System Independence for Interactive Computer Graphics Application Programs

H.G.Bown, C.D.O'Brien, R.E.Warburton and G.W.Thorgeirson
Communications Research Centre
Department of Communications
Ottawa, Ontario

ABSTRACT

This paper discusses the design of a general purpose interactive graphics system. The system is intended for use by application programmers, and its purpose is to simplify the writing of interactive graphic programs. The major system design goal is to achieve a high degree of environment independence through software portability and the concept of a virtual display terminal. The paper presents the requirements of such a terminal and develops a set of commands for addressing it. These commands are refered to as Graphical Task Instructions (GTI). A conceptualization of a general purpose graphics system is presented where the intent is to separate the application dependent and system dependent functions. The paper outlines the advantages and disadvantages of a dual processor graphics system where one processor is dedicated to the provision of the virtual display terminal and the other processor is responsible for the application program execution.

ABRÉGÉ

Le présent document traite d'un système graphique interactif universel conçu à l'intention des programmeurs d'applications en vue de simplifier la rédaction des programmes graphiques interactifs. Le but principal de ce système est de réaliser un haut niveau d'indépendance vis-à-vis des conditions d'utilisation grâce à la portabilité du logiciel et au concept d'un terminal d'affichage virtuel. Le document présente les exigences d'un tel terminal et expose un ensemble de commandes permettant d'y accéder. Ces commandes sont appelées "Instructions relatives aux tâches graphiques". On présente une conceptualisation d'un système graphique universel qui vise à séparer les fonctions qui dépendent des applications et celles qui dépendent du système. Le document expose les avantages et les inconvénients d'un système graphique à deux processeurs dont l'un sert au terminel d'affichage virtuel et l'autre à l'exécution du programme d'applications.

INTRODUCTION

This paper discusses methods to achieve a high degree of environment independence in the design of a general purpose interactive computer graphics system. A graphic system exhibits environment independence when the hardware and software implementation details are made invisible to the application programmer. Environment independence also implies that new advances in the evolving computer graphics technology can be accommodated with a minimum of programming effort.

Most currently available graphics programming systems (particularly manufacturer supplied graphics software packages) utilize the hardware of an interactive graphics systems in an environment dependent manner [1,2,3,4]. These systems differ considerably in design and performance and can be classified either as stand-alone systems or terminals of varied intelligence on a timesharing port. These systems reference their particular hardware in a machine dependent manner and as such, application programs cannot be transported from one machine to another without considerable re-writing of software. Recently [5,6], attempts have been made to develop systems which are easily programmable and incorporate software which is both portable and exhibits a high degree of graphical device independence.

A VIRTUAL DISPLAY TERMINAL APPROACH

A requirement of graphic system environment independence is software portability. This can be achieved by utilizing either a standardized base language such as FORTRAN or by developing a specialized portable graphics language by utilizing a translator writing system. The latter approach is recommended because it presents an opportunity to develop a language syntax that is particularly well suited to handle the interactive environment associated with an interactive computer graphic system. An interactive computer graphics language called 'IMAGE' [6] that meets the above requirements is currently under development at the Communications Research Centre.

An additional requirement for environment independence is independence of the system from the display terminal hardware being utilized. The dividing line between user application software and system software as presented in Figures 1 and 2 suggests a solution to this problem. It is proposed that this hardware independence can be achieved by defining a virtual display terminal with specific capabilities. All communications with this virtual display terminal are made in such a manner as to be independent of any particular realization of the virtual display terminal. For example, an application program is unaware of the technique being employed in the virtual display terminal when it requests that a line, character or symbol be

generated. In fact, the application program is unaware of whether a random access refresh, a raster refresh or a storage display is being utilized. One virtual display terminal may perform the function of vector and character generation by hardware, whereas another may perform the same functions entirely by software. In addition, a set of instructions must be provided to enable the terminal to be referenced in a hardware device independent manner [6]. A proposed set of commands for addressing such a virtual terminal is presented in Table 1. These commands are referred to as Graphical Task Instructions (GTI) and are subdivided into seven different catagories as shown below.

I. Display Generation
II. Co-ordinate Specification
III. Graphical Modifiers
IV. Status Mode Setting
V. Subpicture Definition
VI. Display File Modifiers
VII. Interactive Device Control

The GTI instructions have been defined in such a manner as to be independent of any particular coding scheme. Different coding schemes would simply require modifications to the GTI code generator and decoding routines as indicated in Figure 2. The GTI instructions form an extensible set allowing for future expansion to accommodate new hardware and software innovations.

GRAPHIC SYSTEMS CONCEPTUALIZATION

A conceptualization of a general purpose graphics system is presented in Figure 1 where the intent is to separate the application dependent and system dependent functions. Current graphics systems are readily represented by this conceptualization. The application programs are usually written in some high level language like FORTRAN, EULER/G [7], or GINO [5]. The complexity of the various systems software modules indicated in the boxes does not depend upon whether the system is stand alone or an intelligent terminal to some larger computer. These software modules are written only once for a particular class of hardware and vary in complexity depending to a large extent on the sophistication of the graphics hardware available (e.g. vector generator, character generator, and transformation hardware). The definition of a virtual display terminal permits the separation of these functions into independent processes. A particular realization of a virtual display terminal may be implemented in a single processor system, but the separation of functions suggests a dual processor system design (Figure 2). This is achieved by dedicating one processor to the task of providing the virtual display terminal capability while the other processor is responsible for the application program execution. The class of the communications link between the two processors and the relative sizes of the

processors define whether the system is stand alone or a satellite graphics system.

The advantage of a single processor system for the realization of a virtual display system is its lower cost in hardware, software and communications. The dual processor system provides the advantage of faster performance due to the display housekeeping and I/O device handling being performed by the second processor. Also the graphics terminal is now a complete system utilizing the second processor and can be driven by any computer which outputs GTI, thus providing a high degree of flexibility and standardization.

CONCLUSION

The implementation of a dual processor system as presented in Figure 2 is nearing completion at the Communications Research Centre. The terminal dependent software is being written in MACRO 11 assembler for a PDP 11/10 computer with 8K core memory. The terminal independent software is being written in IMAGE [6] and/or FORTRAN for the PDP 9 and PDP 11/40 computers. For both the PDP 9 and PDP 11/40 implementations, the communications link between the host computer and the PDP 11/10 virtual graphics terminal is a 16 bit parallel interface. The GTI commands transmitted over this link use a particular choice of opcode and parameter data structure well suited for decoding and storage on a 16 bit computer (e.g. PDP 11/10).

This approach to achieving environment independence suggests that the translator for application programs emit GTI code, thus providing a standard code for addresing graphics terminals, not dissimilar to the use of ASCII code for addressing most alphanumeric terminals. Thus, graphical system independence can be achieved through:

1) application program portability made possible through a portable translator writing system or a standardized base language,

2) and, the use of a virtual terminal concept utilizing the graphical task instructions as presented in Table 1.

Application programs could then be moved from machine to machine and could take advantage of future GTI graphic terminals of more speed and power without requiring reprogramming.

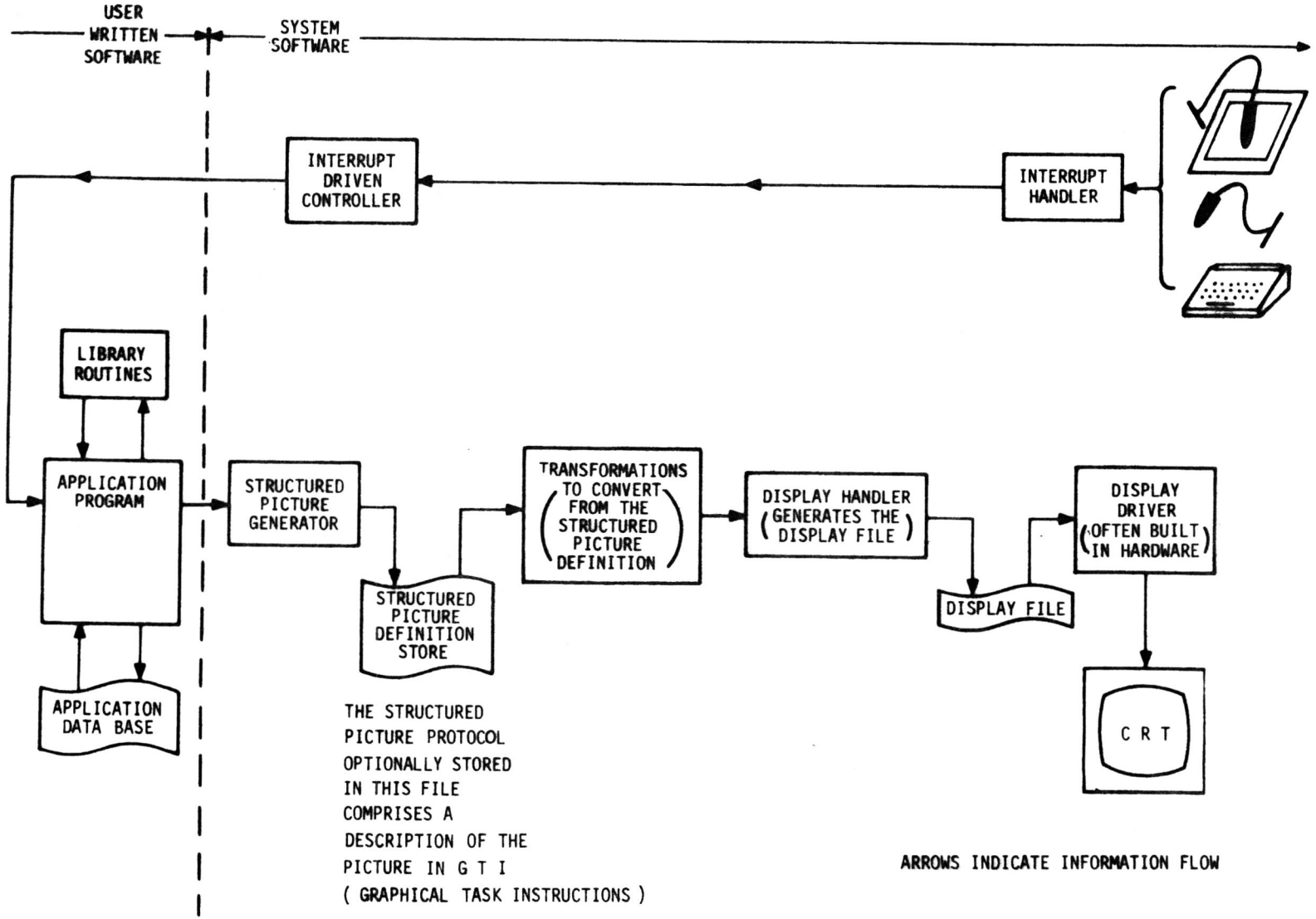

Figure 1 Conceptualization of a Graphics System

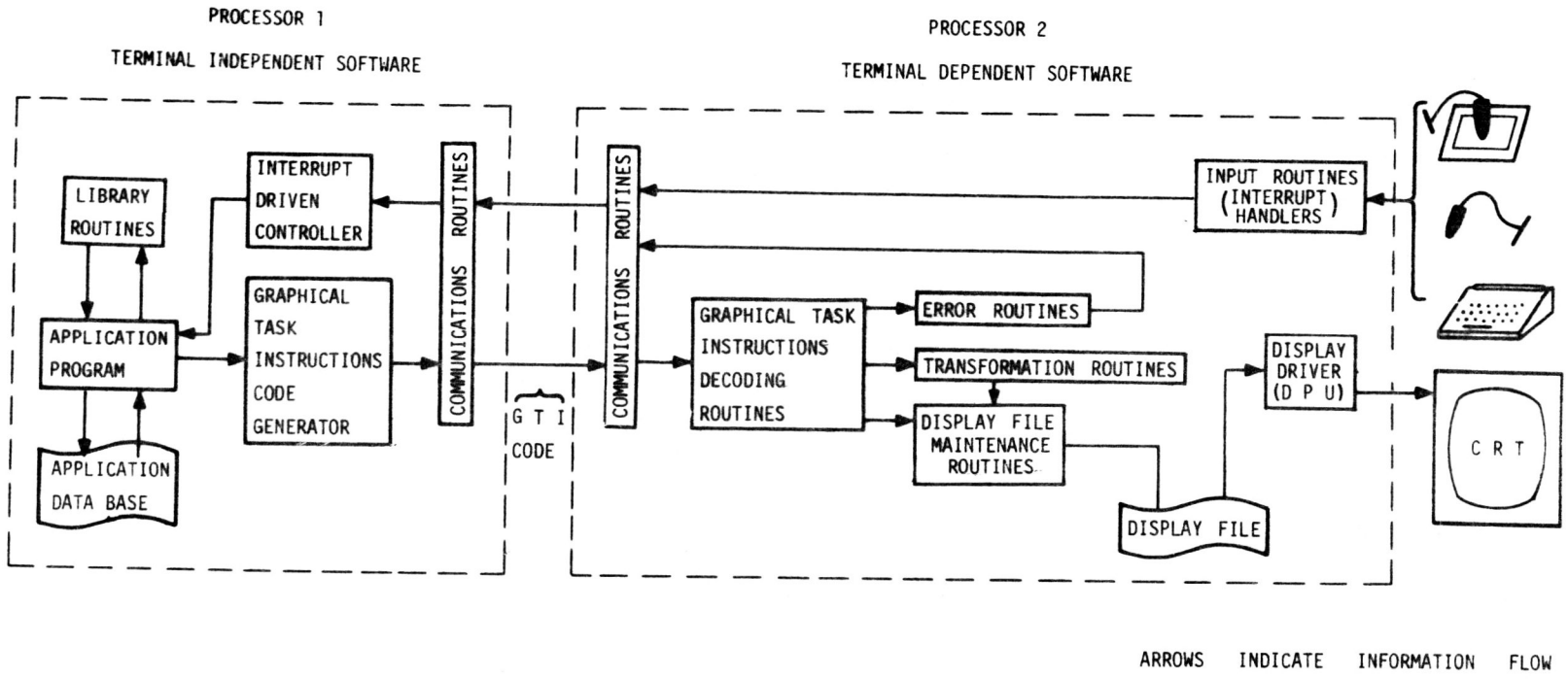

Figure 2 A Dual Processor Graphics System

TABLE 1

GRAPHICAL TASK INSTRUCTIONS (GTI)

MNEMONIC	FORMAT	DESCRIPTION
I. Display Generation Instructions		
LINES	opcode n rel x1 rel y1 : rel xn rel yn	- draw a sequence of concatenated lines each having displacements xi and yi respectively. Lines are defined on a conceptual page , transformed and then drawn on the screen.
LINES THRU	opcode n abs x1 abs x2 : abs xn abs yn	- draw a sequence of concatenated lines through specified points on the conceptual drawing page; then move the origin of the conceptual page to either the current beam position or else to a point specified by SET.
LINE TO	opcode abs x abs y	- draw a line from the current set beam position to the point (x,y) on the page.
SPECIFY COORD. (SET)	opcode abs x abs y	- set the drawing beam to the point (x,y) in the current page co-ordinate system.
CHARACTERS	opcode n char 1,2 : char n	- draw a sequence of characters beginning at the current beam position. ROT & REFL modifiers have a limited effect on char. strings.
ARC	opcode m parm 1 : parm n	- draw an arc using the given parameters. 'm' specifies the mode. (An arc is defined by 3 x,y points, or 2 points and a slope, etc.).
POINT	opcode	- draw a point at the current beam position.
SYMBOLS	opcode sym #	- draw a symbol from the library of symbols.

II. Co-ordinate Specification

PAGE	opcode xmin xmax ymin ymax	- defines the user co-ordinate system & specifies values for unit x & y displacements on the screen.

III. Graphical Modifier Instructions

ROTATE	opcode angle	- rotate a graphical item by the stated angle.
REFLECT	opcode parm	- reflect graphical items about the line y=a*x where a=tan(parm)
SCALE	opcode factor	- scale graphical items by a factor.
TRANSLATE	opcode dx dy	- translate graphical items by an amount dx in the x-direction & dy in the y-direction.
WINDOW	opcode xmin xmax ymin ymax	- specifies a region for the display of graphical items with clipping being performed on all graphical entities not fully in the region specified.
WITHIN	opcode xmin xmax ymin ymax	- specifies a region of the page which graphical items are to be mapped onto, with the graphical items being specified in terms of conceptual page co-ordinates.
END MODIFIER	opcode	- remove the last specified graphical modifier from the stack of current graphical modifiers.

IV. Status Mode Setting Instructions

INTENSITY	opcode level	- specifies the intensity level for graphical items (0 - 1).
COLOUR	opcode value	- specifies the colour of a graphical item.
LINE TEXTURE	opcode type #	- specifies line texture (solid, dash, dot-dash, etc.)

FLASH ON	opcode	- specifies that graphical items to follow are to flash repeatedly when being displayed.
FLASH OFF	opcode	- disable FLASH.
CHARACTER SET SPECIFICATION	opcode set #	- specifies the character set to be used.
SYMBOL SET SPECIFICATION	opcode set #	- specifies the symbol set to be used.

V. Subpicture Definition Instructions

BEGIN SUBPICTURE	opcode subp #	- delimits the beginning of a new subpicture.
END SUBPICTURE	opcode	- delimit the end of a subpicture definition.
USE SUBPICTURE	opcode subp #	- cause a subpicture to be re-used.

VI. Display File Modifier Instructions

a) delimiting

INVIS	opcode	- specify the beginning of an entity pending later display.
VIS	opcode	- add INVIS entity to the display.
TAG	opcode	- delimit a picture for identification.

b) control

ERASE	opcode begin tag # end tag #	- delete specified tagged objects from the display file.
CLEAR	opcode	- clears the display file.
ON	opcode	- initiates the display.
OFF	opcode	- suspends the display.
WINK	opcode	- cause an ON/OFF/ON sequence for interaction acknowledgement.

VII. Interactive Device Control Commands

DEVICE ON	opcode device #	- enable the specified virtual device.
DEVICE OFF	opcode device #	- disable a specified device
ASSIGN DEVICE	opcode device # funct #	- define the device to be used to perform the specified function.
SET MARKER POSITION	opcode x pos y pos	- specifies the position at which a marker will appear on the current page.
MARKER MODE	opcode code	- specify the marker constraints 0 marker on 3 horizontal 1 marker off 4 vertical 2 fixed 5 free
	opcode angle	- constrained to a tilted line
SKETCH RESOLUTION	opcode factor	- set the minimum significant sketcher position change.
KEYBOARD ACTIVATION CHAR	opcode n char 1,2 : char n	- enable specified characters for use as activation characters on entry.
X & Y IDENTIFIER POS. REQ.	opcode	- request the return of the co-ordinates of the last identifier interrupt.
MARKER POS. REQ	opcode	- request the return of the marker position.

- -

The following GTI codes are returned from the graphics terminal.

IDENTIFIER RETURN TAG	opcode tag #	- return the tag number for the object interactively selected.
IDENTIFIER RETURN POS	opcode x pos y pos	- return the x & y co-ordinates indicated by the identifier.

MARKER POS. REQ.	opcode x pos y pos	- return the current x & y marker position.
SKETCH POS. RET.	opcode xpos ypos	- return an x,y point visited by the stylus from the queue of such points.
CHARACTER STRING RETURN	opcode n char 1,2 : char n	- return the input character string terminated by the the activation character.
PUSHBUTTON RETURN	opcode pb #	- upon a pushbutton interrupt return the pushbutton code.
VALUATOR SETTING RETURN	opcode valu # value	- upon a significant change in valuator setting return the value.
ERROR REPORT 1	opcode error #	- error report from the terminal software.
ERROR REPORT 2	opcode error #	- error report to the terminal software.

REFERENCES

1) "GRAPHIC-15 Programming Manual", DEC-15-ZFSA-D, Digital Equipment Co., Maynard, Mass.

2) "BASIC/GT Language Reference Manual", Digital Equipment Co., Maynard, Mass.

3) Bown,H.G., Hartman,W.A. and Warburton,R.E., "Applications of the Interactive Computer Graphics Language, ICPL", NRC 3rd Man-Computer Communications Seminar, May, 1973.

4) "GRAPPLE Language Reference Manual", Bell Nothern Research, Edition 4.0 Sept. 1973.

5) Woodsford,P.A., "GINO: Graphical Input/Output", University of Cambridge Computer Aided Design Group, June, 1969.

6) O'Brien,C.D., "IMAGE - a language for the Interactive Manipulation of a Graphics Environment", M.Eng. Thesis, Carleton University, Ottawa, Canada, 1975.

7) Newman,W.M., Gouraud,H. and Oestreichr,D.R., "A Programmer's Guide to PDP/10 EULER". Univ. of Utah, Report No. UTEC-CSc-70-105, June 1970.