FINAL REPORT OF WORK COMPLETED UNDER NAS8-37139 FOR THE PERIOD

December 24, 1987 through November 15, 1992

SUBMITTED TO:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MARSHALL SPACE FLIGHT CENTER, ALABAMA

SUBMITTED BY:

EARTH SYSTEM SCIENCE LABORATORY
UNIVERSITY OF ALABAMA IN HUNTSVILLE
HUNTSVILLE, ALABAMA 35899

Dr. Richard T. McNider
Michael Botts
Don Moss
Helen Conover
Evan Criswell
Sara Graves
Danny Hardin

(NASA-CR-193860) [DEVELOPMENT OF A NAMING CONVENTION AND DATA MANAGEMENT SYSTEM] Final Report, 24 Dec. 1987 - 15 Nov. 1992 (Alabama Univ.) 107 p

N94-20268

Unclas

G3/82 0201212

Final Report of Work Completed Under NAS8-37139

Prepared by:
Helen Conover
Danny Hardin

1.0 INTRODUCTION

This contract covered work in data management, acquisition, and analysis for the Earth Science and Applications Division (ESAD) of the Space Science Laboratory of Marshall Space Flight Center (MSFC), NASA. Under this contract, a naming convention and data management system were developed for ESAD data sets, data visualization and standard data format issues were investigated, and and specific data analysis and management needs were addressed for several ESAD projects and data sets. In the following sections, details of the research performed over the duration of this contract will be summarized.

2.0 IDIMS

A centerpiece of this research effort has been the development of the Interactive Data Integration and Management System (IDIMS), an end-to-end system for cataloging, archiving, and retrieving scientific data used within ESAD. This task has progressed from an evaluation of data management needs, through recommendation of a standard naming convention for scientific data sets, to development of the IDIMS database and user interface. Data managed under IDIMS includes: data from aircraft sensors, radars and sounders, many different satellite instruments, surface observations, and models and analyses.

The IDIMS user interface is a series of menus which guide the user to specify data attributes. These attributes are used to locate data of interest, or to provide the metadata for a data file being archived. If the case of data retrieval, a query is submitted to an ORACLE relational database management system, and the names of files which matched the search criteria are displayed. At this point, the user may select files to be transferred into the McIDAS system for visualization, or he may save the list of files, either on-line or on paper, to use with his own analysis software, or other tools.

IDIMS was originally developed to maintain an inventory of ESAD data stored on the Engineering Applications and Data System (EADS), an MSFC institutional computer system made up of IBM mainframes, a Cray super-computer, and near line mass storage. The system was later extended to provide inventory capabilities for data kept off-line, as well. Because IDIMS was designed for use on the IBM mainframes, it has a character-based user interface, which is rather primitive by today's standards. Two prototype graphical user interfaces were also developed, for personal computer and UNIX workstation platforms.

2.1 Scientific Data Set Naming Convention

IDIMS was developed to maintain an inventory of ESAD data, and to retrieve data from IBM data files into McIDAS, the principal data visualization and analysis tool in use within ESAD. Standard file names used by the atmospheric scientists indicate data sets to be included in the inventory. Thus, all a scientist must do to make sure his data is cataloged in IDIMS, and easily retrieved into McIDAS, is to name the data according to convention.

A uniform, but flexible, data set naming convention was developed in cooperation with ESAD scientists. This naming convention allows a user to indicate the contents of a scientific data set, while conforming to the constraints of the MVS operating system. The various file name fields indicate data source and sensor, date and time of data collection, project or field campaign which collected the data, and, for McIDAS files, data format. One file name field is generally left for the user to specify. This naming convention was designed for data from many different instruments, and has been extended as necessary to incorporate new projects, sensors, and data sources, as data from these have been acquired. "NASA/MSFC Earth Science and Applications Division Data Set Naming Convention", a document describing the naming convention in detail and listing valid field values, has been submitted with previous quarterly reports for this contract.

2.2 On-line Help

One requirement in the development of a data management system was the provision of on-line help. This has been implemented at several levels, from careful system design to the provision of detailed on-line instructions. Each menu panel is designed to be as clear and easily understood as possible. For example, where the user is required to type dates or times, the required format (eg.'MM-DD-YY') is displayed next to the entry field. Also, error checking is enforced, which does not allow a user to leave a menu field containing an incorrect value; if a user may indicate his menu choice by specifying a letter between 'A' and 'M', the software will not allow him to enter a 'P'. In addition, more extensive context-sensitive help was developed. Each menu panel has an associated help panel, so that when the user presses the "Help" key, a screen will be displayed containing specific instructions or explanations related to his current location within the IDIMS menus. Written instructions on the use of IDIMS are also provided; a copy of the document entitled "How to use IDIMS" was included with a previous quarterly report for this contract.

2.3 Integration With McIDAS

One of the most important features of IDIMS is its integration with the McIDAS data visualization and analysis system. The nature of IDIMS as an end-to-end system allows the user to locate data of interest and transfer it into McIDAS for display, in one step. Before IDIMS was developed, users retrieved data into McIDAS by submitting requests to a designated member of the local McIDAS operations team. One reason for this was that the McIDAS software provided few security features, and any user's designated McIDAS areas could be inadvertently over-written. As part of its integration with McIDAS, IDIMS provided this security feature, so that any particular IDIMS user can only load data into his own McIDAS areas.

2.4 Data Archiving

The data archive feature of IDIMS was designed to fit several scenarios. In every case, the user is guided through the menus to generate a file name that conforms to the ESAD naming convention. The user may then rename an existing IBM data file, thus entering it into the IDIMS data management system for later retrieval. He may also specify a McIDAS "area" to be saved to an IBM data file under the new name. Alternatively, he may use IDIMS as a name generation tool only, then leave the IDIMS menus to allocate a new file with the name just generated.

The IDIMS data archive feature allows a user to enter a data set into the ESAD data management system without going through a formal data submission process, or manually providing metadata to a data manager. It also allows users to archive their own data sets from McIDAS, again without requiring the help of a designated member of the local McIDAS operations team.

2.5 Tape Catalog

During the term of this contract, it became clear that on-line or near-line storage was not required for all ESAD data sets. A new requirement was levied, that IDIMS provide the capability to catalog off-line data as well. To that end, ESAD scientists were surveyed concerning the data they held on tape, that was not also stored on the EADS mass storage system. New ORACLE tables were created to hold information about the data tapes, and metadata was collected from the scientists. IDIMS was modified to query metadata concerning both data files and data tapes, and to display to the user the results for both on- and off-line data.

Because data tapes are not automatically entered into IDIMS, as EADS data files are, scientists must be surveyed periodically for tape information, and that information must be manually entered into the database.

2.6 User Education and Support

The IDIMS development team has provided periodic demonstrations in the use of IDIMS, and often provides individual training for new users. We also provide copies of the "NASA/MSFC Earth Science and Applications Division Data Set Naming Convention", and "How to Use IDIMS", upon request. In addition, a team member has always been available to help users with problems and questions, and to update IDIMS menus and the "Naming Convention" document when data from new projects, platforms, or sensors are acquired.

2.7 Maintenance Issues

In addition to the major development areas described above, periodic modifications to IDIMS have been necessary, due to changes to EADS system software, or to ORACLE or McIDAS upgrades. These software updates have been noted in quarterly reports to this contract, as appropriate.

2.8 Prototype Graphical User Interfaces

With the advances in user interface technology, investigations were made into providing IDIMS capabilities on another computer platform, which allowed the use of a graphical user interface (GUI). GUI's provide many ease of use features not generally available in character-based interfaces, including "point-and-click" mouse interaction, and scrolled windows, which allow the display of much more information on one screen. A graphical IDIMS could thus avoid the navigation through many menu levels necessitated by the original IDIMS user interface. Both personal computer and UNIX workstation platforms were investigated, and a prototype IDIMS GUI was developed on each.

2.8.1 OS/2 Prototype

Use of OS/2, a multitasking operating PC system with a window-type graphical user interface, became widespread within ESAD, because McIDAS was ported to this system. During the summer of 1991, a prototype IDIMS interface was designed for OS/2, and initial menu windows were created. However, this work was not continued for several reasons. Because of the

immaturity of the operating system, few development tools were available. Also, the Windows interface became available for DOS, so that there was no standard PC operating system within the division. Lack of a local area network (LAN), and an OS/2 server machine to host the IDIMS application combined with the above factors to render an OS/2 IDIMS impractical at that time. Many of these problems have now been surmounted, however, so that future development of an OS/2 IDIMS may be called for.

2.8.2 UNIX Prototype

The UNIX operating system, X-Windows user interface, and TCP/IP network protocols, provided a mature and popular set of standards on which to base development of an IDIMS GUI prototype. UNIX/IDIMS was designed to allow the use of more metadata for cataloging scientific data sets, to be more flexible and easier to navigate than the original IDIMS, and to allow the user to "browse" reduced resolution versions of image data, before deciding which files to retrieve.

A study of user requirements was conducted before beginning design of the new user interface, and design ideas were presented to the ESAD Technical / Management Information System (TMIS) Committee for review. During the spring and summer of 1991, development of a functional UNIX/IDIMS prototype was begun. This prototype provides all the functionality of IDIMS, with the addition of new features, including user selection of geographic area of interest from a map displayed on the screen, and the capability to save queries from session to session. An initial browse image display capability was also developed.

With the advent of the UNIX-based EADS II, or the ESAD acquisition of a UNIX server machine, continued development of UNIX IDIMS may also be appropriate.

3.0 DATA MANAGEMENT FOR SPECIFIC PROJECTS

Another task performed under this contract has been assisting with data management for field programs. Though related to the larger task of managing the division's data, this task also requires attention to particular details for each project. We have worked with scientists on the COHMEX, GLOBE, and ABFM projects to assure that the various types of data for each project are properly cataloged, and stored in formats and on media which make them readily available to both local investigators and others requesting the data. Specific tasks have included archiving data to EADS mass storage, cataloging tape data, assisting scientists with development of local project data bases, consulting on conversion of data to a standard data format, developing data translation routines, and providing information on data cataloging and storage options. A requirement common to many different projects is the development of software to convert raw data files to McIDAS format data files. In general, each sensor produces data in a unique format, so that a different piece of software is required to convert each type of data. During the term of this contract, conversion software has been developed for data from such diverse sensors as rawinsonde, VISSR Atmospheric Sounder, and AVHRR.

4.0 DATA VISUALIZATION AND ANALYSIS

During the progress of this contract, visualization tools other than McIDAS have become more widely available. In conjunction with data management issues, contract personnel have also participated in the evaluation of visualization capabilities within ESAD. This work has included determining users' needs, evaluating various hardware platforms and software

tools, as well as demonstrating visualization tools to scientists, and instructing the scientists in their use. Several visualization and analysis tools have been developed locally, or at other NASA centers. Under this contract, a printer driver was developed, which is used to produce color prints of OS2-McIDAS images on the HP PaintJet printer.

5.0 STANDARD DATA FORMAT EVALUATION

Use of a standard data format, useable with many different visualization and analysis systems, would greatly simplify data management and data sharing within ESAD. Under this contract, two case studies have been conducted, utilizing the Common Data Format (CDF) and the Hierarchical Data Format (HDF). The data set implemented in each case was the Special Sensor Microwave/Imager (SSM/I) data set.

5.1 Description of Effort

Portions of the SSM/I data set were obtained from the WetNet project at MSFC. These data existed on 6250bpi 1/2 inch reel-to-reel magnetic tapes. Due to memory constraints on the workstation, only five orbits of data, at approximately 2.5Mbytes each, were copied from the tape onto hard disk. Each record of SSM/I data consisted of 1784 two-byte integers. Some fields contained multiple items in compressed form. A FORTRAN read program was provided with the data.

The first case study involved CDF. The CDF software was obtained from NSSDC. This was loaded onto a Silicon Graphics 4D/340VGX workstation running Unix. At first there were problems because CDF had not been tested on this version of the operating system. This required modifications to the CDF software. When these were completed, all CDF routines compiled and ran correctly. A CDF skeleton-table was generated which defined 34 variables for the SSM/I data set. The SSM/I read routine was combined with the CDF library to produce a translator which input an orbit of SSM/I data and output a corresponding CDF. The translator consisted of approximately 1400 lines of FORTRAN. The resulting CDF was perused with the three CDF utilities, CDFlist, CDFbrowse, and CDFinquire. All of these worked perfectly. A CDF could be produced with any of the five orbits of SSM/I data, but only one could be kept on disk at a time due to storage constraints.

The HDF case study was conducted next. The HDF software was obtained from the National Center for Supercomputing Applications (NCSA) via ftp. This was loaded onto the same workstation as that used for the CDF case study. HDF consists of a library of C routines with both a FORTRAN and a C interface. This similarity with CDF allowed the reuse of much of the software used in the CDF case study. In particular, the translator used in the CDF case study was modified for the HDF test. All CDF references were removed, retaining the read routine and program skeleton. New routines were written which took the compressed SSM/I data and placed it into an HDF. This was accomplished in less time than that required for CDF because of the reusable code.

Since HDF is designed for array or gridded data sets, the 34 SSM/I variables were organized into an array structure using the HDF slice utilities. This two-dimensional array was considered an HDF Scientific Data Set (SDS). One SDS contained one record of SSM/I data. The resulting HDF file consisted of one orbit of these data sets. The HDF routines worked without a hitch. The main concern was that HDF currently stores all data as 32 bit, IEEE format, floating point numbers. There was not a way to mix data types within an SDS.

5.2 Recommendations

CDF was more cumbersome to use than HDF. It appeared that the original version was written for a VAX system, and was later modified for ports to Sun and other UNIX machines. This has resulted in software that has been patched and is not efficient. The best thing about CDF was the almost unrestricted manner in which one can represent metadata. The CDF "skeleton table" structure allows the user to define as many global and variable attributes as he wishes, all of which can be used to describe the data.

HDF was a joy to work with, mainly since every routine used worked perfectly. The software is easily accessible via ftp, and has many examples which can be edited to produce tools specific to the data set of interest. The disadvantage of HDF is in its support of only one data type, IEEE floating point.

A detailed report was produced fully documenting the findings and recommending HDF.

Final Report for NAS8-37139
Prepared by Don Moss

I developed software and operating procedures for the WetNet project to use in producing and distributing SSM/I data. Software was developed on two platforms, EADS IBM mainframes and PC's. Specifically:

Wrote new and modified existing McIDAS processing software:

To read from disk instead of tape, to save processing time.

Implemented binary data packing to reduce storage space by 90%, allowing it to run on EADS.

To correct navigation anomalies.

To detect errors in orbital elements.

Conversion of orbital elements into the parameters needed by McIDAS.

To automatically renumber areas to conform to WetNet numbering specifications.

To ingest Level 1B SSM/I data from NESDIS.

To process Level 1B data instead of Wentz data.

To schedule data ingest.

To perform certain McIDAS functions in DOS on a PC.

To refine navigation of McIDAS areas.

To edit binary McIDAS files.

To eliminate manual entry of certain parameters in production of McIDAS areas.

Quality control and filtering of Level 1B data.

Secured sources of satellite orbital elements.

Developed repair procedures for damaged WetNet magneto-optical disks to correct problems such as deleted directories and cross-linked files.

Participated in design of WetNet MOLEVEL file and system of performing field updates to magneto-optical cartridges.

Downloaded source code and McIDAS metadata for distribution via WetNet MO disks.

Analysis of Level 1B data consistency.

Final Report of Work Completed Under NAS8-37139

Dr. Mike Botts
1/20/93

A. EVALUATION OF THE VISUALIZATION ENVIRONMENT AT ESAD

I was brought on-sight to NASA Marshall Space Flight Center's (MSFC) Earth Science and Application Division (ESAD) in June 1990, in order to evaluation the existing visualization environment at ESAD and to implement efforts to correct any deficiencies within this environment. Much of the initial efforts concentrated on evaluating the scientific computing needs of the division scientists and determining whether the existing visualization environment provides the proper tools for meeting these needs. Some of the deficiencies have been corrected using "off-the-shelf" software available for the Silicon Graphics (SGI) computer. Other deficiencies have been or will need to be corrected by in-house development efforts. Much consideration and energy has been spent preparing for this development effort, so that development will occur in a highly directed fashion and within an integrated, flexible, and expandable environment.

At the beginning of my activities, the visualization environment at ESAD consisted primarily of the McIDAS turnkey image display system (mainframe & PC), with minor supplemental application programs running on I^2S, Stardent, and PC platforms. Although a few scientists were satisfied with the present visualization environment, many expressed moderate to extreme frustration with the lack of adequate visualization tools to meet their needs. The findings, as reported in Botts [2], were summarized below:

- McIDAS provides many visualization needs, but not all
- Many scientists have abandoned McIDAS because of difficulties of use, or because it does not meet their needs
- Scientists often turn to uncoordinated and generally inadequate development on PC's to try to meet their needs
- Some advanced prototype tools have been developed on Stardent, but development has never progressed to a stage of a general useful tool

Ineffective use or abandonment of existing visualization tools has resulted from:

- Lack of Integration of Tools
- Lack of User-Friendly Interfaces
- Lack of Coordination & Archiving of Software Development
- Incompatible Data File Formats
- Lack of Simple Output to Video or Print
- Tools Not Available to Meet Many Visualization Needs

Outside of specific needs by various scientists, general needs of scientists at ESAD included:

- Ability to easily move data between visualization and analysis tools
- Easy to learn/easy to use applications
- Ability to integrate and fuse different data types from various sources

- Ability to interactively probe and analyze data, not just visualize
- Flexibility to quickly add new features into existing tools
- Balance of hardware environment between high-powered & personal workstations
- 3D important in many applications

The importance of developing within a flexible, integrated visualization environment (DAVE) was discussed in Botts [2]. In order that software development at ESAD result in flexible, extendable, and portable code, much effort has concentrated on defining the standards and tools under which this development will proceed. Transfer of data between various file formats (e.g. McIDAS, SGI, X, PC-based, and YUV) has been accomplished. Candidates for Common File Formats (e.g. CDF, netCDF, and HDF) have been evaluated. Common Data Structures to be used at ESAD are being development and implemented (Botts [4]). Standards have been evaluated and selected for operating systems (ATT UNIX V), windowing systems (X/Motif), programming languages (ANSI C, C++, and FORTRAN 77), and 3D graphics libraries (SGI GL). A tool for building Graphical User Interfaces (GUI), UIM/X, has been evaluated and selected.

It was also determined that furthering the visualization and analysis capabilities within ESAD was limited by inadequate and inappropriate hardware. In partiular, the ESAD was highly PC-bound, whereas many of the required tools required Unix-based graphics workstations. Part of the improvement of the visualization environment at ESAD has focused on evaluating graphics hardware platforms, and recommending proper hardware directions. Although much initial work had gone into this task, the primary drive for this change came in the form of the Scientific Computing Facility (SCF) Working Group, which I chaired (Botts [6]; Phillips & Botts [4]).

B. McIDAS EVALUATION AND DEVELOPMENT

Being the primary visualization tool at ESAD, there was a concentrated effort to learn and evaluate the applications and development environments of McIDAS. This involved learning McIDAS on the mainframe and PC as a casual user would learn, as well as delving, to some degree, into the intricacies of more advanced McIDAS programming and development. In addition, much effort has been concentrated on the future direction and development of McIDAS, particularly with regard to the McIDAS port to a UNIX-based graphics workstation, as well as the continued development of McIDAS vis5d into an application tool. This effort included two trips to SSEC at the University of Wisconsin, for the purposes of understanding the future path of McIDAS and expressing the future needs of ESAD within the McIDAS environment. Mike Botts coauthored a small report to Jim Arnold and Greg Wilson, entitled "Suggestions for a UNIX Based McIDAS", outlining needs and concerns of ESAD.

In addition to "core McIDAS", a 3D, Unix-based prototype, Vis5D, was also developed at SSEC by Bill Hibbard. Along with Paul Meyer at ESAD, I evaluated Vis5D for usefulness and for extensibility. Efforts directed at locally expanding Vis5D, and extending over several months, determined that although the tool was useful, it was not easily extendable.

C. VISUALIZATION SOFTWARE EVALUATION AND DEVELOPMENT

It was determined that many of the visualization and analysis needs were not being met with McIDAS, and that the situation would worsen with the movement into the EOS era. Much effort was put into evaluating off-the-shelf software, and in introducing scientists

to the capabilities that presently existed. Visualization tools that ere evaluated extensively included Advanced Visualization System (AVS), IRIS Explorer, ApE, ELAS, FAST, Vis5D, SpyGlass, VolumeVis, and LinkWinds. Many of these programs partially met many of the needs at ESAD; however, all had limitations and certainly no single package would meet all the needs of such a diverse group of scientists.

Initial efforts to introduce scientists met with poor results, primarily because of either difficulties in getting data into the tool, difficulties in learning the tool, or not having adequate hardware for running the tool. However, as more SGI workstations have come into ESAD, many scientists have begun to request and work successfully with several of these tools. In particular, LinkWinds, which is under development at JPL, has proved a very useful tool for meeting a wide range of analysis and visualization requirements at ESAD.

In addition to "off-the-shelf" software, some of the efforts to meet the scientists' needs has required in-house development. In particular, the MASS development to be discussed below under "Projects" was one such effort. Unfortunately in most cases, anything other than small-scale in-house development was not practical, because of either limited personnel or inadequate hardware. This particularly became the case once it was determined that the Stardent computer would not be appropriate for development activities (as described below).

As discussed above, efforts were directed toward evaluating appropriate development tools and standards. Those selected included the evaluation of UIM/X for X/Motif Graphical-User Interface (GUI) development, and SGI IRIS Graphics Library (GL) for 3D graphics. Under my direction, we also served as a Beta site for SGI's Inventor software, which greatly speeds development of 3D applications, and IRIS ImageVision Library (IL) which provides ready-to-use image processing routines for development activities.

D. HARDWARE EVALUATION

Computer hardware capabilities and requirements were a major part of the evaluation of scientific visualization and analysis needs at ESAD. It was important to evaluate not only the CPU and graphics power available with various platforms, but also the computer environment as a whole, including the availability of existing software and the ease with which software development could be accomplished on the platform. Hardware evaluation included literature research and vendor demos, as well as on-site evaluation of loaner workstations. Workstations that were consider included those from Stardent (Stellar/Ardent), Sun, SGI, HP, DEC, and IBM.

A summary of the hardware evaluated on-site follows: A Stellar GS 100 workstation was purchased by ESAD under this contract. This was upgraded, under the contract, to a Stardent GS 2500 after the merge of Stellar and Ardent. An SGI 4D/50G was leased for an extended period of time, an SGI 340VGX and an SGI Crimson VGX were on loan for periods of 6 months and 2 weeks respectively. In addition, HP 7000 Series and IBM RS 6000 workstations were brought in for evaluations for periods of 1-2 weeks. All of these were evaluated for graphics performance, ease of use, ease of development activities, and for speed of running numerical models that were in use within ESAD.

A more extensive hardware evaluation was conducted under the SCF Working Group activities, as discussed below and reported in Botts et al [6]. For the SCF activities, a hierarchical workstation environment was proposed which included vector-based supercomputing, scalar-based supercomputing, file servers, high-end visualization

workstations, and low to mid-range personal graphics workstations. It was determined that workstations from SGI, met the primary needs for scalar-based supercomputing, file servers, and high-end visualization platforms, as well as being very good choices for low to mid-range personal graphics workstations. Since that time, the computing environment at ESAD has progressed from one leased SGI to 25-30 SGI of various power.

E. STARDENT WORKSTATION MANAGEMENT AND USE

Before I began activities under this contract, UAH had leased a Stardent GS 2500 for evaluation as a future development platform for ESAD. This platform provided powerful computational and graphical functionality required for meeting visualization demands, and included the Advanced Visualization System (AVS), developed by Stellar. Experimental development activities at SSEC were also underway on the Stardent platform. Several activities, such as the development of Vis5D at SSEC and the use of AVS at ESAD, proved in concept the importance of advanced graphics workstations for visualization activities.

However, in 1991, the evaluation of many factors gave cause to a growing concern regarding the use of the Stardent Computer as a feasible platform for development. A primary factor was the lack of development tools, particularly the absence of a industry-accepted graphics library. This required the developer to spend a very large portion of development time writing and rewriting modules for doing every minor graphics operation. The lack of a graphics library which was optimized for the platform has required the developer to have intimate understanding of the Stardent architecture in order to optimize rendering algorithms for that particular platform. In addition, optimizing for one algorithm (e.g. polygon rendering) created difficulties for other operations (e.g. transparency).

Furthermore, the stability of Stardent had become a serious issue over the last 6 months. For this reason, we began investigating other potential platforms for visualization development and applications, including the Silicon Graphics (SGI) 4D Series and the IBM RISC 6000. The Iris Graphics Library (GL) was one major advantage of these platforms, in addition to their high graphics performance. We began working on porting usable parts of prototype code developed on the Stardent to the SGI platform. However, in most cases, we found it more effective and efficient to implement some of the prototype's functionality by rewriting the code using Iris GL. The downfall of Stardent as a workstation vendor in late 1991 justified our fears of Stardent's stability and accelerated our activities for moving development to the SGI.

Because AVS was spun off as a separate company, evaluation of the AVS package continued. Since AVS has broken away from Stardent as a separate company, with AVS running on many platforms, the AVS application package has more potential for success. We also began a parallel evaluation of Explorer, a similar visualization environment from SGI.

In January 1992, at the request of ESAD and after the downfall of Stardent Computer, UAH began investigating the feasibility and legality of stopping the Stardent lease. However, because of wording within the contract and because Stardent had sold the lease to an independent leasing company, it was determined that we were not legally able to stop the lease at that time, and the leasing company was not willing to negotiate in good faith.

For the reasons stated above, activities on the Stardent platform shifted from development to a operational. Ironically, scientists began using the Stardent

workstation more actively after the demise of Stardent, primarily because their needs for visualization had been realized. Many scientists have been using AVS, Vis5D, and X-sect, developed by Paul Meyer, and have been produced several videos. In addition, a customized program which displays 12 years of MSU temperature anomalies on the Stardent was developed at SSEC, and became a key component in the debates regarding the realities of global warming. The Stardent has also continued to serve as a source for hard disk storage, as a 9-track tape reader, and as a video recording platform.

F. SILICON GRAPHICS MANAGEMENT

Under this contract, I was partially responsible (along with John Parker and Paul Meyer) for the system management of all SGI systems brought in for evaluation. This included user support, user instruction, network functionality, and software installation and maintenance. The active periods included 2 years for the 4D/50G, 6 months for the 4D/340VGX, and 2 weeks for the Crimson VGX.

G. DATA OUTPUT

In addition to hardware and software issues, much energy has been expended in evaluating solutions to data input/output between computers and video/hardcopy devices. An Abekas A60 video frame store unit, capable of "grabbing" or playing 720 frames at video rates, was brought into ESAD for evaluation (Meyer and Botts [3]). The Abekas has been successfully used for recording several video sequences of scientific visualization scenes, as well as "grabbing" and digitizing video frame ssequences from Shuttle video cameras and from a VHS recording of a tornado. Images from the tornado video were captured and composited in order to create a panorama of the sky during the tornado. Shuttle video has been captured in order to process and study images of cloud-top lightning. Digital images can be freely passed in either direction over ethernet between the Abekas and the SGI or Stardent workstations.

For color printing of computer generated images, the TOYO TPG3100 Thermal Printer and the Mitsubishi S340 Sublimation Printer, have undergone testing with Stardent, SGI, and McIDAS generated images. The TOYO has proved to be quick, but with poor color reproduction, whereas the Mitsubishi provides very high-quality color reproduction, with some sacrifice in speed. The dye sublimation process, which is available on several printers, has been recommended to ESAD for visualization hardcopy reproduction.

In order that file transfer can be done rather transparently to the user, several file format converters have been developed, including converters to and from SGI.rgb, targa, yuv, xwd, and McIDAS.

H. PROJECTS

Several individual activities warrant individual discussion. Four in particular demanded more effort than many of the activities that were completed under this contract. These included SCF Working Group, MASS development, SSMI development, and the Molniya orbit video. Other projects, such as the band compositing of MAMS imagery , Huntsville tornado panorama creation from video, and capturing and processing of shuttle lightning imagery took more moderate efforts.

The SSMI development effort occurred in 1990. At the request of Roy Spencer, application software was developed, which allows processing and evaluation of SSMI data on the Stardent. These programs effectively read and transform Wentz format SSMI data into a more usable format, and then allow bands of the data to be interactively mixed and composited into RGB components for viewing on the Stardent. The conversion routines also allow transformation of the SSMI data through Principal Component Analysis. These programs were written in a generic and modular fashion, in order that they might be easily incorporated into other present or future application packages. This project is also serving as a testbed for more sophisticated future development in C within the Stardent environment and within a GUI environment (AVS).

In the first quarter of 1992, I created a four minute 3D computer animated video illustrating the unique features of the Molniya Orbit. This included several views of single and triple orbits, as well as a view of earth from the satellite during full orbital passes. This animation was presented at the AMS conference by Dr. Stan Kidder.

From the summer 1991 until March 1992, I was highly involved as chairman of the SCF Working Group which has been tasked with evaluating the computing requirements of the five SCFs at MSFC and recommending general and specific recommendations for meeting these requirements. This effort will result in a Technical Memorandum (Botts et. al.[7]. Ron Phillips has coauthored this document with Mike Botts, specifically providing knowledge about computer manufacturers, discussing current and future hardware/software directions and system requirements, gathering the information about the SCF candidate systems, and assiting in the formulation of general options and specific system recommendations. With slight modificat, the concept in this document have since become the standard document for organization of all ESAD computing facility requirements.

Finally, Botts and Phillips have developed the initial Multidimensional Analysis of Sensor Systems (MASS) prototype for visualizing co-registered data from airborne flights, ground-based field mills, satellite imagery, volumetric radar data, and McIDAS geographical boundary data. Dr. Doug Mach (UAH), Pat Wright (USRA), and Dr. Kevin Knupp (UAH) provided fifteen minutes of CaPE airborne field mill, ground-based field mill, and radar data acquired on 16 March 1991. Anthony Guillory (NASA) provided color McIDAS satellite imagery of the east central Florida area. The prototype, written on the Silicon Graphics (SGI) using the Graphics Library (GL), proved the feasibility of real-time graphical interaction with these data types.

This prototype proved the feasability of coregistering and viewing several distinct data sets within a single window, and providing the researcher with the ability to interactively alter the view and appearance of this data. The graphical techiniques including line and surface rendering, display of irregularly spaced of ground-based field mill data, texture mapping of images on a plane, and volume rendering of 3D radar data. In addition to developing the prototype, much initial detail has been given to the tentative structure and layout of the application. While expandability of both visual and analytical were a goal for the application, a major consideration was the development of an interface that is simple and intuitive for a scientist to use.

An informal demo of the MASS prototype was presented, with positive feedback, to Dr. Greg Wilson as well as to CaPE and EOS scientists at Marshall. This prototype is also serving as an example of capabilities requires within the next MIDDS operational system for Space Shuttle launch support.

I. TRIPS

June 1990: Meeting with SSEC (Madison, WI) re: McIDAS directions

November 1990: Visualization '90 Conference (San Francisco, CA) to evaluate general state of visualization

November 1990: McIDAS Users Group Meeting at SSEC (Madison, WI.)

June 1991: Visualization workshop at NASA Headquarters (Washington, D.C.)

January 1991: American Meteorological Society (AMS) Conference (Atlanta, GA) for evaluation of visualization directions within meteorological field

August 1991: ACM SIGGRAPH for evaluation of graphics hardware and visualization software (Las Vegas, NV)

October 1991: SGI Developers Forum (San Francisco, CA), for prerelease information on SGI hardware and software directions and for tutorials on graphics development

J. PUBLICATIONS FROM CONTRACT

1. Meyer, P. and M. Botts (September 1990). *Suggestions for a UNIX Based McIDAS*, report to NASA/MSFC/ESAD.

2. Botts, M. (January 1991). *Recommendations for Establishing a Data Analysis and Visualization Environment (DAVE) at the Earth Science and Applications Division, NASA Marshall Space Flight Center*, white paper submitted to NASA/MSFC/ESAD.

3. Meyer, P. and M. Botts (January 1991). *Summary report to Division on Video Capabilities and Needs*, report to NASA/MSFC/ESAD.

4. Botts, M., 1991: *The Importance of Data Abstraction and Standard Data Structures in Visualization Development*, submitted to Visualization '91.

5. Phillips, R. and M. Botts, 1991: *Computer Industry Directions*. **VisTech, Issue 1**.

6. Botts, M., 1992: *Defining and Satisfying the Computing Requirements of the Scientific Computing Facilities at NASA/MSFC.* **AIAA Space Programs & Technology Conference**, March 23, 1992.

7. Botts, M., R. Phillips, J. Parker, and P. Wright, 1992: *Functional Requirements Document for the Earth Observing System Data and Information System (EOSDIS) Scientific Computing Facilities (SCF) of the NASA/MSFC Earth Science and Application Division*, **NASA TM 4392**, pp. 101.

8. Botts, M., 1992: *Visualization for Atmospheric and Global Change Research at the NASA Marshall Space Flight Center*, **IEEE Visualization '92**, October 22,1992.

APPENDIX C

SUMMARY REPORT TO DIVISION ON VIDEO CAPABILITIES AND NEEDS

by

Paul J. Meyer

and

Michael E. Botts / UAH

January 29, 1991

OVERVIEW

One of the most urgent and compelling needs of the division at the current time is the enhancement of our video capabilities. The present video-out capabilities have proven extremely useful for conference and management presentations, as well as for data interpretation. However, the current method of producing these video sequences is very cumbersome and inefficient. Many research teams have expressed a need for video recording, as well as video frame grabbing. These same research teams have further expressed frustration with the currect methods of producing video sequences.

Currently, all video recording must be performed through the McIDAS system within the McIDAS room. This method has several limitations:

Long, or even moderately long, frame sequences must be produced in piece-meal fashion, and then spliced together at the MSFC video shop;

Difficulty in equipment setup and operation requires experienced personnel to assist in the video production;

Images coming from any source other than McIDAS must be converted, transported, loaded into McIDAS areas, and then recorded; If the frame sequence is longer than 30 to 70 frames, this process must be repeated ad-infinitum, with constant user assistance, until all frames have been recorded; This process is one that has been described as taking from half to four days time;

These limitations and annoyances have been a severe deterent to further use of our video output capabilities. With the current state of technology, video recording of computer generated animation can be simple, reliable, and efficient, requiring very little interaction of expert personnel.

Another drawback of the current video capabilities is the total inability to perform video frame grabs. Requirements from the research scientists have indicated needs for not only single frame grabbing, but the need for grabbing a sequence of frames from a video source, as well.

VIDEO NEEDS

Based on the requirements of the scientists, we have identified the following video needs:

The ability to easily record animation from many sources, including McIDAS workstations, PC-McIDAS, general PCs, and graphics workstations;

The ability to record long animation sequences with minimal difficulty;

The ability to easily route video signals from multiple sources to multiple outputs; This currently requires repeated recabling, depending on the application;

The ability to accurately grab individual frames (or fields), or sequences of consecutive frames;

The ability to produce clean, photo-quality prints of video screens from workstations or PCs utilizing the local area network;

VIDEO SOLUTIONS

We have begun studies, and have requested the assistance of MSFC TV to address these needs. We have determined that the following items would solve these needs:

Frame Record/Frame Grab server
Accessories: Frame Sync, Video- Decoder, Encoder

Video matrix (switcher)

Networked photo-realistic quality printer

Professional quality player/recorder

The most immediate needs are for the frame record and grab server, and the video matrix device. These two facilities would provide invaluable service to a wide range of projects within the division, and require immediate attention. The other needs are long term, and/or project specific.

Frame record/grab server

Three options are available for recording and grabbing video sequences. The evaluation of the options described below assumes that the editing VTR's are available to ES41 (assuming MSFC purchases the Grumman leased equipment at the end of the EADS contract), and include:

1. Frame by Frame sequencing -

   This method involves frame by frame recording directly onto an editing VTR by means of an animation control device. This involves a sequence of pre-rolling the VTR, recording a single frame, and repeating this process continuously for each frame until all frames are recorded. Video frame grabing would require the same frame-by-frame process. This system would incorporate a Personal computer, with an AT VISTA card, as well as an animation controller, and requires the presence of an editing VTR. This is the most inexpensive solution. However, this method has several disadvantages, including excessive wear on VTR's due to the stop and go action, the great amount of time required to transfer images to the deck, and the greater possibility of low quality animation resulting from bad sync and frame dropouts. Approximate cost assuming continued presence of the editing VTR's solution is $25,000.

2. PC based video server with laser video disc -

This method entails PC control over an eraseable laser video disc (LVD). Images would be sent to the PC over ethernet, stored on the magnetic disk, then transferred through software to the LVD. At this point the animation sequence could be played and reviewed in real-time, as well as recorded on any video recorder. Video frame grabbing would consist of direct record of a sequence in real-time from any video player device to the LVD, followed by transfer to the PC. This is a moderate to high priced solution. It has the following advantages; (a) does not require an editing VTR, rather, any video source or device complying with NTSC standards is sufficient, (b) does not require frame-by-frame recording or grab of video, (c) it has up to 55 minutes of stored animation per disc, (d) it could be employed as an image animation archive, and (e) the required paint package allows for some pre/post-processing of the images on the PC. Disadvantages include: (a) incorporates many hardware and software components requiring extensive time for integration as well as higher probability for component failure, (b) it requires extensive expertise and training for use, and (c) it cannot be controlled by the user from his desk. Costs for the complete configuration including software is approximately $50,000, but does not include the cost of integration.

3. Abekas digital recorder -

The Abekas solution is a simple, single-component, standalone system. Like the solution above, it allows transfer of data to the device over a network and direct record onto any VTR. However, unlike the above system, it is simple in operation and easy to integrate. It allows for direct record (grab) of video data from an RGB, or NTSC source and allows for preview and review of the stored data. It has the ability to not only access video frames, but also accesses at the field level (1/60th of a second). It can be operated from the users workstation via ethernet, as well as from its own keyboard. Other advantages include: (a) elimates one major step in the recording/grab process (transfer between PC disk and LVD), (b) there is virtually no integration time involved, (c) training and need for user expertise is minimal, (d) sequences can manually be reviewed (rolled forward and back, or stopped), allowing this to be used as an interactive tool to review and edit the animation sequence before it is ever recorded. Disadvantages are limited animation storage (750 or 1500 frames), and the slightly higher initial cost. Approximate cost for this system is $65,000 for the A60 and A20 units.

After careful consideration of the user requirements and the capabilities of the above options, the Abekas solution is considered to be the most viable device for meeting the needs of the division. The slightly added cost is more than offset by the minimal integration effort and level of expertise required for use.

Video Matrix

A video matrix is required to eliminate the need for constant recabling of the video equipment with every application. It further eliminates the need for expert intervention in video recording, grabbing, or printing. The system needs to be capable of expansion and should be keyboard controlled. We may also need to look into an audio capability. At the present time, we need a minimal configuration of 8 inputs and 16 outputs (8x16 matrix). We need to procure a system which is able to be expanded as our needs increase. Advantages include multiple routing of input to several output devices. This would allow a user to easily dub video tapes to multiple output decks. For example, it could allow video feed to the MSFC center, a monitor in the McIDAS lab, a VTR, etc. MSFC TV is assisting us in determining our needs for the matrix. Approximate cost of an 8x16 matrix and keyboard control is $15,000.

Networked Printer

Division scientists would like a capability to produce very high quality prints of scientific results from their PC's, the Stardent computer, and other machines on the Local Area Network. Printers are available and continue to be marketed which may meet this need. These printers work through a thermal emulsion process and produce photo-realistic quality images. These prints may be used for conference presentations as well as journal articles, and division propaganda for future funding. This capability is necessary within 18 months and currently costs from $20,000 to $30,000. Very expensive options are also available for approximately $80,000.

Professional Quality VTR

Our VHS video systems currently are limited in their capability. We need to pursue purchasing a system which is of professional quality and has all three play speeds, and is able to have an external sync source attached. We may also want to pursue the purchase of Betacam format VTR. The Betacam systems are used by MSFC TV for their internal work, and are a very high quality recording system. Costs for a VHS system may be up to $2,000, while prices for a Betacam system ranges from $15,000-30,000. With an Abekas system in the lab, the purchase of a Betacam editing deck could be forgone by using a portable Betacam VTR from MSFC TV.

Abekas Evaluation

We have been evaluating the Abekas solution for the past several months. Another lab at MSFC has allowed us to utilize their machine until all their equipment arrived. This has been most productive. We have been able to demonstrate to the researchers the ability of the Abekas for both video data record capability and video data frame grab. We plan on producing a video sequence of global temperature anomalies which is over 700 video frames in length. Previously this took approximately 5 days to perform. We expect that this should now take on the order of 4 to 5 hours maximum, with user intervention. As this process is streamlined, this type of video storage capability should be able to be performed in less than 2 hours with trivial user intervention.

Also demonstrated was the ability to perform video frame captures. The lightning group has captured video sequences taken from Space Shuttle missions and performed single frame black and white prints, as well as transferred the data to the Stardent computer for archival on magnetic tape. They plan on performing some image analysis routines on this digitized data. Over 600 frames of data have been already digitized in a few hours. Previously, two months were spent in attempting to capture a handfull of video frames.

As scientists desktop PC's are connected to the Local Area Network, they will be able to use the Abekas system as a video production facility. One can easily envision a scientist creating a long movie sequence from their PC McIDAS system. Some of the software is in place to achieve this goal. Or, one may envision non-McIDAS data converted to an animation sequence. These video sequences would be extremely useful in disseminating information at conference presentations.

The Abekas device allows animation rates to be user controlled either from a control console or from a computer system which is connected to the Local area network using the TCP/IP protocol suite. Actually, almost all functionality of the control console may be done remotely. The unit has performed well, there are some minor problems with the unit in the looping mode (due to magnetic head seek time from end of loop to beginning of loop, a slight hiccup occurs). Abekas engineers have informed us that this problem goes away with the 1500 frame version of the A60 (it keeps two frames in buffer). There is also a time-code trigger mode which we have not yet explored that may remove this problem as well. The engineering staff at Abekas have helped us tremendously with product setup, an ethernet control problem, etc. They apparently stand behind their product. Overall, the evaluation has been extremely favorable. While the unit is a costly item, we believe great use may be made of this device by division researchers. In the product demonstration given to the division, the users were interested in the device and its capabilities. We already are working with five different individuals with the use of this device. We are also trying to use this device to perform some simulation experiments (what a polar orbiting satellite would see at nadir as it orbits the earth). Video connections for this device are shown in the attached figure.

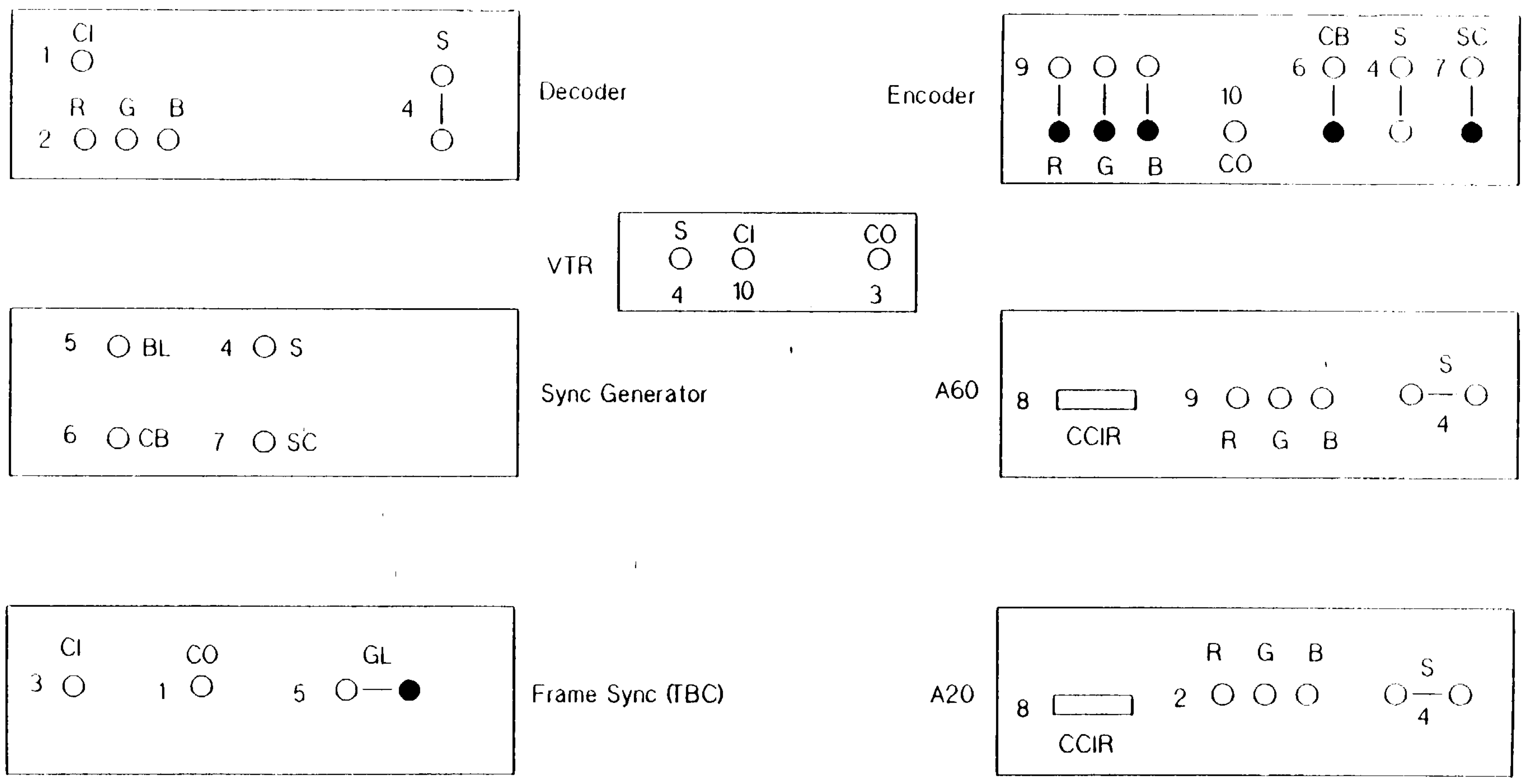
Again, we believe that digital video frame storage and capture is something the division requires to augment our research capabilities. The Abekas device seems to meet these needs.

CONCLUSIONS

Clearly, there is a great need for upgraded video capabilities within the division. Two immediate needs that have been recognized are the video frame grab/record server and a video matrix (switcher). We have determined that these needs can be met with an Abekas A20/A60 unit plus accessories and with a high quality video matrix (Grass Valley or equivalent). The Abekas units are approximately $60,000 and the matrix is approximately $15,000. Estimated cost for accessories is $10,000. A networked photo-realistic quality printer is needed within 18 months. We can get by with the current VTR systems we have available, but we do need to pursue obtaining a professional quality VHS deck and possibly a Betacam VTR.

This document does not address all the video needs of the division scientists. They regularly request new capabilities and functionality which has not yet been addressed in the scope of this document. One such need is the ability to store sequences of video data from field program activities on a permanent media in some form of compressed data format. This issue and others will need to be addressed in a separate document at a later date.

Video connections for Abekas digital video frame store unit



Legend

○ Connection	R - Red	CO - Composite Output
● Terminated Connection	G - Green	CB - Composite Blanking
○—○ Loop Through	B - Blue	BL - Black Burst
	S - Sync	SC - Subcarrier
	CI - Composite Input	GL - Genlock
	CCIR - Digital Video interface	

Connection Summary

1 - 2	6 - 2
2 - 2, RGB	7 - 2
3 - 2	8 - 2
4 - 10	9 - 2, RGB
5 - 2	10 - 2

SUGGESTIONS FOR A UNIX BASED McIDAS

By
Paul J. Meyer and Michael E. Botts
September, 1990

OVERVIEW:

As the Mission to Planet Earth concept comes into fruition within NASA, Data and Information Systems are needed to handle the large volume of data which will be available to the scientist. The Distributed Active Archive Centers (DAAC's) as part of the EOS concept are already being implemented. Very little work has been done to define the Scientific Computing Facilities (SCF's) which the scientists shall utilize to perform their research. The McIDAS system could be used as a major component of the SCF at MSFC. The system, however, does require upgrades to meet the needs of the EOS investigators at MSFC. Functional hardware elements necessary for an MSFC SCF are depicted in Figure 1.

The immensity and diversity of these databases necessitates the application of scientific data visualization methods in order to adequately verify and analyze these data. The development of the Man Computer Interactive Data Analysis System (McIDAS) at the University of Wisconsin-Madison Space Science and Engineering Center (SSEC), under the support of MSFC, has been a major advancement in the field of data visualization within the atmospheric science community. Continued improvement and expansion of McIDAS, within the IBM mainframe, PC, and graphic supercomputer environments, is vital to the visualization needs of the ESAD at MSFC. However, while a fully integrated solution to all needs is a desirable ideal, attempting to handle all present and future ESAD visualization and analysis needs solely within McIDAS, will result in the inability of ESAD to satisfactorily meet these requirements. McIDAS is a vital component of a highly integrated set of applications tools for scientific understanding. Within that framework, we note the following requirements for McIDAS to meet the ESAD needs of the future.

PORTABILITY:

McIDAS currently is an operating system in and of itself, and performs too much low level system activity. McIDAS needs to be an applications software package which is independent of hardware and computer operating system constraints. The UNIX operating system was designed, and has proven to be, a hardware independent operating system. There are two UNIX standards, the AT&T and the Berkeley interface definitions. UNIX was developed by AT&T Bell labs, and there are a larger number of AT&T based UNIX implementations. In most implementations of UNIX, Berkeley extensions are typically added to augment the communications capabilities of the AT&T version. McIDAS should have operate as an applications package on top of the AT&T System V Interface Definition (SVID) Release 3 or 4. It is important in any port of McIDAS to any flavor of the UNIX operating sytem (e.g. AIX) that the system level calls do not rely on implementation specific calls for that version. Rather, it should rely on calls which are defined in the interface definition.

Within the programming environment, we support two programming languages FORTRAN 77, and C. The C language is a very portable language, and was designed to be used with the UNIX operating system. It negates the need for the use of Assembly language programming, which is very system and hardware dependent. In addition, all device dependent code needs to be rewritten with portability in mind. Portability within code can be accomplished by four primary techniques: 1) Place all device dependent calls within libraries, 2) use a graphics user interface (GUI) which is portable, 3) utilize system calls to UNIX, instead of writing special code to perform file copies, file deletions, date, time, etc., 4) use standard library calls for I/O, memory allocation, etc.

COMMUNICATIONS:

Standardization of communications protocols are necessary in order for McIDAS to share and interact with other applications environments and hardware environments. In order to be completely effective the communications strategies within McIDAS should be programmed at the applications layer of the Open Systems Interconnect 7 layer reference model. Any communications which are not performed at the applications layer, while they may be slightly more efficient, do not comply with new industry standards and should be shunned. Applications written at layer seven tend to be highly portable, for example the telnet and ftp protocols are on most vendors hardware. Currently, the TCP/IP protocol under IEEE 802.3 (ethernet) is the industry wide standard. OSI is the next protocol to be supported by the industry.

With an effective communications strategy, McIDAS shall be able to communicate with other devices on a network for the purpose of data transfer as well as distributed computing. Some of these devices include; Super computers, Super graphics workstations, archival devices, hardcopy units (e.g. video, print), Personal computers. Using TCP/IP in the proper fashion, McIDAS should also be allowed to mount remote devices for data access. This would be accomplished using the Network File System (NFS) developed by Sun Microsystems, allowing the scientist to temporarily make McIDAS think it has access to the entire data archive. This would be extremely powerful when working with large climatological data sets. Figure 2 illustrates a concept for a TCP/IP and NFS based McIDAS system.

INTERFACING:

Interfacing involves not only user interfaces, but interfaces to other applications and data management packages as well. It is highly recommended that the front-end to McIDAS employ a standard and portable graphics user interface (GUI), preferably based on the X-Windows Version 11 release 4. Examples of such are Motif, Open Look, Presentation Manager, NASA's TAE+, and ApE from Ohio Supercomputer center. ApE has the added advantage of serving as a distributed computing environment. These interfaces need to be investigated thoroughly and quickly to determine the present and future directions.

The importance of the GUI is not only to provide a friendly environment for the user, but also to facilitate portability between systems. In addition it provides a common user interface for all other application packages as well. As McIDAS is ported to an UNIX environment, these GUI's need to be kept in mind, and the proper hooks need to be integrated within the McIDAS environment. These hooks should be such that as GUI's advance in appearance and functionality, it would be easy to implement changes.

At the present time, McIDAS does not allow other applications packages or data management facilities to be easily integrated. The UNIX environment should provide the capability to open McIDAS to other applications software. One such example is the desire to add sophisticated image processing capabilities to the McIDAS environment through the integration of ELAS or other packages. Application interface hooks need to be incorporated to McIDAS to provide an easy mechanism for other applications software to utilize the power of McIDAS and vice-versa. For example, work on the Stardent computer with the McIDAS system has allowed us to more easily tie applications packages together.

EXPANDABILITY:

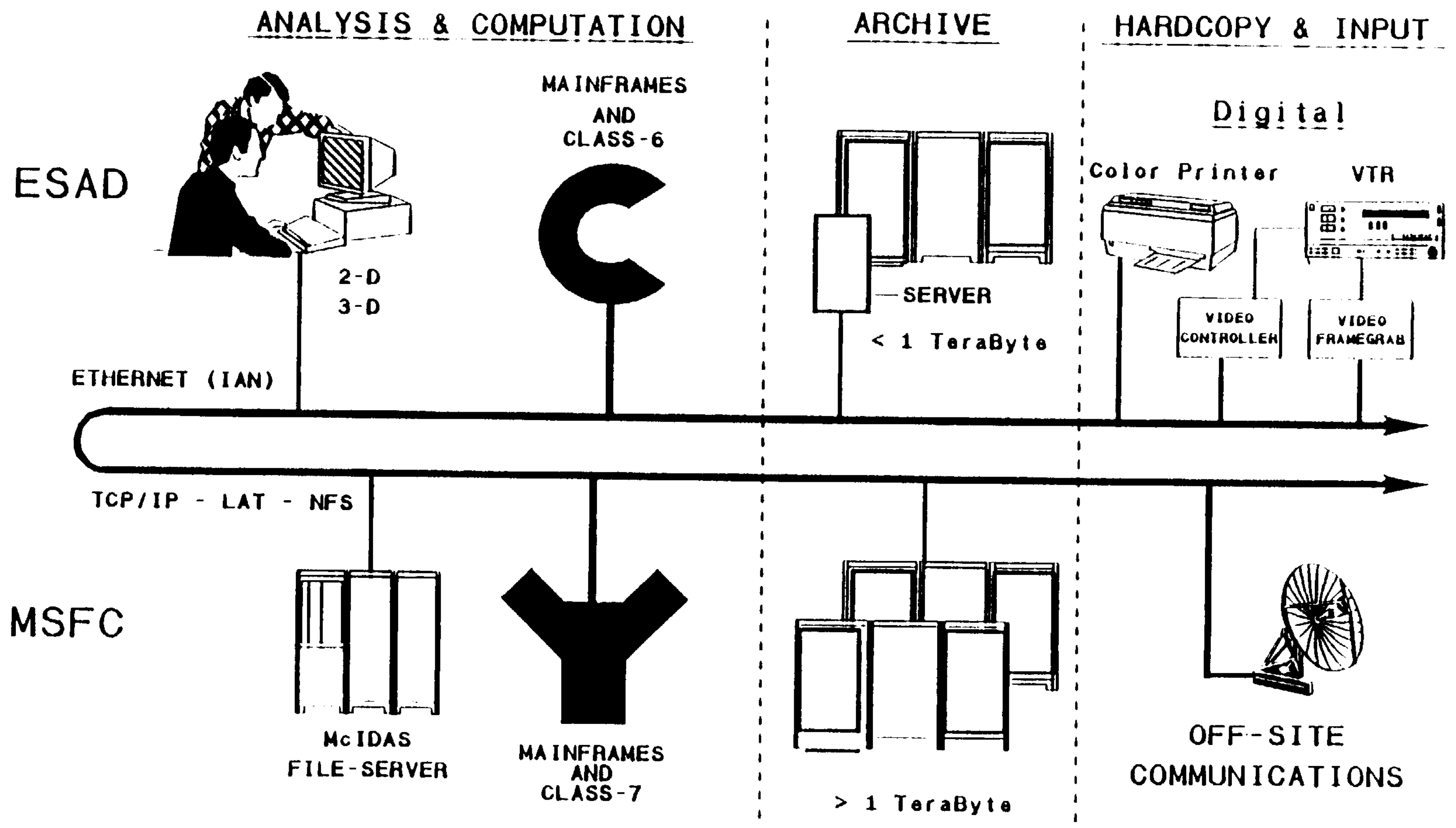
McIDAS, while it is a very functional and useful system, needs to be expandable. Two factors address the expandability, one is the data and the other is code modularity. Limitations in the current implementation include data file size limits, the data storage tends to be restricted to full word integers, difficulty and limitations in implementing new schema types, and the inability to address other data formats from other applications.

Code modularity is important to both portability and for ease of upgrades or algorithm changes. This would allow a developer outside of McIDAS to more easily incorporate McIDAS capabilities into his code, as well as incorporate state-of-the-art algorithms into the McIDAS environment.

CONCLUSION:

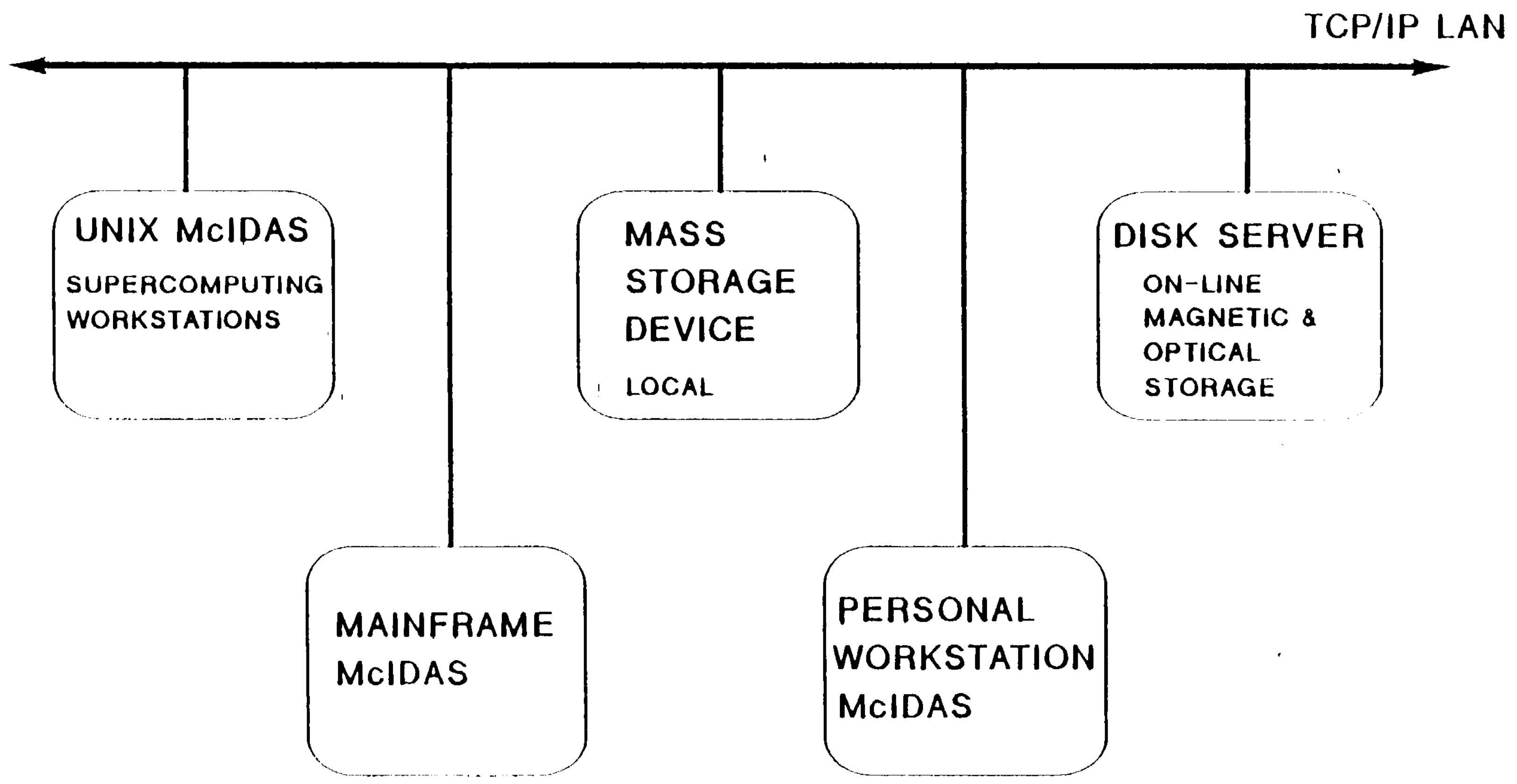
A UNIX port of McIDAS is essential to its continued growth, use, and functionality. Now is the time to implement a truly portable, expandable, and user friendly version of McIDAS. Portability necessitates adherence to the AT&T UNIX standard, TCP/IP communications protocols implemented at the applications layer, the use of standard GUI's, and programming with portability in mind. SSEC should strive for a device independent architecture, for the computational aspects and graphical display devices. It should be opened up for the purposes of distributed processing and distributed data management, as well as the integration of other applications software.

FUNCTIONAL HARDWARE ELEMENTS FOR AN INTERDISCIPLINARY EARTH SCIENCE AND APPLICATIONS DATA SYSTEM



Meyer /ES43/MSFC/NASA 5/10/90

CONCEPT FOR TCP/IP & NFS BASED McIDAS SYSTEM


TCP/IP LAN
UNIX McIDAS
SUPERCOMPUTING WORKSTATIONS
MAINFRAME McIDAS
MASS STORAGE DEVICE
LOCAL
PERSONAL WORKSTATION McIDAS
DISK SERVER
ON-LINE MAGNETIC & OPTICAL STORAGE

NASA/MSFC MEYER/10 4 89

THE IMPORTANCE OF DATA ABSTRACTION AND STANDARD DATA STRUCTURES IN VISUALIZATION DEVELOPMENT

Michael E. Botts, Ph.D.

Atmospheric Science and Remote Sensing Laboratory
Johnson Research Center
The University of Alabama in Huntsville
Huntsville, AL 35899

ABSTRACT

In recent years, many users and developers of scientific visualization tools have recognized and worked to solve problems associated with data format incompatibility. Data incompatibility can appear when one attempts to bring several different data sets into a single application program, or when one attempts to analyze a single data set using several different application programs. Much effort has gone into the development of common data formats for the storing and archiving of scientific data. These continuing efforts and the acceptance of one or two of these formats are very important to solving incompatibility issues associated with the storage and retrieval of scientific data sets.

However, once data are read from storage devices into application programs, there are presently no standards for describing how these data are to be stored and accessed within CPU memory. Although data incompatibility within CPU memory has not proved as detrimental as incompatibility between data storage formats, the adaptation of data abstraction techniques and the development of standard data structures (SDS) within application programs can provide many benefits for future visualization development. These benefits include simplification of format conversion routines, re-utilization of data input/output routines during program development, and perhaps most important, the ability of several visualization programs to access and manipulate the same data structure within shared memory.

BACKGROUND

Data Incompatibility and Standard Data Formats.

In the past, decisions regarding formats were generally made with little regard for standards. Those involved in the creation or archiving of data generally developed customized formats consisting of a header of information specific to that data, plus an array of data organized in a manner consistent with the order in which the data was obtained. Developers of application software, likewise, created additional formats customized for the informational needs and efficiency of the program.

submitted to IEEE Visualization '91, March 1991

These practices created a virtual flood of data storage formats resulting in the need for continual data conversion and the requirement for duplicate data sets in storage.

Several data formats have been developed in order to provide possible standards for data storage. Although rigid, custom formats tend to be more efficient in storage requirements and access rates, the acceptance of a standard format generally requires that it be flexible, complete, and self-describing. This may result in a format that is less efficient and possibly more cumbersome, but these trade-offs more than compensate for the problems created by the lack of standards.

The development of standard formats is of little importance if these formats are not accepted and utilized by both data archive facilities and application software developers. NASA, as well as many other large research organizations, has recognized the need for such standards, and has begun to evaluate potential format standards for archiving the flood of data anticipated with the advent of the Earth Observing System (EOS), "Mission to Planet Earth" [3]. Formats which appear to have gained some acceptance as standards within the scientific visualization community include HDF [4], CDF [1,9], netCDF [6,7], and BUFR [2,8]. The continued improvement of these formats, and the extensive use of these formats by data archivists and application developers alike, will greatly assist the development of visualization tools.

Data Abstraction.

On a parallel front, software developers are becoming increasingly aware of the importance of data abstraction in application programming. Data abstraction involves the encapsulation of data from the application program through the treatment of data as objects or coherent structures. When properly implemented, the use of data abstraction within software development eases software development and maintenance by (1) simplifying coding in large software efforts, (2) grouping data into more logical entities, (3) allowing the treatment of different data types as a single data entity, (4) localizing the dependence of the data structure within the program, and (5) isolating the data structure from any dependence with the application program. In essence, data structure and program structure are not tightly interwoven, allowing for modification to either data or program modules without affecting each other.

The treatment of data as an abstract entity within an application is an underlying premise of object-oriented programming, and is generally rigidly enforced in this paradigm. Data abstraction within the C programming language is not enforced, but can, and should, be implemented through the definition and use of data structures. The extensive use of data abstraction within application development would greatly enhance and simplify the creation of visualization tools. The use of data structures in C and the move to object-oriented programming languages is to be encouraged.

In additional, future development of visualization tools would benefit greatly by the development of standard data structures for scientific data. No standards exist for the structure of data after it has been accessed from disk and transferred into CPU memory. This paper proposes the initiation of efforts to establish such standard data structures.

BENEFITS OF STANDARD DATA STRUCTURES

The main benefits of establishing standard data structures are four fold: (1) the use of an intermediate standard data structure greatly simplifies present and future data format conversion; (2) the same modules for reading, writing, and accessing standard data structures can be re-utilized in any future application development; (3) data analysis and visualization modules designed for one application can readily be incorporated into new applications, and most importantly, (4) the use of standard data structures within different application modules allows for the creation of a true visualization environment in which several applications can access and act upon the same data residing in shared memory.

Data Format Conversion.

The use of intermediate data structures greatly simplifies the development of utilities for data format conversion. Without the use of intermediate data structures, conversion between *n* number of formats would require *n!/(n-2)!* modules. For example, to convert between just 12 formats would require 132 modules. In contrast, the use of intermediate data structures, as illustrated in Figure 1a, requires only two modules for each format, or *2n* total modules; one module to read into the structure and one to write out of the structure. Thus, for the example above, only 24 modules would be required to convert between 12 formats. One example of this principle is a collection of image file conversion utilities, Extended Portable Bitmap Toolkit (pbmplus) [5] .

More importantly, if the intermediate data structures are standard structures to be used within application programs, the same modules written for data format conversion can also be used to read data into and out of the application program (see Figure 1b). One obvious benefit of this is to reduce the amount of redundant programming which often accompanies application development. A second benefit is that once the above modules are written, any of the given data formats can be read into and written out of any future application program.

Re-Utilization of Modules.

If properly designed and implemented, the use of standard data structures should greatly reduce the development effort required within any given facility. As discussed above, modules developed for data format conversion can be used for all data I/O requirements for all future applications. In fact, development of these I/O modules should be the first step in implementing the use of standard data structures.

The next phase of implementation should be the development of generic routines for querying, accessing, and subsampling the data within the standard structures. Where appropriate, these routines could then be used repeatedly within new or modified application programs.

Finally, routines developed for performing specific analytical or visualization functions, such as image display, histogram generation, 3D surface rendering, texture mapping, etc., could be re-utilized within any future application development which adheres to the use of standard data structures. Eventually, a large suite of re-usable routines would exist for performing common and specific functions required within visualization. As new algorithms or more efficient implementations are developed, these could easily replace related routines within existing applications, without the need to sort out any entanglement of the existing application program and the data.

Sharing Data Between Applications.

Once several application programs are developed which adhere to the principle of standard data structures, the possibility of a truly integrated visualization environment can begin to be realized. Within this environment, all application programs read and write all file formats existing on storage devices. More significant, however, is the possibility for each active application to access the same data residing within shared CPU memory, as illustrated in Figure 1c.

For example, consider an application which generates a 3D surface and then texture maps an image onto this surface. Suppose that once the texture map is applied to the surface, the user realizes that the image requires some image processing which is not available within the active application program. Within an integrated visualization environment, the user could simply activate a separate image processing program, which could process the image data already residing within shared memory, and then replace that data with the newly processed data or return a pointer for the new data to the original application. With such use of interprocess communication and shared memory, no reading to nor writing from data storage would be required.

Without the use of standard data structures within these two applications, such a scenario would be difficult, if not impossible. The user would instead be required to load the image processing program, reread the image file into memory, process the image, write a new image file out to disk, and then reread the new image file into the first application. If the two programs do not read and write common data file formats, an additional step of converting the data between file formats would also be required.

IMPORTANT CONSIDERATIONS FOR STANDARD DATA STRUCTURES

It is recommended that standard data structures be developed for the following data types:

Images
n-dimensional Gridded Data
Text.

In addition, standard auxiliary data structures should also be defined for:

Color Palettes
Navigational Information
Processing History
Timing
Equations.

In order to be acceptable for a large number of applications and users, a standard data structure should have the following minimal characteristics:

flexible: in order to meet the needs for efficiency and to meet hardware or windowing requirements, the data structure should be flexible to data ordering and data type;

self-descriptive: the data structure should contain all data required for understanding and accessing the data within the structure, and should provide pointers to other required data structures (e.g. navigational structures, color palettes, etc.);

complete but compact: all required information should be available, but consideration should be given to the most compact means of storing this data; unions should be employed where applicable, to minimize unused storage requirements;

simple: components within the data structure should be easily recognized with simple, short names; complex hierarchy and complex definitions within the data structure should be avoided;

sequential and branching: pointers to structures should be provided allowing sequences and trees of structures to be defined;

expandable: pointers should be provided to user-defined auxiliary structures to allow for additional information of user or data specific requirements;

A beta test example for a common data structure for image data is provided in Figure 2a. In this structure, most of the required descriptive data is included at the beginning. Define statements for specific properties are included in Figure 2b. This data structure allows flexibility in the ordering of the data, including the relative order of RGBA components, the scan order, and the direction of scan. These ordering defines could be determined by either the original ordering of the data file format, or by the ordering requirements of the application program.

In this data structure, the actual buffer for storing data is contained within a union. This allows flexibility for storing all types of data, without requiring more storage space than required for that particular type. Memory is not pre-allocated within the data structure, but is instead allocated as needed by a utility module which access the dimensional and typing information from the data structure. A "bad_flag" variable is also provided within the structure allowing for definition of a bad data or missing data indicator.

Finally, the example structure provides pointers to auxiliary or sequential data structures which may or may not be required for a particular application. The navigation structure would provide information regarding type of projection, coordinate system used, the type of data navigation employed, and the coordinates required for navigation. Similarly, the color map structure would provide, when required, the related look-up-table (LUT) or color map information for the data. The pointers, *next* and *previous*, allow sequences of images to be defined within the structure, whereas the pointers, *previous*, *left*, and *right*, allow definition of binary trees of images. Finally, the pointers, *aux1* and *aux2*, could provide access to user-defined structures for application or data specific information.

The CDS_IMAGE data structure, as well as common data structures for other data types, are being developed at NASA Marshall Space Flight Center's (MSFC) Earth Science and Application Division (ESAD). These data structures will be utilized for all new in-house visualization development within ESAD, and will be tested for flexibility, ease of use, and adaptability.

CONCLUSIONS

The development of standard data file formats is important to halting and reducing the growing problems associated with data format incompatibility. However, no standards exist for defining data once it is read into CPU memory from storage. The visualization community should begin efforts to establish standard data structures for scientific data. Such standards would essentially render data format incompatibility issues obsolete, would greatly aid re-utilization of code, thereby reducing redundant programming, and would provide the opportunity for several application programs to access and manipulate the same data residing within shared memory. This, in essence, would allow the development of a truly integrated visualization environment, in which all application tools can act harmoniously as one.

Certainly, the principles of common data structures outlined in this paper should be adhered to within any facility undertaking application development of any kind. It is hoped that a concerted effort can be initiated for the purpose of establishing national and international standards for data structures within the visualization community.

ACKNOWLEDGEMENTS

The author has benefitted from the continuous discussions with Paul J. Meyer and John V. Parker at NASA Marshall Space Flight Center and from input of Alan Barclay at the University of Alabama in Huntsville. This work is supported by NASA under contract NAS8-37139.

REFERENCES

[1] K. Blackwell, "A Primer for the Common Data Format", CDF version 2.0, NSSDC/STX, Goddard Space Flight Center, February 1991.

[2] European Centre for Medium-Range Weather Forecasts (ECMWF), "Binary Universal Form for Data Representation: FM 94 BUFR Collected Papers and Specification", February 1988.

[3] A. Fleig, "Data Exchange Format Standards", *presented at the V0 DAAC System Engineer Meeting*, November 1990.

[4] National Center for Supercomputing Applications (NCSA), "NCSA HDF version 3.1", via anonymous FTP at ftp.ncsa.uiuc.edu (128.174.20.50), July 1990.

[5] J. Poskanzer, "pbmplus/README" via anonymous FTP at expo.lcs.mit.edu (18.30.0.212), December 1990.

[6] R.K. Rew, "netCDF User's Guide", *NCAR Technical Note, NCAR/TN-334+1A*, December 1990.

[7] R.K. Rew and G. Davis, "NetCDF: An Interface for Scientific Data Access", *IEEE Computer Graphics and Applications*, pp. 76-82, July 1990.

[8] J.D. Stackpole, "GRIB and BUFR: The Only Codes You Will Ever Need", Preprints, *AMS 6th Conference on Interactive and Processing Systems for Meteorology, Oceanography, and Hydrology*, Anaheim, CA, pp. 23-30, 1990.

[9] L.A. Treinish, "The Role of Data Management in Discipline-Independent Data Visualization", *SPIE/SPSE Symposium on Electronic Imaging Science & Technology*, February 1990.

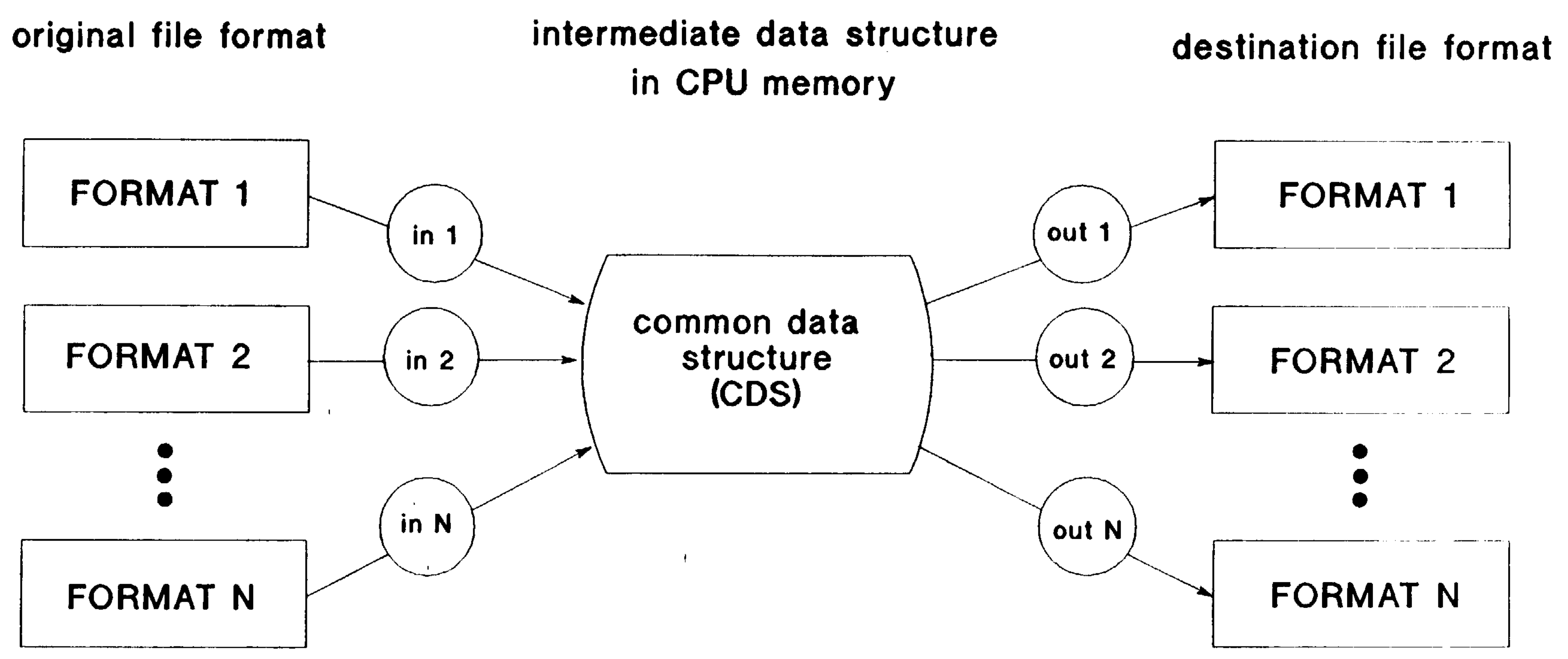


Figure 1a. Use of Intermediate Common Data Structure for Data File Format Conversion

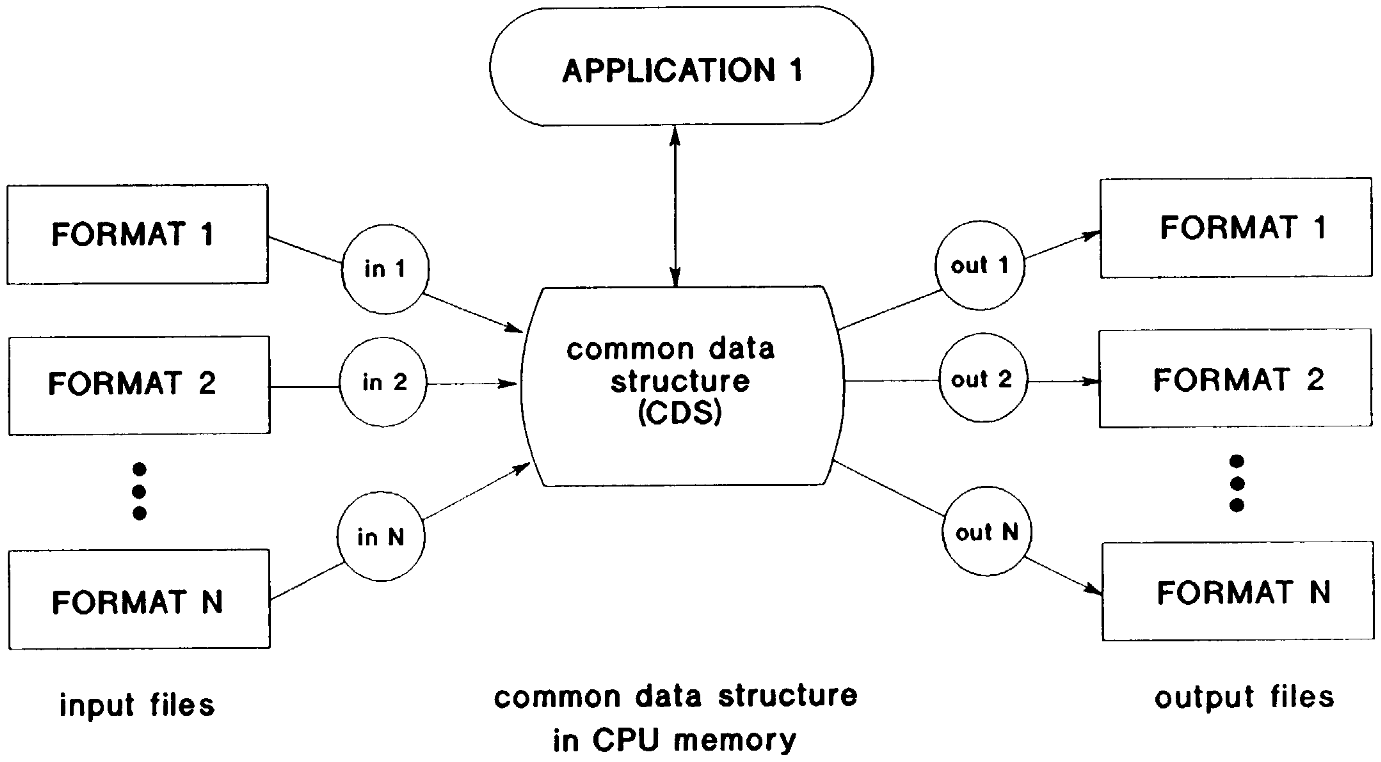


Figure 1b. Use of Common Data Structure for Application Program Input/Output

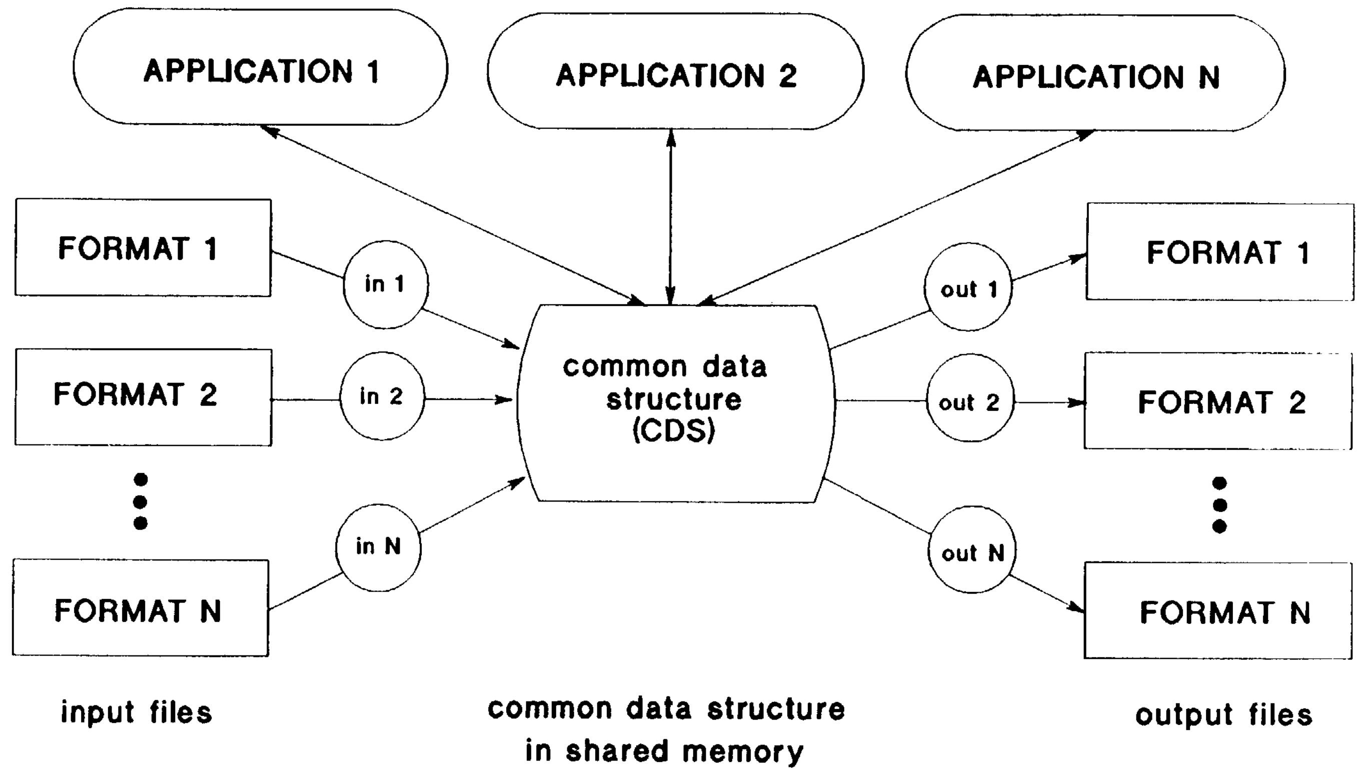


Figure 1c. Use of Common Data Structure in Shared Memory for Multiple Applications

```
/* define CDS_IMAGE structure */

typedef struct cds_image {

        char            file [NAME_MAX];        /* file name                            */
        int             fd;                     /* file descriptor (open)               */
        FILE            *fp;                    /* file descriptor (fopen)              */

        int             width;                  /* image width                          */
        int             height;                 /* image length                         */
        int             planes;                 /* 1 = bw, 3 = rgb, 4 = rgba            */
        int             bits;                   /* depth (1, 8, 24, 32 bits)            */
        long            size;                   /* image size in BYTES                  */
        int             seq;                    /* number of images in file             */

        Ulong           min;                    /* minimum pixel value                  */
        Ulong           max;                    /* maximum pixel value                  */

        Ulong           magic;                  /* format magic number                  */
        int             i_type;                 /* type of image: RGB,CM                */
        int             compress;               /* compression type: none, RLE          */

        int             format;                 /* format of data (SLB, SBL, etc)       */
        int             scan_order;             /* scan order (SCAN_DOWN/UP)            */
        int             rgb_order;              /* arrangement of RGBA bands            */

        int             datatype;               /* see data types above                 */
        union {
                Uchar   *ucbuf;                 /* UCHAR data   */
                char    *cbuf;                  /* CHAR data    */
                Ushort  *usbuf;                 /* USHORT data  */
                short   *sbuf;                  /* SHORT data   */
                Uint    *uibuf;                 /* UINT data    */
                int     *ibuf;                  /* INT data     */
                Ulong   *ulbuf;                 /* ULONG data   */
                long    *lbuf;                  /* LONG data    */
                float   *fbuf;                  /* FLOAT data   */
                double  *dbuf;                  /* DOUBLE data  */
        } buf;                                  /* union for actual data                */
        union {
                Uchar   uc_flag;                /* UCHAR data   */
                char    c_flag;                 /* CHAR data    */
                Ushort  us_flag;                /* USHORT data  */
                short   s_flag;                 /* SHORT data   */
                Uint    ui_flag;                /* UINT data    */
                int     i_flag;                 /* INT data     */
                Ulong   ul_flag;                /* ULONG data   */
                long    l_flag;                 /* LONG data    */
                float   f_flag;                 /* FLOAT data   */
                double  d_flag;                 /* DOUBLE data  */
        } flag;                                 /* bad data flag                        */

    CDS_NAV             *nav;                   /* navigation structure                 */
    CDS_AUX             *aux1;                  /* auxiliary info structure             */
    CDS_AUX             *aux2; / CDS-CMAP       /* color map structure                  */
    CDS_IMAGE           *previous;              /* pointer to previous image            */
    CDS_IMAGE           *next;                  /* pointer to next image                */
    CDS_IMAGE           *left;                  /* pointer to left branch               */
    CDS_IMAGE           *right;                 /* pointer to right branch              */

} CDS_IMAGE;
```

FIGURE 2a. Common Data Structure for Image Data (CDS_IMAGE).

```
/* define data types */

#define     UCHAR         0     /* unsigned char = Uchar     */
#define     CHAR          1     /* char                      */
#define     USHORT        2     /* unsigned short = Ushort   */
#define     SHORT         3     /* short                     */
#define     UINT          4     /* unsigned int = Uint       */
#define     INT           5     /* int                       */
#define     ULONG         6     /* unsigned long = Ulong     */
#define     LONG          7     /* long                      */
#define     FLOAT         8     /* float                     */
#define     DOUBLE        9     /* double                    */


/* define format arrangement of pixel array */

#define     SLB           0     /* Scan-Line-Band format     */
#define     SBL           1     /* Scan-Band-Line format     */
#define     BSL           2     /* Band-Scan-Line format     */


/* define scan line order */

#define     SCAN_DOWN     0     /* scans from top to bottom  */
#define     SCAN_UP       1     /* scans from bottom to top  */


/* define rgb (interlace) order */

#define     RGBA          0     /* red, green, blue, alpha   */
#define     ABGR          1     /* alpha, blue, green, red   */
#define     ARGB          2     /* alpha, red, green, blue   */
#define     RGB           3     /* red, green, blue; no alpha */


/* define compression type */

#define     NO_COMPRESS   0     /* No compression            */
#define     RLE           1     /* Run-Length-Encoding       */
#define     HUFFMAN       2     /* Huffman compression       */
#define     LEMPELZIV     3     /* LempelZiv compression     */
```

FIGURE 2b. Defines for CDS_IMAGE parameters.

RECOMMENDATIONS FOR ESTABLISHING A DATA ANALYSIS AND VISUALIZATION ENVIRONMENT (DAVE) AT THE EARTH SCIENCE AND APPLICATIONS DIVISION, NASA MARSHALL SPACE FLIGHT CENTER

**Report to the
Earth Science and Applications Division (ESAD),
NASA Marshall Space Flight Center**

Dr. Michael E. Botts

January 10, 1991

ACKNOWLEDGEMENTS

Many of the ideas presented in this report were developed through numerous discussions with scientists throughout the Earth Science and Applications Division at NASA Marshall Space Flight Center. Frank and open discussions of their challenges and frustrations have been very helpful. Of particular importance have been the continuous discussions with Paul Meyer, who has been involved with the data analysis and visualization challenges at ESAD for several years. Many of the ideas presented here are either directly, or indirectly, his ideas, or were created in the midst of stimulating discussions with Paul. I hope that this report will prove helpful in providing an overview of the challenges facing ESAD and in consolidating many helpful ideas into a workable plan for meeting these challenges. This work has been performed under NASA contract NAS8-37139.

TABLE OF CONTENTS

TABLE OF ACRONYMS

ABFM -	Airborne Field Mill
AIS -	Airborne Imaging Spectrometer
apE -	applications programming Environment developed at the Ohio Supercomputer Center
AVIRIS -	Airborne Visible and Infrared Imaging Spectrometer
AVS -	Applicaion Visualization System developed at Stardent Computers
CAD -	Computer-Aided Design
CAT -	Computer-Aided Tomography
CFD -	Computational Fluid Dynamics
CPU RAM -	dynamic Read Access Memory residing within a computer
DAVE -	Data Analysis and Visualization Environment
ELAS -	image processing software developed at NASA Stennis
EOS -	NASA's Earth Observing System
ESAD -	Earth Science and Applications Division at NASA Marshall Space Flight Center
FAST -	Computational Fluid Dynamics software developed by NASA Ames Research Center
GIS -	Geographical Information Systems
GL -	Graphics Library developed by Silicon Graphics
GLOBE -	NASA's Global Backscattering Experiment
GRASS -	Geographical Resource Analysis Support System developed by the U.S. Army Engineering Research Laboratory
GUI -	Graphical User Interface
HIRIS -	High Resolution Imaging Spectrometer

IDIMS -	Image Database/Image Management System developed at NASA Marshall
I/O -	Input/Output
I^2S -	International Imaging Systems
JPL -	Jet Propulsion Laboratory, Pasadena, CA
LINKWIND-	Linked Windows software developed at JPL
LUT -	look-up-table for colors on computers
McIDAS -	Man-Computer Interactive Data Access System developed at SSEC
MSFC -	NASA Marshall Space Flight Center
NASA -	National Aeronautics and Space Administration
NCAR -	National Center for Atmospheric Research
NCSA -	National Center for Supercomputer Applications
PC -	Personal Computer
SGI -	Silicon Graphics Inc.
SPAM -	Spectral Analysis Manager developed at JPL
SSEC -	Space Science and Engineering Center at the University of Wisconsin
TAE+ -	Transportable Application Environment developed by NASA
TCP/IP -	Transmission Control Protocol / Internet Protocol
UNIX -	standard operating system developed by AT&T
vis5D -	volume visualization software developed at SSEC
1D -	one dimensional
2D -	two dimensional
3D -	three dimensional

OVERVIEW

The decade of the '90s will see an explosion of data available to Earth system scientists in the Earth Science and Applications Division (ESAD). Already, scientists within a given project, are being overwhelmed with more data than they are able to sufficiently integrate and analyze. With the advent of the Earth Observing System (EOS) "Mission to Planet Earth", scheduled for the middle of this decade, and with the explosion of data being generated by numerical models on todays supercomputers, the amount of data available to Earth system scientists is expected to increase one thousand fold by the end of the decade.

Data in present and future applications can typically be characterized as being large, multidimensional, multiband, multiparameter, multisource, multiformat, multicoordinate, and multiperiod. The problem facing the scientist of this decade is how to archive, retrieve, integrate, and analyze such data. Present methods at ESAD for database management and data analysis are insufficient for dealing with such large and diverse databases. Available methods must be improved, and new methods developed, to deal with the present and future data analysis needs.

The immensity and diversity of these databases necessitates the application of scientific data visualization methods in order to adequately verify and analyze these data. Although scientific data visualization incorporates knowledge and developments from several well established fields, including image processing, three-dimensional (3D) computer graphics, computer-aided design (CAD), volume rendering, geographical information systems (GIS), and graphical user interfaces (GUI), the adaptation of these developments into a visualization tool for the scientist is still in its infancy. Presently, no single tool is available that can handle all of the visualization needs of the Earth systems scientist; yet, scientists at ESAD and other facilities are becoming increasingly flooded with larger and more complex datasets which require visualization.

To cope with this dilemma, scientists are presently forced to either: (1) adapt their data to fit into visualization tools which are inappropriate for analyzing the data, generally with poor results; (2) expand or modify their existing visualization tools to handle the data requirements, a process which is costly in time and finances and which is generally in contrast to a scientist's desires or skills; (3) apply several independent visualization tools to the data in order to analyze different aspects of the data, or (4) give up using present visualization tools in lieu of more traditional, but inadequate data analysis tools.

The third option probably provides the most effective means of analyzing present data using state-of-the-art visualization methods, and is typically the solution in use at most large research facilities today. However, besides requiring the scientist to learn several different hardware and software systems, the necessity of employing several different visualization packages results in serious problems resulting from the lack of integration between these packages. Data must generally be painstakingly converted to different formats, and often transported by tapes or over networks, in order to be used by visualization systems. The lack of integration between visualization tools, many of which are very much established at a particular facility, is a major challenge facing Earth systems research facilities in this decade.

In addition, many application packages in use at ESAD and other facilities, have very poor user interfaces, resulting in inefficient use or even abandonment by many scientist who might otherwise benefit from these tools.

The Earth Science and Applications Division (ESAD) at NASA Marshall Space Flight Center will be at the forefront of the efforts to ingest and analyze Earth systems science data. In fact, the dataset requirements at ESAD are typical requirements of the general Earth systems science community. Thus, the visualization environment at ESAD serves as an excellent example of the challenges facing other Earth system science facilities in this decade.

The development of the Man-computer Interactive Data Access System (McIDAS) at the University of Wisconsin-Madison Space Science and Engineering Center (SSEC), under the support of MSFC, has been a major advancement in the field of data visualization within the atmospheric science community. NASA MSFC/ESAD has made a major commitment in time and finances to the development of McIDAS. That commitment should not be wasted. In fact, continued improvement and expansion of McIDAS, within the IBM mainframe, PC, and graphic supercomputer environments, is vital to the visualization needs of ESAD at MSFC.

However, while McIDAS is a vital part of present and future visualization needs at ESAD, it does not presently meet the total visualization needs at ESAD, nor does McIDAS provide the appropriate environment within which all future visualization needs can be met. While a fully integrated solution to all needs is a desirable ideal, attempting to handle all present and future ESAD visualization needs solely within McIDAS, has and will continue to result in the inability of ESAD to satisfactorily meet these needs.

Scientists at ESAD are, out of necessity, increasingly relying on the use of visualization software which has been custom developed or is readily available from sources outside of the McIDAS arena. These off-the-shelf solutions have been successful in solving some immediate visualization needs. Furthermore, non-McIDAS visualization tools will become increasingly vital to the data analysis needs of ESAD. However, the lack of integration between these important applications packages at ESAD has resulted in inefficient, and at times aborted, use of the present visualization environment within the division. Without the development of a highly integrated visualization environment at facilities such as ESAD, scientists at these facilities will increasingly experience frustration and ineffectiveness at their ability to analyze present and future data.

It is proposed, that specific application packages, such as McIDAS, become important components of a broader Data Analysis and Visualization Environment (DAVE), which will incorporate other off-the-shelf visualization tools, as well as new tools developed to meet specific needs at ESAD and other facilities. This environment would serve to integrate several powerful, state-of-the-art visualization tools in the fields of image processing, 3D computer graphics, animation, volume rendering, and GIS, as well as database management. These tools would be bound in a highly integrated environment by common data structures, and a portable, user-friendly GUI, making the integration appear seamless to the user. Furthermore, the user would have the option to run specific applications on a single workstation, or as a distributed process across several computers on a network. The author feels that DAVE would greatly benefit the scientists at ESAD, as well as other scientist throughout the Earth systems science community.

In the following report, the deficiencies of the present analysis and visualization environment at ESAD are discussed in more detail. The author then proposes the establishment of an initiative for the development of DAVE, a highly integrated, yet flexible and expandable, scientific visualization environment for ESAD and the Earth Systems science community. Included in this initiative are the following key points:

(1) This initiative does not advocate a major effort toward the development of a new visualization package (Ala McIDAS), but advocates instead, a much smaller effort designed to integrate applications packages or modules which already exist, or which might be developed under future project-dependent efforts;

(2) The main goal of this initiative is to establish a team which will (a) recommend the standards under which DAVE should be developed, (b) oversee and assist the in-house, and funded, development of analysis and visualization tools for ESAD, for the purpose of integrating these tools into the environment, (c) archive and document existing and future software development in order to minimize redundant programming, and (d) assist in the distribution and implementation of DAVE within other governmental, educational, and industrial institutions.

(3) For the individual scientist and the division as a whole, getting the immediate job done is, by far, the major priority; developing a long-term integrated computing environment is of secondary importance. However, in many cases, short-term development could, and should, progress with long-term goals in mind.

(4) This report is not a proposal for a major funding effort. Many of the tasks outlined below could be implemented within existing contracts. However, it is anticipated that future funding might be requested by various contractors with respect to this initiative. The DAVE project team would be responsible for establishing the guidelines and for monitoring such development efforts.

BACKGROUND: DATA VISUALIZATION ISSUES

DATA CHARACTERISTICS

Typical data sets for a given project at ESAD can be characterized as large, multidimensional, multiband, multiparameter, multisource, multiformat, multicoordinate, and multiperiod. These characteristics and the demands that such complex data sets place on a computing environment, are discussed in detail in *Appendix A - Data Characteristics.*

For example, a particular project such as NASA's Global Backscattering Experiment (GLOBE) might incorporate various data sets from airborne, satellite, and ground-based platforms. Each of these platforms can be viewed as having different local coordinate systems which are translated through space and time relative to the other platforms. In addition, data sets from a single platform might be derived from several instruments, or sources, on that platform. For example, instruments scheduled to fly aboard GLOBE aircraft include three Lidars, several particle counters, and a wide range of instruments for measuring temperature, pressure, and relative humidity along the aircraft path. The intent of the GLOBE team is to integrate such data with data derived from satellite imagery and ground-based radar. These data range from 1D scalar to 2D and 3D scalar and vector data. In addition, these data have different sampling times and different periods of sampling, ranging from 1 second for various airborne instruments to 5 minutes for ground-based radar data. The major challenges of the analysis of such data are two-fold: (1) to develop means of relating such diverse data in both time and 3D space, and (2) to develop the means of analyzing and interrogating the data in such a way as to obtain insight into the events being studied.

The size, complexity, and diversity of data in Earth systems science, necessitates the application of data visualization tools in order to more effectively validate, correlate, and analyze these databases. The same characteristics of the data also place much greater demands on the hardware and software which comprise the visualization tools.

VISUALIZATION TOOLS

Data visualization is rapidly proving the solution to the scientists' needs for analyzing and understanding large, complex data sets. Although still in its infancy, visualization is actually the unification of several traditionally independent, well-developed disciplines. These include image processing, 3D computer graphics, animation, GIS, volume rendering, and CAD. An informative discussion on the power and limitations of each of these fields is presented in *Appendix B - Components of Computer Visualization.* The present challenge for the developer and user of visualization environments is the integration of these individual tools into a single tool or environment which will allow the analysis and visualization of the types of complex data sets discussed earlier.

Analytical and visualization tools required for handling such complex data sets are in their embryonic stage. At present, no single visualization tool is capable of meeting all the demands of a complex project such as GLOBE. Short term solutions to such analytical needs will surely require the parallel use of several visualization tools, possibly running on various hardware platforms.

The ideal long-term solution involves the development of an integrated visualization environment, which is computationally and graphically powerful, flexible to different data types and different computational platforms, interactive and user-friendly to the investigative scientist, and employs state-of-the-art methods in the various fields of image processing, 3D computer graphics, GIS, animation, and database management. With a properly designed environment, the complexities of integrating the above datasets and passing them between application tools, would become transparent to the user. The Earth system scientist could then concentrate on verifying and analyzing the data, and not on the difficulties of managing and integrating the data.

GRAPHICAL USER INTERFACES (GUI)

The importance of GUI's is often overlooked in application programming or large program development. The GUI is the vital communication link between the user and the application software. They are particularly important for inherently graphical applications, such as visualization. A GUI generally involves the use of multiple windows with user input handled through the use of a mouse, in some cases combined with an option for command line input. In addition to providing windows for displaying images, graphics, or text, the GUI might consist of graphical widgets for file selection, rotation of viewing angles, histogram viewing and manipulation, look-up-table (LUT) editing, and input of parameter values through the use of sliders or dials.

A well designed, user-friendly GUI offers many advantages for the user of application software: (a) it can greatly decrease the time required to learn the application package; (b) it improves user efficiency for casual and power users alike by removing the need to remember and type many complicated command line symbols and parameter values; (c) it can decrease user input errors resulting from typing mistakes; and (d) it can provide a familiar user interface when the same GUI is used for different application programs. In addition, the use of a well designed, standard, portable GUI during program development serves three other very important functions: (a) it greatly speeds development time, since up to 75% of applications programming time is often spent developing the user interface; (b) it greatly increases portability of the applications program, since graphical interfacing is the major source of hardware incompatibility between programs; and (c) it can allow the programmer to transparently combine two or more applications programs into a single visualization program. Furthermore, some GUI's also allow distributed processing and distributed data management, whereby different programs and databases residing on separate hardware devices can all act as a single coordinated processing environment.

It is highly recommended that the frontend to any visualization environment employ a standard and portable graphics user interface, preferably one based on the X-Windows Version 11 Release 4 standard. Examples of GUI tools are Motif, Open Look, Presentation Manager, NASA's TAE+, NASA AMES' FAST ToolKit, and aPE from the Ohio Supercomputer Center. An ideal GUI for visualization is one that is user friendly, is based on existing computer standards, is highly portable to existing and new platforms, is open to expansion (e.g. new widgets, etc.), allows distributed graphics processing and distributed database management, and is easily incorporated into new and existing application programs. Since the GUI is the constant link to the user and the "glue" that holds a visualization environment together, it is important that available GUI's be thoroughly evaluated to determine the best choice for a given visualization environment. Once selected, the GUI should serve as the outer shell for all new application development and for the incorporation of existing programs into the visualization environment.

CURRENT VISUALIZATION ENVIRONMENT AT ESAD

As an example of the challenges facing Earth systems science facilities in the 1990's, current and future visualization challenges at ESAD are examined below. The current visualization environment at ESAD consists of the following systems: McIDAS (mainframe and PC based), International Imaging Systems (I^2S) image processing system, a prototype 3D visualization package "vis5D", Stardent's proprietary Advanced Visualization System (AVS) visualization environment, and various user specific application packages primarily developed for the PC and workstation environments. More recently incorporated for evaluation (November 1990) are NASA Stennis' ELAS image processing software, JPL linkwinds linked windows package, NASA Ames' FAST computational fluid flow visualization software, and the aPE visualization environment from the Ohio Supercomputer Center.

Each of these systems provides important computing and visualization capabilities to ESAD. However, their effectiveness in solving the needs of the scientists have been hampered by several factors:

(1) **Lack of Integration:** There is often the need to enhance the capabilities of one visualization system with complementary capabilities from another. For example, the image processing capabilities of McIDAS are rudimentary, at best; the need to utilize the image processing power of ELAS or I^2S for McIDAS images is one that is often recognized at ESAD, but one that is rarely satisfied.

The reason is generally traced back to a lack of integration. Data generally must be passed back and forth between different application software and different hardware. This requires that data first be saved in a file, then converted into a data structure compatible with the second application package, passed across the network, then converted again to account for byte-swapping requirements between different hardware, and then brought into the second application package for analysis. If the data then needs to be returned to the first application package for reuse, the above process must be repeated in reverse. In addition to requiring the scientist to learn two very different application packages, the above process is very time consuming and annoying to the scientist, who should not be required to deal with the details of data and system incompatibility.

(2) **Lack of User-Friendly Interfaces:** The lack of a user-friendly interface for a software package generally manifests itself in many ways: (a) the time required to learn the rudimentary operations of the system is measured in several days to weeks, instead of one to a few hours for user-friendly systems; (b) there is a constant need to refer back to manuals for command names, parameter values, command line structure, and command sequences for casual, and even power, users; (c) there are an abundance of errors during

analysis associated with typing mistakes or improper order of commands or parameter values; (d) too much of the users' concentration is on the operation of the system, rather than on the data and its analysis; (e) expert status on the system may require six months to a year to achieve and is accomplished only by a few users; and (f) there is a large degree of frustration and apprehension experienced by potential first-time and casual users, often resulting in a lack of use of the system by many who could otherwise benefit from its power.

On the other hand, user-friendly systems generally employ: (a) pop-up menus which guide the user to the appropriate commands and options; (b) various widgets that query the user for appropriate parameter values; (c) on-screen help menus that are easy to search and display; (d) simple mouse-driven means for indicating regions of interest, changing image viewing parameters, or drawing or labeling in overlay planes; and (e) options for bypassing menus and widgets by entering command line input. A few computer users fear that user-friendly interfaces will be restrictive and less flexible for power users. However, a well designed user-friendly interface will, in most cases, increase power users' productivity, as well as the productivity of the casual or first-time user.

Several of the visualization tools at ESAD, including McIDAS, ELAS, and I^2S, are command line driven and lack a user-friendly interface. Others, such as vis5D and Xsect, have poorly defined interfaces. No user interface exist for the transfer of data between different application packages. The lack of well designed, user-friendly interfaces for the visualization environment at ESAD is resulting in inefficient use of the visualization tools available, and even worse, a total lack of use by many scientists who might otherwise benefit from these tools.

(3) **Lack of Coordination and Archiving of Development:** Most small software tools at ESAD are developed with only short-term, specialized application in mind; however, many of these tools could be useful for many applications if properly developed. Furthermore, those software tools which have been developed for general use are often poorly documented and poorly archived. Potential users of these tools are either unaware that they exist, or they are not able to use or modify these programs for their own use due to poor documentation within the program. All of these conditions result in redundant programming (i.e. "reinventing of the wheel"), as well as ineffective use of existing tools.

(4) **Lack of Simple Data Output to Video or Print:** The need to communicate the results of data analyses to other scientists, management, or the public, requires the capability to output individual images or sequences of images to print or video. Although these capabilities exist to some degree at ESAD, the use of these capabilities is presently cumbersome and tedious. These difficulties result from the lack of a general integrated capability, which would allow output to print or video to easily accomplished from within any

visualization tools at ESAD. The video requirements at ESAD are discussed in more detail in the report, *Summary Report to Division on Video Capabilities and Needs* (Appendix C).

The combination of these four factors is resulting in an inefficient use of the visualization tools at ESAD. The scientists should not be required to spend time and energy worrying about the complexities of software integration, user-interfaces, or interfacing with output devices. When the scientists must constantly build, rebuild, or relearn their tools before applying them, there is often very little time, energy, or enthusiasm left for using these tools for the purpose for which they were developed. In many cases, frustration and the lack of time required to cope with these details have resulted in limited use of these tools by the scientists.

Clearly, if ESAD is to meet the demands of verifying and analyzing the complex data sets facing the earth systems scientist in the 1990's, a visualization environment must be developed which is user-friendly, highly integrated, portable, flexible, and more diverse. In addition, this environment must be brought up-to-date with regard to user interfaces, data structures, state-of-the-art algorithms, computer standards, and optimization of newer compilers and processors. This environment should be open and flexible enough to easily adapt to new application packages and to changes in the state-of-the-art within many different fields.

McIDAS has been, and will remain, a very vital part of this environment. However, McIDAS does not meet all the present visualization needs of ESAD. The McIDAS system is weak in the fields of image processing, 3D graphics, 3D volume interrogation and visualization, GIS, and graphical user interfaces. Considering the very rapid advances to the state-of-the-art within each of these fields, and considering the limited financial and manpower resources for program development at the University of Wisconsin SSEC, it is unreasonable to expect that McIDAS can meet all of these needs in the future. Furthermore, although McIDAS was originally developed to be an operating system in itself, McIDAS is neither open, flexible, nor portable enough to serve as a visualization environment under which other visualization tools could function. It is therefore proposed that McIDAS be incorporated as a single vital component in a more flexible visualization environment, to be described below. The McIDAS port to UNIX is vital to this effort. The paper *Suggestions for a Unix Based McIDAS* (Appendix D) offers suggestions to consider during a porting of McIDAS to Unix.

RECOMMENDATIONS FOR THE DEVELOPMENT OF A DATA ANALYSIS AND VISUALIZATION ENVIRONMENT (DAVE)

From the discussions above, it is apparent that no single visualization application program can solve all the visualization needs of the Earth systems scientist. Within any single discipline, many individual visualization tools exists, each providing valuable tools for the scientist in that field. In addition, Earth systems science incorporates several traditionally independent disciplines, including geology, atmospheric sciences, biology, environmental sciences, and chemistry, to name a few. Powerful visualization tools have been developed which are specific to each of these disciplines; it would be unrealistic and detrimental to attempt to replace these tools with a single visualization program that would be expected to solve all visualization problems for each of these disciplines.

Furthermore, the scope of visualization is very broad, encompassing as discussed above, the fields of image processing, 3D computer graphics, GIS, volume rendering, GUI, and database management. The state-of-the-art in each of these fields is expanding too rapidly for any single program development team to be able to continually incorporate up-to-date technology into such a program.

THE CONCEPT OF DAVE

The ideal long-term solution is a visualization environment which will allow integration of several readily available state-of-the-art application packages into a "single" visualization tool. Instead of relying on a single source for new developments, a research facility can then rely on the entire computer and scientific community as a source for new ideas and technologies.

It is proposed that ESAD establish an initiative for the purpose of developing a Data Analysis and Visualization Environment (DAVE) to meet the specific needs of ESAD and the Earth systems science community in support of NASA's EOS Mission to Planet Earth. The proposed visualization environment, is envisioned as one which is:

(1) **user-friendly** with a common graphical user interface (GUI) for all applications which makes extensive use of pop-up menus, widgets, and help menus, and renders the complexities of integration invisible to the user;

(2) **flexible** to different data types and demands;

(3) **open** and **expandable** in order to allow easy incorporation of user-specific applications, or improvements to application tools in the future;

(4) **portable** to a large number of computer platforms;

(5) capable of **distributed processing** and **distributed database management** over ethernet or hyperchannels;

(6) **optimized** for todays' compilers and processors;

A possible scenario illustrating the use of such an integrated visualization environment is as follows:

(1) user types command to begin execution of the visualization environment "dave";

(2) environment opens a window for messages, a separate window for the user to enter command line input if desired, and pops up a menu showing initial options (for example: "data", "McIDAS", "ELAS", etc.);

(3) user clicks mouse on option for "data" (or types command in command line window, if preferred), which pops up submenu offering choices "search", "copy", "load" or "exit";

(4) user clicks "search", which loads the database management application (IDIMS ?), with a common GUI, allowing the user to search and locate a sequence of images from a particular satellite sensor for a particular day and region;

(5) user has option to click "copy" to transfer a copy to local disk drive or another storage device, or "load" to load the data into shared memory (i.e. within CPU RAM); user clicks "load" which automatically applies an appropriate data filter and loads data into shared memory in the appropriate data structure;

(6) user clicks "exit" to leave and unload database manager, and returns to initial menu offering choices for data processing;

(7) user clicks "McIDAS" to load and initialize the McIDAS application package which has been given the necessary memory locations and other necessary information regarding data stored in shared memory; the McIDAS application program is now fronted by the common GUI, offering the user a series of user-friendly menus of options and widgets for parameter input, while still allowing command line input when preferred;

(8) user proceeds to process data using McIDAS commands, with the option to bring more data into shared memory using McIDAS commands;

(9) user requires more image processing power than is available in McIDAS; within McIDAS, user clicks "exit" to leave McIDAS and start a new

application, or clicks "programs" to start a new application program while keeping McIDAS operational;

(10) user clicks "image processing" to load image processing program (ELAS ?), with common GUI, and proceeds to process the data which has remained in shared memory using menu driven or command line driven commands in ELAS;

(11) user then exits ELAS and returns to McIDAS to continue processing of the same data within McIDAS;

(12) after analysis and processing, user requires output to video and print for an upcoming conference; user clicks "output" which pops up menus which guides the user through the printing and video procedure; except for possible insertion of a video tape, all procedures for printing and video output can be handled from the users console;

DEVELOPMENT ISSUES

The main challenge facing the development of such an environment is, of course, the challenge of integrating programs which expect different data formats and which may in fact be running on different hardware platforms. Such integration challenges can be solved by the use of:

(1) **compatible data formats** and **data filters** between different application packages;

(2) **shared memory** using **common data structures** between applications; Note that the term "data format" is reserved in this report for the format of data residing on disk or mass storage, whereas "data structure" indicates the structure of the data in RAM memory and implies data abstraction as in C or an object oriented programming language.

(3) **interprocess communication**;

(4) **reliable network communication**;

(5) **distributed processing** and **distributed database management** across the network;

(6) extensive use of the **windowing environment**;

(7) application of a **user-friendly, portable GUI** shell controlling all interprocess communication and data sharing;

(8) the use of **computer standards**, standard system calls, and standard libraries.

Several development efforts can serve as helpful examples for the development of DAVE. Two of these include the apE visualization environment, developed at the Ohio Supercomputer Center, and FAST, developed at NASA Ames Research Center. Source code is available for both of these packages, thereby providing programmers with hands-on examples of successful development.

The apE is a programming environment designed especially for visualization. Similar to Stardent's AVS, it provides the tools for constructing networks of user-defined modules along which data can be piped. Since individual modules can reside on other hardware systems, it also allows for distributed graphics processing. Of particular interest to the development of DAVE are apE's portability and its graphical user interface. apE is presently supported for a wide range of visualization machines, including the Cray (X-MP, Y-MP: UNICOS), Silicon Graphics, Stardent, IBM (Risc 6000: AIX), Convex (C-1, C-2), Sun microsystems, NeXT, Hewlett-Packard, Digital Equipment (ULTRIX), and Apple MAC II (A/UX). Furthermore, apE's graphical user interface, "face library", supports four window standards, including X windows Version 11 Release 4, SGI's GL, Sunview, and NeXT. Since each module may potentially execute on a different host, binary data (where required) is communicated in a machine-independent form over TCP/IP protocol. However, such details are invisible to the user or programmer of apE. apE is designed to pipeline data along several user-defined modules and like Stardent's AVS environment, is not intended as a means of integrating two or more large applications packages. Still, it provides an excellent example for the development of distributed processing, and the use of easily portable code in several windowing environments.

FAST was developed by NASA Ames in order to transparently integrate several Computational Fluid Dynamics (CFD) programs which had previously been developed. Before the development of FAST, these individual programs each provided needed functionality for the visualization of CFD data, yet were not able to communicate with one another and suffered from incompatible data types. Like apE, the concepts of distributed processing and proper development of GUI's are illustrated in the development of FAST. In addition, however, FAST provides an excellent example of how several independent applications packages can be integrated into a single functional package through the use of shared memory, data filters, and interprocess communication.

The programs which might initially be integrated into DAVE have been selected based on their widespread use in the NASA and other Earth systems science communities, their portability, and on the availability of source code. These include:

(1) McIDAS - image display and animation system developed at University of Wisconsin's Space Science and Engineering Center (SSEC), partially under contract to NASA MSFC/ESAD, for the purpose of analyzing satellite imagery and other meteorological data;

(2) ELAS - powerful image processing applications program developed by the Earth Resources Lab at NASA Stennis for the purpose of analyzing satellite imagery;

(3) GRASS - Geographic Resource Analysis Support System developed by the U.S. Army Construction Engineering Research Laboratory which includes extensive functions for vector digitizing and display, raster analysis and display, and image processing;

(4) LINKWINDS - linked multiple windows program developed by JPL for the purpose of analyzing and correlating multiparameter, multidimensional data;

(5) 3D graphics - modified vis5D program or other application software for investigating structure in three dimensional datasets, using surface rendering, texture mapping, and animation;

(6) volume visualization - software package to be determined, which would allow detailed examination of three dimensional structure in data, based on volume rendering methods; primarily of use when the ability to view structure in detail is more important than interactivity;

(7) NCAR graphics - collection of modules for generating video and print output from various types of scientific and meteorological datasets;

(9) numerical modeling hooks - modules which would allow easy incorporation of numerical model output into the visualization environment, and would allow for real-time viewing and steering of numerical models;

(10) FAST - an integrated collection of modules developed at NASA AMES for the purpose of steering and visualizing computational fluid dynamics simulations in a distributed processing environment; program will also serve as functional model which integrated several somewhat incompatible programs into a single functional unit using concepts of shared memory, interprocess communication, and common data structures;

(11) SPAM - a Spectral Analysis Manager developed at JPL for the purpose of analyzing megaband image data, such as that obtained from AVIRIS (128 - 224 bands) or EOS HIRIS (196 bands); can be used to define a spectral signature for each pixel which can then be compared with known spectral signatures for minerals, vegetation, etc;

(12) IDIMS - database manager for archived data;

Other applications packages under consideration include AEGIS from Delta Data Systems, YNOT from Unidata, and several visualization tools from the National Center for Supercomputer Applications (NCSA). In addition, other small and large software tools could be integrated into or developed under DAVE. The integration of any tools into DAVE would be based on the need for their functionality within ESAD and the Earth systems science community, as well as the need for integration with other tools. Full integration of every visualization tool may not be necessary nor desired, particularly if that tool is typically used as a stand-alone package. In such cases, it may only be necessary to provide software for data import and export between DAVE and these packages.

DEVELOPMENT PHASES

The development stages outlined below have been designed to provide immediate solutions to the needs of ESAD and the Earth systems science community during each phase of development, while keeping a constant eye on the final goal of an integrated and portable visualization environment. The development of the environment and the integration of the above programs into the environment will occur in three phases:

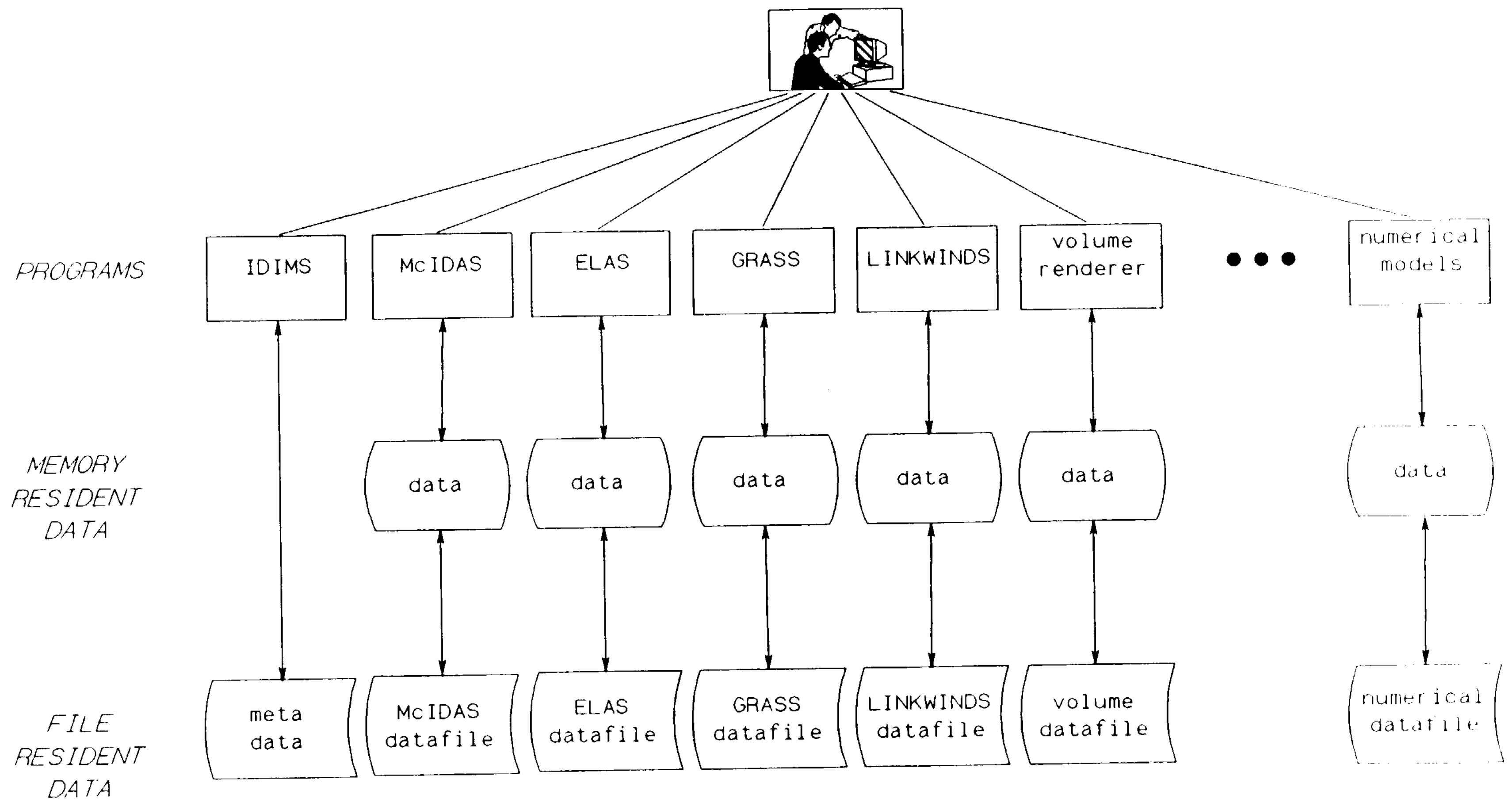
(A) PRESENT CONFIGURATION:

The present lack of integration between the above programs is depicted in Figure 1a. The environment is characterized as consisting of several separate, non-interactive programs, each of which depends on its own data format. The user interacts with each application independently. In order for data from one application to be used by another, the data must be read from data storage, translated into the new format, written as a new file in data storage, and generally transferred over network to a different hardware platform. The user, himself must then exit the first program, physically relocate himself to the other hardware platform, initiate the new application program, and finally read in and process the data. Generally, several different versions of the same dataset exist on the same disk or throughout the network. The majority of the program interfaces are command line driven and do not have a user-friendly interface. Translation between data formats and transfer between systems is totally by brute force, with no user interface.

(B) PHASE I:

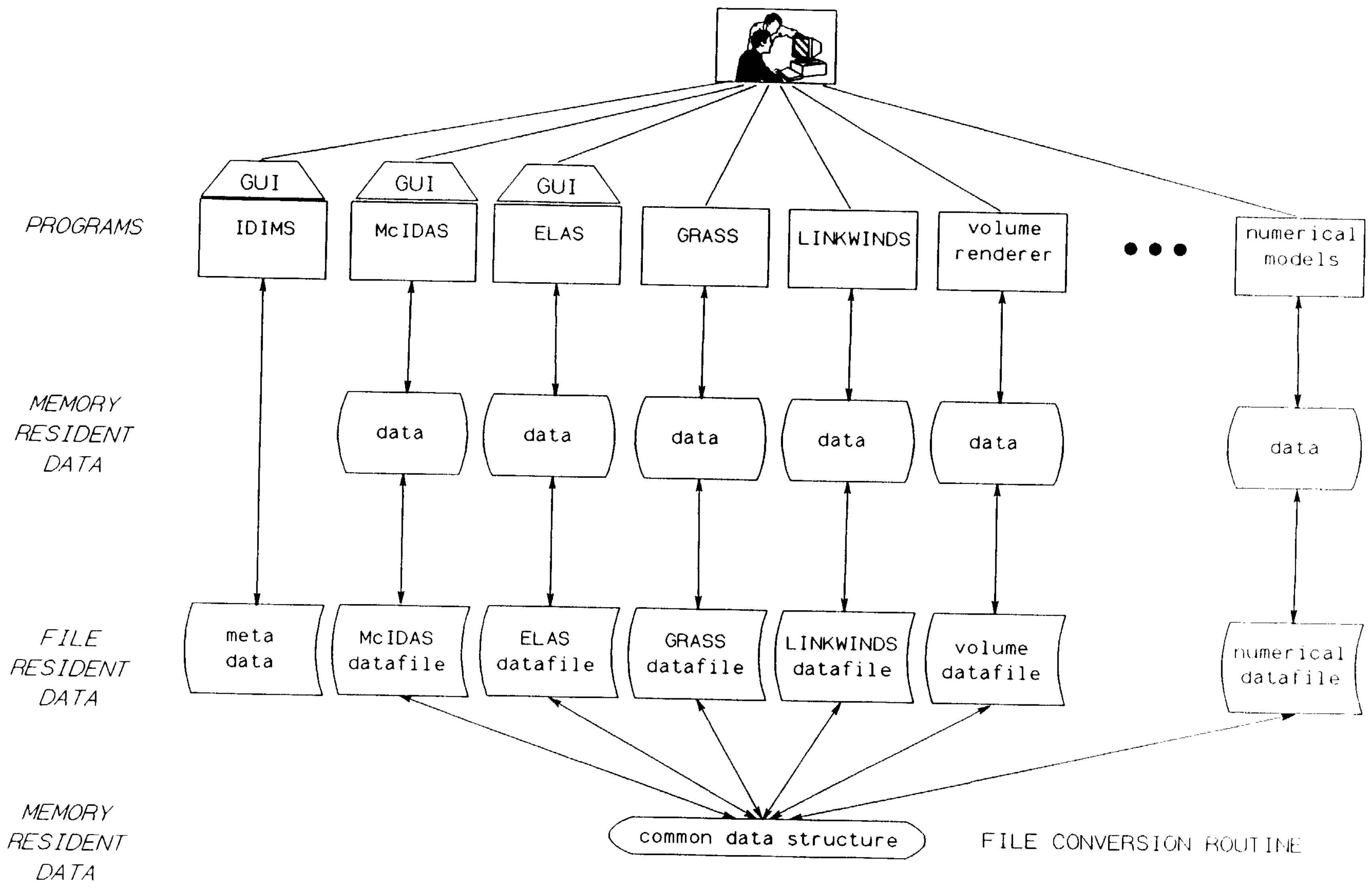
As depicted in Figure 1b, Phase I development accomplishes four major tasks: (a) establishment of standards for GUI's, windowing environments, file names, and program architecture; (b) development of user-friendly GUI's for major programs; (c) development of common data structures for all types of data (e.g. images, uniform and nonuniform grids, and trajectories); and (d) development of a user-friendly data conversion module using this common data structure.

The establishment of standards for the DAVE allows immediate development to begin in a fashion that will insure portability and extendibility of code. The development of GUI's for programs such as McIDAS and ELAS should result in an



PRESENT ENVIRONMENT

Figure 1a



PHASE I

Figure 1b

immediate increase in the effective use of these programs. The development of common data structures has two-fold importance. First, these data structures will serve as a common intermediate structure for the data conversion program. Using common data structures for such a module requires that only two modules be written for each data format desired (i.e. one for reading the data format into this structure, and one for writing the data format from this structure). Second, these data structures will serve as the basis for data storage within shared memory in phase III. The combination of user-friendly GUI's and the ease of data exchange between different applications will greatly expand the range of tools available to scientists.

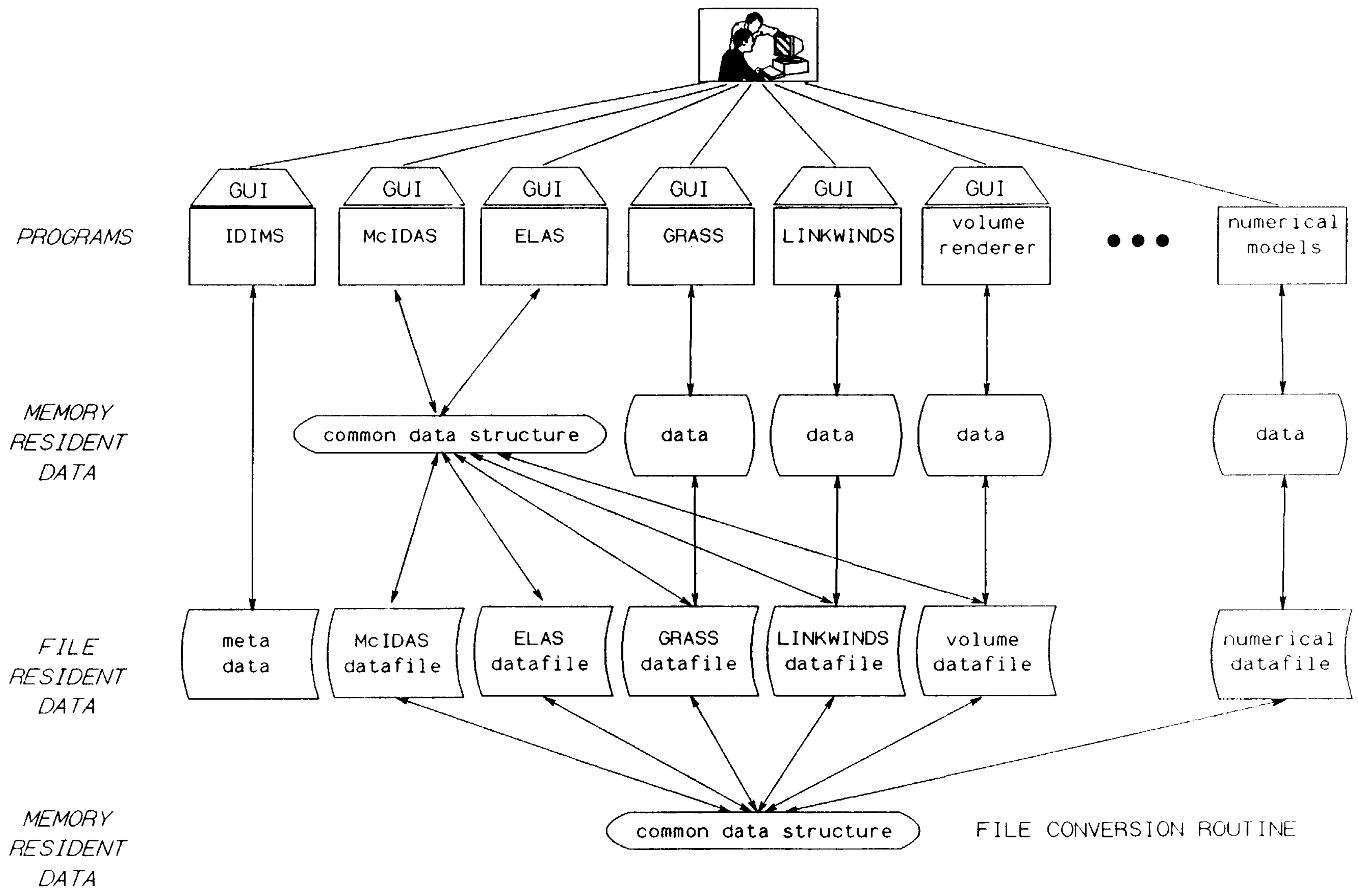
(C) PHASE II:

Phase II, as depicted in Figure 1c, consists of several tasks: (a) continued development and refinement of GUI's; (b) establishment of interprocess communication hooks between various application programs to allow information exchange between routines; (c) implementation of shared memory between critical programs; (d) development of filters to allow critical programs to read and use data from the common data structure in shared memory; and (e) development of a user-friendly module for controlling output to print or video (not shown).

Phase II will begin to allow true interaction between application modules, as well as provide increased productivity through the use of user-friendly GUI's for most programs. The common data structure and routines developed in Phase I for converting to and from various file formats, now function as data I/O modules for some application programs. Some applications would not require the user to exit the present program, in order to access functions existing on another application program. Database management would become a part of the visualization environment, by providing the capability to search for and load into shared memory, the appropriate data for graphics processing. Numerical models would begin to output data in the common data structures and common datafile formats. The video/print module would allow easy input/output to and from these devices from within application programs.

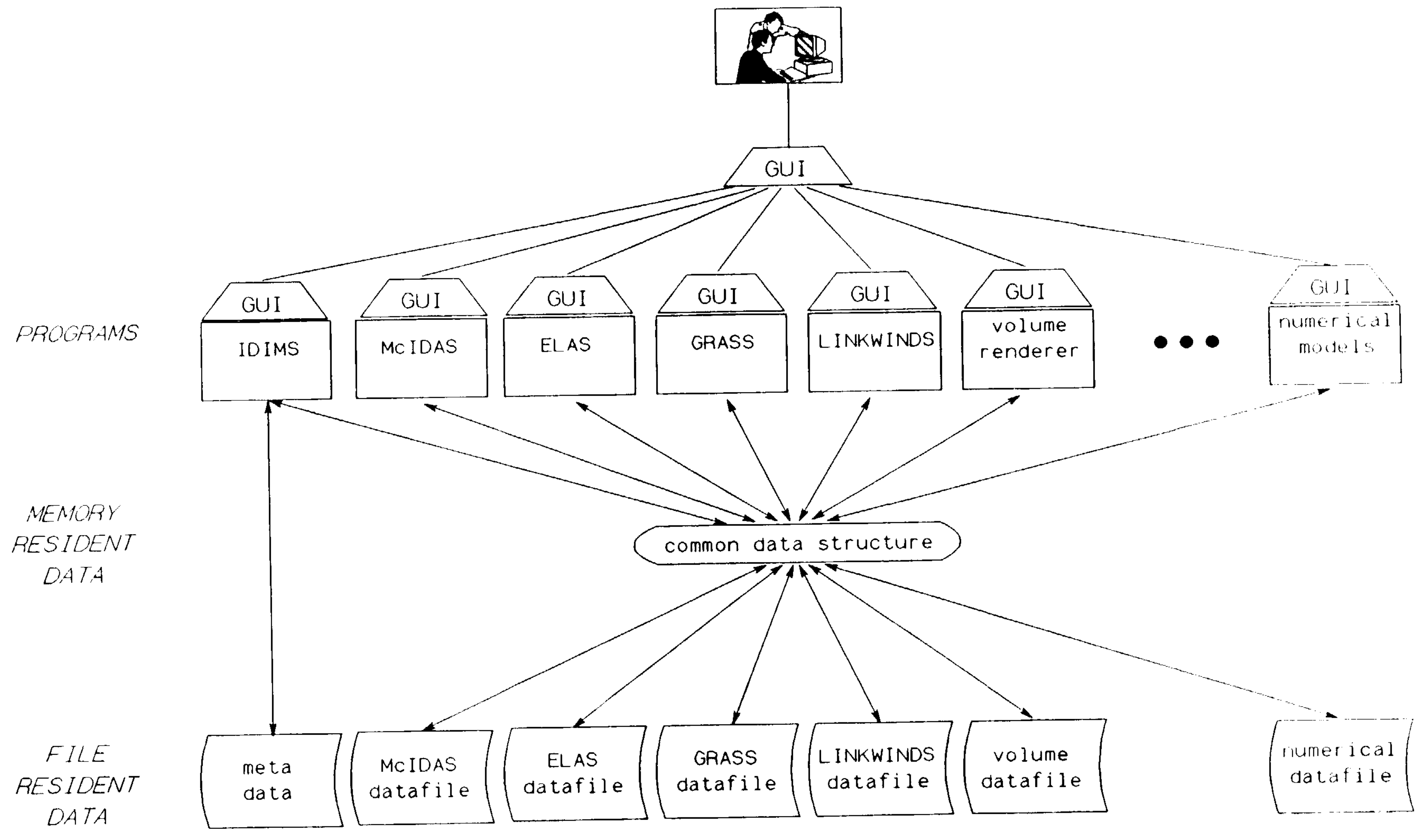
(D) PHASE III:

The final phase, depicted in Figure 1d, completes the integration of all modules into the environment, as well as provide a single interface through which the user can access all applications available. From this interface, the user can search databases existing on the network, retrieve datasets and load then into shared memory, and then use the processing power of all the application programs as if they were a single program. Processing could begin to be distributed across the network, so that compute intensive operations might be transparently and interactively off-loaded to the Cray or other devices, while continuing other graphics processing at the users' workstation. Output to video or print would be easily setup and activated by simple menu-driven modules. Numerical modeling could be incorporated directly into the



PHASE II

Figure 1c



PHASE III

Figure 1d

visualization environment in order to provide immediate visualization of results and possible interactive steering of the modeling.

The final configuration of DAVE can also be viewed as a series of shells seperating the user from the complexities of program and data integration. As shown in Figure 2, the inner shell consists of the various data file formats stored on disk, tape, or mass storage. The next shell consist of the common data structure which resides in shared memory within computer RAM. The use of a common data structure in this shell buffers each application program from the complexities of data incompatibility. The application programs interact with the data in shared memory through the use of filters or as a result of modifications within the "data read" modules of the application program. The application programs within this shell consist of database management and video/print modules, as well as individual visualization tools. Each application program is enclosed within its own user-friendly GUI, which directs the operations of that program. The final shell surrounding the application specific GUI's is the master command module which controls interprocess communication and shared memory management. The user interacts with this shell through the master GUI.

KEY TASKS

The development of DAVE can be divided into several somewhat independent tasks. The ability to divide the development effort into small independent tasks is important to the success of DAVE. This allows the development of DAVE to proceed without overwhelming the minimal resources for in-house development at ESAD, and without the need for a large and costly development effort with a single contractor. It is foreseen that several contractor and in-house development teams might independently complete individual tasks under the auspices of the DAVE project team. The completion of each task would provide immediate benefits to the visualization environment, while also bringing the DAVE environment closer to completion.

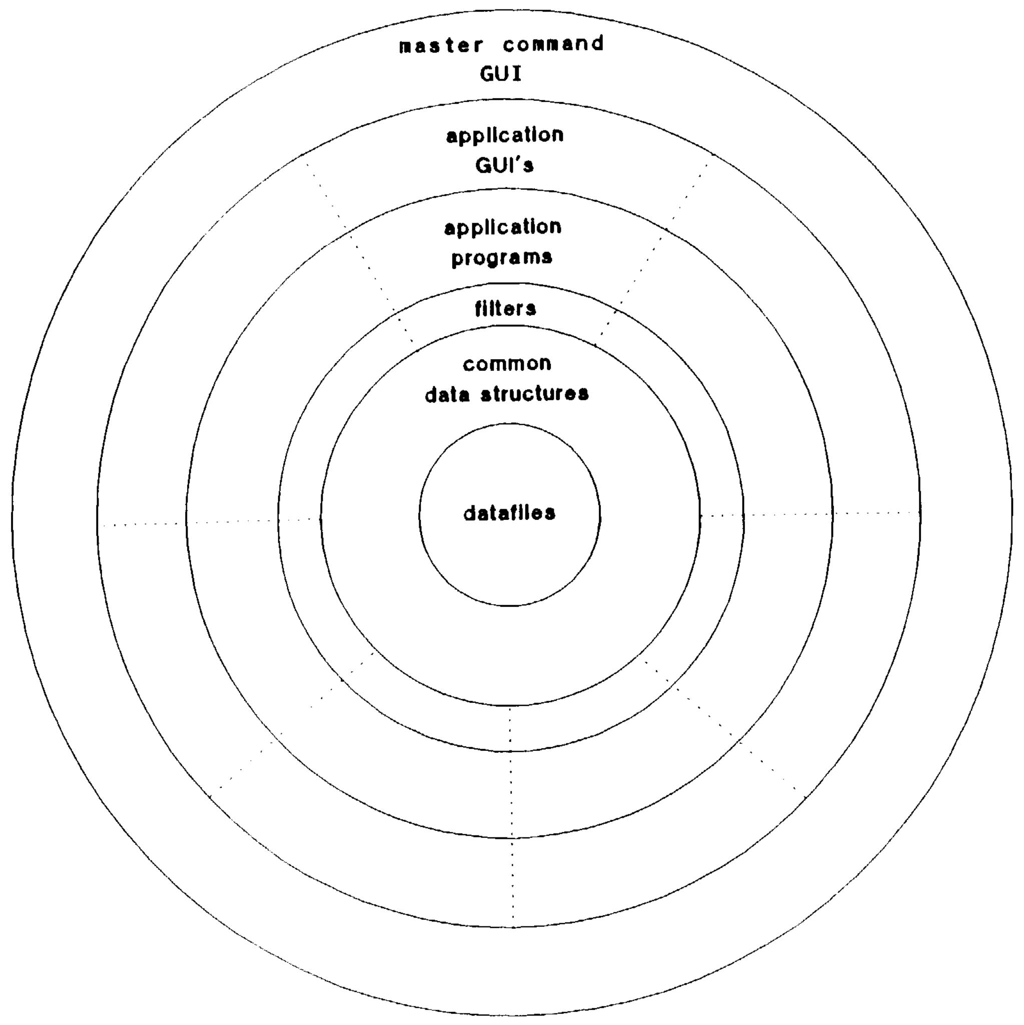
The major tasks that have been identified include:

(1) **Investigation and Recommendation of Development Standards** for GUI's and windows, for common data structures and data formats, and for portability of programs;

(2) **Development of Data Conversion Module** using the concept of an intermediate common data structure;

(3) **Development of GUI's** for individual application packages, such as McIDAS, ELAS, and IDIMS;

(4) **Development of Data Filters** for each application packages, allowing them to directly read, write, and use the common data structure residing in memory;



SHELL CONCEPT
OF DAVE

Figure 2

(5) **Development of a Video/Print Module**, which is user-friendly and flexible;

(6) **Development and Implementation of a Master Command Module**, which employs shared memory, distributed processing, and interprocess communication, and allows for final integration of all desired modules.

Several of these major tasks can further be divided into independent subtasks, which can be completed with smaller contractual or in-house efforts.

PROPOSED FUNCTIONS OF THE DAVE PROJECT TEAM

A project team should be created to oversee the development of DAVE. The DAVE project team should not function in any manner which inhibits timely development of programs designed to meet immediate short-term project needs. Instead, the functions of the DAVE team should be:

(1) **To investigate and recommend standards** by which the development of DAVE should adhere, when possible. These standards should be developed in order to increase, rather than restrict, software portability. Furthermore, these recommended standards should not inhibit other project teams from developing nor purchasing non-standard hardware or software, if such tools are more capable of solving immediate project needs.

(2) **To oversee in-house and funded development** which is geared toward completing specific tasks of the DAVE initiative.

(3) **To assist other project development teams** in developing tools which could be compatible with DAVE.

(4) **To archive and document existing and future software** in order to maximize its use and reuse, and to minimize redundant programming.

(5) **To assist in the distribution and implementation of DAVE** within other government, educational, and industrial institutions.

The DAVE team should meet on a regular basis and should assign specific tasks to relevant team members. The team should seek potential sources for funding of in-house development of DAVE, as well as encourage and assist outside contractors who wish to obtain funding for completing any of the project tasks outlined earlier. Any outside development of DAVE specific tools should progress under the auspices of the DAVE team.

CONCLUSIONS

The present data analysis and visualization environment at ESAD suffers from four major drawbacks, including (a) the lack of integration between various computer analysis and visualization tools, (b) the lack of efficient, user-friendly interfaces, (c) the lack of coordination and archiving of development, and (d) the lack of simple video/print output capabilities. These deficiencies have resulted in inefficient use of the visualization tools available to the scientist at ESAD, and in many cases, the total abandonment of these tools.

A Data Analysis and Visualization Environment (DAVE) has been proposed, which would allow several large and small application packages to be integrated into a single user-friendly environment. As proposed, this environment would allow transparent interfacing and sharing of data between these applications package, as well as the possibility for distributed processing across the network. As envisioned, DAVE would be portable to a large number of computer platforms, flexible to different data types and demands, and expandable to allow straight forward incorporation of new tools in the future.

It is further proposed that a DAVE project team be initiated for the purpose of recommending standards under which DAVE will be developed, and to oversee and assist in-house and funded development of visualization tools which might be incorporated into the DAVE environment. It is recognized that getting the immediate job done is the primary concern of the scientist and the division as a whole, and the DAVE project team should assist, and not inhibit, such efforts. However, when short-term development will not be inhibited by adhering to DAVE standards, the project team can help insure that these new tools can be incorporated into the visualization environment for future use by other scientists.

The amount of data available to the Earth systems scientist at ESAD and other facilities will increase tremendously in this decade and decades to come. The demands on the scientist analyzing this data will increase proportionally. Computer visualization will surely play a large role in providing the scientist with the tools for analyzing this flood of data. The state-of-the-art in data analysis and visualization, and in software and hardware development, will advance very rapidly in this decade. If present and future tools cannot be easily integrated into a user-friendly data analysis and visualization environment, scientists at ESAD will become increasingly frustrated by their inability to effectively use the tools at their disposal.

The proposed Data Analysis and Visualization Environment (DAVE) provides a realistic environment through which the Earth systems scientist at ESAD and other facilities can meet present and future demands. Establishment of a DAVE project team would provide the mechanism by which this environment will be developed and maintained.

APPENDIX A

DATA CHARACTERISTICS

APPENDIX A - DATA CHARACTERISTICS

Typical data sets for a given project in the Earth systems science community can be characterized as large, multidimensional, multiband, multiparameter, multisource, multiformat, multicoordinate, and multiperiod. These characteristics, and the problems associated with each, are discussed in more detail below:

(1) **Large:** Single data sets in the Earth systems sciences can typically range from 10 to 100Mb. Analyzing and visualizing such large data sets generally require greater computational power, larger memories, and more powerful graphics capabilities. While class 6 and class 7 supercomputers, such as the Cray-XMP/44, offer the computational power needed for filtering and massaging these large data sets, the visualization and interactive manipulation of these data generally requires either the application of graphics supercomputers from such manufacturers as Stardent or Silicon Graphics, or the creation of customized video capabilities (e.g. McIDAS workstations). Of equal importance to the computational or graphics power of hardware, is the availability, or ease of development, of powerful software required to interactively manipulate and visualize such large databases.

(2) **Multidimensional:** Data sets can range from point source data, consisting of scalar measurements at a single location, to four dimensional data, consisting of data measured within three spatial dimensions over time. These data might be in the form of two-dimensional (2D) images from satellites, radars, lidars, or other sources, or 2D or 3D grids of measured or numerically calculated data, such as temperature of the atmosphere or wind speeds. These data sets might be further grouped into time sequences, adding the dimension of time to 2D or 3D spatial images or grids. Other data sets might consist of measured or derived paths through space or time, such as trajectories of air particles derived from wind data, or the flight of a sensor-carrying aircraft. As a further complication, these grids or paths of data points could be either uniformly or nonuniformly distributed in space or time.

The multidimensionality of these data sets necessitates the use of various visualization methods, depending on the desired application, as well as the dimensional characteristics of the data. For example, single 2D multispectral images can generally be analyzed adequately using digital image processing techniques which have been employed in remote sensing applications for over two decades. Analyzing a time sequence of these images, however, requires animation capabilities found in graphics supercomputers or customized video systems. While time sequences of 3D grids of data can be viewed on video terminals connected to mainframes or supercomputers, which has been done with mainframe McIDAS, interactive steering and visualization of 4D data require the graphics power of graphics supercomputers, possibly connected to more computationally powerful class 6/7 supercomputers.

(3) **Multispectral:** A multispectral data set consists of multiple arrays of data which are closely correlated in space and time, but which differ slightly in their measured values. Satellite imagery is a classic example of multispectral data, in which each band represents radiance values of a given location within different bands of the spectrum. Each spectral band is directly correlated to the other bands in space and time, however, each band represents a different spectral response for this location. This differs from multiparameter data sets in which each parameter represents a measurement of distinctly different physical characteristics at the same location.

Multispectral databases are generally analyzed using image processing tools. These range from simple filtering and multispectral compositing, to more powerful statistical transforms and classifications. One of the present and future challenges in image processing involves the need to deal with databases which consist of very large number of bands. In the past, spectral bands have been rather wide and limited in number (typically 8-10, usually less than 20 bands). However, some present available airborne databases, such as the Airborne Imaging Spectrometer (AIS) and the Airborne Visible and Infrared Imaging Spectrometer (AVIRIS) generate images with 128 and 224 bands, respectively. The High Resolution Imaging Spectrometer (HIRIS) to be flown on EOS, will provide images with 196 spectral bands. Present image processing techniques do not adequately allow analysis of such images, in which each pixel essentially represents a semi-continuous spectrum.

(4) **Multiparameter:** Like multispectral data, multiparameter data is characterized as consisting of multiple arrays of data which are closely correlated in time and space. However, the measured or derived values in each array may consist of very different physical parameters or observations. For example, an atmospheric database of this type might consist of temperature, pressure, wind speed, relative humidity, and other measured or derived parameters. While these data sets could, in some cases, be analyzed using image processing techniques or by graphically overlaying these data sets on top of each other, a more appropriate visualization tool might employ the ability to interactively view and correlate these parameters through the use of multiple, linked windows.

(5) **Multisource:** In addition to single data sets being large, multidimensional, multispectral, and multiparameter, a particular application might incorporate and attempt to integrate data sets from several sources. A particular project or field program can incorporate various data sets measured from a multisensor platform. For example, NASA's Global Backscatter Experiment (GLOBE) involved remote and *in-situ* measurements from an aircraft platform, including three on-board Lidars, several particle counters, and a wide range of instruments for measuring temperature, pressure, and relative humidity, as well as aircraft position, aircraft speed, and sun angle. These

data sets might then be integrated with equally diverse data sets from other ground, aircraft, or satellite based sources. In addition, scientists might wish to correlate these data with data numerically derive from theoretical models.

Besides the analytical and visualization challenges inherent with each data source, the integration of data sets from several sources creates additional demands. Some of the major challenges arise from the need to **spatially and temporally** correlate data sets which might have different spatial and temporal resolutions, different instrument locations, and different look-directions. Added to these difficulties is the need to validate and investigate the relationships between several very distinct data sets. These might include sequences of 2D multispectral, multi-axis images, point source measurements of scalar values and spectra along the 2D path of the aircraft, paths of air particles derived from wind measurements, and 3D grid data derived from such sources as ground-based radar or numerical calculations.

(6) **Multiformat:** Differences in data formats can be attributed to two main causes. The first results from the inherent differences in the physical nature of the data and the means by which the data was obtained. Data can be in the form of lines, polygons, surfaces, images, trajectories, 2D or 3D (uniform or non-uniform) grids, or even text, depending on the information the data conveys and on the means by which the data was obtained. In order to be effective, a visualization tool must be able to integrate all of these data forms in a manner that allows them to be viewed and correlated together.

Secondly, within each of these data types, data sets may be different in the way the data is organized and ordered in a file, or perhaps in the format of header or navigational information. These differences result from variations in hardware and application software, as well as from different information requirements and preferences of various investigators. It is generally helpful, particularly for the purpose of data archiving and transport, to establish a standard format for each of the above data types. The standardization of data formats should not, however, restrict the use of visualization tools which do not explicitly recognize the standard formats. Software filters for converting application-specific data to and from the standard format are straight forward to write and use. These filters can often be placed within the Input/Output (I/O) functions of a specific application, thereby eliminating the need to store two versions of the same data.

(7) **Multicoordinate:** The need to interrelate multiple coordinate data is becoming increasingly more common in the Earth systems science community. Multiple coordinate systems arise from remote sensing of data from various platforms, such as aircraft, satellite, and ground-based platforms, as well as from differences in the inherent nature of the sensing device. For example, while typical satellite imagery might be inherently Cartesian, other data sets, such as 3D radar or global climate data, are more inherently spherical or conical in nature. In addition, any GIS data, whether

it consists of imagery, atmospheric pressure contours, or topography, can be stored or displayed in various map projections, such as Mercator, polar, or Mollweide.

Transformations between these various coordinate systems and projections are easily accomplished using readily available algorithms. Many visualization tools require the user to convert all data to some common coordinate system and projection, usually a Mercator projection within a Cartesian coordinate system, prior to its use by the visualization program. However, if data is transformed prior to importing the data into the visualization environment, the resolution and accuracy of the data are generally corrupted before its use for further contouring or rendering in the visualization routines. It is better to allow greater flexibility of data in the visualization environment, and to delay coordinate transformation until just prior to the display of results. This flexibility does not exist in the present versions of core McIDAS or McIDAS vis5d.

An additional demand on the visualization environment, is the need to interrelate data taken from different platforms, resulting in coordinate systems with potentially different frames of reference. For example, ESAD projects such as GLOBE and the Airborne Field Mill (ABFM), attempt to interrelate and coordinate data taken from airborne, satellite, and ground-based platforms. Since these three platforms are in relative motion, it is necessary to coordinate these in temporal, as well as spatial domains.

(8) **Multiperiod:** The need to interrelate time-dependent data is often further complicated by the presence of different sampling periods between the various data sets. For example, one ESAD data set may consist of ABFM data, spaced at 5-50 samples/sec, flying on an aircraft, whose parameters are sampled about once each second, and related to ground-based radar data, which might be sampled once every 5 minutes. In order for these databases to be adequately correlated and visualized together, it is essential to subset or interpolate these data into a common sample period. Since subset sampling and interpolation can potentially corrupt the data, it is important to thoroughly investigate various methods (e.g. averaging, Fourier transform, etc.) before application of any time sampling routine within a visualization environment.

APPENDIX B

COMPONENTS OF COMPUTER VISUALIZATION

APPENDIX B - COMPONENTS OF COMPUTER VISUALIZATION

Although the field of scientific visualization is still in its infancy, it is actually the unification of several traditionally independent, well-developed disciplines. These include image processing, 3D computer graphics, computer animation, geographical information systems (GIS), volume rendering, and computer aided design (CAD). The power and limitations of each are discussed in the following sections.

(1) **Image Processing:** The traditional role of image processing involves the evaluation of multispectral data sets in which each band consists of a 2D image of the same location, recorded at essentially the same time, but differing in the portion of the spectrum observed. A single multispectral data set can range from a single band to 224 bands, while typical data from satellite platforms range from 8 to 20 bands.

Image processing tools have proven very useful for filtering noise and enhancing images, in order to visually optimize extraction of information from single images (or single bands). Even more useful information has been extracted by compositing several bands (or ratios of bands) into a single color image. However, the most powerful tools developed in the field of image processing involve the application of various statistical methods in order to maximize the amount of information obtained from <u>all</u> bands in the data set. These include a large variety of techniques for classification and recognition of similar spectral responses based on data from all bands, as well as methods such as principal component analysis which transform the data for maximum discrimination of differences in spectral response.

In addition, many other powerful tools have been developed within the realm of image processing. Some involve the ability to determine temporal changes based on comparison of images taken at various times, the ability to spatially transform images into various projections, the ability to create large mosaics from several smaller images, and the ability to determine 3D relief of surfaces based on transformation and analysis of stereo pairs of images.

All of these tools, and particularly those which utilize powerful statistical methods, should play an important part in the visualization of the large, complex data sets described earlier. Although image processing tools have traditionally been applied primarily to image data, these same methods should be incorporated into the visualization of all types of data. For example, these might include 2D and 3D grids of multiparameter data (e.g. temperature, pressure, water pressure, etc.), volume data, and contoured data which have been texture mapped onto one or more 3D surfaces.

As discussed under "Multispectral Data" earlier, one of the present limits of image processing is the inadequacy of present techniques to handle megaband data, such as the 224 band AVIRIS airborne data, or 196 band

HIRIS data scheduled to be flown on NASA's EOS. For such data sets, statistical classification based on all bands, will become increasingly important, but will demand even greater computational power than required of present image processing systems. There will be a greater need for the development and improvement of image processing tools, such as NASA JPL SPectral Analysis Manager (SPAM), which allow the generation of a "complete" reflectance spectrum for each pixel or region, and to compare this spectral response with known spectra of various likely materials.

The second future concern in image processing involves the adaptation of image processing utilities, which have traditionally been applied to 2D imagery, to 3D databases, such as ground-based radar images, or 3D multiparameter data. Convolutions and classifications which rely on "nearest-neighbor" determinations will prove to be the most challenging to extend to three dimensions.

The final concern in image processing is how to incorporate these traditionally stand-alone capabilities into a more general visualization environment. In order to be fully effective, image processing tools will need to be fully compatible with, routines for surface rendering, volume visualization, or texture mapping onto 3D surfaces.

(2) **3D Computer Graphics / CAD :** The field of 3D computer graphics, and the interrelated discipline of CAD, have been the most rapidly advancing computer technologies of the last decade. In the 1980's, advances in computer graphics were driven by and rapidly incorporated into two rather distinct arenas; one being military flight and battlefield simulators, and the second being 3D animation for television and industrial presentations. The 1990's promises to hold equally rapid incorporation of 3D computer graphics into the field of scientific data visualization.

Advances in computer graphics during the 1980's provide a substantial base for the development of important visualization tools. However, in many cases, the algorithms and application programs developed for the needs of the 1980's, don't completely meet the specific needs of the visualization community today. Visualization tools need to be interactive and accurate in order to be effective. Some lighting models, such as the Phong model, produce very photo-realistic renderings of 3D surfaces, but are too computationally expensive for present use in interactive visualization environments. Furthermore, many commonly employed algorithms for defining 3D surfaces do not consider the level of accuracy required for scientific visualization and do not efficiently define surfaces for coarse grid data commonly found in measured or derived scientific data [Gallagher and Nagtegaal, 1989]. In order to adequately evaluate or develop visualization tools, it is important to remain up-to-date with the rapidly advancing computer graphics technology and to evaluate these advances with regard to the particular needs of the visualization environment.

(3) **Animation :** Most scientific data are transient in space and time. This is particularly true of data in Earth systems science. The ability to animate, or view this data over time and space, is a very vital component to any visualization tool. Systems for creating video animation for television or industrial presentations are generally more concerned with producing very photorealistic images than with rendering images interactively. In such systems, the user is generally able to view motion only after the individual frames have been transferred to a video editing system. Creating animation in such a manner may be adequate, and in some cases preferred, for presentation videos of scientific data.

However, in order for the scientist to interact with his data or to steer a simulation run, an effective visualization system must have the ability to quickly render surfaces based on the user's input, and to animate these results immediately on a video screen. This is generally accomplished in one of two ways: (a) render the individual frames of the animation sequence, and to sequentially store these frames in a large frame buffer for rapid playback; and (b) display the scenes in "real time" as they are being rendered. The first method is limited in the number of individual frames that can be stored in the buffer, and thus limited in the length or resolution of the animation sequence, while the second method is limited in the size and complexity of the models to be rendered. Which of these methods is employed is generally determined by the computer hardware available and by the particular needs of the scientist.

The need for interactive animation in the visualization environment, therefore requires much shorter rendering times than traditional animation systems provide. This must be accomplished by increasing the graphics performance of the computer hardware and software, or by optimizing rendering routines such that they meet the specific needs of scientific visualization.

(4) **Volume Visualization :** Many scientific data sets are volumetric; that is, they consist of a 3D grid of data values which are intended to represent a measured or calculated property within a volume of space. Examples include a 3D conic array of atmospheric radar data, or numerically derived values for atmospheric water content within a 3D thunderstorm model. A 3D volumetric data set might also be derived from a sequence of 2D images, such as medical Computer Aided Tomography (CAT) scans, or from a sequence of airborne LIDAR images.

In order to visualize a 3D volume of data, one must create some representation of the data (e.g. lines, dots, surfaces, etc.), that can be rendered to produce an image. There are three main techniques presently employed for interactively viewing such data. These include (a) contouring along slices, (b) generation and rendering of surfaces defined at some threshold value, and (c) volume rendering of voxels (volume elements).

Starting with its origin in geographic mapping, the technique of generating contour plots for various planes within volumetric data has been in use for several decades. Important recent improvements to this technique included more accurate contouring routines, the application of color to distinguish various contour levels, and most importantly, the ability to study changes in the contours in real-time while interactively moving the slicing plane through the data or while the data itself is changing over time. The ability to move the contour plane through space or time within the data set allows one to develop a feel for the 3D structure of the data. The use of multiple orthogonal slicing planes enhances this capability further. However, contouring techniques rely greatly on the user's constant interaction and on the user's ability to infer three dimensionality from 2D views.

The generation and rendering of 3D surfaces greatly enhances the ability to delineate three dimensional structure within the data. These surfaces are generally defined so that they indicate the location of a user-defined threshold value for a given parameter within the 3D volume. Various colors and degrees of transparency can be assigned to each surface such that several surfaces, representing different parameters or different thresholds within the same parameter, can be viewed at the same time. Furthermore, various lighting models, such as flat shading, Gouraud, Phong, or ray tracing, can be applied according to the limits constrained by the hardware and according to whether the researcher's needs are for interactivity or realism.

An ideal visualization tool should allow one to combine surface rendering with surface contouring. This ability provides three advantages. First, the user has the option to employ whichever technique is best for analysis or presentation of the data. Second, two related or independent parameters can be visualized and compared using one surface. For example, topographic data might define the shape of the surface while ground water moisture might be represented by color contouring on this surface. Third, data can be contoured along any shaped surface, oriented in any direction. For example, this would be useful for comparing ground-based 3D radar data with 2D radar data taken from a aircraft undergoing side-to-side and up-and-down motion, as well as pitch and yaw. In such an example, the surface would be defined by the pointing direction of the radar in time, which in turn is controlled by the position and tilt of the aircraft. If ground-based radar is contoured along this surface, it can be compared directly with the airborne radar.

The third method of visualizing a volume of data is commonly referred to as volume rendering and assumes that each data point in the grid represents the data value for a discrete volume in 3D space. Based on the associated data value, these volume elements, or voxels, can be assigned values for color and transparency. By assigning various values of color and transparency to different levels in the data, images can be rendered which retain very detailed information within the data. Such detail is often lost with the surface rendering techniques discussed earlier. The primary disadvantage of volume rendering is that the technique is presently computationally expensive, thus limiting its effectiveness in an interactive mode. Still, this technique could be valuable for many visualization applications where data detail may be more important than complex animations. In addition, new hardware power, as well as new rendering algorithms, are rapidly removing previous limits of volume visualization.

(5) **Geographical Information Systems (GIS):** The most important contributions of GIS to the visualization arena involve database management of highly variable data sets, geographical navigation of various data sets, and the integration of raster and vector display data.

VisTech

A Technical Newsletter of
the Visualization and Database Development Team
of the Remote Sensing Branch (ES43)
of the Earth Science and Applications Division
of NASA Marshall Space Flight Center

Edited by
Michael E. Botts

Issue 1 *October 17, 1991*

Welcome to the first issue of VisTech. VisTech is the mechanism by which the Visualization and Database Development Team will provide technical information to the division. Whereas the function of the sister newsletter, Vis News, is to provide information on the activities of the development team, VisTech will cover technical information on a wide array of topics, such as industry directions and evaluations of hardware and software. In addition, VisTech will include tutorials on such topics as Unix, vi, ftp/telnet, the use of locally developed software, etc.. These tutorial issues in particular might be worth storing in a notebook for future reference.

Each VisTech will focus on a single topic and will be released at irregular time intervals. The present issue is an excerpt from the preliminary SCF Working Group document, and discusses the directions of the computer industry as compiled by Ron Phillips and myself. It is based on continuous perusal of a large number of computer journals and newsletters, some of which are referenced at the end. We felt that such information might be helpful to anyone involved in or concerned about hardware purchases or software development. For your sake and ours, most future issues of VisTech will not be this long. If anyone should have any comments, rebuttals, or additions, please send them to me and I will try to publish them in future issues. Thanks, MEB.

COMPUTER INDUSTRY DIRECTIONS

Ron J. Phillips (UAH)
Michael E. Botts (UAH)

GENERAL DIRECTIONS

Standardization.

Standardization of computer hardware and software is a trend that began towards the end of the 1980's. It began as a reaction to the rapid introduction of dissimilar, proprietary systems introduced in the early 1980's from start-up and established companies. This push to standardize has been driven by customer demands for system longevity and industry risks associated with introducing proprietary technology.

The trend toward standardization has manifested itself in the development of open architectures and cooperative alliances between computer companies, many of whom were originally competitors. Further standardization efforts include increased use of off-the-shelf hardware components and the adoption of software standards for operating systems, programming languages, graphics libraries, windowing environments, and networking protocols. These alliances and standards will be discussed in more detail in the following sections on "Industry Alliances" and "Industry Standards". Consideration of developing standards and their acceptance within the computer industry is vital for ensuring longevity, upgradability, and interoperability of hardware purchases, as well as aiding portability and expandability of software development efforts.

CPU Trends.

The Central Processing Unit (CPU) is the primary computation engine of a computer. Therefore, its speed and characteristics drive the maximum theoretical performance of any computer platform. In reality, the actual performance of a computer running a given application is dependent on a large number of factors, including I/O rates, the amount of cache memory available, the performance of the graphics subsystem, the data transfer rates between memory and the CPU or graphics subsystem, and the proper balance of the performance and data flow rates throughout the entire system architecture. Because of potential "bottlenecks" within computer architectures, most computers do not approach full CPU potential for a given application. Still, it is important to consider the present and future directions of CPU technology when evaluating possible computing platforms.

Historically, CPU performance for a given cost has increased by an order of magnitude every 6 to 7 years. As an example, CPUs with 1 million instructions per second (MIPS) were prevalent by 1980, only to be replaced by CPUs near 10 MIPS by the mid 1980's. By the end of 1991, CPUs with performance of near 100 MIPS per chip are anticipated. Industry analysts predict that 1000 MIPS per chip will be available before the end of the decade. This logarithmic increase in computer power places additional importance on the upgradability of computer hardware and on software development with an eye to future power.

Within the last few years, most microprocessor CPU designs began the shift towards the RISC (Reduced Instruction Set Computer) architecture as compared to earlier designs employing a CISC (Complex Instruction Set Computer) architecture [Leibowitz,1991; Hennessy and Joupp, 1991]. Whereas CISC instructions vary in length (i.e. the number of bits in the machine language instruction), all RISC instructions are the same length allowing them to be processed in a "pipelined" fashion (similar to an assembly line), effectively allowing the CPU to execute one (or more) instructions per clock tick and thus raising the CPU's MIPS rating. Pipelining and a small instruction set of same-length instructions are central features of the RISC architecture. Figure 1 illustrates the logarithmic increase of CPU power used in supercomputers, mainframes, minicomputers, and microprocessor-based computers (i.e. workstations and PCs). The figure shows a sharp break and rapid increase in the power of microprocessors beginning in 1985, and corresponding to the increased power provided by RISC-based CPUs. The

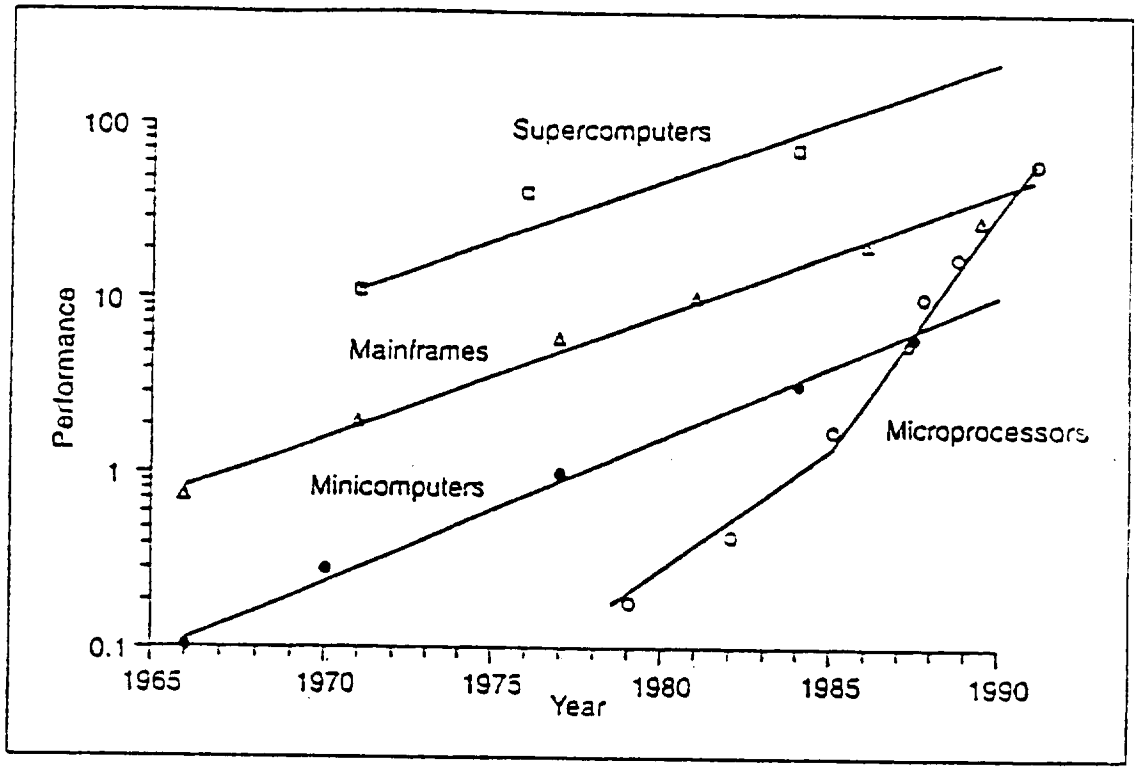


Figure 1. Trends in microprocessor and mainframe CPU performance. The increased rate of performance for microprocessors in 1985 is associated with the introduction of RISC technology [Hennessy and Joupp, 1991].

performance of CISC-based CPUs has continued growing at the rate exemplified by the pre-1985 segment of the microprocessor curve.

The CISC architecture is exemplified by the Intel 808x and 80x86 family of microprocessors (used in the IBM PC and PS/2 lines) and the Motorola 680x0 family (used in the Apple Macintosh lines). These processors have grown logarithmically in processing power over the years but not at the same rate as the newer RISC architectures. Every new generation in these families has maintained, for the most part, compatibility with previous generations. Unfortunately this compatibility requires that the instruction set grows with each new generation leading to more inefficient instruction execution. To combat this, CISC manufacturers are employing RISC-like features in newer generations.

The RISC architecture is exemplified by the Mips R-series (R2000, R3000, and R4000), the Intel i860 family, the Motorola 88000 chip set, the Sun SPARC family, the IBM RS/6000 chip set, and the HP/Apollo PA-RISC chip set [Corcoran, 1991a; Smith, 1991a; Smith, 1991b; Iverson, 1991]. Though most of these newer CPUs actually have an instruction set that compares in size to earlier CISC designs, they are still considered primarily RISC architectures since they try to execute at least one instruction every CPU clock tick. Newer RISC designs attempt to execute multiple instructions per tick by employing "superscalar" and "superpipelining" techniques that add multiple instruction pipelines and more instruction processing stages to each pipeline.

Clearly the choice of CPU that a computer manufacturer uses is critical if they want to maximize performance growth potential and minimize future compatibility problems. Typically, the CPU characteristics (other than speed) are not visible to application users nor to high-level language (FORTRAN, C, etc.) programmers. However, the customer should be concerned with potential obsolescence and incompatibility of their purchased hardware should the computer manufacturer choose a different CPU line in future products. Figure 2 shows a timeline of past, present, and future CPU utilization for key manufacturers of PCs and workstations.

Parallel Processing and Distributed Processing.

Multiprocessing, or parallel processing, involves simultaneous computation on two or more CPUs within a single computer. In theory, multiprocessing increases the computing power by *n* times, where *n* is the number of parallel processors. The major benefits of parallel processing are two-fold. First, on multitasking/multi-user workstations, separate tasks and users may be running on different CPUs simultaneously and thus not competing with each other for CPU time. The second benefit comes from using a parallelizing compiler on application code such that during execution, separate pieces of the code run simultaneously on different CPUs thus giving more processing power to the application.

It is important to differentiate between asymmetric multiprocessing, in which each processor basically works independently, and symmetric multiprocessing, in which the processors are tightly coupled by the operating system [Forbes, 1991a; 1991b]. Asymmetric processing is greatly limited in its ability to take full advantage of multiple CPUs, whereas in tightly-coupled symmetric processing, there is a near-linear relationship between the number of processors and the level of performance [Forbes, 1991b]. The general trend in recent years is to provide a growth path for additional CPUs within a given computer platform and to progress from asymmetric to symmetric processing [Baum and Winget, 1990].

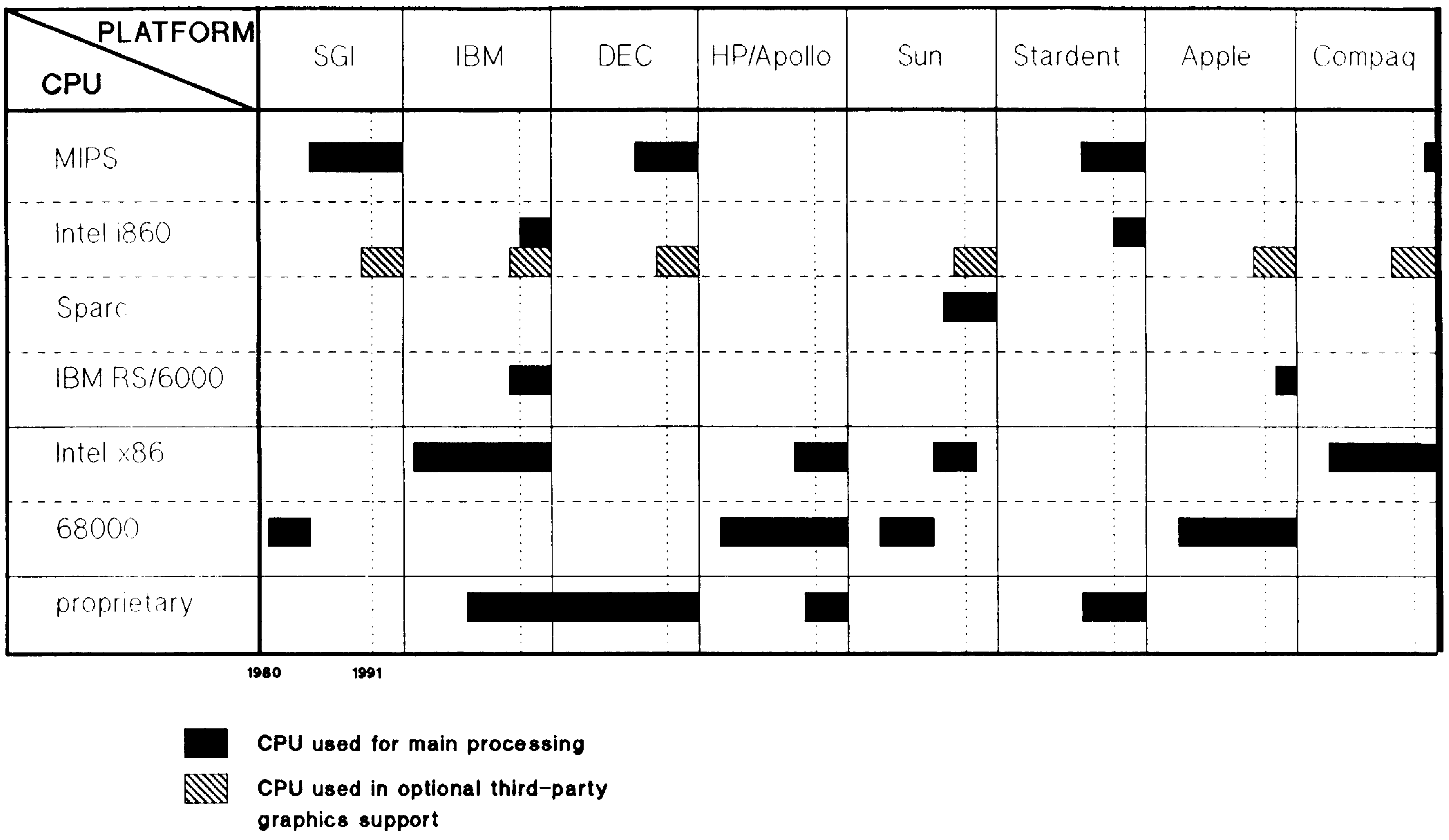


Figure 2. A CPU history plot showing the usage of various CPUs by workstation vendors. Within each column, the time scale ranges from 1980 to 1994 with the dashed line representing 1991.

Of the major workstation vendors, only SGI and Stardent provide symmetric multiprocessor workstations and have provided these since 1988 [Montgomery, 1991]; Sun will be introducing a multiprocessor SPARCserver by the end of 1991, but it will not be symmetric until the release of the Solaris 2.0 operating system in mid 1992 [Corcoran, 1991b]; in August 1991, IBM announced its high-end 32-processor Power Visualizer which utilizes the IBM RISC 6000 as its front-end. Also introduced in 1991 were a few first-generation massively parallel processing (MPP) servers, such as the Wavetracer, which consists of thousands of processors and utilize other workstations for frontend processing [Smith, 1991d].

The same benefits of multiple CPUs can be achieved even if the CPUs are not in the same physical computer. Distributed processing allows different computers to share their workload as well as their file systems, assuming that the proper connections and software exists on the machines. Unlike parallel processing, distributed processing introduces more compatibility and management issues since networking and dissimilar machines are usually involved. Distributed processing provides far more flexibility regarding upgrades and expansion since the number of computers that can be added is limited primarily by networking constraints. The network speed is also the biggest bottleneck since it is typically several orders of magnitude slower than the bus speed in a multi-processor system. Having a mixture of machines with varying performance characteristics ideally allows for a closer coupling between application run-time requirements and hardware availability than does a multi-processor system.

Importance of Software.

The availability of application software and software development tools should be a critical concern when making purchasing decisions for computer hardware. Computer power is useless unless it can be directed towards solving the needs of the computer user in a timely and efficient manner. In order to be of use, application software to meet the users' needs must either be available for that computer platform or it must be developed.

If it is necessary for the user to develop software, then it is important that one consider if tools are available for the particular hardware under consideration. In addition to possible languages requirements such as C, C++, Pascal, Ada, and FORTRAN, development tools might include graphics and image libraries, graphical user interface (GUI) builders, network toolkits, video toolkits, Computer Aided Software Engineering (CASE) tools, and optimization/parallelization compilers. For development needs, it is also helpful if the vendor supplies example source code illustrating how to access particular features of the system. These development aids become increasingly important considering the added complexities introduced by windowing standards, multiprocessing, distributed processing, and the integration of GUIs with interactive 3D graphics. Many hardware manufacturers have invested considerable effort and finances to provide the necessary software and libraries through in-house development or third-party support, giving computer buyers far more to consider than just hardware capabilities.

For scientific visualization, there are three types of software products available. The first type includes vertical application packages which are designed to meet the specific needs for a smaller group of scientists. For example, programs such as FAST, ELAS, VoxelView, and ARC/INFO are designed to meet the specific needs of computational fluid dynamics (CFD), image processing, volume rendering, and geographical information systems (GIS), respectively. The second type of visualization software includes application packages which provide a collection of tools for meeting a wide range of general visualization needs, such as image processing, 3D graphics, and volume rendering, but allow limited capabilities for user-customized development. These packages include Precision Visuals PV Wave, Wavefront's Data Visualizer, Spyglass Transform, and Sun's SunVision software [Kriz, 1991; Mercurio, 1991; Burns, 1991]. The third type of visualization software includes environments which allow custom design of visualization tools by the user or software developer. Within these environments, several functional modules can be linked by the user using a graphical interface, in order to rapidly create a custom application to meet the specific needs of the scientist. Having begun with SGI's Conman [Upson,1990], this concept has rapidly grown in popularity and is now incorporated in AVS, Inc.'s Application Visualization System (AVS) [Upson, 1990, Mazor, 1991], SGI's IRIS Explorer [Pope, 1991; Gorey, 1991, Mazor, 1991], IBM's Data Explorer [Mazor, 1991], and the Ohio Supercomputer Center's apE [VandeWettering, 1990, Upson, 1990].

Merging of Personal Computer and Workstation Environments.

Workstations were introduced in the early 1980's as a result of the successful introduction of personal computers in the late 1970's. The personal computer set the stage for the "one user - one computer" philosophy that ran counter to earlier uses of mainframes and minicomputers; the workstation was conceived as a minicomputer packaged in a personal computer for scientific and engineering applications. As CPU performance increased and prices dropped, personal computers approached the computational power of low-end workstations while low-end workstations became comparable in price to personal computers. The current trend in the industry is to provide a single platform which couples the diverse, user-friendly software of a personal computer with the computational power, connectivity, and graphics capability of a workstation. Significant efforts are underway to allow the cohabitation of DOS and Unix operating systems on the same platform and to allow the interchange of DOS and Unix application software between these environments. These efforts include the actions of the Advanced Computing Environment (ACE) consortium and the IBM-Apple alliance, to be discussed below, as well as the release of Sun's Solaris operating system.

Merging of Workstation and Supercomputer Technology.

As a result of increases in CPU power, parallelization, and compiler technology, workstations are now capable of performing compute-intensive tasks traditionally reserved for supercomputers only a few years ago. This trend towards incorporating supercomputer-class power into the graphics workstation environment began with the introduction of the Ardent Titan and the Stellar GS superworkstations in the late 1980s. These platforms included vector processing hardware similar to that used in supercomputers with the added functionality of high-speed graphics for displaying the computed results. Since that time, other manufacturers, such as SGI, IBM, and HP, have introduced high-performance superworkstations based on scalar hardware, believing that most application code cannot be effectively vectorized.

As shown in Figure 1, RISC-based CPU performance is growing at a faster rate than CPU performance of traditional supercomputers. The supercomputer architecture of the future will probably not rely on faster processors for increased performance, but will instead gain increased performance through MPP architectures with thousands of interconnected CPUs, as exemplified by the Connection Machine offered by Thinking Machines, Inc. and the MasPar system offered by Digital Equipment Corporation (DEC). For full-scale supercomputing applications, analysts predict that MPP systems will "reach the crossover point" - achieve equality with traditional vector systems - around 1994 or 1995 [Smith, 1991c]. Additionally, many "supercomputers" of the future will actually consist of an array of individual workstations networked together and operating under a distributed processing environment.

INDUSTRY STANDARDS

Operating Systems.

UNIX, DEC's VMS, the IBM-PC operating systems (DOS, OS/2, DR-DOS), and Apple's Macintosh operating system are the prevalent operating systems found on personal computers and workstations today, while UNIX, VMS, and IBM's MVS are the prevalent operating systems found on minicomputers and supercomputers machines. Based on the number of ports to different architectures and current trends of manufacturers, UNIX has become the de-facto standard for workstation and supercomputer class machines. UNIX is also available for the PC environment through Santa Cruz Operation's (SCO) Open Desktop. In a recent move, Sun Microsystems announced the availability of the latest version of their UNIX operating system, Solaris 2.0, on PCs [Burns, 1991b, Bucken, 1991].

UNIX currently comes in two major forms: AT&T System V (ATT SV) and Berkeley System Development (BSD). Most manufacturers have developed flavors of UNIX based on ATT SV, with Berkeley extensions related to communications and connectivity. Few companies continue to operate strictly under BSD due to the early momentum in UNIX support gathered by AT&T. The POSIX standard is a government-imposed UNIX standard based on ATT SV with Berkeley extensions. Recently, two major alliances have formed in order to standardize and promote the growth of UNIX: UNIX International (UI), founded by AT&T, and the Open Software Foundation (OSF), formed by IBM and DEC with the backing of most major system vendors. Both comply with POSIX and offer additional functionality for distributed computing support and distributed resource management support. Although most manufacturers have a different name for their UNIX operating system, the functional differences between these Unix flavors have become relatively minor due to standards compliance. In addition, the OSF has recently developed on the Architecture Neutral Distribution Format (ANDF) which promises to extend the advantages of off-the-shelf or "shrink-wrapped" software to architectures that are not binary compatible [Serlin, 1991]. This would allow software developers to develop a single version of a UNIX application, which would then run correctly on any other OSF UNIX machine regardless of the vendor.

DOS was developed for the IBM Personal Computer as a single-user, single-tasking operating system. Due to the success of the PC and the demands of the users, DOS has grown in capability but remains primarily a single-user, single-tasking operating system.
The Windows environment is a graphical user-interface shell operating within DOS designed to bring a limited form of multi-tasking into DOS. OS/2 was developed as a separate effort to perform true multi-tasking on the upper end of the PC line. Unfortunately OS/2 has not met with the anticipated development and market success. The future directions of these operating systems is uncertain in light of the

recent creation of new alliances, such as the ACE and the IBM-Apple alliances, and the destruction of old alliances, such as IBM-Microsoft.

In a move that perhaps best illustrates the rapid changes occurring in the computer industry, DEC announced in September 1991 that they will soon license their once closely-held VMS operating system in a RISC-based form to other computer manufacturers in an effort to recapture the market shift towards open, standards-based systems such as UNIX [Vizard and Ballou, 1991; Vizard, 1991b]. The performance of the RISC-based system, due sometime in 1992, is expected to be three to four times greater than the next generation CISC-based VAX system scheduled for release by the end of this year. In a clear sign of the gathering strength towards UNIX as a standard computing environment, DEC has scheduled to deliver a POSIX compliant interface for VMS by the end of 1991 so that VMS can operate essentially as UNIX to ease portability of applications across VMS and UNIX environments.

Windowing Environments.

Originally developed on workstations at Xerox PARC in the early 1970's, windowing environments have spread to most personal computers and all workstations starting commercially with the Apple Macintosh in the mid 1980's. During the mid 80's, most manufacturers created their own version of a windowing environment. By the late 1980's and early 1990's, the X window environment emerged as the standard for UNIX workstations. The success of X is partly attributed to its development as a networking protocol allowing graphical connectivity to a wide variety of machines. Since X purposely defines only the protocol for distributed windowing graphics and not the look and feel of the computer display, two standards which define the look and feel have emerged: OSF X/Motif and UNIX System Laboratories' (USL) X/Open Look. Table 1 illustrates the vendor and third-party support for these two standards on different platforms. Clearly, OSF X/Motif has become the window interface of choice among the majority of Unix-based hardware vendors [Burgard, 1991; O'Connell, 1991]. It is important to note, however, that these two standards do not imply incompatibility; any X window-based environment can run programs compiled with either standard. In a network environment, X provides greatly enhanced capabilities for both text and interactive graphics display relative to that available for simple text-only terminals.

With the breakup of the Microsoft and IBM team and the creation of a new alliance between IBM and Apple, the future direction of windowing environments on the PC is less certain. Clearly the combination of the undeniable market success of Microsoft's Windows environment, the market failure of OS/2, and the support of Microsoft's future Windows NT environment within the ACE consortium, will provide strong impetus for the Windows API. However, more PC users in the future may be enticed over to the Unix/X windowing environment as a result of the incorporation of DOS applications into the Unix operating system under efforts of the ACE and IBM-Apple alliances. This transition is being aided by the existence of user-friendly, "Mac-like" Unix interfaces, such as the icon-based X.desktop from IXI Limited and the Workspace tools from SGI and Sun [Hansen, 1991]. The IBM-Apple alliance may introduce a new windowing interface for PC in the future. However, this operating system is not scheduled to be released until at least 1993, at which time the ACE X/Motif and Windows NT environments, and Sun's Solaris operating system, should have become well entrenched.

VENDOR	MOTIF	OPEN LOOK
IBM	V	T
DEC	V	T
HP/Apollo	V	T
Unisys	V	
Sun	T	V
Solbourne	V	V
Compaq	V	
Dell	V	
Prime	V	
Data General	V	
Silicon Graphics	V	
MIPS	V	
NCR	V	
AT&T	V	V
Wang	V	
NEC	V	
Hitachi	V	
Commodore		V

V - Vendor supported T - Third-party supported

Table 1. Vendor and third-party support for Motif and Open Look X window-based interfaces [Burgard, 1991].

Text: Hardcopy and Screen.

The 1980's saw the emergence of "What You See Is What You Get" (WYSIWYG) capability for textual and simple 2D graphic editing and output. By the mid 1980's, the PostScript (PS) page-description language (PDL) from Adobe Systems had emerged as the leading standard for describing resolution-independent text and 2D graphics for hardcopy output in black and white and in full color [Barkes, 1991]. For simple text output (e.g. program or data listings), there are no special requirements for printing to a laser or dot-matrix printer, and many manufacturers, such as Hewlett-Packard, offer more affordable printers with proprietary and simpler PDLs.

With the emergence of the X window standard, the X font capability has become the standard for simple text display on UNIX-based workstation screens. For more complex text display and manipulation, Display Postscript (DPS) has emerged as a standard. When DPS is supported on a platform, it generally coexists as an option to the X fonts. There are few other font technologies, most notably TrueType from Apple Computer, that provide resolution-independent scalable fonts that can be generated at the speed required for interactive use on a workstation display. PostScript leads the text presentation market primarily because it has provided a consistent implementation for display on workstation screens and on a wide variety of hardcopy devices.

3D Graphics.

No defacto standard has yet emerged for interactive and static 3D graphics. The present X standard was developed as a 2D text and graphics protocol, and has proved inadequate for real-time 3D graphics. This is particularly true for color graphics, or for 3D X display over the network [Hayes, 1991; Hess, 1990]. Protocol Extensions to X (PEX), a protocol for improving 3D graphics over X is under development. The PEX protocol is based on the Programmers' Hierarchical Interactive Graphics System (PHIGS), a standard developed primarily for Computer Aided Design (CAD) applications and is not well-suited for the rapidly changing geometric information found in scientific visualizations [Jenkins, 1991b]. PEX, like X, requires that each manufacturer support and implement the standard on their hardware, and as of now, it is not clear whether PEX will achieve the industry endorsement anticipated by the developing committee. Although not often used as the primary 3D graphics language for scientific applications, PHIGS is presently available on most workstation platforms. Likewise, Sun's XGL library is based on PHIGS+ [Johnson, 1991].

Renderman is an established standard developed by Pixar that encompasses both a photorealistic renderer and a textual means to express complex 3D scenes. It was designed and is used primarily for generation of static images or non-interactive 3D graphics and not for scientific visualization.

A few manufacturers of graphics workstations have implemented proprietary interactive 3D graphics capability, most notably Silicon Graphics' (SGI) IRIS Graphics Library (GL). Undoubtedly the increased success of SGI as a manufacturer of graphics workstations has been in part due to the endorsement of the IRIS GL language by graphics programmers. Unlike PHIGS, GL is designed for rapidly changing geometries and is specifically tailored to high-speed graphics hardware. GL has been enhanced considerably over the years while maintaining backward compatibility, an important issue when considering a 3D graphics capability. In the past, the GL included not only graphics presentation capabilities but a windowing library as well. In addition, the GL has in the past been tightly coupled with the SGI Geometry Engine (GE) hardware. These two factors have inhibited the acceptance of the GL as a true standard for 3D graphics.

However, in recognition that the IRIS GL provides primarily a high-speed interactive rendering capability, SGI has been concentrating efforts on establishing the GL as a 3D graphics rendering standard. These efforts include decoupling earlier GL-based windowing protocols from the GL and developing a software implementation of the GL. In September 1991, SGI announced that it would license the GL to all vendors and would develop an enhanced version that would support the X windows 3D standard [Corcoran, 1991a]. Participating in the announcement were DEC, Compaq Computer Corp, and Intel Corp. who pledged their support for the GL, with Intel announcing its intention to integrate the GL into its i860 chip, [May, 1991b]. Microsoft had previuosly licensed the GL and announced intentions to make the GL available to the PC and workstation arena through its NT Windows. Similarly, IBM had previously licensed the GL, and incorporated it into its RS6000 series, while Du Pont Pixel Systems has recently been made available for SPARC platforms, including the Sun, through its PX/GL [Du Pont Pixel Systems, 1991]. SGI has created an advisory committee, which includes members from IBM, DEC, Compaq, and Intel, with the task of guiding the development and future of the GL. The agreement between DEC and SGI includes plans for incorporating PEX into the GL, and for support of the GL under DEC's VMS and Ultrix operating systems [Grygo, 1991b].

Networking.

In the workstation domain, the Transport Control Protocol/Internet Protocol (TCP/IP) is a firmly established standard on UNIX-based systems with connectivity available on more proprietary systems such as Crays, VAXes, Macintoshes, and PCs. It is typically used with Ethernet as the connectivity hardware, delivering 10 megabits per second communication capability. TCP/IP support includes standard network file-copying software (ftp) and remote login capability (telnet). Higher-level TCP/IP-based tools include the Network Filing System (NFS) for transparent remote file sharing capability and the Network Queueing System (NQS) for remote process capability.

A more recent development in the hardware connectivity domain is the Fiber Distributed Data Interface (FDDI) that increases the communication performance to 100 megabits per second, a factor of ten over Ethernet [Green, 1991]. The present standard utilizes fiber optics for connectivity, but recent committees have been formed to produce an FDDI standard using traditional twisted-pair wiring. This would appreciably lower the cost and complexity of upgrading to FDDI. A similar development is the High-Performance Parallel Interface (Hippi) that delivers a performance of 100 megabytes per second, eight times faster than FDDI and eighty times faster than Ethernet. Hippi is an increasingly popular standard for connecting supercomputers and high-speed graphics platforms to deliver real-time distributed visualization capability.

The International Standards Organization (ISO) has defined an Open Systems Interconnection (OSI) model that was intended to become a standard for communication connectivity but has yet to be realized on the same scale as TCP/IP and Ethernet [Varney, 1991]. Other proprietary systems have their own connectivity ardware and software such as DEC's DECnet, Novell's Netware and Microsoft's ANman for the IBM PC, IBM's SNA, and Apple's AppleTalk.

NDUSTRY ALLIANCES

dvanced Computing Environment (ACE).

n Spring 1991, over 20 hardware and software vendors announced the creation of he Advanced Computer Environment (ACE) consortium, formed to develop and upport an environment which would essentially merge the PC and Unix vorkstation environments. Since then, the consortium has rapidly grown to more han 85 members, including Digital Equipment Corp. (DEC), Compaq Computer orp., Microsoft Corp., Mips Computer Systems, The Santa Cruz Operation (SCO), Silicon Graphics (SGI), NEC Corp., Prime Computer Inc., Wang Laboratories, and Zenith Data Systems, among others [May, 1991a; Ballou, 1991; Flack, 1991, Smith, 991a; Porter *et al*, 1991].

As illustrated in Figure 3, the ACE consortium standard will support two hardware architectures: the Intel 80x86 PC standard and a new definition of RISC-based ardware called the Advanced RISC Computing (ARC) specification, which ncludes the MIPS R3000/R4000 CPU. In addition, ACE will support two advanced operating systems, either of which will operate on both platforms: one Unix-based, based on the SCO Open Desktop and OSF/1, and the other PC-based, consisting of the Microsoft Windows NT operating system. Furthermore, ACE will be compatible with both PC- and Unix-based networking services, including comprehensive interoperability with Novell, Banyan, and Microsoft networking services, and TCP/IP, SNA, and DECnet protocols. First-generation ACE compliant workstations are already available from SGI and DEC, with more system releases expected from other vendors in late 1991 and early 1992. The ACE compliant SCO Open Desktop Unix and the developers' version of Microsoft's Windows NT operating system is scheduled for release in late 1991 [Grygo, 1991a; Johnston, 1991].

SPARC International.

SPARC International was founded in 1989 as an independent association of corporations, institutions, and individuals with an interest in the standardization and evolution of the Scalar Processor Architecture (SPARC) technology [Hubley, 1991]. Developed by Sun Microsystems, the SPARC is available at low cost and has been implemented by many international hardware manufacturers creating low-end, affordable RISC computers. One of the main design philosophies for the SPARC CPU was performance scalability with little change to the architecture. This has held fairly true over the years but the design has not lent itself well to current efforts by Texas Instruments and Sun to produce a higher-speed superscalar SPARC CPU, called the Viking, which would be capable of running 50 to 100 Mips [Corcoran, 1991b, Wilson, 1991]. The relationship between Sun and some SPARC vendors in SPARC International was recently soured by Sun's surprise announcement to its dealers and value-added resellers (VARs), that they could not sell any SPARC-based computers other than laptops and mainframes if they wished to continue business with Sun [Poole, 1991].

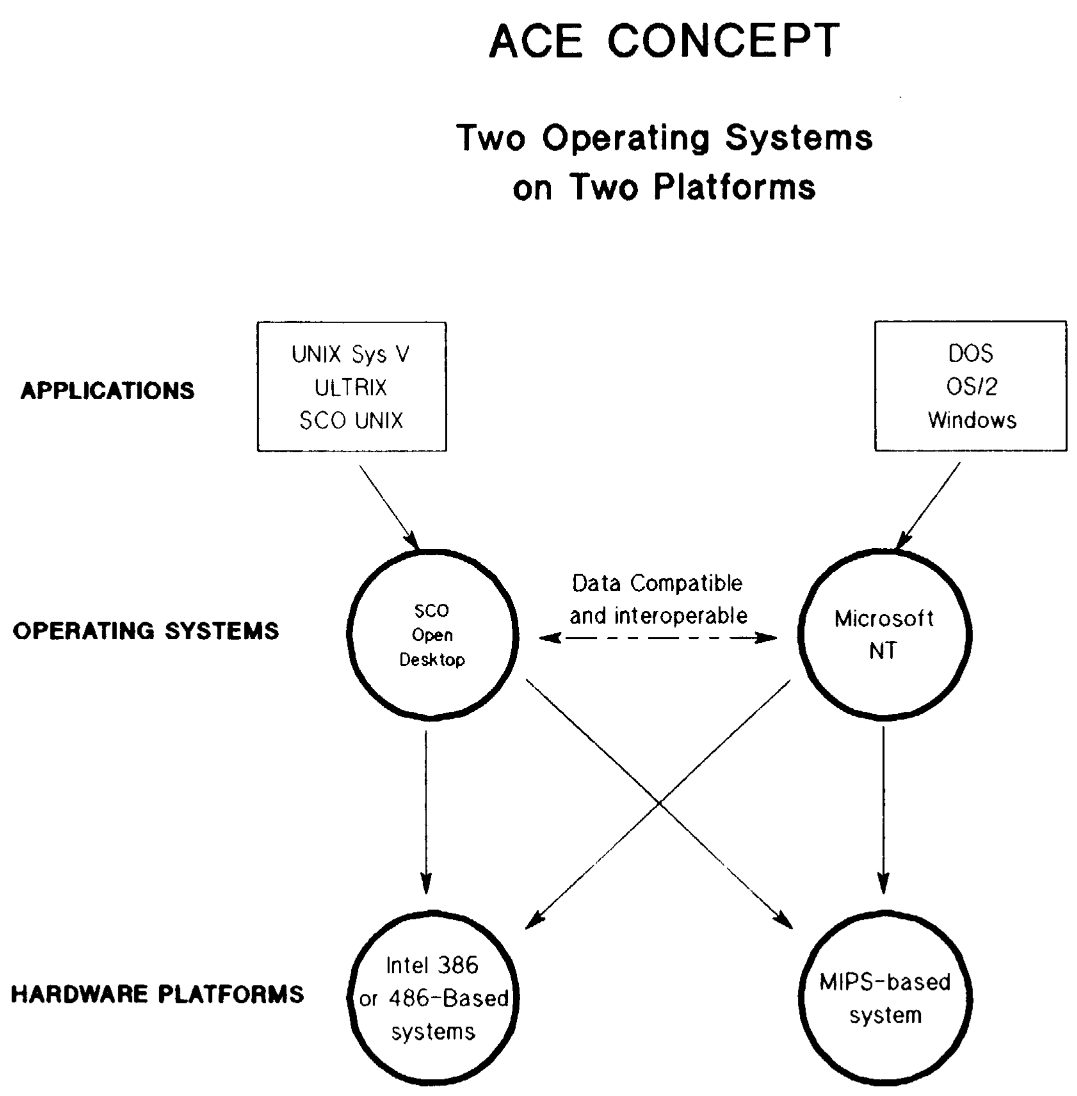


Figure 3. The four corners of the ACE concept; Both SCO Unix and Microsoft NT operating systems will be available for MIPS-based and Intel x86 PC platforms [DEC, 1991].

Apple-IBM Alliance.

In a surprise announcement in the spring of 1991, Apple Computer and IBM announced an alliance to develop a new operating system for the IBM RS/6000 line and as-yet unreleased Apple and IBM workstations running with a single-chip implementation of the RS/6000 chip set [Vizard, 1991a; Quinlan and Scannell, 1991; Scannell and Quinlan, 1991; Jenkins, 1991]. The single chip implementation is to be manufactured by Motorola with assistance from IBM and marketed by Motorola. The operating system will be developed primarily with object-oriented technology from Apple and from Metaphor, a company previously purchased by IBM to develop an object-oriented operating system for an earlier unannounced IBM platform. A special emphasis on multimedia capabilities using Apple Computer's QuickTime audio/video technology should serve to differentiate the new operating system from current systems. It is unclear whether this new system will incorporate backwards compatibility with DOS, IBM OS/2, and Apple's Macintosh OS. The companies have announced intentions to produce object-oriented software for AIX, OS/2, and Macintosh platforms. Many analysts view this alliance as a reaction to the threat of the ACE consortium as well as a reaction to the present PC software monopoly held by Microsoft [Vizard, 1991a]. The earliest anticipated release of the new operating system and the single-chip RS/6000 implementation, called the Power PC, is 1993 [Jenkins, 1991].

REFERENCES

Ballou, M-C. (1991), *ACE formation rocks desktop system architecture alliances,* **Digital Review**, June 17, 1991, pg. 1.

Barkes, K.G. (1991), *Clear sailing for PostScript*, **DEC Professional, Vol. 10**, No. 9, September 1991, pg. 64.

Baum, D.R. and J.M. Winget (1990), *Parallel graphics applications*, **Unix Review, Vol. 8**, No. 8, September 1990, pg. 50.

Bucken, M. (1991), *Sun targets Windows NT*, **Software Magazine, Vol. 11**, no. 12, October 1991, pg. 40.

Burgard, M. (1991), *Who's winning the GUI race*, **Unixworld**, August 1991, pg. 80.

Burns, M. (1991a), *Scientific visualization wars*, **Supercomputing Review**, October 1991, pg. 31.

Burns, M. (1991b), *Sun's new operating system straddles PC/workstation border*, **Supercomputing Review**, October 1991, pg. 13.

Corcoran, C. (1991a), *MIPS launches R4000 RISC chip*, **InfoWorld, Vol. 13**, Issue 40, October 7, 1991, pg. 25.

Corcoran, C. (1991b), *SGI licenses IRIS library*, **Infoworld, Vol. 13**, Issue 38, September 23, 1991, pg. 21.

Corcoran, C. (1991c), *Sparcservers await Solaris 2.0 to go symmetrical*, **Infoworld, Vol. 13**, Issue 39, September 30, 1991, pg. 6.

Du Pont Pixel Systems (1991), *Insight: GL on SPARC*, **Computer Graphics World, Vol. 14**, No. 10, October 1991, pg. 48+.

Flack, D. (ed.) (1991), *ACE Reshuffled into DEC*, **UnixWorld, Vol. VIII**, No. 10, October 1991, pg. 15.

Forbes, J. (1991a), *Multiple processors: are two heads better than one?*, **Personal Workstations, Vol. 3**, No. 2, February 1991, pg. 32.

Forbes, J. (1991b), *The Silicon Graphics approach*, **Personal Workstations, Vol. 3**, no. 2, February 1991, pg. 38.

Gorey, K. (1991), *IRIS Explorer: The example/the product,* **IRIS Universe**, No. 17, pg. 34.

Green, P.E. (1991), *The future of fiber-optic computer networks*, **Computer, Vol. 24**, No. 9, September 1991, pp. 78-87.

Grygo, G. (1991a), *DEC pledges to market NT operating system; MIPS launches R4000*, **Digital Review**, October 7, 1991, pg. 1.

Grygo, G. (1991b), *DEC backs Iris library in graphics workstation war*, **Digital Review**, September 23, 1991, pg. 1.

Hansen, A. (1991), *The desktop of the future arrives*, **UnixWorld**, May 1990, pg. 97.

Hayes, F. (1991), *X rated in 3-D*, **Unixworld, Vol. VIII**, No. 10, October 1991, pg. 54.

Hennessy, J.L. and N.P. Joupp (1991), *Computer technology and architecture: An evolving interaction*, **Computer, Vol. 24**, No. 9, September 1991, pp. 18-29.

Hess, M. (1990), *Flexing PEX*, **SunTech Journal, Vol. 3**, No. 1, Winter 1990, pg. 72.

Hubley, M. (1991), *SPARC International: guiding the market*, **Personal Workstation, Vol. 3**, No. 6, June 1991, pg. 77.

Iverson, W. (1991), *Allies rally around new MIPS chip to confront Sun, IBM, and HP; Also support Microsoft OS/2 3.0 and Intel x86 series microprocessors,* **Supercomputing Review**, May 1991, pg. 11.

Jenkins, A. (1991a), *IBM, Apple detail roles in joint desktop system plan,* **Digital Review**, October 7, 1991, pg. 1.

Jenkins, A. (1991b), *PEX takes X into the third dimension*, **Digital Review**, April 22, 1991.

Johnson, S. (1991), *XGL dynamic acceleration*, **SunWorld**, October 1991, pg. 71.

Johnston, S.J. (1991), *NT to run 16-bit software on ACE systems*, **Infoworld, Vol. 13**, Issue 38, September 23, 1991, pg. 8.

Kriz, R.D. (1991), *PV-Wave Point & Click*, **Pixel**, July/Aug. 1991, pg. 28.

Leibowitz, M.R. (1991), *The new generation of RISC*, **Unixworld**, August 1991, pg. 70.

May, C. (ed.) (1991). *SGI joins forces in ACE to build industry standard*, **Silicon Graphics World, Vol. 1**, No. 1, July/August 1991, pg. 10.

May, C. (1991b), *SGI to share IRIS Graphics Library*, **Silicon Graphics World, Vol. 1**, No. 3, November 1991, pg. 1.

Mazor, B. (1991), *Imaging at SIGGRAPH Las Vegas: environments in spotlight*, **Advanced Imaging**, September 1991, pg. 26.

Mercurio, P.J. (1991), *The Data Visualizer*, **Pixel**, July/Aug. 1991, pg. 31.

Montgomery, J.I. (1991), *Premier parallel performer*, **Digital Review**, September 9, 1991, pg. 35.

O'Connell, B. (1991), *GUI Update*, **DEC Professional, Vol. 10**, No. 3, March 1991, pg.60.

Poole, G.A. (1991), *Sun booms, SPARC vendors fume*, **UnixWorld, Vol. VIII**, No. 7, July 1991, pg. 15.

Pope, D. (1991), *Visualization news; tools in the news*, **Pixel**, July/Aug. 1991, pg. 10.

Porter, S., B. Robertson, and A. Doyle (1991), *News in Brief*, **Computer Graphics World, Vol. 14**, No. 10, October 1991, pg. 14.

Quinlan, T. and E. Scannell (1991), *Apple, IBM ready to finalize joint venture*, **Infoworld, Vol. 13**, Issue 38, pg. 1.

Quinlan, T., E. Scannell, and K. Scott (1991), *IBM, Apple ink historic deal*, **InfoWorld, Vol. 13**, Issue 40, October 7, 1991, pg. 1.

Scannell, E. and T. Quinlan (1991), *IBM, Apple set to launch deal*, **InfoWorld, Vol. 13**, Issue 39, September 30, 1991, pg. 1.

Serlin, O. (1991), *Unix inches closer to off-the-shelf software*, **Unixworld**, September 1991, pg 33.

Smith, N.P. (1991a), *Allies rally around new MIPS chip to confront Sun, IBM, and HP; Also support Microsoft OS/2 3.0 and Intel x86 Series microprocessors*, **Supercomputing Review, Vol. 4**, No. 5, May 1991, pg.11.

Smith, N.P. (1991b), *Intel's new 2.5 million-transistor i860XP - big impact on massively-parallel market, weaker prospects for workstations*, **Supercomputing Review**, June 1991, pg. 6.

Smith, N.P. (1991c), *New NERSC Director pursuing CRAY-3 program, seeking active role in Cray research MPP plans*, **Supercomputing Review**, October 1991, pg. 12.

Smith, N.P. (1991d), *Wavetracer: Low-cost MPP power*, **Supercomputer Review, Vol. 4**, No. 10, October 1991, pg. 18.

Upson, C. (1990), *Tools for creating visions*, **Unix Review, Vol.8,** No. 8, September 1990, pg 38.

VandeWettering, M. (1990), *apE 2.0*, **Pixel**, Nov./Dec. 1990, pg. 30.

Varney, S. (1991), *Global networking to push OSI momentum*, **Digital Review**, October 7, 1991, pg. 47.

Vizard, M. (1991a), *IBM signs up Apple to trump ACE*, **Digital Review**, July 8, 1991, pg.1.

Vizard, M. (1991b), *Posix Interface, RISCserver tools further Unix integration*, **Digital Review**, September 23, 1991, pg. 1.

Vizard, M. and Ballou, M. (1991), *Open VMS to come as DEC licenses RISC-based version to rival vendors*, **Digital Review**, September 23, 1991, pg. 1.

Wilson, R. (1991), *Sun unveils Viking superscalar design*, **Electronic Engineering Times**, Issue 657, September 2, 1991, pg. 1.