A Parallel Accelerator for Generating Virtual Studio Sets

A. V. Sahiner, P. Lefloch, A. D. Nimmo, Y. Paker

ABSTRACT

Recent developments in digital video techniques and the use of computers in video and television production provide new opportunities for programmes. Key technologies which embrace these developments are advanced computer graphics techniques for photorealistic rendering, image analysis and animation. Parallel processing provides the potential for practical utilization of these techniques. We describe current research in Electronic Set Design using high performance graphics workstations and a new parallel processing platform, for generating virtual studio sets.

1.1 Introduction

With the recent developments in digital video media there are many opportunities to use computers in video and television production. Computer generated logos and animated objects are now common in programmes we see every day. Advanced illumination models and rendering techniques, like ray tracing [5] and radiosity [2] permit the economic use of high-quality computer-generated images in the production process. However, the practical utilization of such techniques require high performance computation engines. Parallel processing has made the development of such engines possible. Multiple processor hardware platforms built using high-performance processors such as Intel i860, transputers, and the Motorola MC96002 digital signal processor are already being used for this purpose [3]. The European Community MONALISA project, funded under the RACE II program is developing and integrating software and hardware technologies to enable the use of computer generated sets in telvision studio production and post-production. The use of virtual sets has the potential to dramatically reduce the costs of production and post-production, whilst maintaining and increasing image quality. Current practice requires physical construction and dismantling of sets before and after each programme, which is an expensive operation. A demonstrator for the MONALISA project, called the ELSET (ELectronic SET) system, is currently under development.

The ELSET system, in addition to the generation of high quality graphics and real-time mixing facilities, will aim to provide image analysis based tools for foreground and background seperation, camera tracking and 3D model building.

The ELSET hardware platform comprises a multiprocessor accelerator hosted by a high-performance Unix-based graphics workstation. An X Window System user interface based on the OSF/Motif toolkit provides access to the ELSET system; some of the ELSET tools run on the graphics workstation, while the accelerator is used for real-time camera tracking, real-time mixing, and high performance image processing and image synthesis. Unix provides a convenient front end to the ELSET system bringing in the benefits of open systems. As a result, the ELSET hardware can be readily networked with other systems, and commercially available software can easily be integrated with ELSET functionality.

1.2 The Hardware Accelerator

The hardware accelerator comprises a pool of Motorola MC96002 digital signal processor boards closely coupled with an intelligent frame buffer system, shown in Figure 1.1. Each processor board has a MC96002 processor, 2 extension ports, and 2 banks of static RAM on it. The frame buffer consists of memory boards with a capacity of 128 Mbytes each. The processor pool is organized as a number of VSB Bus based clusters. Each cluster has a gateway processor with high speed access to the frame buffer. FIFO buffered I/O processors are used to connect gateway processors to the frame buffer. There is also a purpose-built mixing device for use in the accelerator.

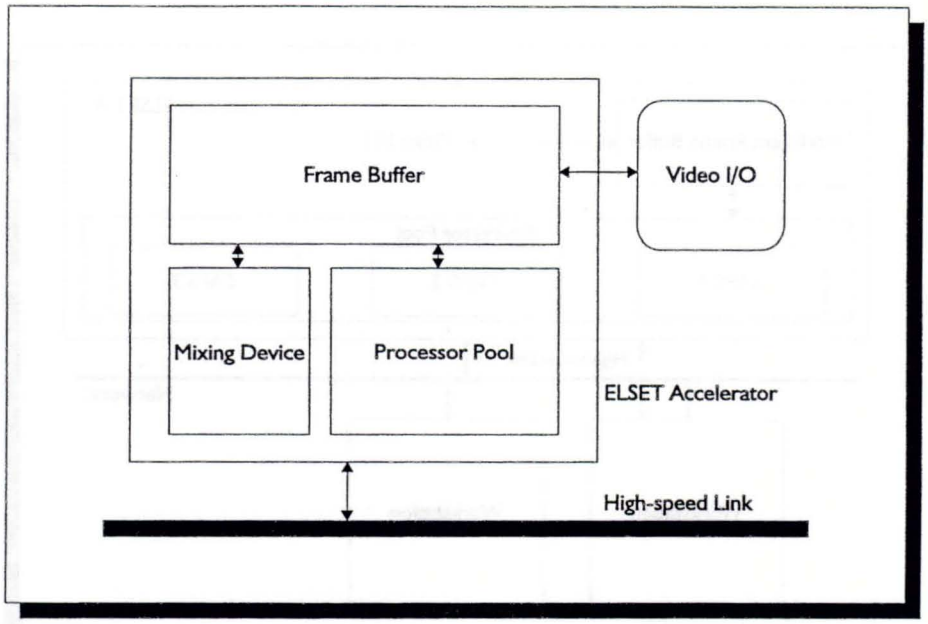

FIGURE 1.1. Hardware accelerator for ELSET system

A VME-based subsystem running on a Motorola MC68040-based processor board controls the frame buffer and the related I/O activities. This subsystem is supported by the real-time OS9 operating system. Control of the peripherals for recording and displaying in real time, data transfer to and from the storage, and interfacing to other computer systems constitute some of the functionality provided by this subsystem. The frame buffer subsystem can manage several independent data streams, input, output and input/output, in real-time, up to a data rate of 400 Mbytes/second.

1.3 The Processor Pool and Compute Server

The utilization of the processor pool as a high-level accelerator is based on the *Self Adapting Parallel Servers* (SAPS) model for parallelism [4]. In this model, the parallelism is encapsulated within compute-server objects. A server, when requested, executes multiple copies of an associated sequential procedure in parallel in Single Program Multiple Data (SPMD) mode. The data is provided by the client within the service request. The inter-

face between the application programs, as clients, and the parallel servers is conveniently hidden in the procedure call mechanism which is a well-understood facility to develop modular programs.

Figure 1.2 illustrates a scenario where the processor pool runs three servers each with different functionality; since the processor pool is closely coupled with an intelligent frame buffer, the servers have high-speed video I/O capabilities. The services provided by the servers are shared by two programs running on separate workstations on the network. There is also an optional high-speed link provided between a workstation and the hardware accelerator. It must be noted that the number of processors used by each server and their internal organization is completely transparent to the application programs running on the workstations. The system software for ELSET provides the framework to create, run and utilize these servers.

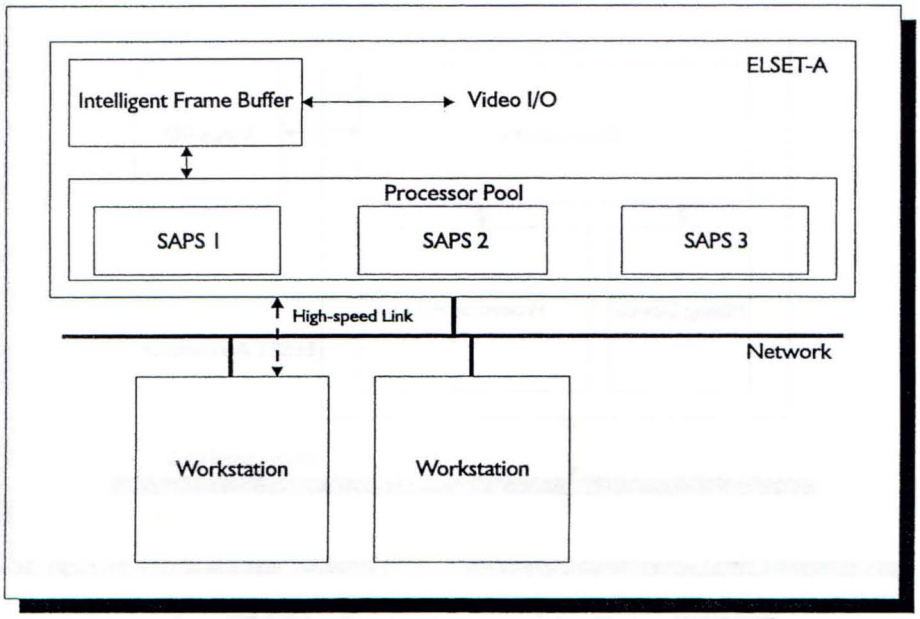

FIGURE 1.2. Compute-server using the SAPS model

1.4 SPMD Parallelism and the SAPS Model

SPMD mode parallelism provides a convenient framework for developing parallel application software using sequential programming techniques. This allows application software already developed for sequential machines to be converted for parallel architectures, without undergoing major changes. Consequently parallel program development is simplified; the bulk of a parallel program can be written in a conventional sequential style.

Under the SAPS Model the creation of server blueprints is template based. A sequential procedure, function $F()$, automatically inherits the properties for SPMD mode parallelism when interfaced to a standard template. Note that parallelism is introduced by running multiple copies of the function, one copy per available processor, on the target multiprocessor, shown in Figure 1.3. Each copy runs with a different fragment of D as its data.

There are five basic operations which are crucial for SPMD mode parallelism:

- decomposition of the application data D into data fragments D_i
- modification of the sequential program, for the insertion of communication primitives
- replication of the function on available processors
- distribution of the data fragments to the running copies of the function
- collection of the partial results R_i and recomposition of the final result R

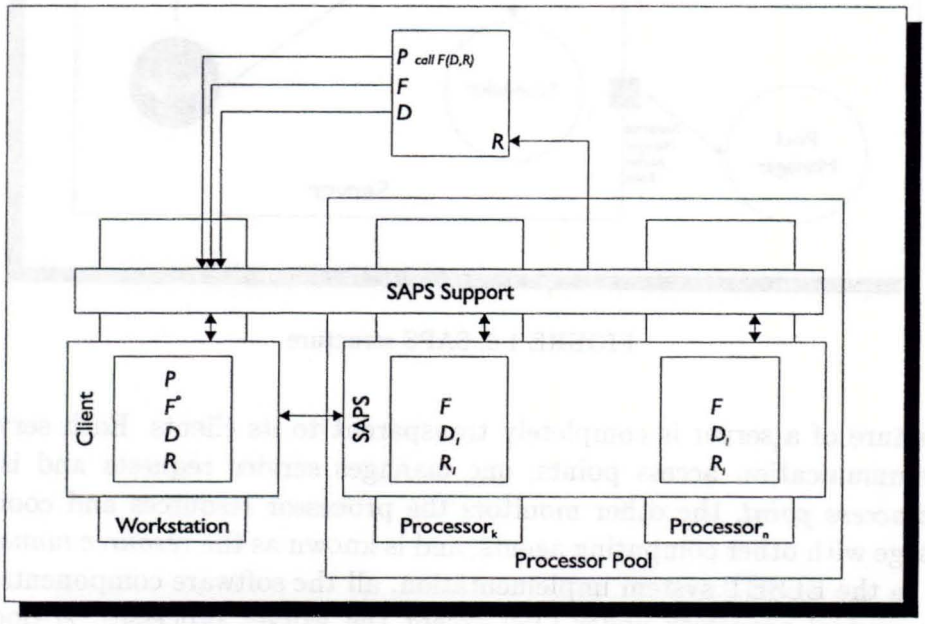

FIGURE 1.3. SAPS model for SPMD parallelism

In the SAPS model these operations are encapsulated within servers. The service provided by each server is supported by a user package to hide the details of service request and service provision interaction and data decomposition. The user package is a library of procedures which can be linked to client programs. The user package permits the service request to look like an ordinary procedure call.

The communication requirements for data distribution to the task force of a server is achieved via the callback mechanism. Callbacks are remote operations carried out by servers on shared data objects. Data handles, which are passed by the client to the server within the service request, are used for this purpose. The shared object layer allows the distribution of client data according to a specified decomposition and enables the collection and recomposition of results.

1.4.1 Server Structure

A server has a multi-process structure, shown in Figure 1.4. A dispatcher process and a pool of workers form a process farm where the dispatcher farms out work to workers and

each worker, when it becomes idle, requests more work from the dispatcher process. It is the multiplicity of the workers that provide parallelism within the server.

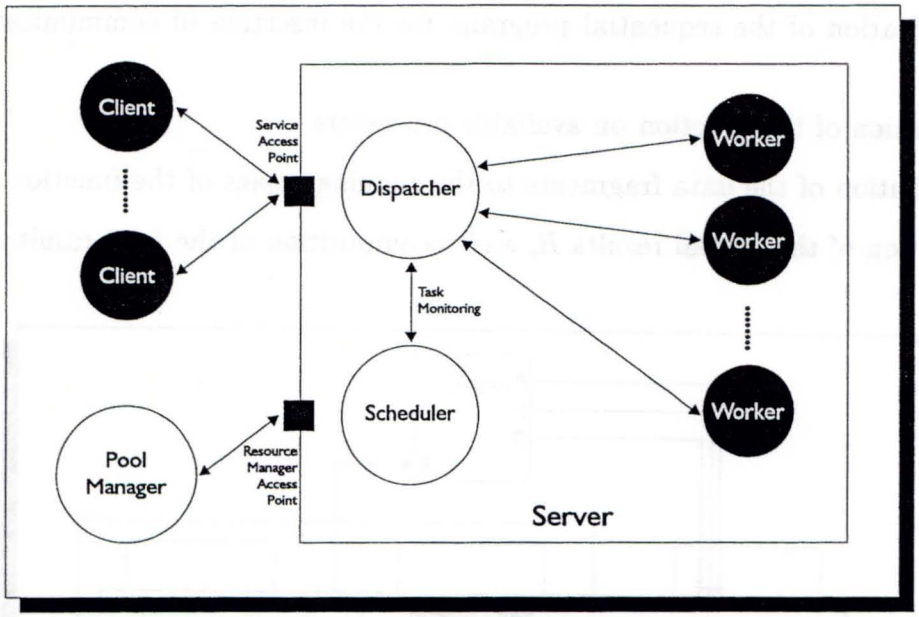

FIGURE 1.4. SAPS structure

The structure of a server is completely transparent to its clients. Each server has two message communication access points; one manages service requests and is known as the *service access point,* the other monitors the processor resources and coordinates its resource usage with other computing agents, and is known as the *resource management access point.* In the ELSET system implementation, all the software components for servers run on the control processor under OS9 except the worker processes. A pool manager and a name server are also implemented for an effective sharing of the pool by multiple servers.

1.4.2 Shared Data Objects

Under the SAPS model, shared data objects are introduced as a means of making the decomposition details transparent to servers. It is one of the important properties of a server, that it can be built using a standard software template. Decomposition transparency is one of the preconditions for this, otherwise different decomposition strategies would require different templates. In addition, the shared data objects serve to meet additional communication protocol requirements between a server and its clients. The RPC protocol, although very flexible, is not powerful enough to serve as a protocol for parallelism by itself.

The shared data object layer provides a flexible and uniform framework for the management of data within the ELSET system. A shared data object encapsulates client data which is accessible by server processes through a set of operations. These operations are invoked only by those processes which have data handles on the respective objects. Data handles are created when a rule, or a set of rules, is registered with an object. The client

then passes these handles to the server in the service request. Server processes can propagate data handles among themselves by message passing. Figure 1.5 illustrates how input and output data handles are used by a server, to access data in the frame buffer, for data distribution and recomposition of results.

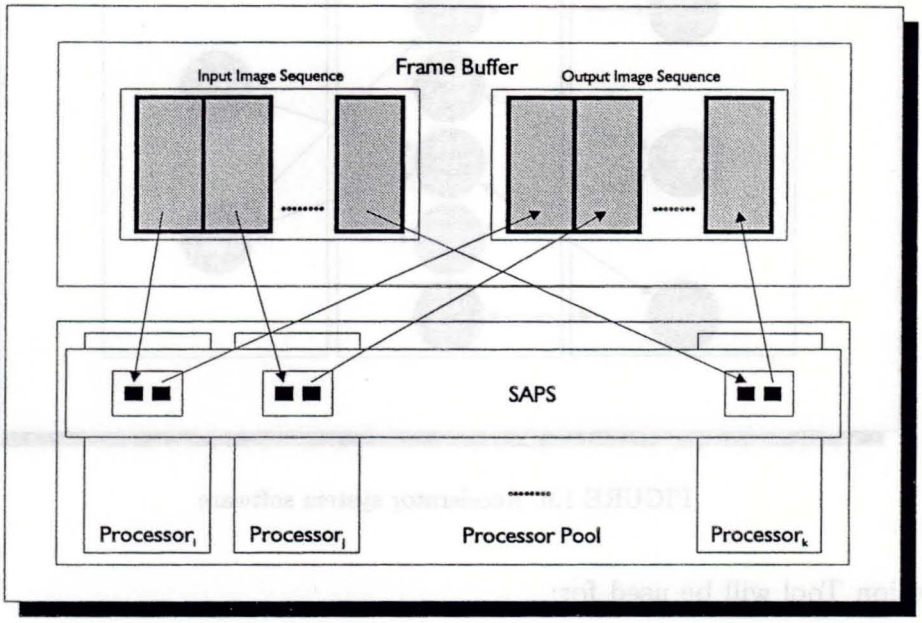

FIGURE 1.5. Data distribution using invocations on data handles

1.5 ELSET System Software

The system software modules for the ELSET system are being developed to run under three separate operating environments. The user interface runs on the Unix-based graphics workstation; processor pool management and server management run under OS9, which acts as the front-end to the intelligent frame buffer system; the parallel server execution environment runs under W-Kernel on the processor pool. W-Kernel is an operating system kernel built specifically for the MONALISA project, to execute on the MC96002-based processor pool. Figure 1.6 shows the main system software modules and their relationships. Interfaces between various software modules are implemented either as Unix/OS9 communications or as OS9/W-Kernel communications. Unix/OS9 communications are programmed using TCP/IP sockets. For OS9/W-Kernel communications, a remote procedure call mechanism is used.

1.5.1 The Graphics Host

The software running on the graphics host consists of three OSF/Motif toolkit-based tools; the Object Browser, the System Adminstration Tool and the Command Tool. The object browser is used to create and register data objects with the ELSET system. This tool will later be extended for data decomposition purposes for parallelism. The System

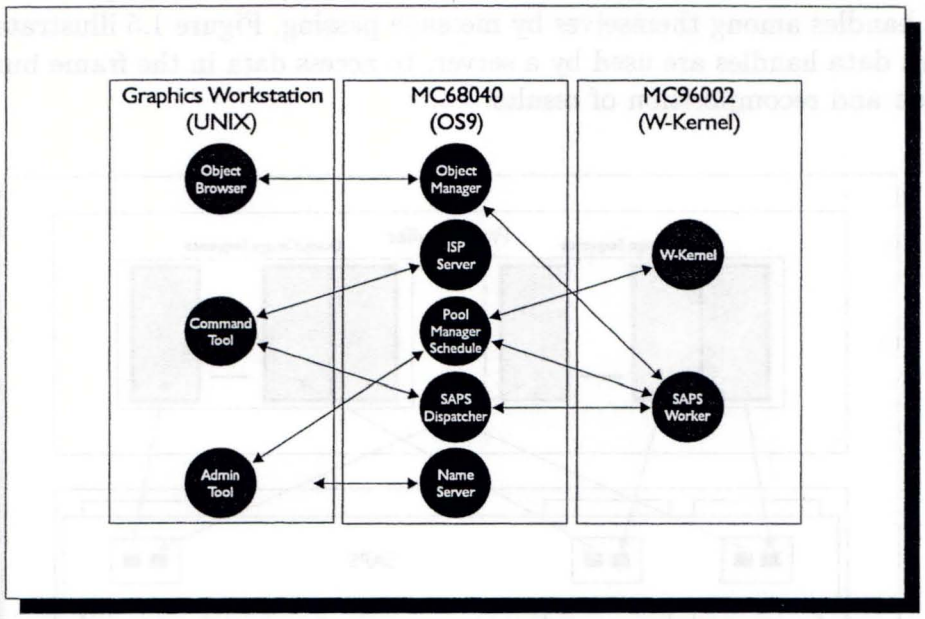

FIGURE 1.6. Accelerator system software

Adminstration Tool will be used for:

- ELSET system initialization
- starting system management servers
- switching between operational modes (real-time and non-real-time)
- starting parallel servers and browsing the server library for this purpose
- monitoring the processor pool
- monitoring OS9 activities

The Command Tool will be used for:

- utilising intelligent frame buffer functionality for video I/O
- assigning application tasks to parallel servers running on the processor pool
- managing real-time usage of the ELSET system for camera tracking and mixing

1.5.2 The OS9-based Host

OS9 [1] provides the host environment for utilization of the intelligent frame buffer, the processor pool, and the mixing subsystem. The system management processes for the ELSET system therefore run in this environment.

The object manager keeps track of all the data objects registered with the ELSET system. The information kept within each object description includes data type and caharcteristics as well as the location of the object, and its decomposition properties. The

data required by the parallel servers during execution are fetched with the aid of the object manager. Object manager has an interface with the Browser running on the graphics host.

The ISP server is the system process that provides the services related to the usage of the ISP system (the intelligent frame buffer system). These services include operations such as reading a video sequence from a video input device, writing a video sequence to a video output device, and accessing the frame buffer in a random frame by frame manner. The command tool running on the graphics host is the front end to the ISP server.

The pool manager is responsible for the initialising the processor pool and includes operations for resetting and booting W-Kernel. It also keeps track of processor loads, processor states and slot states. The resource management of parallel servers is done by the scheduler. It creates the workers for a particular server on a number of processors, and when required it terminates them. The administration tool, running on the graphics host, provides the user interface to the pool manager and the scheduler.

The SAPS dispatcher is the process which carries out the distribution of the task load among the workers of a parallel server. It receives service requests from the applications running on the graphics host and dispatches the tasks accordingly to the workers running on the processor pool. The name server process is used for communication purposes; processes running on the graphics host fetch the port addresses for the OS9 system servers using the name server. It is also used for creating the ELSET system servers running under OS9.

1.5.3 W-Kernel

W-Kernel is an operating system kernel which provides process creation and process management functionality for parallel server execution on the processor pool. It also incorporates a communications layer for ISP to OS9 communications. W-Kernel, as an operating system kernel, provides the basic facilities to run slot-based processes. Since there is no hardware support for memory management, neither board-level nor chip-level, each process is run in a statically mapped memory environment called a slot. A program that will run in a particular slot has to be compiled and linked to be run explicitly in that slot. It is not possible to change dynamically, the target slot for an executable object. This is not a significant limitation for our purposes since we will build the parallel servers to run in predetermined slots.

1.6 User Interface

The user interface for the ELSET accelerator is provided by the X Window System and the OSF/Motif toolkit, incorporating a selection of widgets such as buttons, lists and drawing areas. The system administrator can boot, reset, and reboot processors in the pool by pressing respective buttons and selecting the required command. The status of each processor can be viewed in a window. The servers are started simply by selecting the required server in a browser and marking the *start server* command in the commands list.

Utilization of the ELSET system is achieved with the Command Tool which incorporates a work area where various modules with different functionality can be connected to

perform particular tasks. There are three classes of modules:

- frame buffer modules are used to control video I/O devices and manage frame buffer files
- active SAPS modules are used to utilize parallel servers running on the processor pool
- SGI modules are used to utilize the facilities provided by the graphics workstation

These modules are copied into the work area by simply selecting them from the *module browsers.* For example, using a combination of different modules, a sequence of images can be captured from a camera and stored in a file in the frame buffer; these images can then be processed by a SAPS in parallel and results can be displayed on a monitor and also recorded on a video recorder. The user of an ELSET system deals with the accelerator functionality at a module level and all the mechanisms for parallelism and message communication are completely transparent.

1.7 Conclusion

An accelerator has been built to meet the computational requirements of compute intensive graphics and image processing tasks of a computer-based virtual studio set generation system for television programme production. The accelerator is based on a pool of high performance processors, tightly-coupled with an intelligent frame buffer system. A client-server based computational model for parallelism, the SAPS model, is used to achieve parallelism in a modular fashion using sequential code. The model also supports an object-oriented scheme for data distribution which allows parallel servers running on the processor pool to use the fast video I/O capabilities of the ELSET hardware. The SAPS model provides a framework to transform the ELSET processor pool into a compute-server for computationally intensive graphics and image processing tasks. It enables servers to be built in modular fashion using sequential code and it supports an object oriented scheme for data distribution. Utilisation of servers is similar to the concept of parallel software libraries in the sense that customized software which incorporates parallelism is shared or reused by different application programs. However, here the shared software is not linked to individual application programs but its functionality is provided as an active service by a run-time computing agent. Development work is continuing, for the implementation of radiosity, image analysis and image synthesis techniques, specifically to take advantage of the ELSET hardware and system software.

Acknowledgements:

We wish to acknowledge the contributions of the other full partners in the MONALISA consortium; BBC, Daimler-Benz, DVS GmbH, University of the Balearic Islands, University of Hannover, Seimens, Thomson CSF-LER, VAP GmbH. We also wish to thank cooperating partners Silicon Graphics and TDI for consortium hardware and software provision and support. This work is supported by the European Community RACE II programme, project number R2052.

1.8 References

[1] P. Dibble. *OS9 insights, an advanced programmers guide to OS9/68000.* Microware Systems Corporation, 1988.

[2] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. In *Proceedings of SIGGRAPH '84*, pages 213–222. ACM Press, July 1984.

[3] P. Pitot. The VOXAR project. *IEEE Computer Graphics and Applications*, 13(1):27–33, January 1993.

[4] A. V. Sahiner. *A computational model for parallelism.* PhD thesis, The Polytechnic of Central London, 1991.

[5] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6):343–349, June 1980.