To be published in the Proceedings of
the AEC Scientific Computer Information
Exchange Meeting, New York, N.Y.,
May 2-3, 1974.

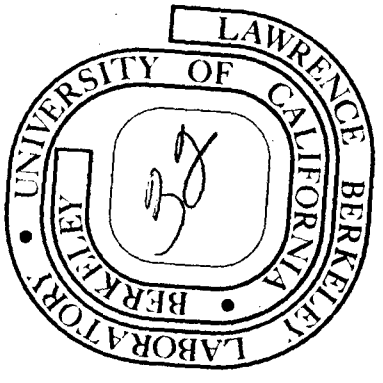
LBL-2675

CONFERENCE PROGRAMS WITH INTERACTIVE GRAPHICS

Donald M. Austin

March 1974

Prepared for the U. S. Atomic Energy Commission
under Contract W-7405-ENG-48



LBL-2675

DISCLAIMER

Table of Contents

Conference Programs with Interactive Graphics

Donald M. Austin
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

ABSTRACT

Multi-user conference programs provide interaction between users at remote terminals through the mechanism of a time-sharing or multiprogrammed host computer system. Extension of the conference program idea to include terminals with graphic input and output capability will provide a more natural medium of interaction and information exchange for a large class of problems.

The major problems to be solved in developing this type of network graphics facility are the interfacing to a variety of graphics terminals through a device-independent graphics system, providing reasonable data transmission rates necessary for interactive graphics, and the design of a suitable man-machine interface language to handle a variety of problem areas. Operating system requirements for both shared-program and shared-file conference systems are investigated, and the implementation of such a system based on the BKY 6000 operating system at LBL is explored.

I. INTRODUCTION

Conference programs are interactive programs which allow several users to interact with each other through the mechanism of a time-sharing or multiprogrammed host computer operating system. Representative examples are the single-host MOTIF program on Dartmouth University's DTSS [1] and the FORUM program of Institute for the Future which runs on the ARPANET system under TENEX [2]. These programs provide textual communication between participants with the additional advantage of having computational and data base facilities of computer systems as an integral part of the activity. On a network of multiprogrammed system with remote terminals, users may interact with each other and a data base in a real-time environment or on a delayed basis via a storage file mechanism.

Graphics is a natural extension to text based conference programs. The availability of graphics terminals has reached the state that this extension is both practical and useful for problems in which communication by pictures is the natural method. Interactive graphics applications cover most areas of problem solving, but the programs are usually written for specific systems and a relatively small number of these satisfy both criteria of extensibility and transportability.

The major problems to be solved in developing this type of network graphics facility are the interfacing to a variety of graphics terminals, providing reasonable data transmission rates necessary for interactive graphics, and the design of a suitable man-machine interface language to handle a variety of problem areas.

II. APPLICATIONS FOR INTERACTIVE CONFERENCE GRAPHICS

Some more or less specific examples of interactive conference graphics programs will help define the problems associated with implementation of such a system. Applications may be separated into two rather broad categories which characterize the nature of the interactive graphics - display with commentary and game situations.

A. Display with Commentary

Perhaps the simplest applications of interactive conference graphics are the analysis programs which display a picture (graph, set of data, schematic, flow chart, etc.) and allow the users to select features of interest and comment on them. A further extension of this category includes features such as windowing, zooming, overlaying plots and guiding the analysis by command menus and parameter setting. Many existing interactive applications could be extended to permit easy and natural communication of ideas between remote users. Conference manipulation of graphical data bases, such as urban planning, transportation network design, or architectural design, is an area which overlaps this category and the game situation area.

B. Game Situations

This category includes programs in which input from more than one user is required. Obvious game situations are those usually associated with the term, such as chess, economic modeling games and other decision-testing applications. A broader definition encompasses teacher-student and question-answer applications in a graphics based problem area.

III. DEVICE-INDEPENDENT GRAPHICS SYSTEMS

A. Types of Terminals

In order to be very useful, conference graphics programs require that the host computer be able to communicate with a variety of interactive graphics terminals and hard-copy devices. These terminals can be classified as follows:*

1. "Dumb" terminals, which perform i/o only.
2. "Semi-intelligent" terminals, which have a limited instruction set display processor and local memory.
3. "Intelligent" terminals, which have central processors as well as display processors and local memory.

For the first class, the host computer must speak to the terminal in its language. There is no sub-picture capability and each change in the display must be done in the host computer. An example of this type is the Tektronix 4012.

The semi-intelligent terminal has a limited subpicture capability and at least a "start address" operation code, so that portions of a display can be changed selectively without re-transmitting the entire picture. Translation of the graphics data into display commands must still be done by the host computer.

The intelligent terminal has a fairly powerful computer and is capable of performing many graphics operations locally, including translation of graphics data into display commands, simple transformation

---

*cf. Ref. 3 for a review of terminal classifications

of subpictures and editing.

In a conference system utilizing a variety of terminals, most operations will be reduced to the lowest common denominator, for if each user is to have an identical display all operations on the graphics data structure necessary to generate a new display must be done by the host for the lesser endowed terminals. However, a reduction in data transmission can still be realized by utilizing the full power of each type of terminal in the conference. For example, suppose a zoom operation is called for. For "dumb" terminals, the appropriate transformation of the data structure, translation into terminal display commands and the transmission of the new picture to the terminal must be carried out by the host. For the intelligent terminal, however, only a simple parametrized zoom command need be transmitted and all the other operations can be carried out locally.

Graphics input devices available fall into three categories:

1. Character input devices, usually a keyboard with 6, 7, 8 or 12 bit character codes.
2. Numerical input devices, such as potentiometers, or function keyboards.
3. Two-dimensional input devices, such as light pens, joy-sticks, mice, tracking balls, data tablets, thumb wheel cursors.

(There exist some three-dimensional input devices, such as the Lincoln Wand and 3-D joy sticks, which form a fourth category, but these are usually too exotic to be useful with 2-D terminals.)

B. Requirements for Device-Independent Graphics Systems

Given the above constraints it is evident that applications programs for conference graphics should be based upon a device-independent graphics system. Such a system consists of general high level routines for creating displays such as grid, smoothed curves, etc., plus some low level routines for translating a graphics data structure into terminal-specific display instructions. Ideally, the system should meet the following requirements:

1. Allow full use of available hardware features, such as character generators with variable sizes, fonts and orientations, and vector generators with variable line widths and intensities.
2. Allow for support of several devices simultaneously, including hard-copy devices operating in parallel with the various types of terminals.
3. Allow for high level graphics operations such as picture subroutining and incremental display modification.
4. Allow modular selection of high level routines and have small memory requirements for low level routines.
5. Allow for the various categories of input devices.

A common implementation of a device-independent graphics system employs a high level intermediate display language with a set of graphics commands in either a fixed-length format, such as

OP CODE	X_1	Y_1	OP CODE	X_2	Y_2

or a string format, such as

BREAK	OP CODE	X_1	Y_1	X_2	Y_2	...	X_n	Y_n	BREAK

Translation of the intermediate display file into device specific commands can be done as a separate job step or in line by specifying at load time the proper library of low level subroutines. Through picture subroutining, a mixture of the two methods can be used.

The study by the Network Graphics Group for the ARPA computer network covers most aspects of device-independent graphics protocol [4]. The protocol proposed by this group is to be implemented at various levels of sophistication, and provides features for interfacing with all types of terminals.

IV. OPERATING SYSTEM REQUIREMENTS

In order to implement conference programs on a host computer system, the operating system must contain certain features. Two possible conference systems will be discussed - the shared-program system and the shared-file system. The applications possible with these two systems share considerable overlap (the shared-file system has more general possibilities), but operating system requirements differ considerably for the two.

A. The Shared-Program System

For a single set of related applications, particularly in game situations, the most efficient method of conferencing is the shared-program system. In this system, multiple terminals are connected to a single job running at a single control point (thus a single user operating system is even suitable if one has the resources to tie up a host computer with interactive jobs). Features required of the host operating system are:

1. Multiple-terminal interface.
2. Multiple-terminal connection to a single job.
3. An interrupt or polling capability which allows the host computer to service any one of the connected terminals with reasonable response time (including log on and log off).
4. For interactive graphics programs, a device-independent graphics system and an appropriate set of interpretors.

The third requirement is perhaps the most troublesome. The concept is simple enough - it requires that the program be informed by the terminal handler whenever any terminal logs on or off the conference program.

In addition, it requires that the program be able to post reads to all connected terminals and be activated (rolled into central memory) when any input is forthcoming.

Typically such a program would consist of an applications module, a high level graphics module, an executive module and a set of interpretor modules, as depicted in Figure 1. The applications module operates on some data base to produce interesting data, which is fed to the graphics module for creation of a display file. The executive module directs input to the interpretors for translation and transmission to the terminals. Terminal response is fed back to the executive for further action.

B. The Shared-File System

The shared-file system is somewhat more general than the shared-program system in that communication is between separate jobs, each of which may include different applications programs and data bases. Features required of the host operating system are:

1. Multiple-terminal interface.
2. A multiprogrammed or time-sharing system.
3. Special file types accessible by more than one program simultaneously.
4. An interrupt or polling facility which allows a host program to service any of the conferee's with reasonable response time (including log on and log off).

This system is the basis of the conference systems mentioned in the Introduction (DTSS [1] and FORUM [2]) and seems to suit a wider variety of operating systems than the shared-program system.

A schematic of this system is depicted in Figure 2. The executive module reads the input files from the terminals and creates a global

display file with the conference picture. Each user application program has read-access to the global display file, and can create a display in any manner desirable, whether split screen, overlay or even disregarding some subset of conferees. The executive program holds the conference together by performing the following tasks:

1. Polls the system for new conferees and allows them to join the conference. Conversely, it detects conferees who leave the conference and keeps the conference informed of the participant status.
2. Maintains a global display file with each particpant's current display by polling them for updated inputs.
3. Offers help and advice to individual conferees on demand.
4. Takes votes, gathers statistics and runs questionaires.
5. Provides back-up for important files (including hard copy) and performs restart procedures.

V. CONFERENCE POLITICS

In some conference environments there is a need for a chairperson function to guide the activity, while other environments may be totally democratic in structure. Any required dictatorial functions may either be embedded in the conference executive (i.e., the chairperson may be connected directly to the conference executive) or command facilities for conference direction may be honored for only one prespecified terminal (e.g., the first terminal joining the conference).

The democratic conference can be realized by a network structure with no explicit executive as shown schematically in Figure 3. Here each program contains a mini-executive with read access to every other participant's display file. The mini-executive routines consist of terminal-to-display file interpretors, display file-to-terminal interpretors, and polling facilities. This style of conferencing comes close to the sort of facility offered by computer networks, such as the ARPANET, where host-to-host file transfer protocol has been established [4].

VI. A PROPOSED IMPLEMENTATION

The computer center at LBL offers the following facilities relevant to interactive conferencing:

1. Interconnected CDC 7600, 6600 and 6400 with over a billion (60 bit words) of on line mass storage.
2. A terminal handler system being expanded to 256 terminals with data rates up to 9600 bps.
3. A variety of interactive terminals, including Tektronix 4012's, DEC GT40's, CDC 250 VISTA consoles, plus several hardcopy devices.
4. ARPANET connection (soon).

The BKY operating system currently allows an implementation of a shared-file system through a facility called "shadowed" COMMON files. This facility allows a job to capture a COMMON file created (and temporarily released) by another job, obtain read-only access (i.e., SHADOW the file) and return it to the system. The originating job then recaptures the file by the COMMON operation and retains write access. Anything written on the file can be immediately read from the shadowed file. Thus in Figure 2, the executive program shadows all the input files for the connected terminals, and all the terminal programs shadow the global display file simultaneously. Polling is accomplished by periodically reading the system File Name Table into executive program memory space and checking a list of prespecified file names for users logging on or off the conference. By maintaining an updated list of file pointers, the programs can determine when new input is available on a given file.

On the BKY 6000 system, interactive jobs are automatically rolled out of memory after a period of inactivity. Thus, while the terminal programs can be rolled in on demand, the exeuctive program, which is not connected to a terminal, must execute a recall loop in order to relinquish the central processor to other jobs. This becomes unnecessarily expensive for long periods of inactivity. One solution, albeit a rather clumsy one, is to have a chairperson terminal connected to the executive program. It is then the chair's responsibility to insure that response time is maintained at a reasonable level. A much more elegant solution is to provide a peripheral processor (PP) program which resides in one of the 20 PP's attached to the 6600. This PP program can perform the polling function by "waking up" the executive program when new input is forthcoming. Going one step further, the same PP program is capable of doing direct memory-to-memory block transfers, eliminating the need for auxilliary storage files (at least for input from the terminals, which tends to be smaller than the global display file).

The shared-program system has already been implemented for the primitive Berkeley Remote Facility, and a new system under development for the implementation of the ARPANET connection at LBL.

VII. CONCLUSIONS

Conference programs with interactive graphics on a variety of terminals offer a useful method of communication between users at remote sites. The problems involved have for the most part been solved in one way or another, and all that remains is fitting the pieces together into a coherent system.

The shared-program system allows several terminals to connect to a single job, offering features usually associated with conference or game situations, where all users are interacting with the same data base. The addition of graphics broadens the applications possible with this system to include many problem areas not feasible with text-only systems.

The shared-file system connects several interactive jobs and thus provides several host-sized computer facilities to the conferees. This system is in fact a natural extension of computer networks and is considerably more general than the single-host, shared-program concept, since only the graphics and file transfer protocols need be specified. Program languages, analysis programs and data bases available to the users can be as varied as required for a particular application.

ACKNOWLEDGEMENT

Work performed under the auspices of the U. S. Atomic Energy Commission.

REFERENCES

1. McGreachie, J. S., Multiple Terminals Under User Program Control in a Time-Sharing Environment, Comm. ACM 16, 10 (Oct. 1973), 587-590.

2. Amara, R. and Vallee, J., FORUM: A Computer-Based System to Support Interaction Among People, Institute for the Future, Menlo Park, Calif. 94025.

3. van Dam, A., Intelligent Satellites for Interactive Graphics, Proceedings of AFIPS, 42 (June, 1973) 229-238.

4. Michener, J. and Sproul, B., Proposed Network Graphics Protocol, Network Graphics Group Note No. 5, NIC No. 19933, ARPA Network Information Center, Stanford Research Institute, Menlo Park, Calif. (Oct. 1973).

FIGURE 1. DIAGRAM OF THE SHARED-PROGRAM ENVIRONMENT

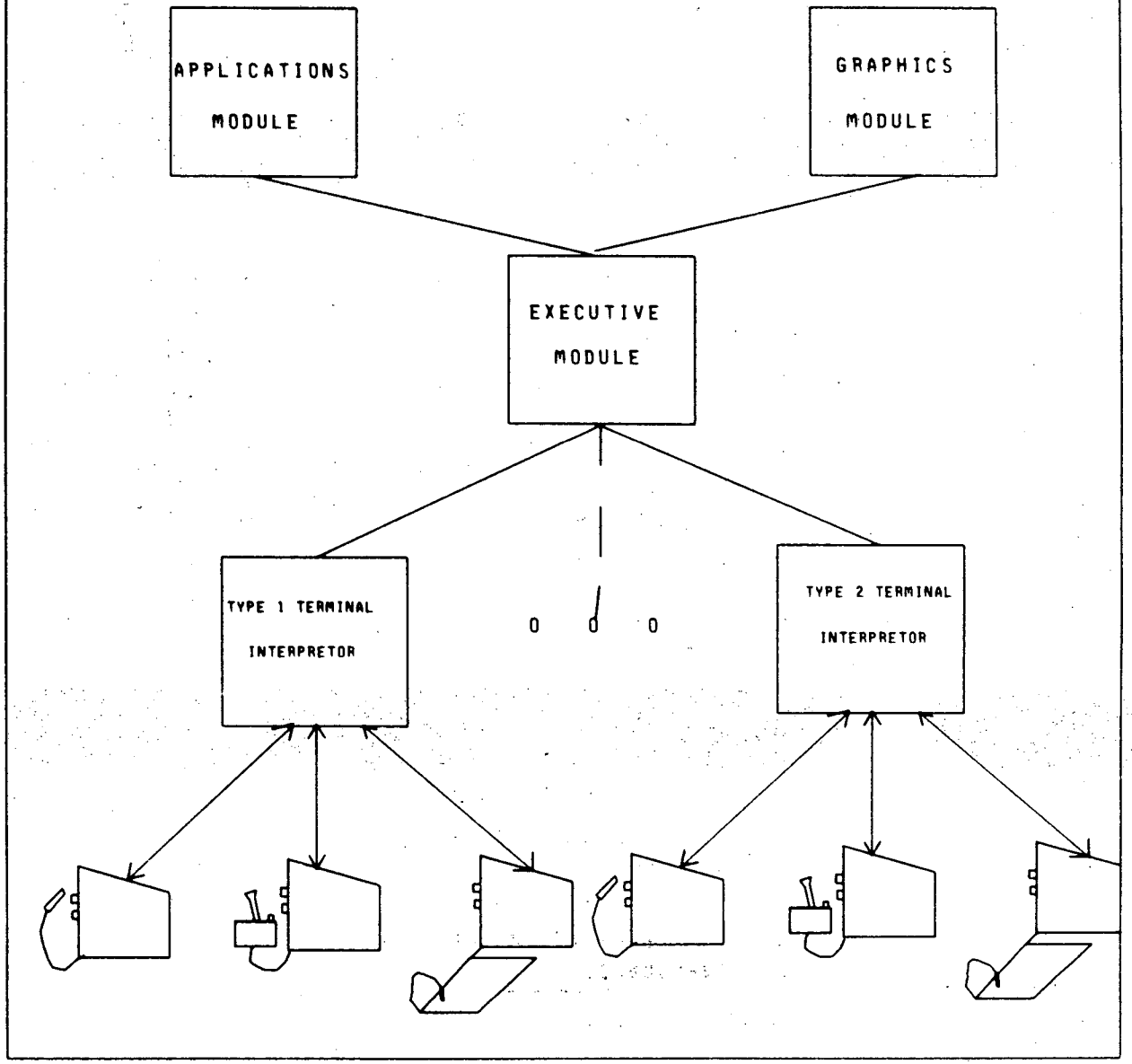


FIGURE 2. DIAGRAM OF THE SHARED-FILE ENVIRONMENT

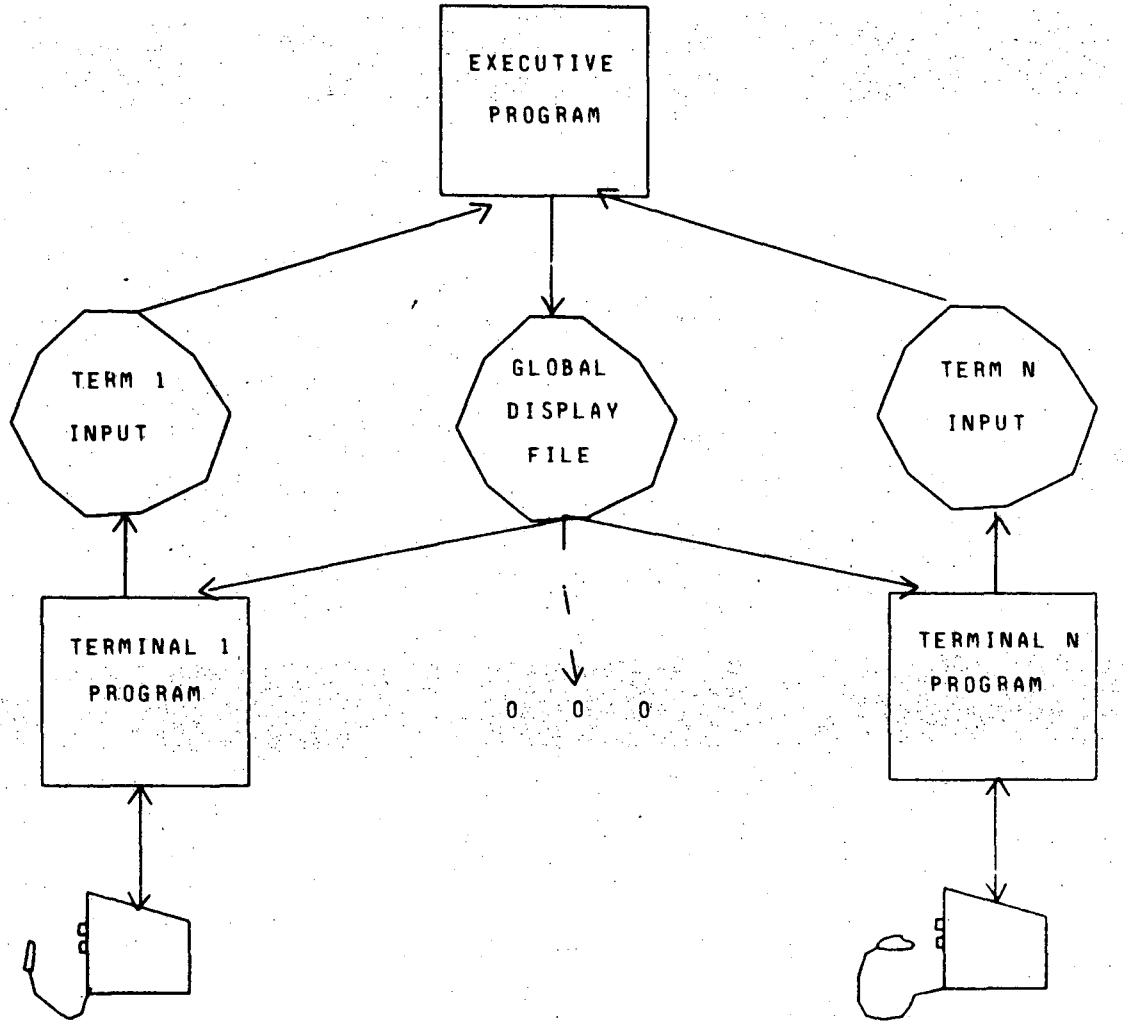
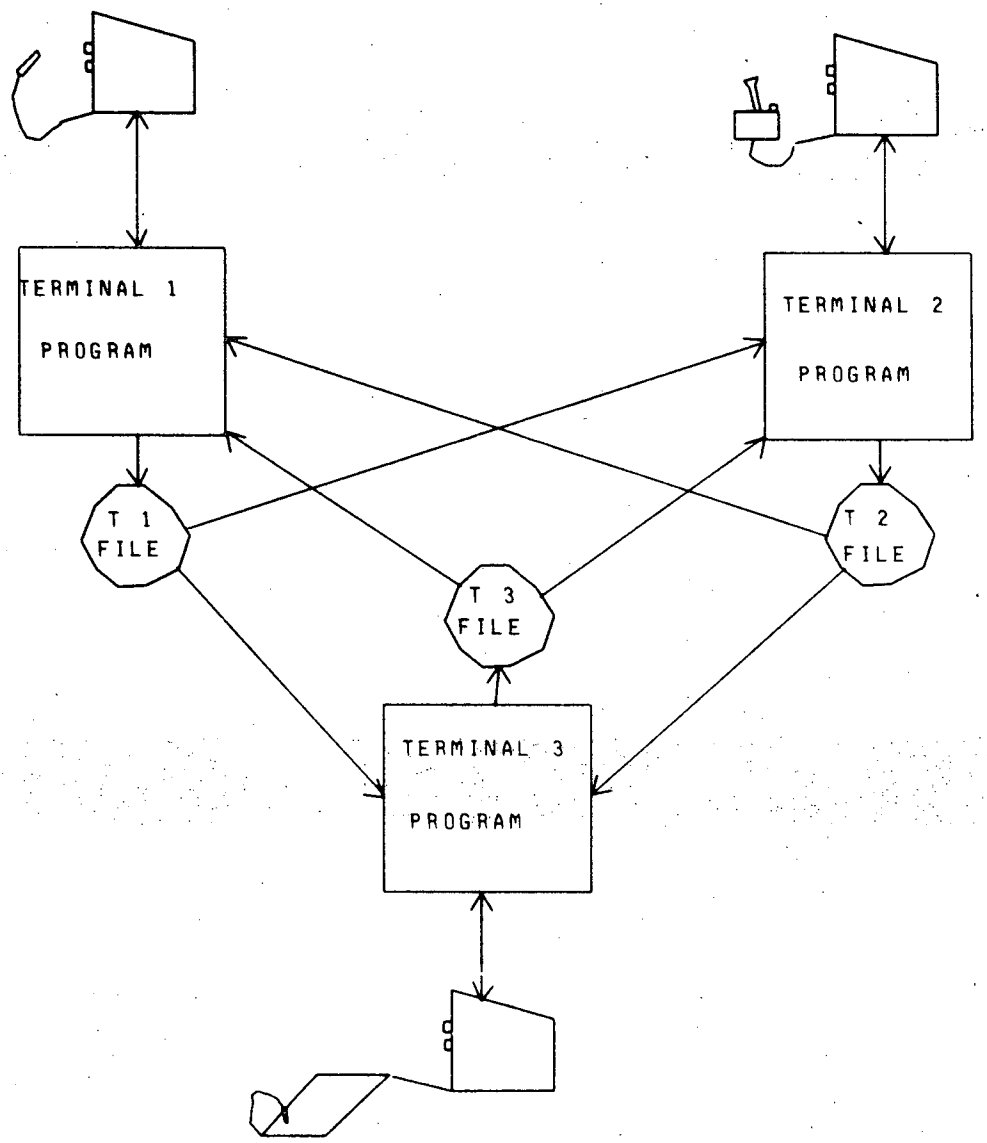


FIGURE 3. DIAGRAM OF THE DEMOCRATIC ENVIRONMENT



TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720