currently being used to output data to the CRT for viewing.

In a single-bit-plane system where there is one bit per pixel, each bit maps directly to a corresponding pixel, or picture element, which is either turned off or on in a monochrome display. In color systems or gray-scale monochrome systems, there may be more than one bit corresponding to a particular pixel on the screen but these groups of bits may be arranged in memory planes or in a linear address space.

In systems where the display memory is larger than the frame buffer used for the current display, the display processor can change the display in one of three ways. It

can write new data into the frame buffer from commands received from the host CPU, it can redefine different areas of the display memory as the current frame buffer, or it can transfer blocks of pixel data into the current frame buffer from other portions of the display memory. Some processors can also use display memory to store other information such as instructions, tables and other data required to manage the display even though they will never be output to the CRT.

In displays where each pixel is defined by multiple bits to specify color or gray scales, problems may arise in managing the bit-mapped display memory. The traditional approach in attempting to overcome these problems has been to separate the display into bit planes where each plane corresponds to a bit in the mapped pixel. However, traditional microprocessor addressing schemes are designed to access bits as part of a word, whereas a graphics system often needs to access and change a single bit in a pixel. For a conventional microprocessor to handle these pixel bit changes, it must first select a plane corresponding to the bit to be changed, access the word (either eight or 16 bits within that plane) and perform a read/modify/write operation. If two bits within that same pixel need to be changed, two such operations must be performed. If the processor cannot execute these operations within the time

constraints of the video refresh rate, an intermediate color will appear as an artifact on the screen.

It is not possible for processors to modify multiple bits in a pixel by sequentially accessing each memory plane within the time needed for flicker-free display. This is due to the increasing resolution of graphics displays and the increasing number of colors which must be displayed. With higher resolution there are more pixels to process and less time to do the processing. One way to overcome this limitation is to process bit planes in parallel, but dedicated hardware must be used to do this. This approach increases system cost because of the increased chip count, pin count, printed circuit board area and power consumption, but it produces a constant pixel-processing rate for any selected pixel depth.

AMD's Am95C60 is designed to access four bit planes in parallel. It uses 64 lines which can be grouped as four 16-bit buses so it can address four 16-bit words in parallel, or sixteen 4-bit pixels, on each memory access. Multiple QPDMs can be cascaded to handle 8-, 12- and 16-bit pixels, and so on.

With this type of architecture, the process of updating a bit map, such as in a block transfer, is largely dependent upon the accessing and processing speed of the QPDM and not upon the pixel bit depth. For example, a read/modify/write

operation could require 800 ns for a 16-bit word, or 50 ns per pixel, independent of the number of bits per pixel.

The National DP8500 raster graphics processor is also designed to access pixels in parallel. Unlike the Am95C60, the DP8500 must use a separate subprocessor, the DP8510 BitBlt processing unit, for each bit plane. The DP8510 can act as a slave either to the RGP or a general-purpose microprocessor. The BPU has a control register for storing commands, a barrel-shifter for aligning pixels on bit boundaries and a logic unit for BitBlt operations. A single RGP can control up to eight BPUs but buffers would be needed to control more. If buffers are used, any number of BPUs can be controlled by a single RGP. The RGP is responsible for sending instructions and enabling or disabling the BPUs as needed. Upon commands from the RGP, the BPUs will move data within a plane or within several planes, move data from one plane to another or move data from one plane to multiple planes, and also execute logical operations on that data.

2.8.3 Packed Pixel Architecture

A newly developed architecture for planar organization of the bit map is the packed pixel architecture. In this architecture, the processor addresses the bit map as a single, linear address space, rather than as an address space multiplied by some number of bit planes. The number of possible bits per pixel is therefore limited to the

number of bits in the processor's data word size. A 16-bit processor could therefore address a bit map with 16 bits per pixel. Any combination of bits in a pixel could be changed with one read/modify/write operation. For example, the 16-bit data word could be set to contain two 8-bit pixels, four 4-bit pixels and so on. The packed pixel architecture is well-suited for 32-bit architectures since it offers fast pixel processing for a specific pixel depth along with a much lower pin and package count than a parallel processing approach.

Three ICs which use the packed pixel architecture are the TMS34010 graphics system processor from Texas Instruments, the 68490 raster graphics display processor from Motorola and the 82786 graphics controller from Intel. The TMS34010 is a full 32-bit graphics processor which includes display memory and video control circuitry.

2.8.4 Use of Barrel Shifters

Regardless of the memory organization, internal architecture or speed of a graphics processor, the processor must be able to efficiently move groups of bits within the bit map and perform logical operations on them. There are many ways in which bits can be shifted within words and across word boundaries. This becomes important when a pattern of bits which crosses a word boundary must be moved to another location where it may also cross a word boundary,

but in a different position relative to that boundary than it was in its original location.

The use of a barrel shifter is one way of aligning bits in particular locations across word boundaries. The barrel shifter can take two words containing a bit pattern which crosses their boundary and output a single word containing the partial but properly positioned bit pattern for the destination word. To complete the pattern transfer, additional shifts are performed by inputting a single additional source word per shift and always shifting by the same amount. The Am95C60 and TMS34010, for example, contain internal barrel shifters. National Semiconductor has placed the barrel shifter in its DP85010 BitBlt processing unit, dedicating a barrel shifter to each memory plane.

The National DP8510 BPU is dedicated to bit boundary alignment and logical operations and has a 16-word first-in/first-out buffer. In addition to accessing pixels as 16-bit words in a plane, the BPU has a pixel port that allows the system to pull out any particular bit within a word. When multiple BPUs are used, it allows instant access to the bits in each plane. These bits, which constitute a pixel at the specific depth of the system, can then be used as data words by the DP8500 RGP or the host processor. This feature is useful for image-processing tasks that must perform a large number of very fast operation on pixel data.

2.8.5 Function Implementation in Hardware Versus Software

One of the fundamental differences between graphics processors concerns their design philosophy as to what functions should be implemented in silicon and what functions should be left to the user to program in software. It is a choice between speed and flexibility. In each case, the processor relies on a stream of instructions from a host CPU or a specific geometry engine. These instructions, namely the display list are written in terms of the processor's instruction set and once they are supplied, the processor can run the display list independently.

The instruction set for AMD's QPDM are more specifically oriented toward graphics processing, and sets of its registers are specifically oriented for video control, display configuration and window control. It is also the most dependent on an external processor. The instructions themselves execute microcoded algorithms for drawing functions such as line, arc and circle, and fill operations. Clipping, anti-aliasing, color search and even line style are all covered in the instruction set.

National's DP8500 RGP, on the other hand, combines features of the very hardware-oriented instruction set of the QPDM and the very software-oriented approach of TI's GSP. The DP8500 incorporates many hardware draw instructions including polygon, curve and the most widely used line drawing algorithm, the Bresenham algorithm. But

the 8500 also supports moves and arithmetic functions (except divide) and uses a conventional stack in memory.

The TI graphics system processor, being architecturally the most similar to a general-purpose microprocessor, relies on software algorithms to perform its graphics functions. One architectural feature that distinguishes this processor from others is a 256-word on-chip instruction cache. Many graphics algorithms are highly iterative and consist of relatively short loops, so even a few instructions cached on-chip can contribute a great deal to speed and efficiency.

2.8.6 Reliance on Host CPU

The host CPU can place instructions in the bit map, or the CPU can feed instructions directly to the QPDM's instruction FIFO buffer. To avoid tying up the CPU for too long, AMD recommends use of a universal DMA controller, especially when multiple QPDMs are used. The DMA controller can keep a steady stream of instructions, which have been placed in a memory area by the CPU, flowing to the QPDMs while servicing requests for one or more controllers. To read and modify contents of the bit planes, controllers must be addressed separately via chip selection logic. Instructions, however, are transmitted to all QPDMs simultaneously.

The National 8500 family uses a scheme that is similar conceptually, except that the 8500 RGP is itself the master in terms of graphics system control. Specific graphics operations, as well as general processor functionality, are concentrated in the RGP with specific control for each individual bit plane allotted to the 8510 BPUs.

In case of high-end systems, one RGP can control many bit planes via individual BPUs, which can be used without the RGP but with a general-purpose microprocessor. For low-end applications, it is possible to use a microprocessor controlling one BPU that can be multiplexed between a small number of bit planes.

In the case of TI 34010, it can operate independently of the host CPU. While a system design may designate a master system CPU, this chip can address system memory as directly as its bit map memory and, conversely, the host CPU can directly address the bit map.

2.9 Conclusion

The design of the architecture of an interactive graphics system is very application dependent. Functionality, speed and flexibility dictate the architecture design used. In the design of high-end graphics systems and systems aimed at specific applications, the bit-slice processor approach along with the graphics

pipeline concept are being widely used in the industry. Graphics equipment manufacturers are customizing portions of the hardware to streamline the architecture and overcome the limitations of off-the-shelf graphics devices. There is a tendency towards customizing the graphics processor to enhance the parallelism and concurrency of graphics tasks. Architectures employing concurrent and parallel use of graphics processors and the use of custom graphics processing modules are being developed. This has given way to the design of private bus structures for these architectures.

For low- and medium-end graphics systems, standard VLSI graphics processors and graphics controllers are being used. These chips support a large variety of graphics functions. The manufacturers of these processors and controllers are trying to standardize the functions in an attempt to establish their particular chip set as the industry standard. The support logic for a graphics processing system such as memory and CRT control logic functions are being integrated into these chips to streamline the architecture in terms of cost/performance, speed, power requirements and minimization of total parts count in the system. Each of these architectures represents a different design philosophy with its own unique trade-offs, advantages and limitations.

CHAPTER III

SOFTWARE DESIGN

The software infrastructure of a graphics system must be designed to meet the ever-increasing demands for higher performance, greater interactive usability and new applications. Graphics devices currently available have dissimilarities which make it difficult to design a single system which will run optimally on many of them. Also, there is a high risk of early obsolescence because of the high rate of growth in hardware technology. To minimize these risks and take full advantage of the capabilities of the graphics devices available today, the system designer must design the hardware and software architecture very carefully.

This chapter discusses the various concepts in designing the user interface for graphics software. In addition, designing the software to be device-independent allows greater portability and better usability for the end user. Various standards relating to graphics software have been adopted or are being reviewed by international committees on graphics for future standardization. These standards are influenced by the hardware architecture design

and graphics IC design. An overview of these standards, their purpose and the relationship between them is discussed. While designing the software for an interactive graphics system the designer must consider whether to customize software or develop the software around a particular standard. Many graphics chips have integrated special hardware features to support these graphics standards.

3.0 Types of User Interface Designs

The ease of use of the graphics system, including fast response to user input and feedback to the user are important considerations in designing the software for a graphics system. There are three primary types of user interfaces: command-driven, menu-driven and graphics-driven interfaces.

3.0.1 Command-Driven Interfaces

Command-driven graphics systems are best-suited for experienced users. Since a command-driven system offers very few prompts or choices, the user is expected to know which programs are available and how they are to be used. The system usually responds with an error message if a command is not recognized or if it is used incorrectly.

The main advantage of using a command-driven interface is speed. A limited number of keystrokes are needed to initiate any action and there is no need to pass through multiple levels of menus. This type of a system is usually unfriendly in the sense that it is difficult for a new user to quickly become familiar with the system.

3.0.2 Menu-Driven Interfaces

Menu driven interfaces offer, in a hierarchical manner, all possible choices which the user can select. A menu typically imposes a temporal order, where one choice or action must be taken before another. The selection of one menu item may bring up another menu, which may bring up yet another menu until a final selection allows a particular task to be executed. Thus, the user is not required to remember the name of any command or program. Menu-driven interfaces make system usage easy for the novice but frustrating for the experienced user who must move through several screens of prompts before reaching the desired function.

3.0.3 Graphics-Driven Interfaces

Graphics-driven interfaces provide the user an interface which is both friendly and non-restrictive. Rather than presenting items in temporal order, graphics-driven systems present to the user a spatial order

of possible actions which are represented by icons. Icons are pictorial representations of actions so a user can simply select an action by positioning a graphics pointer over an icon. The usability of such an interface is enhanced by having the visual interface oriented towards the selection of objects rather than forcing the user to execute a set of procedures to implement a certain function.

The use of windows is one example of a visual interface. Window-based interfaces provide the user with immediate visual feedback for each action taken. A user can view windows of data from different applications simultaneously, transfer data from one window to another, or access another application.

3.1 Device-Independence

Software applications for a graphics system must provide a means of interfacing to other systems. The graphics system must be capable of accepting input from other systems and providing output in a manner which can be easily used by other devices and systems. These characteristics are explained by the concept of device-independence.

Device-independence is a characteristic of a graphics support package that allows a single application program to run on different display devices. The graphics support

package is accessed by the application program to generate graphics output to, and obtain graphics input from, the graphics device(s) to which the system is connected. The software package maps the application program's graphics requirements into the specific graphics format of the target device.

This mapping is performed by modules called device drivers. A separate device driver is required for every graphics display which is supported. An application program can then send output to a new display device by simply loading the device driver for the target device with the application program and the device-independent support package. Device-independence also lowers the impact of obsolescence on graphics application development. For example, the designer can develop an application on a storage tube device, even though the application is targeted for a beam-directed or color raster display.

Current international standardization efforts are aimed at a common interface among device-independent software packages. The primary goal of device-independence is for a single application program to generate similar images on more than one type of display device. The application program would do so by accessing device-independent routines to build an image or obtain graphics input. In addition, the application program references a graphics device as a virtual graphics device and all graphics-related computation

in the device-independent routines is oriented toward this virtual device.

The virtual device interface is the communication link between the device-independent routines and one or more graphics device drivers. Each driver must map the device-independent commands passed across the virtual device interface into the specific commands understood by the target graphics device.

3.2 Graphics Standards

The concept of device-independence, portability of software applications and graphics data from one system to another and a need to standardize software and hardware architectures has led to the need for graphics standards. Graphics standards can be divided into two basic types:

(a) Low-level, device-oriented standards that interface applications-oriented software to device drivers, and,

(b) High-level, applications-oriented standards.

These two levels of standards when used together allow the application software transparent access to input devices and display hardware.

3.2.1 Low-Level Graphics Standards

Low-level device standards which are commonly used today are the Computer Graphics Interface (CGI) and the Computer Graphics Metafile (CGM) standards. There is also another graphics standard known as NAPLPS (North American Presentation Level Protocol Syntax).

3.2.1.1 The North American Presentation Level Protocol Syntax. This standard defines the method for encoding visual information, including both text and graphics. This encoded information is then used to drive video-text devices.

3.2.1.2 The Computer Graphics Interface. The Computer Graphics Interface (CGI), formerly called the Virtual Device Interface (VDI), provides the definition of a device-independent interface to device-dependent graphics equipment and services. CGI is the proposed computer graphics interface standard designed to act as a lower level device interface protocol to hardware devices. CGI has been designed such that programs using the high-level, applications-oriented Graphics Kernel System (GKS) can use its capabilities as a device driver.

CGI provides an interface to a virtual device which frees the application programmer from concerns about system devices. It interfaces directly to the device drivers available on the system hardware. Other kinds of graphics routines can also be written which incorporate CGI,

including non-standard vector and pixel routines. CGI includes the concept of the bit map and allows direct use of pixel arrays for BitBlt operations. CGI can also define bit maps on- and off-screen which is very important in implementing window environments.

CGI standardizes the interface between graphics systems and graphics devices by generating graphics primitives. CGI is also responsible for controlling the capabilities of graphics devices and obtaining graphics input.

3.2.1.3 The Computer Graphics Metafile. The Computer Graphics Metafile (CGM), formerly called the Virtual Device Metafile (VDM), is a mechanism for storing and transmitting graphic images. In addition to being device-independent, the graphics information in a metafile can be reprocessed and modified without regenerating the entire stored picture.

Another type of graphics metafile, the GKS metafile, has been proposed and is currently being reviewed for standardization by the American National Standards Institute (ANSI). The Computer Graphics Metafile stores a representation of the final image in a more compact manner than the GKS metafile can. It also offers three means of encoding the same information: character encoding, binary encoding, and clear-text encoding. Character encoding is compact and ready to transmit using standard networking file-transmission software, whereas binary encoding is simple to generate and interpret on a given host.

Clear-text encoding, by contrast, is intended for easy reading, editing, and debugging, but this method of encoding affects hardware performance.

In addition to offering a choice of encoding schemes, CGM permits storage of more than one image in a single metafile. Because these images are individually delimited and separated, they are randomly accessible. Their display in no way depends on their location in the file. CGM can also include non-standard graphical and non-graphical elements in an image file.

3.2.2 High-Level Graphics Standards

At present, the two most important application-oriented standards are the Graphics Kernel System (GKS) and the Programmer's Hierarchical Interactive Graphics System (PHIGS).

3.2.2.1 The Graphics Kernel System. The Graphics Kernel System Standard (GKS) defines the interface between an application program and a graphics model and it also shields the application software from differences among various computers and graphics devices. GKS is a two-dimensional graphics standard which is intended for simple graphics application environments, such as personal computer-based software applications.

3.2.2.2 The Programmer's Hierarchical Interactive Graphics Systems. The Programmers Hierarchical Interactive

Graphics System (PHIGS) is similar to GKS but it is a more extensive hierarchical interface for graphics applications such as computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), simulation, and modeling applications. PHIGS offers a higher level environment, targeting interactive design applications. Approved as a working draft, PHIGS is being considered presently by international graphics committees for implementation as a graphics standard.

3.2.2.3 Comparison of PHIGS and GKS. There are areas of applications where either PHIGS or GKS can be used. PHIGS, however, is targeted at high-end, highly-interactive, highly-dynamic programming applications. GKS was designed to be used with a broad spectrum of hardware devices ranging from plotters and printers to interactive workstations. The primitives of PHIGS are the same as the primitives of GKS, but PHIGS enables the full exploitation of the capabilities of high-end workstations. PHIGS is a three-dimensional standard as compared to GKS which is a two-dimensional standard. However, a three-dimensional extension to GKS, called GKS-3D, has been proposed and is being considered by graphics standards committees.

GKS-3D aims to provide a standard graphics application environment for end users, and supports a range of machines, from mainframe systems to microcomputer systems. GKS-3D can operate in a much less-costly environment than PHIGS.

Efficient display of three-dimensional objects requires algorithms to handle perspective, parallel projections, clipping, and hidden-line and surface removal. In GKS-3D, world coordinates are transformed to device coordinates through three intermediate coordinate systems.

Unlike 2D and 3D wire-frame graphics, which are primarily vector operations, solids modeling applications are very pixel-intensive. Using smooth shading, rather than specifying line endpoints or polygon edges and filling in between them, requires floating-point operations on each individual pixel. PHIGS as currently specified is a wire-frame standard.

Presently there are no approved three-dimensional standards. The three approaches which are being considered for possible standardization are the extension of GKS to GKS-3D, the use of PHIGS and the adaptation of PHIGS to be compatible with GKS. Whatever standard is adopted, however, 3D graphics primitives are needed for today's graphics applications and they will have to encompass both pixel and vector operations.

3.3 Graphics System Software Design Considerations

The software of an interactive graphics system must offer increased productivity to the end user. Therefore, execution speed and the efficient use of computer resources

are important factors in graphics software design. The rapid changes occurring in the computer graphics industry require that the development cycle of the graphics system be as short as possible in order to keep pace with changing technologies. Product life cycles are also becoming shorter as technology advances. These challenges put pressure on the system designer to simplify and shorten the process of designing, testing, and producing a new system. These constraints are severe, because investment in graphics hardware and software could be considerable and must be insured against obsolescence.

The first step in designing the software infrastructure is to define the functional and communication requirements of the system. This step parallels the hardware architecture design process. The second step is the selection of the hardware which meets the defined software and hardware architecture requirements. The type of graphics processor, frame buffer, display resolution and the operator interface are determining factors in the amount of processing power which the system will have. Selection of the graphics devices that will be used with the system must also be completed.

The graphics system designer must offset system functionality with control between the graphics processor and other intelligent graphics devices in order to achieve device-independence. This is done in order to produce a

flexible software infrastructure. This strategy also allows for ease of software development for the system and increases the life of the product through easier software maintenance and support. This approach, however should, be optimized such that it does not affect the performance requirements of the system being designed. In designing the software architecture for a system the designer can use one or more of the approaches discussed below.

The first approach is to develop software that is specifically written for the system hardware. In the process of developing a high-performance graphics application which uses the latest hardware technology, the graphics devices are selected before software development. This is done to achieve optimum efficiency, by customizing the graphics software for the selected devices and their interfaces to the system. Such an approach may or may not result in device-independence since system performance and software customization are prioritized. This approach requires a considerable software development effort in terms of time and cost.

The second approach is based on the software architecture of a standards-based graphics software package that offers substantial device-independence. This type of approach may completely address the hardware capabilities of the system and the graphic devices being used. The off-the-shelf, standards-based software package usually does

not provide the performance required from the system. However, there are certain advantages in taking this approach. One advantage is that the standard software package may be able to meet many of the functional specifications of a typical CAD or CAE application without modification. Also, the quality and effectiveness of the software package may have already been established by other users. A standards-based graphics software package provides portability of the software across processors, compilers, and graphics input/output devices. This increases the chances of acceptability of the product in the marketplace.

The third approach is to use a standards-based software package with extensions for optimal operation with selected devices. This allows hardware options to be kept open longer during the design cycle. In this way the system can be prototyped on several devices, and later optimized to run on the devices that are ultimately chosen for the final released product. This eases the debugging of the hardware and the customized interfaces for the graphics devices to be used with the graphics system.

In designing a high-end graphics system using a standards-based software package there are certain disadvantages. Graphics standards address a large number of devices, processors, and applications, so that standards can never be completely optimized for a particular system. This results in the degradation of performance of

speed-sensitive, highly interactive applications. For example, both the GKS and PHIGS standards process graphics primitives through a software pipeline of viewing transformations in order to provide versatility and device-independence. On a powerful, state-of-the-art graphics system this processing can be slow and often redundant. Another disadvantage in using standards-based software packages is that standards take several years to develop and usually lag behind current market needs and hardware technology. Many of the present-day advanced graphics systems use complex, device-dependent data structures which have evolved separately from the standards. Since these devices process and use graphics data differently, device-independence is difficult to achieve.

Other considerations include the selection of the appropriate standard for the system. In 2D applications, for example, it may be appropriate to use the GKS standard whereas in a 3D applications it may be appropriate to use the PHIGS standard. The architecture of the standard software package is also important. Another consideration in graphics software architectures for a high-end system is device-intelligence. Device-intelligence allows an application to run on various graphics devices, by taking full advantage of their special characteristics in the most efficient way. Since standards are not always at the same

level of state-of-the-art hardware, optimizing the overall system software is always needed.

3.4 Hardware Implementation of Graphics Standards

Development of graphics software today is closely related to standards issues and hardware developments since these developments are a direct result of standards definitions which already exist and implementations which have become de facto standards. Although there may be industry-wide agreement to some extent on the kinds of primitives graphics devices should support, there is no agreement on which functions should be implemented in hardware and which ones should be implemented in software. As a result, the current generation of graphics ICs can be grouped into two categories: one which defines graphics functions as a relatively small set of macro instructions and another which emphasizes microprocessor-like programmability. Companies which have selected the second category have had to devote a great deal of effort to software support. This latter category of chips includes the TI's TMS34010 and the National DP8500 chip set, while the former category includes the AMD Am95C60, the Intel 82786 and the Hitachi HD63484.

Standard software interfaces such as CGI and GKS, which will both eventually include window systems, are typically

supplied and supported by third-party software developers who modify them for a given chip. This development will eventually allow software developers to spend less time writing and supporting device drivers for a variety of different terminals, hard-copy devices and board-level controllers.

Highly programmable display processors which have the ability to emulate popular hardware for which a large amount of software has already been written have also helped in establishing standard hardware interfaces. Programmable display processors also allow users to optimize the algorithms for maximum performance. High-end system designers can design their own algorithms to suit their applications. Programmability of low-level devices also allows for innovative algorithms for drawing primitives.

3.5 Graphics Software Development Tools

Software tools and products with ever-increasing sophistication and utility allow a system developer to make various types of graphics hardware functional. In addition, they give applications easy access to graphics display and input facilities--unlike in the past, when applications had to be written for specific hardware devices or they had to support many device drivers to achieve some degree of portability.

There is an increasing number of software development tools the system designer can use for designing applications and user interfaces and for adapting existing applications to produce acceptable graphics output. Since the formal adoption of a GKS standard by the American National Standards Institute (ANSI) and by the International Standards Organization (ISO), a number of versions of GKS have been developed for both Unix and MicroSoft-Disk Operating System (MS-DOS) environments and for various languages, mainly C and FORTRAN. Standard graphics packages already adapted to run under different operating systems make it easier for a system developer to integrate graphics standards with system application software.

There will always be a need for device drivers that interface the application software directly to the hardware since slow system performance may result if the software must conform to all aspects of a particular interface protocol. There are development efforts underway which would make it easier for a software developer to create and support device drivers. One approach used is to create a text file containing user-readable device protocol commands. This file is then converted into a control module which can be accessed by the application software at run time. The user can enter almost any graphics device description into the text file to be converted to a device driver later.

Some widely used de facto standards which do not implement official standards are Plot 10 from Tektronix and Megatek's Template product. These were written to the proposed Siggraph Core standard which was later dropped by the ISO in favor of GKS. Core implementations such as these are still widely used, have a large number of applications and device drivers available and are relatively portable. Core implementations also have 3D functionality which GKS is only beginning to support.

To help software developers address some specific applications areas, some software vendors are offering extensions to existing basic subroutine libraries. These extended libraries are similar to one of the graphics standards such as GKS or PHIGS, but are oriented towards a specific operating system environment with several device drivers available to support a range of displays, hard-copy and input devices.

Other methods exist which link a graphics database to the graphics display. Standards such as GKS and PHIGS, among others, use a display list. PHIGS adds a hierarchical data structure to this display list so that objects defined by the drawing primitives can be easily duplicated and edited and can be more easily linked with a hierarchical database in the host than that which is possible with GKS.

Another development and application tool which is gaining greater acceptance is window environments. Besides

being graphics elements themselves, window characteristics must be taken into account when dealing with system architecture and operating system design and user ergonomics. In addition to fitting windows into a system environment, some convenient means is needed to interface application graphics to the windows.

Although an ANSI committee, designated X3H3.6, has been formed and is discussing the issues of window environments, two ad hoc standards are emerging, one for MS-DOS and another for Unix. These ad hoc standards include a library which would provide a tool-kit for user interface components such as text, panels, menus and buttons so that a user could program a windows application by simply assembling such components from the library. Text and graphics editing facilities specifically oriented for windows applications are also needed to design special tools, add components to the tool-kit library and make modifications to specific elements chosen from the library.

One goal of the ANSI window standards committee is to define a windowing approach which can span operating system environments. Such a windowing system would concentrate less on graphics and more on display management. Thus, the window standard would define a database of users and hardware. Graphics applications will interface with such a windows environment most likely through a user interface management system (UIMS), which will be device-specific and

capable of supporting a lower level graphics protocol such as CGI. Any graphics user application that can communicate with the UIMS via a standard protocol could therefore interact with the windowing system via UIMS, which would also do the actual drawing of panels, text and graphics.

3.6 Conclusion

Graphics system software design must be performed in parallel with the design of the hardware architecture. The design of the user interface is of considerable importance and influences the usability and acceptability of the system. The software must support device-independence to allow the portability of applications software and data used by the system. The designer can either customize the software, construct the software around a graphics standard or a combination of these two. Special applications where algorithms must be fine-tuned to the hardware require custom software. A combination of these approaches might be the optimum solution enabling the designer to exploit the hardware resources of the system while maintaining software compatibility with the rest of the industry.

At present there are two high-level or application-level graphics standards of importance, GKS and PHIGS. GKS does not support 3D graphics, but a GKS-3D standard is being considered. However, PHIGS supports 3D

graphics. There are proposals to make PHIGS more adaptive to the GKS standard. At the lower-level or device-level, the graphics standards of importance are the CGI, CGM and the NAPLPS. In the near future a major standardization effort especially for 3D graphics is expected due to the ongoing efforts of various international graphics committees. Presently, various graphics manufacturers have introduced their own version of the standards, which are likely to undergo a major change after the 3D standards are approved. Many graphics device manufacturers have started to integrate many of the lower-level standards functions into the chips. There appears to be a general consensus in the industry on the functions to be integrated in the graphics chips but there are no established standards. Various software tools for different graphics standards are currently available, which a designer can use to develop software. Graphics software will become more standardized upon the approval of some of the proposed graphics standards in the near future, making it easier for the system designer.

CHAPTER IV

OTHER DESIGN CONSIDERATIONS

This chapter discusses various other design considerations involved in the design of an interactive computer graphics system. The CRT display is predominantly the most important device used in a graphics system to display the images generated by the system. Therefore, it is important for the designer to specify the proper display characteristics which match the system's performance. This chapter summarizes the important considerations for CRT displays.

Human factors and ergonomics have become a very important issue in the computer graphics industry. Several workers unions around the world have raised concerns about potential health hazards a person is exposed to while working in front of computer displays over extended periods of time. A section is devoted to these issues.

The design of a product does not end at the prototype level. To produce a marketable product, the designer must be aware of the rules and regulations of the various safety test houses in the United States and other countries. Many of these requirements are mandatory and affect the design

considerably. An example of this is compliance to Federal Communications Commission (FCC) regulations regarding the electromagnetic radiation levels allowed for a particular product. This can be a difficult problem for graphics display devices where the video rates can be as high as 100 MHz or higher and special shielding, component layout and interfacing considerations need to be followed in designing the hardware. The issues pertaining to these safety regulations are discussed.

4.0 Characteristics of Color CRT Displays

The color CRT raster display is the most widely used output device for computer graphics systems. Various other display technologies such as vector displays, direct-view storage tubes (DVSTs), liquid crystal displays (LCDs) and plasma displays are also used for graphics display, but recent advances in CRT technology for both monochrome and color cathode ray tubes, along with improvements in drive circuits and the development of new phosphors, have made the raster display the most widely used output device. The CRT offers low cost, high brightness, high speed, many choices of phosphors and many sizes and configurations.

In a graphics system, the image to be displayed is stored in the frame buffer. A color look-up table is used to process information in the frame buffer and the bit planes of the frame buffer are transformed into equivalent

red, green and blue color information in the image. This binary information is then converted into an analog red, green and blue (RGB) signal by a digital-to-analog (DAC) converter and provided as input to the color CRT display.

The color display basically consists of the cathode ray tube, the vertical and horizontal drive circuits, and the video amplifier circuits. The video amplifier along with the horizontal and vertical drive circuits control the red, green and blue electron guns of the cathode ray tube. In the conventional color CRT the RGB electron guns are arranged in either a delta or in-line pattern. A beam-forming region followed by a main magnetic lens focuses all three beams so that they pass through the center of the lens. The beams then pass through a screen known as the shadow mask. The shadow mask consists of a mesh-like screen with round or rectangular holes. The electron beams pass through the shadow mask in order to be sorted before striking the phosphor screen to produce the image on the screen.

Selecting and specifying the proper CRT display which complements the graphics system's performance involve a large number of considerations. Most design criteria take into account both user interface factors and hardware limitations of the graphics system.

4.0.1 Size

The size for a CRT display is usually specified in one of two ways. One way is to specify the size of the diagonal length of the CRT, usually in inches. Another way is to specify the diagonal length of the usable area on the faceplate of the CRT. This is expressed in inches visual (V). The usable area on the screen is the area of the glass of the CRT that is coated with phosphor and can be illuminated.

4.0.2 Resolution

Resolution is specified as the size of a matrix of pixels that can be displayed distinctly on the screen. Resolution depends on the bandwidth of the CRT and the drive circuitry. The characteristics of the frame buffer DAC and the internal drive circuits within the display along with the CRT determine the resolution capabilities of the display.

Higher resolution CRTs are continually being developed and will reach a point where improvements will be difficult to notice. As an example, for a 19-inch CRT, 2000 x 1500 pixels is approximately the highest discernible resolution. In high-end graphics applications, for example gate array design in CAD applications, larger resolutions are needed, thereby requiring larger CRT sizes. Current state-of-the-art resolution is typically 1024 x 1024 pixels

with a 60 Hz non-interlaced refresh rate. For monochrome monitors, resolutions greater than 2000 lines are available.

4.0.3 Phosphor

The phosphors used in the CRT affect the color and flicker characteristics of the display. The three criteria used in choosing a phosphor type are luminous efficiency, decay time and color. The decay time of the phosphor is characterized by the persistence of the phosphor. The persistence of phosphors can be classified into three categories: low-, medium-, and high-persistence phosphors. Medium-speed phosphors are generally preferred since they minimize both flicker and smearing. Various phosphors have different color characteristics and the choice of the phosphor can be greatly influenced by the application.

4.0.4 CRT Faceplate

The CRT faceplate is the glass part of the CRT commonly referred to as the screen. Reflections and glare can hurt contrast, while the shape of the tube's faceplate may affect legibility. These shortcomings are especially noticeable in faceplates which are roughly spherical with rounded corners. Some manufacturers, however, have developed square faceplates that are considerably better in corner screen legibility, reduction in reflection of overhead light, and minimal visual pattern distortion. While still roughly

spherical, these faceplates are flatter than those of conventional tubes and their corners are squared off. Unlike spherical faceplates that present a curved surface from every viewing angle, some manufacturers use a cylindrical faceplate to reduce overhead reflections and minimize distortion.

4.0.5 Bandwidth

The display must be able to support the video bandwidth required by the graphics processor. The main factor limiting high performance displays is the driving circuitry which must possess sufficient bandwidth. Circuits capable of video bandwidths of greater than 200 MHz are presently being developed using high speed CMOS, ECL, and GaAs technologies. In addition, fiber-optic links between the processor and display are desirable as compared to coaxial type video cables in order to support fast rise times of the video signals with low loss in signal levels. The fiber-optic link also alleviates shielding and electromagnetic shielding (EMI) problems, which become more severe in high-resolution displays.

4.0.6 Refresh Rate

The refresh rate is the rate at which the image is updated on the screen. Refresh rates of 60 Hz to 75 Hz and

higher are desirable to achieve flicker free images on the screen.

4.0.7 Horizontal and Vertical Synchronization

The horizontal and vertical synchronization rates for the display must be matched to the graphics system. These timings can be programmable from the system end to allow different displays to be attached to the system.

4.0.8 Shadow Mask

The shadow mask in the cathode ray tube is a perforated metal mask. The holes in the metal mask form equilateral triangles resulting in an interlaced pattern with offset holes in both directions. The three electron beams converge on single aperture holes of the metal mask. Each beam impinges on one of three colored phosphor dots (also arranged in a delta configuration) and is shadowed from the other two. Because less than 20% of the electron beam reaches the phosphor, high beam currents are required. Higher resolution masks require higher beam currents to maintain luminescence levels which, in turn, make the mask vulnerable to warping due to high temperatures.

Doming occurs due to the misalignment of the shadow mask apertures and the phosphor dots. This is caused by heating of the shadow mask which becomes critical as resolution goes up. Solutions to this problem include using

prestressed laminates of perforated steel and ceramic or using low temperature-coefficient materials such as invar.

4.0.9 Misconvergence

To display any color other than primary red, green, and blue, at least two electron guns must be on simultaneously. For displaying white, all three guns must be superimposed on the screen. If all three color guns are not co-linear, there is a loss of resolution (blurring) and a color-fringed dot will appear instead of a single white pixel. The problem is most noticeable in CAD/CAM and mapping applications. In especially misconverged displays, two parallel red and green lines may mix and become one yellow line.

One way to compensate for misconvergence is to use a digital dynamic convergence (DDC) technique, which employs microprocessor controlled DACs to adjust a special yoke. Users can then adjust convergence information and store it in EEPROM. Another method is an autoconvergence system which has a photomultiplier inside the CRT. As the electron beams scan the screen, phosphor patterns on the backside of the shadow mask emit light back into the tube. Through a window in the CRT, the photodetector senses this light and outputs the information to a microprocessor. The phosphor patterns on the shadow mask are deposited in locations corresponding to major convergence points on the screen.

The processor adjusts the beam positions at every major point, enabling the display to maintain its convergence specification. Each phosphor pattern is shaped so that the photodetector will output two pulses. The photodetector outputs a pulse when the beam first scans the vertical line and again when the beam scans the diagonal line. From the time difference between the two pulses, the processor can determine the relative positions of the electron beams. Misconvergence of 0.2 mm is barely noticeable, but levels over 0.5 mm become apparent in any monitor and may be unacceptable.

4.0.10 Auto-Tracking

It is desirable for displays to be capable of automatically tracking the horizontal and vertical synchronization signals. This allows the display to track incoming signals and determine the correct synchronization rates. For example, some monitors set for 47.7 kHz may have difficulty operating at 48 kHz unless auto-tracking is used to compensate for the mismatch in synchronization rates provided by the processor.

4.1 Human Factors and Ergonomic Considerations

This section discusses the human factors and ergonomic considerations for a graphics system. It is important that

these factors are considered during the design of the graphics system and also in the selection of graphics devices.

4.1.1 Radiation Hazards

There has been considerable concern on the radiation emitted from CRT display devices due to fact that certain type of emissions are harmful to humans and can result in permanent damage. Workers unions and governmental agencies are advocating tighter specifications on the radiation level being emitted from CRT display devices.

The level of X-ray radiation emission from CRT displays is of concern since small doses can be harmful to humans. The U.S. Department of Health and Human Services (D.H.H.S.) and other international safety and health organizations have established strict standards to which devices using a CRT must conform. All CRT display devices whose cathode voltage on the cathode ray tube exceed a certain level are required to implement X-ray protection circuitry to ensure that no X-rays will be emitted in case the cathode voltage regulation fails. This failure occurs when the CRT cathode voltage increases beyond the rating of the CRT resulting in the emission of X-rays.

Emissions of very low frequency (VLF) magnetic fields have recently gained attention. Swedish standards agencies are in the process of defining allowable levels for CRT

display devices. Agencies in the U.S. are also in the process of formulating similar standards. Approval of such standards will require display devices to provide special shielding measures to contain these emissions which could possibly pose a health risk.

4.1.2 Flicker

Flicker in a CRT display can be observed when the screen display operates at a refresh rate too slow to allow the eye to fuse successive fields into an apparently continuous set of images. Flicker also depends on the type of phosphor used in the CRT. Phosphors of medium to high persistence exhibit less flicker than low persistence phosphors.

4.1.3 Character Fonts and Sizes

The fonts of the character displayed on the screen are dependent on the processor driving the display. Various standards organizations require certain requirements of the characters displayed on the screen. The box size of the character, for example a 9 x 14 matrix, the thickness of the stroke on the screen, and the height of the characters are some of the requirements. A product to be marketed in Germany could be disqualified by the German test houses if certain font characteristics do not conform to these standards.

4.1.4 Contrast Ratio

The contrast ratio is the ratio of luminescence of the character to that of the background, or vice versa. To distinguish characters on a background, contrast of a certain degree is required. On a reverse video (positive) CRT display, this contrast is obtained by illuminating the background more than the characters; on a normal video (negative) display, the characters are illuminated more than the background. The contrast ratio of a display device is a very important characteristic contributing to the legibility of a display. A good figure for this ratio is in the range of 6:1 to 10:1.

4.1.5 Glare

Glare on CRT displays is produced due to reflections off the surface of the display screen. Various standards exist on the extent to which glare is allowed. In low- and medium-end graphics displays the CRT surface is treated with a combination of mechanical and chemical etching. This causes the surface to become rough and diffuse the reflected light in all directions, thereby reducing glare. In higher-end graphics displays anti-glare filters are installed on the faceplate of the CRT.

4.1.6 Labeling and Placement of Controls

There are international requirements on labeling and placement of controls on various electronic products. Certain controls are represented by symbols approved by international committees. Symbols representing hazards are required to be affixed in appropriate places on a device. For example, the auxiliary power outlets on a system must be labeled to caution the user of the potential danger due to the presence of high voltage. Color coding of controls is also important. For example, in West Germany the use of a red colored switch indicates an emergency power switch and cannot be used for any other controls. The color, placement and labeling of controls are also dictated by the ergonomics involved.

4.1.7 Mechanical Features and Requirements

Ergonomic and human factors require that certain mechanical features be present in certain products. For example, the weight of a device or product decides its portability. Graphics displays are required by certain countries, West Germany for instance, to include the tilt and swivel features. These features are also being incorporated by U.S. manufacturers.

4.2 Standards and Agency Approvals

All products in their final and marketable form are required to meet certain legal and regulatory standards. These standards have been set by the U.S. government, consumer and technical organizations and various federal and private agencies. Different countries have comparable standards and agencies. The designer of the graphics system must be familiar with the requirements of these standards to incorporate them into the design of the system if the product is to be successfully marketed.

All graphics systems require agency approvals from Underwriters Laboratory (UL). This is a safety related approval. Design considerations such as insulation requirements for a power transformer in a power supply, thermal ratings of components, safety labeling requirements and other aspects defined in the standard have to considered. In addition, all graphics products must comply with Federal Communications Commission rules. These rules are intended mainly to make certain that the product does not radiate electromagnetic emissions which would interfere with other equipment in nearby surroundings. This is a very critical requirement for graphics systems since the video rates for graphics devices can be 100 MHz and higher. To meet these rules the designer has to incorporate into the system design good shielding and layout considerations. The

products to be marketed in other countries require different approvals. For example, Canada requires the product be approved by the Canadian Standards Authority (CSA).

It is important that the designer take into account these considerations at the start of the design cycle rather than make major modifications to the hardware in order to develop a final, marketable product.

4.3 Conclusion

There is a large variety of graphics input and output graphics devices available. Proper selection of these is influenced by the architecture of the graphics system and the particular application, bus standards and graphics standards followed. The CRT raster graphics display is the most widely used graphics output device. The CRT technology is relatively old but recent technological advances in the construction of the CRT and development of new phosphors have made the CRT raster display difficult to replace. Various other technologies include plasma displays and liquid crystal displays. CRT devices will be the primary graphics display device for a considerable time in the future. There are no standards in the graphics industry for resolution, synchronization rates or voltage levels and therefore it is important for the designer to make certain

that the characteristics of the graphics processor match the display.

The design process of a graphics system goes beyond the definition and design of the hardware and software of the system. A marketable graphics product is expected to meet certain other requirements such as human factors and ergonomic requirements, and compliance with various safety and agency standards. These considerations have a direct effect on the design details of the graphics system.

CHAPTER V

TRENDS IN COMPUTER GRAPHICS

Interactive computer graphics is one of the fastest growing disciplines influencing every aspect of human life. The image-making capability of the computer has reached a sophistication that allows a person with little or no knowledge of computing to use an electronic system for manipulating pictorial data. New heights of realism have been achieved using advanced computers and techniques of computer imaging. These developments have given way to many new applications and have changed several others in the ways in which they were performed earlier.

This chapter reviews some of these developments. Three-dimensional graphics is a rapidly growing area of computer graphics with applications in solid modeling, film making, advertising and many other aspects area of everyday life. An overview of three-dimensional graphics is presented. Various modeling methods of three-dimensional objects are discussed. This chapter also discusses some of the present and future applications of computer graphics.

5.0 Modeling Techniques for 2D and 3D Objects

The final image produced on the screen of a graphics system is always two-dimensional. Three-dimensional graphics refers to the underlying model used to generate a picture, not the picture itself. Although it is possible to generate a picture without a model at all, it would be very difficult to alter the picture in any way if it is not quite what is desired. The graphics system must always hold some representation of the picture, even if there is no actual model. At the most fundamental level a vector display must have a list of vectors to be drawn; a raster display needs a bit map. It may be possible to perform some operations on these directly but most operations will not be possible. It is the ability to alter a model after it has been created, together with the possibility of generating many different views from a single model which account for the importance attached to modeling techniques.

Models may be divided into two basic types: two-dimensional models and three-dimensional models. It is also possible to have models with dimension greater than three, but the final image will always be a projection of the model onto a two-dimensional viewing screen.

5.0.1 Two-Dimensional Models

The computer model of a 2D object consists of a list of vertices to be joined to form line segments. A curve would be approximated by a series of short straight lines. Thus for each line segment, the data which needs to be stored in memory would be a list of the coordinates of the points along the line segment. If the image of the object is to be shaded it is necessary to define polygons to be filled in a certain color. This could take the form of a list of vertices or line segments to be joined, thus defining the perimeter of the polygon, together with a color value. Computer models may also have some hierarchy in order to save space or facilitate manipulation of the image.

Although a curve could be approximated by a series of short straight lines, a large number of line segments would be required to make the curve appear smooth. This would use a large amount of memory. The degree of detail required would depend on the maximum resolution at which the image is to be displayed.

A way of displaying detail greater than that stored in the model is a method called stochastic interpolation. This is a method whereby a complex object is approximated by a crude polygon. For example, the coastline of Florida could possibly be specified by 20 or 30 line segments in the model. When the object is to be displayed on a graphics terminal, extra line segments are generated in a

pseudo-random manner to produce the effects of inlets and bays. Each line segment in the original model is divided in half by the addition of an extra point at its center. This point is then displaced by a randomly determined distance. Thus, the original straight line will form a shallow V-shape. These new line segments are repeatedly subdivided in a similar manner until the maximum line length is less than some predetermined value, at which point the process is stopped. In this way an irregular coastline may be generated from a few input data points. It will not be an accurate map of the coastline of Florida, but will be much more recognizable than a picture formed by joining the original points by straight lines. This technique is based on a process called fractional Brownian motion and the curves are referred to a fractal curves. They are equally jagged at any resolution and therefore have a fractional dimension.

5.0.2 Three-Dimensional Models

Most computer models are based on points, lines and polygons, referred to as graphics primitives. Lines are drawn on a screen by changing the color of pixels, whose centers lie within the body of the line and leaving all the other pixels in their original color. There are many fast methods for performing this process. They mostly involve

starting at one end of the line and tracing the path of the line across the screen to the other end.

The method for drawing a polygon is essentially the same except that pixels inside the polygon boundary are colored. In this case, however, the problem is complicated by the fact that a polygon boundary may cross itself, so that it is difficult to determine whether a given pixel lies within the polygon or not. Some polygon filling methods look at each row of pixels, or scanline, one at a time and consider how they should be colored. The process starts at the leftmost pixel of the scanline, which is assumed to be outside the polygon, and the scanline is followed until a polygon boundary is crossed. Once a boundary has been crossed the pixels are colored until a second boundary is crossed. This process continues until the rightmost pixel is reached, at which point the process starts again on the next line down. If a count is kept of the number of times the boundary has been crossed we can see that the pixels are colored if the count is odd and they are ignored if the count is even.

This method, however, does not handle problems with polygons which have horizontal edges and at the points where the boundary crosses itself, but these can be overcome easily. One method takes advantage of the coherence of the raster picture by assuming that the color for one pixel on a scanline is probably similar to the color for the next pixel

on the scanline. The assumption is wrong when the scanline crosses a polygon boundary, but this happens only a few times on each scanline, even for fairly complicated polygons.

By using these drawing primitives, a simple model can be transformed into an image by drawing raster lines for points which are joined in the model to form a wire-frame model, and filling raster areas where there are polygons defined in the model to produce a solid image. This will produce reasonable pictures for two-dimensional models but will be unacceptable for three-dimensional models where some degree of realism is required.

Three-dimensional image display involves the translation of image coordinates into world coordinates which represent the digital coordinate system used by the graphics system. In high-end systems, world space consists of three axes defined in terms of 32-bit integer or floating-point numbers. All objects to be displayed are mapped into this coordinate space. World coordinates must then be mapped into two-dimensional screen coordinates. If the view of the object is defined such that parts of it would extend beyond the edges of the screen or a plane defined in the z-direction, it must be clipped.

5.0.2.1 Wire-Frame Model. As is the case in two dimensions, a 3D model would consist of points and lines.

However, when an object modeled in this way is drawn, there is no indication of solidity--the image produced will be a wire-frame figure. Since lines have negligible thickness, no other lines will be obscured.

5.0.2.2 Polygon Mesh Model. To avoid the problem in a wire-frame model where every line is visible, a shape's surface must also be modeled, since surfaces hide everything behind them. This suggests the idea of adding facet or polygon information to the point and line model. Models defined in this way are called polygon meshes.

There are algorithms which can be used which avoid displaying surfaces and lines that are hidden from the viewer. This gives the impression of a solid three-dimensional object. This method is useful for describing any object with flat surfaces but a problem arises when the object to be modeled contains curved surfaces. In much the same way that curves may be approximated by a series of short lines in two dimensions, three-dimensional curved surfaces may be approximated by a large number of flat polygons. The disadvantage of this technique is that the representation is only approximate. The resulting error can be made arbitrarily small by using more polygons, but this uses more memory and also increases the amount of processing time necessary to manipulate or display the model.

5.0.2.3 Curve and Surface Specification Methods.

Methods have been developed which define curved surfaces in a way similar to parametric cubic curves. A parametric cubic curve is one for which x, y, and z are each represented by third-order polynomials. Parametric bicubic patches define the coordinates of points on a curved surface by using three simultaneous equations, one for each x, y, and z. Each equation has two variables and terms for all powers of the parameters up to the cube (three dimensions) are used, hence the name bicubic. The boundaries of the patch are parametric cubic curves themselves. In general, these patches can be larger than the polygons which would be required to define a similar curved surface.

A few control points can be stored in the model (in a way similar to cubic curves) in order to define the shape of the patch exactly. When the image is being drawn, extra points may be generated on the surface and used in equations to determine shading, for example. Thus any level of detail can be generated at display time, although only the coordinates of a limited number of points are stored explicitly in the model.

Various methods exist for specifying a curve in two or three dimensions whose shape is determined by a small number of data points. Some of these methods are Overhauser curves, Bezier curves, Hermite curves and B-splines. The

last three of these are examples of parametric cubic curves. They differ in the details of how the curves are specified, but all allow extra points to be generated along the curve if the resolution of the final picture demands it. The decision as to how many line segments should be used does not need to be made until the image is finally displayed on some output device. Instead of joining the points in the model using straight lines, the coordinates of the points may be used in a formula to generate extra points on the curve. This saves memory space since unnecessary coordinates do not need to be stored. The shape of the curve is specified exactly by the control points used in the model. In three dimensions, these ideas generalize to bicubic patches.

Splines are third-order polynomial equations which define complex curves. They are a step above the practice of representing surfaces in terms of polygons, which evolved from the wire-frame representation of solid objects. Polygons only approximate a smooth surface and each one must be defined by the graphics software. Using splines to represent curved surfaces gives a more precise representation that takes the form of an equation and hence less data. Splines provide an approximate data reduction of between 17-to-1 and 78-to-1 over polygon rendering, depending on the complexity of the surface being modeled.

Of the various families of splines that have been defined mathematically, such as cardinal splines, Bessier splines and beta splines, the type most popular in graphics applications is called the nonuniform rational blended-basis, or B-spline. One attractive feature of the B-spline is that the control points that determine the shape of the curve do not lie on the curve, making it easier for the user to see the results of operations performed on the curve.

Splines are highly computation intensive and a great amount of computing power is needed to create the commands sent to the display device. Because there is also very little vector drawing and splines use one or more light sources, it is very pixel-intensive. With new advances in hardware and software technology the required computing power become cost-effective.

5.0.2.4 Procedural Modeling Technique. Another way that a three-dimensional model may be built is by use of specific program procedures, called the procedural modeling technique. In this technique, software is written which defines in general terms how a certain object should be drawn. Certain key parameters are left to be specified at a later stage to produce a particular version of the object. For example, a procedure may be written to produce a model of a window in a building structure; parameters to be

specified later may include the position and size of the window. Large numbers of windows could be produced for a model of an office building, by repeated calls to this procedure. If, at a later stage, the style of all of the windows needed to be changed, it would not be necessary to alter all of the window coordinates in the model manually; a simple change in the procedure as to how windows should be drawn is all that would be required. Procedural modeling is a powerful tool if a feature has to be modified often.

5.0.2.5 Solid Modeling--Constructive Solid Geometry Approach. A slightly different approach to three-dimensional modeling is to work directly with solid shapes. This is called solid modeling. In the techniques described thus far, only the surfaces of three-dimensional objects have been modeled; a solid object would in fact be modeled as a closed hollow shape. Solid modeling techniques, by contrast, may be used to represent the object as an actual solid. Objects may be built using solid primitives such as cubes, spheres, cones and cylinders. This technique is called constructive solid geometry (CSG).

5.1 Modeling of Natural Forms

There are very complex forms found in nature which would take an inordinate amount of time to model by conventional means such as those outlined above. This has

led to another technique of trying to grow models in a naturalistic way but controlled by a limited amount of data specified by the modeling system. An example is the modeling of a range of mountains. To model the mountain surface in sufficient detail would require a large number of polygons to be specified. This technique would be impractical, because of the large number of the large number of polygons required, but under certain circumstances it may not be necessary to model an existing mountain range exactly. Any shape may be sufficient as long as it can be recognized by its overall features. By taking this approach, an overall shape can be specified using a few data points and polygons, and more data may be generated using some random method which can represent data from the physical world reasonably accurately. Objects created in this way are called fractals. Other algorithms which may be used to grow objects starting from a few data points include particle systems and growth algorithms. Particle systems are capable of modeling complex objects as smoke, fire or grass, whereas a growth algorithm is often used to create models of objects such as flowers or spiral shells. These are all complex objects which may generally be described as natural forms.

5.1.1 Fractal Geometry

Fractal geometry is based on the concept that dimensions do not always need to be considered as integers (one, two, or three dimensions), but can be broken into fractions as a means of obtaining functions much closer to the natural world than the Euclidean geometry of point, line and plane.

According to fractal geometry, there are infinite numbers of curves whose dimensions are between one and two. As fractal curves between one and two dimensions show close resemblances to natural forms, fractal surfaces display many more such resemblances. A fractal surface has dimension between two and three. It can look like a mountainside, a cloud or foliage, depending on the dimension.

The most commonly seen fractal surfaces are ones used to represent hills and valleys. Hills are easier to represent with fractals than valleys since valleys are smoother and smoothness is not a quality associated with fractals. A contouring algorithm may be applied to make valley floors sufficiently smooth, instead of having many ravines, or a truncated fractal may be specified, in which an arbitrary zero is set at some imaginary water level and no growth occurs below it. The water itself is then produced by other means.

The simplest way to represent a hill by fractals is to specify three parameters, not including those for

illumination and the viewpoint. One parameter is length. If this parameter is doubled, the altitude of the hills will double. The second parameter involves a choice among a finite number of fractal growth patterns; this parameter is called a seed. The third parameter is dimension. The dimension for a surface will fall between two and three: a dimension closer to two produces relatively smooth hills and a dimension nearer to three produces a range of jagged hills of the same height. To produce a relatively realistic mountainside which is slightly smoothed, a dimension of approximately 2.3 is used.

5.1.2 Polygon Subdivision

One method in which fractal parameters may be used is polygon subdivision. A scene in which all surfaces consist of triangles is a type of model which is often used to represent real-world data which has been acquired automatically. Each triangle can be subdivided into four smaller triangles by connecting the midpoint of its sides. These midpoints can then be displaced by a randomly determined distance to form an irregular surface. Such subdivisions can be continued until a scale is reached in which no triangle has a side exceeding a specified length. The original triangle is now a fractal triangle whose irregular surface consists of many small triangular facets. A quadrilateral can be subdivided in a slightly more complex

way by generating the midpoint of each of the four sides, together with a central point, and displacing these points as before. In this way, a fractal quadrilateral whose surface is composed of many quadrilateral facets is produced.

With just a small amount of fractal data as described above, any number of realistic fractal landscapes can be created. Only color and shading need to be added. Because of the random nature of fractal geometry, these landscapes will be more or less random in appearance. The designer cannot accurately predict the position of a particular feature, but it is possible to set positions for mountain peaks by suitable selection of the initial triangles or quadrilaterals.

5.1.3 Growth Algorithm

Fractal geometry is not the only way that a few mathematical instructions can produce natural forms. Another technique used to create natural forms is a growth algorithm which produces repeated patterns in similar, increasing or decreasing sizes. The growth algorithm, like the fractal techniques it resembles, bypasses the construction of wire-frame models by including dimension in the original parameter specification to the modeling system.

5.1.4 Particle Systems

Tiny particles dots can be generated in various ways to create several different effects. This idea is based on early vector displays, with the electron beam moved such that a line was displayed on the screen or recorded on a photographic film. With particle systems, generated dots recorded as streaks on film, since in normal film-making objects move during the exposure of the picture; the photograph records the path of the movement as well as the shape of the object. With very small dots, individual streaks or blurs merge together and a number of natural forms can be simulated.

Particle systems were first applied to the generation of an explosion and flame effects for the feature film Star Trek II: The Wrath of Khan. The initial explosion of the so-called genesis bomb in this film generated particles which were themselves particle systems. These formed concentric rings on the planet, and the second-level particle systems were generated at various times to simulate the flickering effect of flames. The velocity, radius of generation, circle, ejection angle, size, lifetime and generation rate of particles were varied also.

Most computer-generated images are held in memory as an abstract model, which may be manipulated (positioned, rotated, shaded) by operations at a level above that of the fundamental representation held by the graphics display.

These models may take the form of vertices and lines, polygon meshes or, patches, together with some of the methods which may be used to create the model, such as fractals geometry, constructive solid geometry or polygon subdivision).

5.2 Considerations for Displaying Three-Dimensional Models

There are many methods used to convert a computer model to an image on some form of computer display, but the final image displayed depends upon the hardware available. The initial predominance of vector displays resulted in images based on lines and arcs, whereas more recent advances in raster technology have led to images drawn by filling areas on the screen with one of a wide range of colors. Vector displays can produce only monochrome or simple color pictures which are of limited complexity and fairly poor quality. The discussion in this chapter is mainly concerned with the use of raster displays.

Before displaying a model on the screen, a decision must be made on which parts of the model are to be displayed and whether any overall adjustments are to be made to the

picture as a whole. There are three steps which are used in this process:

(a) By using a process called clipping, a certain area of the image is selected for viewing and only the objects lying within that region are drawn.

(b) The second step involves deciding how much detail should show in the drawing. In this step there is more detail which is removed from the original model to make it easier to display.

(c) The final step involves deciding details about the actual appearance of a particular drawing which are not specified by the model, such as the overall size and the colors to be used.

The three-dimensional model is more difficult to process since it must be converted or transformed to appear on a two-dimensional display. The first step in this transformation process involves determining the volume of space which will be represented by clipping the image against a box. Then the next step involves determining the amount of detail required within this box. The third step is to decide on the position in space from which the model is to be viewed and the degree of perspective which should be in the final image.

5.2.1 Hidden-Line and Hidden-Surface Removal

There are two types of hidden-line and hidden-surface removal, depending on the way the model is drawn. The first method is used for wire-frame drawings and involves removing lines or segments of lines which lie behind visible surfaces. The second method is used where the model is defined as a number of planar surfaces, or facets, defined by polygons, and starts by rejecting all polygons which face away from the viewpoint and are therefore not visible. The remaining polygons are drawn on the display starting at the back of the model, the point furthest from the viewpoint, and working forwards. The result is that distant polygons appear only if they are not over-drawn by nearer obscuring polygons and thus a full three-dimensional effect is produced. There are problems with this method when polygons overlap in awkward ways since this process will involve drawing a large number of extra polygons which are likely to be overwritten by nearer ones.

5.2.2 Shading

The next stage in producing a realistic image is to decide the color to be used for each of the different facets of the model. The simplest technique, called grenade shading, is to color them all the same shade, but this leads to a very plain image in which it is difficult to tell the direction a facet is facing or how it lies relative to

nearby facets. With an actual three-dimensional object, its orientation relative to the light source is determined from the exact shade of the different facets, since facets reflect different amounts of light at different angles. In modeling the 3D object the shade of the facets must be relative to their position the imaginary light source. When drawing a curved object, such as a bottle, there are highlights or specular reflections on the surface where the point light source is strongly reflected. Having only one point light source would give a very sharp high-contrast image, so another factor in shading a facet is the ambient lighting, which causes diffuse reflections.

5.2.2.1 Gouraud Shading. One commonly used method for calculating shading is Gouraud shading. Gouraud shading defines the colors at the vertices of a model's polygons and interpolates the values for the pixels in the x and y positions between them. Interpolation is based on the position and angle of each polygon to one or more light sources. Gouraud shading gives good diffuse shading but may miss surface highlights within a polygon where, for example, one point might catch the reflection of the light source exactly.

5.2.2.2 Phong Shading. Another commonly used method for calculating shading is Phong shading. Phong shading

starts with a uniform color of the surface and calculates the shading for one or more light sources on a pixel basis. This method is slower than Gouraud shading because individual pixel calculation is more involved than interpolation, but it gives a truer representation of surface curves that fall inside polygons.

5.2.3 Illumination

The illumination of the model, which is separate from the model itself and is only decided when the picture is drawn, also gives rise to the problem of shadows that would appear in the real world. The problem of shadowing is relatively simple because it can be calculated in exactly the same way as hidden lines. If a point can be seen from the light source it should be well lit and have specular reflections; if it is hidden from the light source and lies in a shadow, it will show only dimly and will have only diffuse reflections. It is necessary to calculate shadows early in the drawing process, before highlights are added.

More advanced forms of lighting are being developed for specific applications where the restriction to a single point light source is unacceptable. There are modeling systems which can control the light directions, specify the concentration of light from spotlights to floodlights and mask unwanted spillage from all these sources.

5.2.4 Texturing

The methods described thus far are capable of producing an image which shows evidence of lighting in the form of shadows and highlights, but to make the image appear more realistic other methods can be used to provide texture. One method which can be used is to uniformly vary the illumination of the object's surface, which produces a rough texture.

Another method of creating surface detail is the process of texture mapping, in which an arbitrary pattern or picture can be mapped onto the surface of any object. This can actually involve mapping previously computer-generated images onto a surface to create a new image.

The texture mapping process can be used to create different kinds of surfaces from brick walls to patterned bottles, and can also be used with a subtle pattern to give some realism to an otherwise totally flat surface. Texture mapping also improves the quality of an image by giving indications of perspective, since the texture detail is enlarged for objects near the viewer but reduced for objects further away.

5.2.5 Transparency

One difficult area in drawing images is the representation of transparent objects. This requires handling the refraction of light passing through the

transparent object, like a lens, and the attenuation of light from objects behind the transparent object caused by any translucency in the object. The most successful approach is ray tracing which solves most of the problems of hidden-lines, hidden-surfaces and shadows but only at the price of considerable computation time. Ray tracing reproduces the effects of light on reflective surfaces and through translucent, refractive substances, as well as rendering shadows. In ray tracing, the graphics system follows imaginary rays starting at the viewpoint and heading towards the model passing through one of the pixels on the way, and the passage this ray takes through the model will determine the color of that pixel. If the ray hits a solid surface, then a fraction of the ray, derived from the properties of the particular material in question, is reflected off and the rest is absorbed by the material. If this reflected ray heads towards a light source the effect will be to generate a highlight; otherwise, a normal diffuse color will appear--unless the ray strikes another object, in which case the whole process is repeated. The most interesting case occurs when the ray passes through some transparent material. Some of the ray will be reflected, some of it will pass through refracted and the rest will be absorbed. The path of the two new rays is followed as before and the pixel color is derived from a mix of the two rays. The ray may collide with other transparent surfaces

and be split again, so the value for one pixel could be the combination of a number of separate rays. The processing time required for ray tracing does not vary with the number of objects in the scene but it is only dependent on the number of pixels being rendered. This is in contrast to polygon or facet shading, where processing time varies with the number of polygons involved.

5.2.6 Aliasing

One problem which arises in image display stems from the fact that the display is limited to a finite array of pixels and the final picture is really a mosaic of small colored squares. During the computation of the image the model has been sampled only at points which correspond to the center of the pixels, so the detail lying between the points is lost. This causes the problem of aliasing. There are two types of aliasing: spatial and temporal aliasing.

5.2.6.1 Spatial Aliasing. Spatial aliasing involves the loss of information which results when specific points in space are sampled. One obvious effect of this sampling is the staircase effect, which occurs when drawing straight lines or edges at an angle on a display screen which has a finite number of pixels. Spatial aliasing also causes other problems, such as interference effects when thin lines are drawn close together.

5.2.6.2 Temporal Aliasing. The aliasing problem occurs again when individual computer-generated frames are combined to form an animated sequence. The model is sampled at discrete moments in time, leading to temporal aliasing, or aliasing in time, rather than the previous problem of spatial aliasing, or aliasing in space. Besides temporal aliasing between frames in an animated sequence, spatial aliasing within each frame can occur.

5.2.6.3 Anti-Aliasing. The solution to this problem of aliasing is called anti-aliasing. With spatial anti-aliasing, the color for the pixel becomes the average color of a small area of the whole model rather than the color at one particular sampling point. This implies sampling a number of points within each pixel and taking the average of these values. For example, with the ray tracing method of picture construction the rays are actually sent from each corner of a pixel; the final color of the pixel is the average of these four corner values. The sampling which is the cause of the aliasing still remains but the effects are reduced to such an extent that the eye can no longer detect them. The same effect could also be obtained by using smaller pixels in the display, but this is not physically possible with current displays except when recording directly to film.

Anti-aliasing is also used as a solution to temporal aliasing. The solution here is to make each computer frame show the average picture over a short period of time, in much the same way that a film camera shutter is open for a short period of time and captures an average photograph. The camera shutter is open for half the time during camera filming and so the temporal anti-aliasing, or motion blurring, should average the picture over a similar proportion of the total time between frames. All frames such as this must be spatially anti-aliased, so computer generation of animated sequences is a slow and complicated process.

5.3 The Z-Buffer Concept

Shading, illumination, hidden-surface removal and perspective-viewing are all highly computation-intensive techniques which can be implemented more efficiently with the use of z-buffering. These techniques are used to create a three-dimensional image on the screen by processing information in the model which is related to depth, namely the z-coordinate of every pixel.

The z-buffer is a memory that stores the z-coordinate value of each pixel in order to determine its location in the direction perpendicular to the display screen (the z-axis). A graphics system with a 16-bit z-buffer would have 64K possible z-coordinate positions for each pixel.

Each time a pixel is to be written into the frame buffer, its z-value is compared with the pixel already there. If the new pixel has a lower z-value, which indicates that it is closer to the viewer, it is written over the pixel in memory and appears in the image as part of the nearer surface.

The z-buffer value is also needed to calculate perspective in a three-dimensional display. A pixel's z-position in relation to other visible pixels may be used to calculate foreshortening and cause it to be drawn at a slightly different position in two-dimensional screen coordinates to preserve the illusion of depth perspective.

The z-buffer is used to store z-values when calculating translucency. Colors of overlapping surfaces must be averaged not only in terms of relative density, but in relation to the angle of view. For instance, if a thin surface is seen more from the edge of the surface, more z-values will contend for that pixel location. A number of pixels with a given color density value and different z-values trying to occupy the same screen coordinates will result in a pixel whose intensity is the sum of the intensities of the separate pixels.

The z-value is also useful if a curved surface is presented, in which case the visible pixels are at varying distances from the viewer and at different distances and angles from the light source. Here, the x-,y-, and z-values

are used to calculate the angle and intensity with which the light strikes that pixel to determine the resulting color density required for smooth shading.

5.4 Input Methods for 3D Model Information

The input of three-dimensional data into a graphics system is a problem since most input devices such as digitizing tablets or light pens are inherently two-dimensional. This makes it difficult to specify 3D objects by direct methods. Various methods have been developed to speed up the input of three-dimensional information into the model. To produce a complex symmetrical shape it may be possible to draw an outline in two dimensions on a tablet, for example, then rotate this curve in three-dimensional space to produce points on a surface. These points may then be used to define polygons or patches as desired. This can involve considerable saving for shapes such as bottles, since it is easy to draw the outline of a bottle but not so easy to specify points on its surface in three-dimensional space. Shapes produced in this way are called surfaces of revolution.

Another method for 3D data input is the use of swept surfaces. Swept surfaces are created by taking a 2D shape and extending it in a certain direction. For example, a cube can be created by sweeping a square in a vertical

direction. As another example, a curved piece of rope can initially be represented by a curved line. Then the cross-section of the rope is specified at one point on the curve and swept along the curved path.

5.5 Computer Graphics Applications

Applications making use of computer graphics have emerged in almost every aspect of everyday life. Computer graphics applications are extensively used in industry, office and home environments, entertainment, medicine, education, and advertising and the number of applications continues to grow. Some of these applications are described below.

5.5.1 Industrial Applications

Computer graphics forms a prominent and essential component of computer-aided engineering (CAE). Computer-aided engineering refers to the computer-aided design (CAD) and computer-aided manufacture (CAM), testing and repair of engineering products. In addition there are many supporting activities, such as reporting and documenting, and there are also connections to other more traditional computing functions, stock control, management information and forecasting.

CAE methods offer advantages in mass production and quality control by performing repetitive tasks. CAE provides methods of analysis and synthesis. Many manufactured objects have an associated mathematical explanation which can be used in CAE to allow the computer to discover what would happen or, is happening, to the object under stated or applied conditions.

Computer graphics tools assist in speeding up and simplifying experiments. Various design procedures involve some form of the trial-and-error-method, in which the designer changes something and then assesses the change to ascertain if it is better or worse.

Computer graphics is being integrated with applications in image processing, pattern recognition and robotics. The use of more general-purpose robots with more adaptability and sensitivity is being facilitated by advances in artificial intelligence, which allows the computer to understand more about its environment through information received through vision systems, position transducers and tactile sensors. A television camera can deliver to a vision program an overall view of the surroundings for strategic planning of the robot actions, and sensors in the robot end-effector can respond to pressure and orientation for tactical control of the robot actuators. Analyzing the scene picked up by the vision system and finding how the gripper has grasped some object (from the pressure image on

its fingers) are difficult computations that are the subject of much current research. The wider use of robots and the increasing generality of their function depends on advances in this particular area.

5.5.2 Computer Graphics in Film Making and Animation

Computer graphics has started being used extensively in film making. A large number of feature films and series are being made using computer graphics-created animation techniques. Animation is the process of creating dynamic display images with elements that appear to have motion. Animation is also used in simulating real-life scenes in flight, driving, and other vehicle and engineering simulations. With animated images, the intent is to generate images that appear to move smoothly and naturally. Movies and television animate scenes by displaying sequences of display frames with very small differences between them. These minute changes from frame-to-frame blend into smooth motion when many frames are sequentially displayed at rates of over 15 frames per second. Techniques such as data compression are employed and the computers which are used make decisions and update only what has changed rather than the whole scene. Compact representations, high-performance hardware, high-performance software, and selective updating form the basis for computer animation.

Digital Scene Simulation is the trade name of a technique which produces very effective computer-generated film. The Digital Scene Simulation technique can also be used with pictures as complex as 1,000,000 polygons, although the computing power required for such images is tremendous. As an illustration of the magnitude of such computing power required, a 1,000,000 polygon complex image would take six minutes to create on a freon-cooled, $12.5 million Cray X-MP/22 supercomputer. A general-purpose system such as this can be used to calculate the camera movement and object animation as well as the actual image generation. Other less-expensive and slower computers would take significantly longer to create such an image. Even short sequences of film can take years to complete.

5.5.3 Computer Graphics in Fine Art

A large number of images generated by computer graphics techniques have been accepted for exhibition as works of art. Computer art has been compared to photography in the sense that there is one original program, which can produce repetitive images by running the program several times, as is a photographic negative and several prints. When the program incorporates randomness however, the output of one run is very different from another that minor variations of photographic prints are insignificant by comparison.

Many artists have tended to accept the formal tradition of art as the creation of static objects that can be displayed in a gallery or in books. For them the computer is an accessory, like a drawing or painting aid, rather than the central means for both the creation and the display of their work. The continued popularity of abstract art has allowed computer-assisted paintings to be gradually accepted by the art world.

Programming languages which are used among artists include BEFLIX for mosaics and the GRASS and ZGRASS languages. It must be kept in mind, however, that the computer's use as an artistic tool represents a big challenge to artists. It is a difficult tool to master and only those artists who have the ability to program computers can create computer-generated renderings of fine art.

5.5.4 Graphics Design and Advertising

Computer graphics is extensively used in graphics design applications. Computer graphics assists the designer in creating sophisticated graphic layouts. Powerful computer graphics systems are being installed by printing and publishing companies. There is an going fundamental change in the way printed material is designed and reproduced. Electronic retouching is being used to retouch photographs which have scratches or blemishes. The image can be greatly magnified until the individual pixels are

visible. The pixel color can be changed to retouch the digital image of the photograph.

Pagination systems allow the layout artist to assemble a page by displaying a basic layout grid into which both text and images are integrated. Computer graphics systems can turn the entire pre-press operation of editing, typesetting, color separation and retouching into a virtually paperless process.

Computer graphics and animation techniques are being used in advertising, logo design and television commercials. The most frequent application of computer graphics in television is in portraying animated titles and network identification.

5.5.5 Business Applications

Computer graphics applications are widely used in business. There are two types of business graphics: analytical graphics and presentation graphics. Analytical graphics is used in business decision-making tasks. These type of applications are usually connected on-line to a database. Instantaneous access and flexibility are the most important features of decision-support graphics systems. Analytical graphics can be developed on small office-based computer systems with links to a mainframe computer.

Presentation graphics are for much wider usage than analytical graphics. Presentation graphics include data

representation and display in form of charts and graphs. Presentation graphics are much more pictorial than analytical graphics.

5.5.6 Computer Graphics Applications in Science and Simulation

Computer graphics has applications in molecular modeling, cartography and remote sensing. Computer graphics is being used to simulate driving and road conditions in evaluating the performance of automobile design. It is also being used to perform ship navigation simulation and flight simulation. These applications use a combination of the many of the techniques developed in computer graphics such as modeling, simulation, interaction and realism.

5.5.7 Computer Graphics in Medicine

Computer graphics in medicine is still early in its development. From routine X-rays medical imaging has expanded to other areas such as tomography, ultrasound, computed tomography (CT), nuclear imaging, position emission scanning, and more recently magnetic resonance imaging (MRI). Computer graphics has made useful contributions to clinical medicine. For example, accurate views of such complex skeletal structures as facial bones and vertebral components of the spine are critical to making a correct diagnosis and planning subsequent surgery. Surgery can be

better planned, even simulated with interactive graphics, prior to the arrival of a patient in the operating room. Operations are shorter, with more accuracy, and with fewer in-surgery adjustments. High operating-room expenses are therefore reduced. Medical imaging databases provide exchange of medical information among the medical community for educational purposes, diagnostics, and medical research.

5.6 Conclusion

Computer graphics applications have influenced many aspects of everyday life. Computer graphics has emerged as a very powerful means of communication. Future computer graphics systems will use artificial intelligence for the model creation and for effective animation of complex scenes where the computer must understand the physical properties of the objects being drawn.

The large amount of computing power required to create detailed images is not available to most researchers and so much of the work being done in computer graphics is concerned with the use of new and existing technology to create graphics-oriented machines and new types of displays. Considerable research is also being carried out on ways of consolidating the current ideas to generate faster methods of producing the same results.

BIBLIOGRAPHY

Salim S. Abi-Ezzi and Albert J. Bunshaft, "An Implementer's View of PHIGS," IEEE Computer Graphics and Applications, February 1986, pp. 12-23.

Joseph P. Altnether, "High-Speed Memory Takes Fast Chips and Little Delay," Electronic Design, July 10, 1986, pp. 135-140.

Bruce A. Artwick, Microcomputer Displays, Graphics, and Animation, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

John Bennett, "Raster Operations," Byte, November 1985, pp. 187-203.

Jerry Borrell, "Display Functions Migrate to Graphics Devices," Digital Design, May 1984, pp. 66-77.

David Brazier, "CAD/CAM Graphics Today," Manufacturing Systems, April 1986, pp. 40-46.

E.G. Bylander, Electronic Displays, McGraw-Hill, New York, 1979.

S.K. Card, T.P. Moran, and A. Newell, The Psychology of Human-Computer Interaction, Lawrence Earlbaum Associates, Hillsdale, N.J., 1982.

Michael S. Casale and Edward L. Stanton, "An Overview of Analytic Solid Modeling," IEEE Computer Graphics and Applications, February 1985, pp. 45-56.

Horng-Yee Chou, Allen Samuels, and Brian Sackett, "Fortified PC AT Poised for Shaded Solids Modeling," Electronic Design, June 19, 1986, pp. 95-100.

Thomas B. Clarkson III, "VDI Graphics Tools Aid Software Integration," Mini-Micro Systems, July 1985, pp. 109-117.

Terry Costlow, "High-Resolution CRTs Hold Their Lead, Despite the Rise of Compact Plasma Panels," Electronic Design, July 12, 1984, pp. 112-120.

Franklin C. Crow, "A Comparison of Antialising Techniques," IEEE Computer Graphics and Applications, January 1981, pp. 40-48.

Nabil G. Damouny, "VLSI Video-Display Controller Simplifies Video-System Design," EDN, May 30, 1985, pp. 193-200.

David Deane, "Specific Applications Affect Choice of Color-Graphics Systems," EDN, September 20, 1984, pp. 195-198.

Stefan Demetrescu, "Moving Pictures," Byte, November 1985, pp. 207-217.

Casey Dowdell and Anthony Masraff, "Euroboards Propel GKS Into Real World," Digital Design, July 1986, pp. 58-61.

Nick England, "A Graphics System Architecture for Interactive Application-Specific Display Functions," IEEE Computer Graphics and Applications, January 1986, pp. 60-70.

James D. Foley and Andries van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley, Reading, Massachusetts, 1982.

Akira Fujimoto, Christopher G. Perrot, and Kansei Iwata, "A 3-D Graphics Display System with Depth Buffer and Pipeline Processor," IEEE Computer Graphics and Applications, June 1984, pp. 11-23.

Richard Goering, "Standards Converge for CAE/CAD Data and Languages," Computer Design, July 1986, pp. 36-39.

Rafael C. Gonzalez and Paul Wintz, Digital Image Processing, Addison-Wesley, Reading, Massachusetts, 1977.

Phil Gustafson, "Graphics Systems Deliver Upfront Performance," Computer Design, July 15, 1985, pp. 61-65.

Karl Guttag, Jerry Van Aken, and Michael Asal, "Requirements for a VLSI Graphics Processor," IEEE Computer Graphics and Applications, January 1986, pp. 32-47.

Steven Harrington, Computer Graphics: A Programming Approach, McGraw-Hill, New York, 1983.

L.C. Hobbs, "Computer Graphics Display Hardware," IEEE Computer Graphics and Applications, January 1981, pp. 25-38.

Steven H. Leibson, "Graphics-Controller ICs," EDN, February 6, 1986, pp. 104-116.

David Luther and Walter Anderson, "Integrating an Interactive Graphics System," Mini-Micro Systems, July 1982, pp. 175-178.

H.R. Luxenberg and Rudolph L. Kuehn, Display Systems Engineering, McGraw-Hill, New York, 1968.

Gregory MacNicol, "Graphics Workstations Compete for Top Performance," Digital Design, March 1985, pp. 24-28.

Gregory MacNicol, "Graphics Architecture--New ICs Promise Better Performance," Digital Design, May 1985, pp. 53-60.

Gregory MacNicol, "Color Monitor Specmanship: Look Before You Leap," Digital Design, November 1985, pp. 44-48.

Roy E. Meyers, Microcomputer Graphics, Addison-Wesley, Reading, Massachusetts, 1982.

Ian Michaels, "Focus on High-Resolution CRTs," Electronic Design, January 23, 1986, pp. 137-144.

William E. Murray, "Video Display Terminals: Radiation Issues," IEEE Computer Graphics and Applications, April 1984, pp. 41-44.

Ware Myers, "The Ergonomics of Video Display Terminals," IEEE Computer Graphics and Applications, January 1984, pp. 7-15.

Ware Myers, "Staking Out the Graphics Display Pipeline," IEEE Computer Graphics and Applications, July 1984, pp. 60-65.

William M. Newman and Robert F. Sproull, Principles of Interactive Computer Graphics, 2nd ed., McGraw-Hill, New York, 1979.

Randy Nickel, "The IRIS Workstation," IEEE Computer Graphics and Applications, August 1984, pp. 30-34.

Stephan Ohr, "How Silicon ICs are Reshaping the Graphics Picture," Electronic Design, June 26, 1986, pp. 72-80.

Duane M. Palyka, "Computer/Art--Depolarization and Unification," IEEE Computer Graphics and Applications, July 1985, pp. 47-56.

Curtis Panasuk, "Software Standards Will Usher in the Age of Graphics," Electronic Design, July 12, 1984, pp. 94-106.

Chan S. Park, Interactive Microcomputer Graphics, Addison-Wesley, Reading, Massachusetts, 1985.

Robert E. Peterson, Jr., "Flat-Panel Displays," EDN, November 24, 1983, pp. 102-127.

Mike Raghavan and P. Kim Wheeler, "Powerful Support Chips Simplify Graphics Design," Electronic Design, August 21, 1986, pp. 123-128.

Martin Randall and Arun Johary, "Graphics Coprocessor Chip Gives Small Systems a Look at the Big Picture," Electronic Design, May 29, 1986, pp. 159-166.

Martin Randall, "Graphics IC Simplifies Design, Boost Performance," Computer Design, June 1, 1986, pp. 73-77.

Raster Graphics Handbook, Conrac Corporation, Covina, California, 1980.

Michael L. Rhodes, "Computer Graphics in Medicine," IEEE Computer Graphics and Applications, December 1985, p. 11.

David F. Rogers, Procedural Elements for Computer Graphics, McGraw-Hill, New York, 1985.

Daniel R. Ryan, Computer-Aided Graphics and Design, 2nd ed., Marcel Dekker, New York, 1985.

Bruce J. Schachter, "Computer Image Generation for Flight Simulation," IEEE Computer Graphics and Applications, October 1981, pp. 29-64.

Sol Sherr, Electronic Displays, John Wiley and Sons, New York, 1979.

Michael J. Smith, "The Physical, Mental, and Emotional Stress Effects of VDT Work," IEEE Computer Graphics and Applications, April 1984, pp. 23-27.

Bert Speelpenning, "Graphics Terminal Reduces Dependence on External Host," Computer Design, January 15, 1986, pp. 87-91.

Robert F. Sproull, W.R. Sutherland and Michael K. Ullner, Device-Independent Graphics, McGraw-Hill, New York, 1985.

William E. Suydam, Jr., "Graphics Standards Emerge as Firms Strive for Viabilty," Computer Design, December 1985, pp. 157-161.

Robert W. Swezey and Elaine G. Davis, "A Case Study of Human Factors Guidelines in Computer Graphics," IEEE Computer Graphics and Applications, November 1983, pp. 21-30.

Nadia Magnenat-Thalmann, Daniel Thalman, and Mario Fortin, "Miranim: An Extensible Director-Oriented System for the Animation of Realistic Images," IEEE Computer Graphics and Applications, March 1985, pp. 61-72.

Nadia Magnenat-Thalmann and Daniel Thalmann, "Three-Dimensional Computer Animation: More an Evolution Than a Motion Problem," IEEE Computer Graphics and Applications, October 1985, pp. 47-57.

Jack Vaughn, "Solids Modeling: Hardware Grabs the Spotlight," Digital Design, August 1986, pp. 50-53.

Anders Vinberg, "Device-Intelligent Graphics Software," Mini-Micro Systems, July 1982, pp. 227-228.

Clinton N. Waggoner, "GKS-Based Graphics Software Adapts to Changing Technologies," EDN, May 31, 1984, pp. 127-133.

James R. Warner, "Standard Graphics Software for High-Performance Applications," IEEE Computer Graphics and Applications, March 1985, pp. 74-79.

James R. Warner, "Principles of Device-Independent Computer Graphics Software," IEEE Computer Graphics and Applications, October 1981, pp. 85-100.

James R. Warner and Nikolaus J. Kiefhaber, "Implementing Standard Device-Independent Graphics," Mini-Micro Systems, July 1982, pp. 201-208.

Carl Warren, "Graphics Software Brightens Business Look," Mini-Micro Systems, July 1985, pp. 121-129.

Mary C. Whitton, "Memory Design for Raster Graphics Displays," IEEE Computer Graphics and Applications, March 1984. pp. 48-64.

Tom Williams, "Graphics ICs Increase On-Chip Functions and Intelligence," Computer Design, January 15, 1986, pp. 43-48.

Tom Williams, "Super Chips Bring Unprecendented Power to Graphics Display Control," Computer Design, May 1, 1986, pp. 65-82.

Tom Williams, "Graphics Processing Migrates from Host to Workstation," Computer Design, July 15, 1986, pp. 49-57.

Tom Williams, "Hardware Advances Fuel Graphics Standards Controversy," Computer Design, August 15, 1986, pp. 28-32.

Tom Williams, "Graphics Software Rides a Wave of Silicon Advances," Computer Design, October 1, 1986, pp. 38-48.

Andrew Wilson, "Flat-Panel Displays: Still in the Future," Digital Design, December 1985, pp. 54-58.

Andrew C. Wilson, "Graphics Boards Tackle Imaging," Digital Design, June 1986, pp. 84-88.

David Wilson, "Number Crunching Graphics Processor Pushes Pixels," Digital Design, January 1986, pp. 61-64.

Maury Wright, "Color Monitors," EDN, May 31, 1984, pp. 107-122.

Johnson K. Yan, "Advances in Computer-Generated Imagery for Flight Simulation," IEEE Computer Graphics and Applications, August 1985, pp. 37-51.

Matthew M. Zuckerman, "Innovative Display Technologies--Why a Flat Panel When You Can Have a CRT?" IEEE Computer Graphics and Applications, April 1984, pp. 9-15.

VITA

Pramod Chandra was born on July 30, 1957 in New Delhi, India. He attended college at the Indian Institute of Technology, New Delhi, and in April, 1978 he received a Bachelor of Technology degree in Electrical Engineering. He then worked as a Design and Development Engineer for Larsen and Toubro, Ltd., in Bombay, India.

He joined the University of Tennessee, Knoxville in September, 1980 to pursue a Master of Science degree in Electrical Engineering. He served as a graduate teaching assistant from January, 1981 to December, 1982. In June, 1983, he joined International Business Machines Corporation.