This paper discusses a system called DISPLAYTRAN *that interpretively executes* FORTRAN *statements entered at a display console, allowing graphics users to perform unanticipated computations and to more easily debug graphics application programs.*

The relationships among the operating system, the display terminal, and the computing system are discussed, and the major components of this system are described. A command language, the FORTRAN IV *subset, and the graphics language provided for users are presented. Internal operation of the graphic facility is outlined.*

INTERACTIVE GRAPHICS IN DATA PROCESSING

A conversational display capability

by F. W. Gagliano, H. W. Thombs, and R. E. Cornish

Programming implies anticipating all conditions that may arise in the course of solving a problem. Unfortunately, not all problem solving lends itself to this tidy approach. In many cases, each successive step can only be planned after the succeeding step has been completed. Thus, effective use of graphics devices for interactive problem solving requires some means for requesting that a data processing system perform functions not anticipated at the beginning of the problem-solving process. This fundamental problem has been attacked in various ways.[1,2] For example, one system provides for a library of previously compiled computation modules that can be called by the display console operator as needed.[3] However, that approach assumes that the needed computation modules exist.

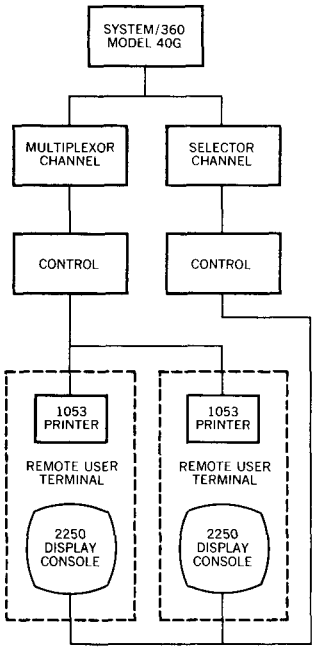
The system discussed here interprets and executes FORTRAN statements as they are entered from the display console.[4–8] For example, if a console operator, after seeing a display of a geometric figure on the screen, decides that he would like to perform an unanticipated computation, he can do so without a separate compilation run. He simply enters FORTRAN statements at the display console, which are interpreted in real time and then executed.

Interpretive FORTRAN execution also ameliorates the problem of debugging for graphics programmers.[9] Syntax errors are revealed as soon as the system attempts to interpret each statement. Also, errors in logic can be corrected more easily because the console operator can stop execution at any point he desires. These facilities are provided for the graphics as well as the computational portions of application programs.

The system discussed here is called DISPLAYTRAN, which takes its name by analogy from QUIKTRAN.[10,11] Like QUIKTRAN, DISPLAYTRAN provides interpretive FORTRAN execution for interactive problem solving. Many of the capabilities of DISPLAYTRAN are useful for graphics applications, although the system is not designed exclusively for graphics jobs. Graphics and other jobs can be entered directly from the console, and batch processing can be done concurrently in a background partition of main storage. The system provides time slicing for jobs done at the display console. General-purpose graphics subroutines are supplied for FORTRAN programmers,[12,13] and can be called from a program being constructed at the display console.

DISPLAYTRAN is one result of studies undertaken jointly by the International Business Machines Corporation and the U. S. Naval Weapons Laboratory.

Figure 1 DISPLAYTRAN computing system



The first part of the following discussion deals with the overall relationships among the display terminal, the computer configuration, and the operating system. The remainder of the paper emphasizes the languages provided, which include a command language, the FORTRAN IV subset, and the graphics language. Also, the manner in which the console operator can call and execute previously compiled subprograms is discussed briefly.

System design

DISPLAYTRAN is executed as a single task under the IBM SYSTEM/360 Operating System (OS/360) capable of multiprogramming with a fixed number of tasks (MFT). It requires a basic main storage partition of 86K bytes, which may be expanded or reduced within limits by overlays of a number of system subroutines. Wherever possible, advantage is taken of the facilities of the operating system, such as interrupt handling, input/output, and system macroinstructions. Although at the time that DISPLAYTRAN was developed, many channel programs had to be written especially for the IBM 2250 display console using the EXCP macroinstruction, available data management access methods were used, such as the basic sequential access method (BSAM) and the basic partitioned access method (BPAM).[14]

One of the primary advantages of DISPLAYTRAN is its ability to provide the user with graphic subroutines that can be used to plot graphs or draw figures under FORTRAN program control. DISPLAYTRAN was developed for an IBM SYSTEM/360 Model 40G (128K bytes of main storage) and with remote terminals, as shown schematically in Figure 1. Each terminal consists of an IBM 1053 printer and 2250 display console. The 2250 has a standard typewriter keyboard through which statements are entered, a display screen that can display up to 52 lines of 74 characters each, and a function keyboard with 32 buttons, each of which signals a specific DISPLAYTRAN system command. The 1053 printer is used for hard-copy output in listing and/or debugging applications.

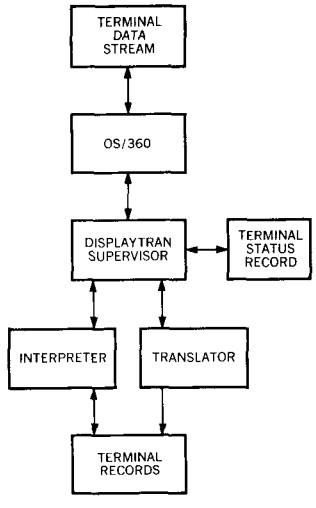
Three major program components had to be developed to pro-

vide the facilities required for multiterminal conversational operations: supervisor, translator, and interpreter. Figure 2 shows the relationships among the programming components of DISPLAYTRAN. The supervisor primarily serves as an interface between the DISPLAYTRAN components and the operating system. All I/O, system, or problem programs are scheduled through the supervisor to the operating system, and all interrupts are posted back to the supervisor. Thus, terminal status and terminal switching are centrally controlled, and other components of DISPLAYTRAN are relieved from handling the multi-user aspects of the system. All system commands to be described here are acted upon by the supervisor. In the Naval Weapons Laboratory two-terminal system, switching of interpretive execution from one terminal to the other is under supervisor control and is based on I/O requests and a time increment. More complex algorithms were investigated, but with the two-terminal system, the simpler approach is adequate.

system components

Associated with each terminal is a buffer called a *terminal record* that contains the terminal FORTRAN program. When a statement is entered into the system, the supervisor calls the translator routine to decode the symbolic statement into a string of operands and operators (i.e., using Polish notation) and stores the results in the terminal record. Because this area is fixed in size, an overflow causes a diagnostic message to be displayed on the display screen. The programmer may then divide the program into smaller subroutines because the system operates on one subroutine at a time.

Figure 2 DISPLAYTRAN programming system



Statements continue to be accumulated until a system command (or overflow) defines some alternate action, such as START. A START command causes the supervisor to give control to the DISPLAYTRAN interpreter, which analyzes the Polish string, distinguishing between operands and operations, and executes the specified operations.

Execution of FORTRAN statements continues in this manner until an END or system command terminates the program. Certain types of statements are handled a little differently by the interpreter. For example, a FORTRAN CALL results in the replacement of the calling subroutine by the terminal record of the called subroutine. In the case of calling a binary subroutine, which does not have a terminal record, the interpreter fetches the called routine and gives it control. Because previously compiled routines have restricted I/O and are assumed to be checked out, time slicing is suspended while they are executing. Standard library routines, such as sine and cosine, fall into this class. Input/output statements are another type that the interpreter handles differently. Defined data are transferred between main storage and the 2250 by the interpreter through a call to the supervisor, which initiates the I/O operation.

User languages

The previous discussion suggests some system aspects of the DISPLAYTRAN command language, the FORTRAN IV subset, the

Table 1 DISPLAYTRAN command language

Command class	*Command*	*Description*
Terminal	SIGN ON	Activate system via user name and number (identification)
	SIGN OFF	Deactivate system
Program control	PROGRAM	Begin constructing current program; destroy other program
	RESUME PROGRAM	Continue constructing current program following use of other commands
	SAVE	Enter user identification and copy current program on disk
	LOAD	Retrieve program from disk via user identification
	PURGE	Delete program(s) on disk via user identification
Execution control	EXECUTE	Begin executing program at specified statement
	STOP	Halt program execution; indicate last statement executed
	CONTINUE	Resume execution after execution error, or GUARD, PAUSE, STOP commands
	RESET	Cancel effects and results of executing a program
	CLEAR	Cancel effects of all previous commands of the "debug" class
Output control	OUTPUT OPTIONS	Specify output for "debug" and "display" command classes
	INCREMENT	Specify increment for generating line numbers differing from system standard
Program modification	MODIFY	Insert, replace, or delete program statement or statements (MODIFY sequence)
	END MODIFY	Terminate a MODIFY sequence and return to RESUME PROGRAM status
	RENUMBER	Renumber all statements in program
Debug aids	SNAPSHOT DUMP	Print leftmost variable of arithmetic assignment statement whenever it changes
	END SNAPSHOT DUMP	Cancel SNAPSHOT DUMP
	BRANCH TRACE	Print origin and destination (line numbers) of every transfer
	END BRANCH TRACE	Cancel BRANCH TRACE
	GUARD	Do not execute guarded statements—to procede, issue CONTINUE command
	END GUARD	Cancel GUARD

Table 1 DISPLAYTRAN command language (cont'd)

Command class	*Command*	*Description*
	SUBPROGRAM TRACE	Print notice when program calls library subroutine, e.g. sine
	END SUBPROGRAM TRACE	Cancel SUBPROGRAM TRACE
Display	LIST	List specified area of program
	DUMP	List variables with current values in specified area of program
	DUMP CHANGES	List variable and value changes since beginning on last DUMP
	UNUSED	List unused program areas and variables not "set"
Miscellaneous	CALCULATE	Enter one arithmetic assignment statement, execute, and print results (do not save statement or results)
	SET	Same as CALCULATE except save results

graphic language, and the compiled routine facility. We now describe briefly how one uses the languages to build and debug programs and to produce displays. Capabilities and restrictions on building and executing binary subprograms are also briefly discussed.

command language

The DISPLAYTRAN command language, listed in Table 1, directs the overall operation of the system. At the present time, there are thirty-one command words, shown in the table grouped into eight functional classes. For purposes of discussion, we shall simply divide the commands into two major groups: system commands and debugging aids.

All commands are entered via the 2250 function keyboard, checked at the time of entry, and the console operator is notified of an incorrect command or command sequence. When the console operator issues a command, it appears on the screen for visual inspection.

In general, system commands are procedural in effect, performing such functions as identifying the console operator to the system at the beginning of his session (SIGN ON) and the complementary function of terminating his session (SIGN OFF). The PROGRAM command enables the console operator to construct his FORTRAN program, and the LOAD command brings in a previously entered program. Other such commands allow programs to be saved or destroyed.

As with commands, FORTRAN source programs are also entered into the system on a statement-by-statement basis. A number of system commands are interspersed with the FORTRAN statements. For example, the command EXECUTE causes FORTRAN statements to be executed. Commands are also needed to resume program

Table 2 DISPLAYTRAN FORTRAN IV

Statement	*Normal sequencing*	*Executable or nonexecutable*	*Order in source program*
$a = b$	Next statement	E	Anywhere except preceding specification statement
ASSIGN n to i	Next statement	E	Anywhere except preceding specification statement
BACKSPACE i	Next statement	E	Anywhere except preceding specification statement
CALL	First statement of a called program	E	Anywhere except preceding specification statement
COMMON	Next statement	N	Must precede all executable and DATA statements
COMPLEX	Next statement	N	Must precede all executable or DATA statements
CONTINUE	Next statement	E	Anywhere, but usually as last statement in DO routine
DATA	Next statement	N	Must precede all executable statements
DIMENSION	Next statement	N	Must precede all executable or DATA statements
DO	Normal DO sequencing, then next statement	E	Anywhere except preceding specification statement
DOUBLE PRECISION	Next statement	N	Must precede all executable or DATA statements
END	Terminates compilation program	N	Must be last program statement
END FILE	Next statement	E	Anywhere except preceding specification statements
EQUIVALENCE	Next statement	N	Must precede all executable and DATA statements
EXTERNAL	Next statement	N	Must precede first appearance of subprogram name in executable statements
FORMAT	Next statement	N	Anywhere except preceding specification statements
FUNCTION	Next statement	N	Only as first statement of FUNCTION subprogram
GO TO n	Statement n	E	Anywhere except preceding specification statements
GO TO i, $(n_1, n_2, \ldots, n_m)$	Statement last assigned to i	E	Anywhere except preceding specification statements
GO TO $(n_1, n_2, \ldots, n_m)$	Statement n_i	E	Anywhere except preceding specification statements
IF (a) n_1, n_2, n_3	Statement n_1, n_2, n_3 if $a < 0$, $a = 0$, or $a > 0$	E	Anywhere except preceding specification statements
IF (t) s	Statement s if t is true; next statement if t is false	E	Anywhere except preceding specification statements
INTEGER	Next statement	N	Must precede all executable or DATA statements
LOGICAL	Next statement	N	Must precede all executable or DATA statements
PAUSE	Next statement	E	Where temporary halt is desired
PRINT	Next statement	E	Anywhere except preceding specification statements
PUNCH	Next statement	E	Anywhere except preceding specification statements
READ	Next statement	E	Anywhere except preceding specification statements

Table 2 DISPLAYTRAN FORTRAN IV (cont'd)

Statement	*Normal sequencing*	*Executable or nonexecutable*	*Order in source program*
REAL	Next statement	N	Must precede all executable or DATA statements
RETURN	First statement or part statement following reference to subprogram	E	Must be placed in subprogram where return to calling program is desired
REWIND	Next statement	E	Anywhere except preceding specification statements
STOP	Terminates execution	E	Where program termination is desired
SUBROUTINE	Next statement	N	Only as first statement of SUBROUTINE subprogram
WRITE	Next statement	E	Anywhere except preceding specification statements

construction after use of other commands and to stop or continue program execution.

Much of the benefit of a conversational system derives from the fact that it permits on-line debugging; therefore, many DISPLAYTRAN commands are used as debugging aids. Notice how many of the commands in Table 1 relate to creating program or variable listings and traces. Many debugging commands are in the class of debugging aids, while others are scattered among the other classes. In general, debugging aids enable the user to monitor the execution of his program and keep informed of changes in values of variables. The output device for debugging commands may be specified as either the 2250 display screen or the 1053 printer; if neither is specified, the 2250 is selected by default. An indication that a debugging aid has been initiated or cancelled is always recorded on the printer.

FORTRAN IV

FORTRAN IV is the second major language that the DISPLAYTRAN user applies, and it consists of the statements, expressions, and operations used to write source programs. Table 2 gives a complete listing of FORTRAN IV source program statements acceptable to DISPLAYTRAN, their sequence in interpretive execution, and their order when used in a source program.

As previously mentioned, the DISPLAYTRAN system does not compile FORTRAN source programs into machine language as is conventionally done on batch processing systems. Rather, a FORTRAN source program is entered into the system, one statement at a time, through the 2250 keyboard. The system translates each statement to a Polish string and stores the string for later execution. DISPLAYTRAN checks each statement for syntax errors, and immediately notifies the user that the statement is in the system and if any errors were detected.

FORTRAN input/output statements are limited to the 2250. Initiated by a START command during the execution phase, the system interprets the Polish strings, and the resulting computation or debugging information is displayed or printed.

Several advantages are gained by using the interpretive method:

- Multiple console operators are more easily serviced concurrently because the system has complete control.
- More extensive debugging is provided.
- The system immediately responds to syntax errors on a statement-by-statement basis.
- A console operator can interrupt a program at any execution point, perform some function, then continue execution.

Storage requirements for Polish strings are less than storage requirements for machine language programs. The main disadvantage of the interpretive method of program execution as compared to the compiler approach is that execution time is increased by at least an order of magnitude.

graphics language

The third language available to the DISPLAYTRAN user is the graphics language, which consists of FORTRAN CALL statements that permit access to a set of subroutines. These subroutines permit the FORTRAN programmer to program the 2250 display unit at the FORTRAN level. The subroutines provided with DISPLAYTRAN are similar to those in the graphics subroutine package described by Rully in this issue, and the organization of DISPLAYTRAN is such that that package could be substituted for the subroutines provided. One of the major differences is the naming capability provided with the subroutines.

With appropriate programming, one can perform such tasks as displaying, deleting, expanding, contracting, or rotating images. Some of the capabilities of the graphics language are summarized here.

Figure 3 Graphics storage areas

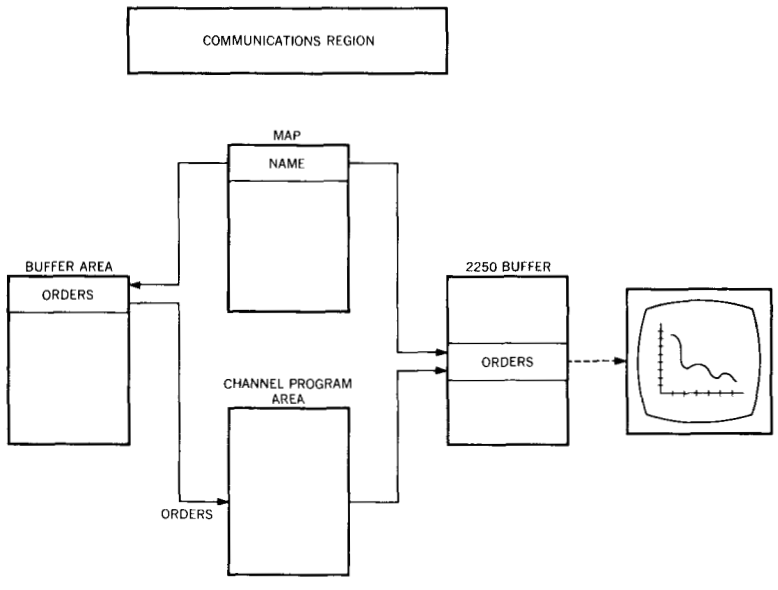
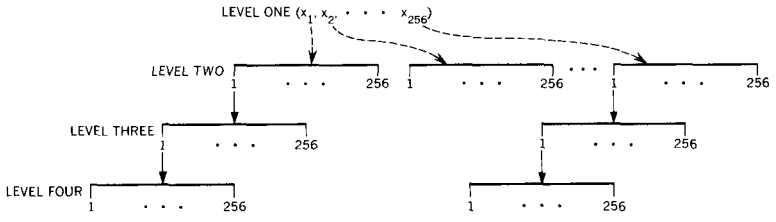


Figure 4 Graphics naming levels



- Draw coordinates—rectangular, polar, logarithmic
- Display points, symbols, or vectors
- Label coordinates and images
- Count numbers of points (or vector ends)
- Display text
- Delay computation while waiting for interrupt
- Identify portion of displayed image touched by light pen
- Tag image 0 or 1 and retrieve tag
- Enter alphanumeric data from keyboard
- Read contents of any portion of the 2250 buffer into main storage

The graphics routines make use of four areas in main storage: a buffer area, a channel program area, a communications region, and an area called *map*. The communications region is for storage of such information as the names of graphic images or grid limits that may be later needed by other routines. An array is provided by the user for each of the other three areas.

In order to display an image on the 2250, the necessary 2250 orders must be constructed by calls to graphics subroutines, which place the orders and their associated data bytes in the buffer area. At the same time, the name to be associated with the image generated by these orders and the number of bytes occupied by the orders is placed in the map, as shown in Figure 3. A pointer to the orders in the buffer area is also placed in map. The console operator supplies a name to a graphic image by calling a subprogram with the name as an argument, and then calling the graphics subprograms that generate graphics orders. There is a capability to delete names and associated images from the system.

The graphics system provides four levels of naming with 256 unique names at each level, as shown in Figure 4. Thus, each image can be identified by referring to its name. Further, a group of images can be identified by referring to the appropriate level in the tree to which the sublevels are attached. Likewise, distinct portions of any image may be broken into components by assigning sublevel names.

When the user is ready to actually have the image displayed, its name is located in map, and the information there is used in constructing a channel program in the channel program area, causing the orders to be transferred from the buffer area to the 2250 buffer and the image to be displayed. The 2250 buffer address to which the orders will be transferred is also placed in the map.

If a console operator detects an image with the light pen, which may be one of many being displayed simultaneously, the name of the image detected is passed back to the FORTRAN program via association between the 2250 buffer address at the time of interruption and the map entries.

compiled subprograms

Although not a language, one other programming facility available to the DISPLAYTRAN user is the ability to call and execute previously compiled subprograms from his FORTRAN interpretive routines. These subprograms must be added to the DISPLAYTRAN-FORTRAN library via a special update program and must adhere to several restrictions:

- Size must not exceed the terminal buffer space.
- Calls are from and returns are to a DISPLAYTRAN program.
- Returns to the DISPLAYTRAN program are made in accordance with specified programming conventions.
- Input/output is not permitted.
- Data must be referenced in FORTRAN CALL statements.
- The subprogram must be debugged.
- Time slicing is not possible.

This facility allows a programmer to mix previously compiled subprograms and interpretive routines as one program. Of course, once a symbolic program is checked out, a programmer can compile that program by a FORTRAN compiler and execute it as a background job.

Summary comment

Begun as an exploratory development in 1964, DISPLAYTRAN has proved itself in operation, and it is continuing to be improved especially in the areas of performance and capability. Being added is the preloading of symbolic programs from a card reader.

For the Naval Weapons Laboratory, which is mainly FORTRAN IV-oriented, the system provides means for efficient FORTRAN program writing, debugging, and maintaining. Graphic displays aid programmers, engineers, and scientists according to their needs. DISPLAYTRAN is a nondedicated system and is compatible with OS/360.

It is possible to modify DISPLAYTRAN to become a production tool instead of an experimental facility. Additional capabilities could be incorporated as well as means for supporting other types of terminals that might be needed in a time-sharing environment. The fact that DISPLAYTRAN is capable of producing useful work makes it desirable to further exploit this system.

CITED REFERENCES

1. E. L. Jacks, "A laboratory for the study of graphical man-machine communication," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **26,** Part I, 343–350 (1964).
2. S. H. Chasen, "Man-computer graphics," *Lockheed Quarterly,* Lockheed Corporation, Marietta, Georgia (Summer 1965).
3. The Program Language ANalyzer (PLAN) with PLAN Graphics Support (PGS) discussed by Chen and Dougherty in this issue.
4. A. D. Parker, "Graphical communications in an on-line system," *On-Line Computing Symposium Proceedings,* UCLA and Informatics Corporation (February 1965).
5. J. D. Joyce and M. J. Cianciolo, "Reactive displays—improving man-machine graphical communications," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **31,** 713–721 (1967).
6. A. Ruyle, J. W. Brackett, and R. Kaplow, "The status of systems for on-line mathematical assistance," *Proceedings of the 22nd National Conference of the Association for Computing Machinery* **P-67,** 151–167 (1967).
7. R. A. Morrison, "Graphic language translation with a language-independent processor," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **31,** 723–731 (1967).
8. R. V. Smith, *The Electronic Coding Pad,* IBM Thomas J. Watson Research Report NC-731, Yorktown, New York (1967).
9. T. G. Stockham, Jr., "Some method of graphical debugging," *Proceedings of the IBM Scientific Computing Symposium on Man-Machine Communication,* 57–72 (1965).
10. T. M. Dunn and J. H. Morrissey, "Remote computing, an experimental system—external specifications," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **25,** 413–422 (1964).
11. J. M. Keller, E. C. Strum, and G. H. Yang, "Remote computing, an experimental system—internal design," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **25,** 425–443 (1964).
12. Graphics Subroutine Package (GSP) discussed by Rully in this issue.
13. A. Hurwitz, J. P. Citren, and J. B. Yeaton, "GRAF: Graphic additions to FORTRAN," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **30,** 553–557 (1967).
14. W. A. Clark, "The functional structure of OS/360, Part III, Data management," *IBM Systems Journal* **5,** No. 1, 30–51 (1966).