1987

An expert system advisor for acquiring graphics devices /

Patricia L. Doe
Lehigh University

AN EXPERT SYSTEM ADVISOR FOR
ACQUIRING GRAPHICS DEVICES

by

Patricia L. Doe

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1987

This Thesis

is Accepted and Approved

in Partial Fulfillment of

the Requirements for the Degree

of

Master of Science

August 12, 1987
(date)

Professor in Charge

Chairman of Department

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. John C. Wiginton, for his support. I also wish to extend my gratitude to my family for all their support. A very special "thank you" to my husband, Larry, and to my sons, Lawrence, Jr. and Christopher.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 3

CHAPTER 4

CHAPTER 6

CHAPTER 7

CHAPTER 8

ABSTRACT

This thesis implements a prototype expert system designed to provide a method by which to choose graphics terminals. The system was developed using Turbo Prolog Version 1.1 on a Zenith PC/158 with 640K bytes of memory. Activities included reviewing the many aspects that need to be considered when choosing either a graphics software package or hardware device. A knowledge-based system assists the untrained user in making decisions in acquiring the necessary tools. Not only does the user have to wrestle with the many tools available but also must understand the environments with which he must interact in his organization.

A review of graphic output most applicable to analysis, presentation, and publication applications is included. Graphics terminals are discussed including medium performance models typically used in analysis. Hardcopy units will vary from single user, low-end applications to high-end, high volume production output devices. The scope of the project has been constrained initially to include a subset of available software and hardware devices.

The development of a graphics expert system proved to be an appropriate goal in applying expert system techniques. Additionally, Prolog was found to be a good choice as the development tool.

CHAPTER 1

INTRODUCTION

1.1 Graphics as a Business Tool

It is recognized that computer graphics can be an excellent tool in engineering, data processing, marketing, planning and financial management. Expansive growth and innovation in the graphics arena can lead customers to delay making acquisition decisions. Making a choice is too confusing, while delaying can cause the loss of an organization's competitive edge.

Customers of business and data representation graphical software are continuously faced with choosing graphical output devices. A well-implemented graphics application can offer increased productivity and profitability. Most organizations find that high pay-off applications require the production of many graphs. To assure success, it is important to match a graphics medium to the application and then match the graphics device to the medium. The power of computer graphics and how to accomplish an application can be obscured by technical jargon foreign to a typical user. One purpose of the following discussion is to provide information about the capabilities of devices and suggest how to match devices and media to applications.

The word graphics can take on a variety of meanings. For the purpose of this discussion, graphics is defined as "any image created with a computer and displayed on a graphics terminal or hardcopy device".

This is represented by such examples as:

- three-dimensional modeling
- contour maps
- charts
- overheads
- 35 mm slides
- statistical charts
- mechanical design layouts
- schematics
- lines, bars and pies
- text or word charts.

1.2 Objectives

In reviewing the many aspects that needed to be considered when choosing either a graphics software package or hardware device, it became clear that a knowledge-based system could assist the untrained user in making decisions in acquiring the necessary tools. Not only does the user have to wrestle with the many tools available but also must understand the environments with which he must interact in his organization. The objective of the next step to be accomplished is the development of a prototype knowledge base assistant which will consider the previously stated questions concerning the acquisition of graphics software and hardware devices. The scope of the project will be constrained initially to include a subset of available software and hardware devices. During the first phase, the user will be able to query the assistant with English sentences which are predefined to the system. Available syntax and semantics are viewed in the online tutorial.

CHAPTER 2

COMPUTER GRAPHICS

2.1 Why Use Computer Graphics?

Studies indicate that graphics have value in aiding executive decision making. Most companies are just starting to take full advantage of what graphics have to offer. According to an 1981 research project conducted by the Applied Research Center at The Wharton School, The University of Pennsylvania, graphic presentation of information had the following effect (24):

- o Meetings were shorter by 28 percent.
- o Group consensus was faster, that is, agreement was reached 79 percent of time as compared to 8 percent of the time when graphics was not used.
- o Decisions were made faster. Approximately 64 percent of participants made decisions immediately following presentations, whereas without graphics decisions were delayed.
- o Participants making presentations were viewed as more credible, effective, and professional.

Project Impact (funded by the U.S. Air Force) sought to find the most fertile area for white collar automation. One area looked at was the preparation of visuals. Project Impact quantified the incidence of different types of visuals as follows (17):

Word Charts and Number Tables	70%
Symbol Charts and Diagrams	20%
Bar/Line/Pie Charts	7%
Artistic Charts	3%

2.2 Issues in Choosing Graphics Software

There are a number of qualities a package should have to maximize the value to its user. In choosing a comprehensive graphics software package, an examination of the specific capabilities required by a customer and his applications is necessary. The following capabilities are particularly important (16):

- user-friendly
- flexible
- multiple plots per page
- device independence
- layout intelligence
- machine independence
- pc access
- chartbooks
- immediate production of executive charts
- integration with existing databases.

Although the initial investment in an extensive, full capability software package may be higher, the added functionality for broader applications can make the investment cost effective over time.

End user friendliness means different things to each of us. In deciding on a graphics software package, the background of the individual needs to be considered. Is it a requirement that both clerical and engineering staff access the same packages? With a potentially wide range of computer literacy, is it better to acquire a command driven system, a prompting system or a menu driven one? Experienced users tend to prefer command driven systems while novice or occasional users prefer prompting or menu driven packages. Which packages are available that offer the flexibility of both menus and commands? Quality levels vary from package to package in both software and hardware capabilites.

Is the package going to be used for throw away, peer, or presentation quality graphics? Is it important that multiple plots per page be an option? If so, how difficult is it to accomplish this requirement? How many charts per page can be produced while still being able to read the lettering? The software must support a wide selection of currently available devices as well as potential future devices. Device independence allows taking advantage of many hardware manufacturers' devices.

Additionally, the software should lay out the chart to the advantage of a hardware device. Graphics devices can be very expensive and not available to all customers. Can the software support nongraphics devices for previewing output?

2.3 Hardware Issues

A related important issue concerns the hardware environment. This includes:

- o terminals and output devices
- o execution environment.

The number of installed graphics hardcopy devices in 1982 was numbered at 383,000. By 1986, 4.4 million devices were installed. Competition among graphics terminal vendors along with their growth created a mature industry. This has caused market segmentation into four primary product groupings. They are:y

(1) low-cost monochrome

(2) low-cost color

(3) high-performance 2-D

(4) personal computer - terminal combination

With the numerous graphics output devices available in today's market, difficulties arise in trying to match appropriate devices to a required application. Colorful displays are attractive but may not meet the application's final objective. Does the application require 35mm slides, transparencies, perhaps a video show, or is black and white hardcopy the most appropriate output? To decide on a device, the application must be defined. Cost and performance tradeoffs in available equipment must affect the selection. Does it need to be publication quality? Is the software portable to a reasonable selection of machines? Many orgainizations are interested in running a package on wide ranges of environments such as an IBM mainframe under MVS/XA, DEC VAX or MICROVAX under VMS, or PCs under PC/MS-DOS .

2.4 Vendor issues

When choosing a potential vendor, the following need to be reviewed.

- o stability
- o committment
- o multiple environments.

It is necessary to evaluate the product/ vendor combination for the following: market presence, reputation, customer base, product line focus (an example is business versus engineering graphics), upward compatibility, record for delivering products, quality assurance, customer support, documentation, training, users' group, maintenance support, telephone hotlines and price, to name a few considerations. Not every customer finds it desirable to create sophisticated charts especially if he is a new or occasional user. Is a chartbook available from the vendor or is it possible to create an inhouse chartbook? Chart layouts that are frequently used, that is, the same title, axis labels and so on should be accessible from a chart library. An important capability of the software is to be able to access data stored in databases. If the software cannot easily access databases, users become frustrated by reentering data. Clearly, an integrated environment where an application can grow to meet the customer's expectation is an attainable goal. This can be accomplished by porting applications among PC, Mini and Mainframe environments. Where would the customer best be served?

2.5 Which Environment is Best?

The mainframe's best use is for production graphics. Typically, this includes:

o Periodic reports -

Predefined charts, stencils or formats are created with an ad hoc application builder and combined with data. The Mainframe environment is preferred for this since it remains the major location of data from many sources and can provide batch facilities to generate daily, weekly, monthly or quarterly reports without manual intervention.

o Central Control of Output Devices -

The Mainframe provides the most cost effective environment for output devices. High-quality , high-volume devices such as laser printers, slide makers and electrostatic plotters are too expensive to attach to PCs or departmental computers.

- Data Access -

  Large databases and financial data still reside on the Mainframe. There remains the requirement for graphics to be where the data exists.

PC graphics are best suited for ad hoc graphics. What is ad hoc graphics?

- "What if" Analysis -

  It is easy and cost effective on a PC because of speed and local control to do "what if" analysis. This applies if the amount of data is relatively small and the customer wishes to cycle through several iterations of the analysis.

- Ad hoc graphics -

  A large use of ad hoc graphics is the development of production or presentation graphics applications. The PC allows creation of chart formats because the environment is conducive to the iterative process of prototyping charts.

- Local control of output devices -

  If output requirements are for relatively low quantity and quality, PC output devices will probably meet customer need while giving him local control over the device. Local devices are typically easy to use as well as able to provide fast turnaround.

- o Text Charts -

  Text charts are examples of graphics that may be appopriate for a PC . Most PC packages do well with this type of chart.

- o Freehand drawing -

  If the customer requires data independent drawings, PC software is available to aid in the generation of these type of pictures. PC devices such as the mouse and digitizing tablets are becoming more common and are especially useful for customers who are not keyboard oriented.

It was mentioned that the integration of these environments was of major interest. How can the integration or porting be accomplished? Smooth integration from one environment to another involves three major issues.

1) File transfer -

   The ability to transfer files down and up in an integrated facility and still be able to retain the graphical integrity of the chart is important.

2) Compatibility of file structures -

A graphics file in command or metafile form should be compatible to both Mainframe and PC environments. Can the PC graphics package understand the Mainframe package? Can the Mainframe understand the PC output?

3) User Interface -

If the customer requires several environments, how easy is it to interface the environments, how can the customer take advantage of the environment and how can the functionality look similiar to the customer? Do the environments look similiar to the customer so that he can avoid learning too many software packages?

CHAPTER 3

COMPUTER GRAPHICS SYSTEMS

3.1 Graphics Systems

A color graphics system is comprised of numerous hardware components and computer software that can create and display graphical information. Graphics system components include:

- o graphics terminals
- o a host computing environment
- o software
- o hardcopy devices
- o input devices

3.2 Graphics Terminals

A graphics terminal consists of a CRT or display and a keyboard. Using a CRT, a user can display graphics, receive textual output such as messages from a host computer or other users, and echo messages that he sends to other user as well as the host. A keyboard is an example of a tool which allows the entering of commands as well as data. Some keyboards are more intelligent than others when they are equipped with control and function keys. Other devices that input graphical information are the mouse, joysticks, thumbwheels, joydisks and tablets. Some devices are locally intelligent. Microprocessors perform many tasks within the terminal, which typically is more efficient because the host host cpu is offloaded.

3.3 Host

A graphical system needs some type of host computer . Several computing environments can be provided. Examples are a mainframe, minicomputer, microcomputer, a local processor or a microchip within the terminal. Today's tendency is to move to the lowest-end computing environment that can support the application. This direction is typically a lower cost environment, and usually provides a lower communications response time. In a distributed processing environment, the computing requirements are dispersed over several computing units and devices. The device best suited for a particular tasks is made available. For example, small tasks can be executed locally by a workstation. Tasks that require significant memory utilization or that are computationally intensive require a larger host environment. Many terminals are locally intelligent and capable of performing graphics commands such as drawing line segments, scaling and rotating graphical objects. An engineering workstation combines a processing unit, a terminal, file storage and input/output devices.

3.4 Programs

A computer requires an instruction set to tell it how to function. This instrtction set can take the form of software or firmware. Firmware such as read only memory chips (ROM) store instructions that cannot be changed. Software is another example of program instructions.

3.5 Output Devices

Output devices are either hardcopy or softcopy. Softcopy is the output that is displayed on a terminal. To view this same output at a later time, it is necessary to recreate it by running the program again or by retrieving it using a utility to reproduce the output or image from a file storage environment.

Hardcopy is typically used in reports for review at a later time away from the softcopy environment. Hardcopy can take many forms ranging from the very basic to very high quality. The most primitive is a line printer. A host attached printer's output is not of high quality and its use is limited to analysis or use by peers. Dot matrix printers used with personal computers display information as closely spaced dots. These devices can display text as well as graphical output.

Pen plotters usually operate with one to eight pens and are vector devices that move from point to point. Although relatively slow, the quality can be very good. However, filling solid areas can be tedious and time consuming. The range is limited to the number of pens that the plotter can hold although users can switch out pens if additional colors are required. These devices are excellent for engineering analysis, drafting and design applications.

Screen copiers copy whatever is displayed on the screen. Screen dumps take the graphics from the screen in rasterized form and send the graphics to the copy unit. A discussion of raster graphics is contained in Chapter 4. The resolution of screen copies tends to be low. The terminal that has the copier attached is locked, that is, the terminal cannot be used while the copying process takes place.

Ink jet technology sprays inks onto special paper. These copiers can provide a higher resolution output than what was on the display. Rasterizers can be used to create inkjet copies and allow the hardcopy device to be shared among a number of displays on a network.

A host rasterizer is in software form and converts graphics primitives to a raster image. This process can be memory intense and requires a large storage area. This can be a disadvantage. Another concern is the large amount of computer utilization in I/O and CPU cycles needed. An example of software that does rasterization is IBM's Graphical Data Display Manager (GDDM), although some of the newer devices are intelligent enough to do the rasterization locally within the terminal. A vector to raster converter (see Chapter 4) receives vectors from a host environment, rasterizes the data and sends it out to a raster device. A vector to raster converter saves CPU, memory and disk utilization.

Many 35 mm slide cameras are available for use in creating slides from screen copies, from the terminal video output and from sending a software generated image directly to the camera.

Expensive hardcopy units can be best used in a shared environment if the operation of the devices are made transparent to the users. This environment eliminates users fussing with paper, and pens, reduces the capital expenditure for devices and spreads other costs out. It also frees the terminal or workstation for continued work while hard copy is taking place.

Hardcopy units can be compared based on the speed of their output.

	PEN PLOTTER	PRINTER
IMAGE QUALITY	high	low to medium
PURE COLORS	up to 8	8
RESOLUTION	250-1000	100-360
SPEED (8.5 X 11)	3-10 min	approx 3 min
COST	paper $.01 acetate $.50	$.04-$.15
EQUIPMENT COST	$1000-$6000	$4400-$11500

Table 3.1
Impact Devices

	ELECTROSTATIC	COLOR ELECTROSTATIC	THERMAL	THERMAL TRANSFER
IMAGE QUALITY	medium	high	low	medium to high
PURE COLORS	1	8	1	8
RESOLUTION	125-400	200-400	50	150-200
SPEED (8.5 X 11)	20 sec	.5 -1 min	20 sec	1 min
COST	$.06-$.09	$.06-$.09	$.04 $.50	$.25 paper $.65 acetate
EQUIPMENT COST	$4400-$7900	$12000-$110000	$1000-$5000	$2000-$13000

Table 3.2
Non-Impact Devices

	INK-JET	LASER	CAMERA SYSTEMS
IMAGE QUALITY	medium - high	very high	very high
PURE COLORS	1	1	unlimited
RESOLUTION	150-200	100 color 240-300 B/W	2000-4000
SPEED (8.5 X 11)	1-4 min	.2-5 sec B/W 20 sec color	film dependent
COST	$.15-$.20	$.06	$7.00-$10.00
EQUIPMENT COST	$1000-$15000	$5000-$390,000	$1500-$200,000

Table 3.2
Non-Impact Devices-cont.

Choosing hardcopy devices can be a series of trade-offs. The purchaser needs to consider:

o resolution versus speed

o speed versus cost per copy

o resolution versus device cost

o cost per copy versus device cost

3.6 Matching Applications with Media

Timeliness in analysis media as well as accuracy make graphical display in engineering, scientific and business applications an effective business tool. An initial review may include softcopy while long term hardcopy for documentation purposes may be the final output.

Many times the type of presentation material is dependent on the environment ,where the presentation will take place. For example, 35mm slides are well suited to formal presentations, giving the impression of being more professional. Darkened rooms, however, usually allow for little participation by the audience. But then slides are considerably easier to transport. Many quality graphics software packages now support the more popular camera systems such as Matrix. Standalone camera systems such as Dicomed or Genigraphics are more oriented to computer graphics artists rather than average end-user access.

Overhead transparencies are accepted in most types of presentations, particularly those where participation by the audience is encouraged. Markers can be used to enhance or highlight information. Transparencies can be produced in several ways. The most common at this time is xerographic copies of word charts. Liquid crystal displays (LCD) have been introduced for direct projections. Pen plotters, ink-jet and color thermal technology have become popular for color output on transparency material. Flipcharts may be created on plotters and electrostatic devices that create drawings of D-size(22" X 34") and E-size (34" X 44"). Use of video displays of computer graphics is becoming more popular. Vendors are available to provide video projection systems for either personal computers or terminal based output.

Publication quality output requires laser printers for black and white or pen, ink-jet or thermal transfer for color.

Table 3.3 provides guidelines for output media.

3.7 Input Devices

The keyboard has long been a standard in inputting information at a terminal although entering certain kinds of data can take time and is tedious. Also keyboard entry of commands to do certain functions may not be the mostnatural way for the user. Examples of input devices other than keyboard are:

- o touch screen
- o data tablet
- o joystick

o trackball

o mouse

o light pen

Analysis	Soft Copy	Raster Terminals
		Storage Tube Terminals
	Black/White Paper	Demand Hardcopy
		Pen Plotter
		Laser Printer
		COM Recorder
	Color Pages	Pen Plotter
		Impact Printer
		Thermal Transfer
		Ink-Jet
		Color Xerography
Presentations	Flip Charts	Pen Plotter
		Color Electrostatic
	Transparencies	Pen Plotter
		Thermal Transfer
		Ink-Jet
		Laser Printer
		Film
	Slides	Film Recorder
Publications	Black/White paper	Pen Plotter
		Ink-Jet
		Laser Printer
		Electro-erosion
		COM Recorder
	Color	Ink-Jet
		Film

Table 3.3
Guidelines for Output Media

3.7.1 Touch Screen

A device developed to be used by anyone regardless of his computer skills, the touch screen allows input to the computer by simply touching the screen at the desired function. The X-Y coordinates of the touched location are sent to the processor for deciphering the meaning. Technologies used for touch screens are infrared, plastic membrane and capacitive. Infrared is prone to activation by dust or smoke and is also very expensive. Plastic membranes are of lower quality and malfunction from moisture and heat. Because capacitive screens are solid glass, they offer durability, visibility and accuracy.

3.7.2 Data Tablet

A tablet consists of a flat surface covering a grid of fine wires and a pen attached to the tablet. These fine wires, or conductors, are activated by the pen tip. Software interpets the meaning of the data provided. Graphics artists or CAD users are more comfortable using this input device for drawing and design applications.

3.7.3 Joy Stick

A joy stick is a lever mounted on a ball-joint mechanism. The movement of this joystick is converted to electrical impulses that control the cursor. While an inexpensive device, users have to be coordinated enough to control the device. Widely used in the computer games market, CAD and analog applications, it has not become widely accepted in other markets.

3.7.4 Trackball

Moving a trackball moves the cursor on the screen in the appropriate direction. Most trackballs are attached to a RS-232C serial port. Because it requires less dexterity than other pointing devices, more users can become adept at its applications. Although used widely in analog applications and computer games, like the joystick, it has not gained wide acceptance elsewhere.

3.7.5 Mouse

Widely accepted, the mouse has allowed computer users with minimal experience to become comfortable with the computer. Interaction among mouse, menus and graphical icons is oriented toward novice or occasional users who forget the syntax and semantics of a command driven environment.

A mouse is connected to a computer by a serial port or adapter card. A trackball on the undersurface moves the screen cursor in the required direction as the user pushes the mouse in the appropriate direction. A button in pushed when the mouse reaches the particular location activating the selected function. Several different mouse designs exist in the current market.

Examples of applications that are designed around the use of a mouse include AutoCad and Microsoft Windows. Applications that require frequent switching between keyboard and mouse or among other input devices can be cumbersome for the user. Also lack of speed and usable software can limit its use as a univeral tool.

CHAPTER 4

TYPE OF DEVICES

4.1 Raster Graphics

One of the two most popular types of display technology for terminals is raster refresh. The other is stroke storage.

Raster refresh is similiar to regular television. Also referred to as scan graphics, raster technology means moving an electronic beam across a screen while transmitting color and varied intensities. For monochrome devices, a single beam is used. For color devices, there are red, green, and blue beams. Other colors are a combination of these three beams. Because the phosphor glow has a subsecond life, the beams must be refreshed. The refresh rate is the number of times per second that the surface is again bombarded by the electron beam.

Character cells used by many desktop displays address blocks of pixels. Bit-mapped indicates that every pixel on the screen is addressable. The quality of a graph is dependent on the number of pixels as well as the rate at which the pixels are refreshed. A raster display consists of rows and columns of dots, called pixels, which can be turned on and off. Flicker is caused by a low refresh rate as well as a interleaved device. Non-interleaved means to rewrite the entire screen at one time instead of writing every other line.

Examples of non-interlaced 60 Hertz devices include:

o IBM 3179-G

o IBM 3270-PC/G

o Tektronix 41xx series.

Examples of interlaced and hence flicker prone devices are:

o Most personal computers

o IBM 3279

o Tektronix 402x series.

An additional cost of up to $2000 can be incurred for the advantage of a non-interlaced 60 Hertz device.

Raster refresh supports multiple colors, erasability, backspacing and panel fill with color. In 1984 the use of raster refresh devices accounted for 90% of CAD/CAE market and most all of medium performance market.

Seven items to look for in raster refresh terminals are:

o display refresh rate

o resolution

o screen size

o number of colors

o computer overhead

o hardcopy availability

o price.

There are three methods of generating graphics on a raster device:

- o character cell graphics
- o pixel graphics
- o vector graphics.

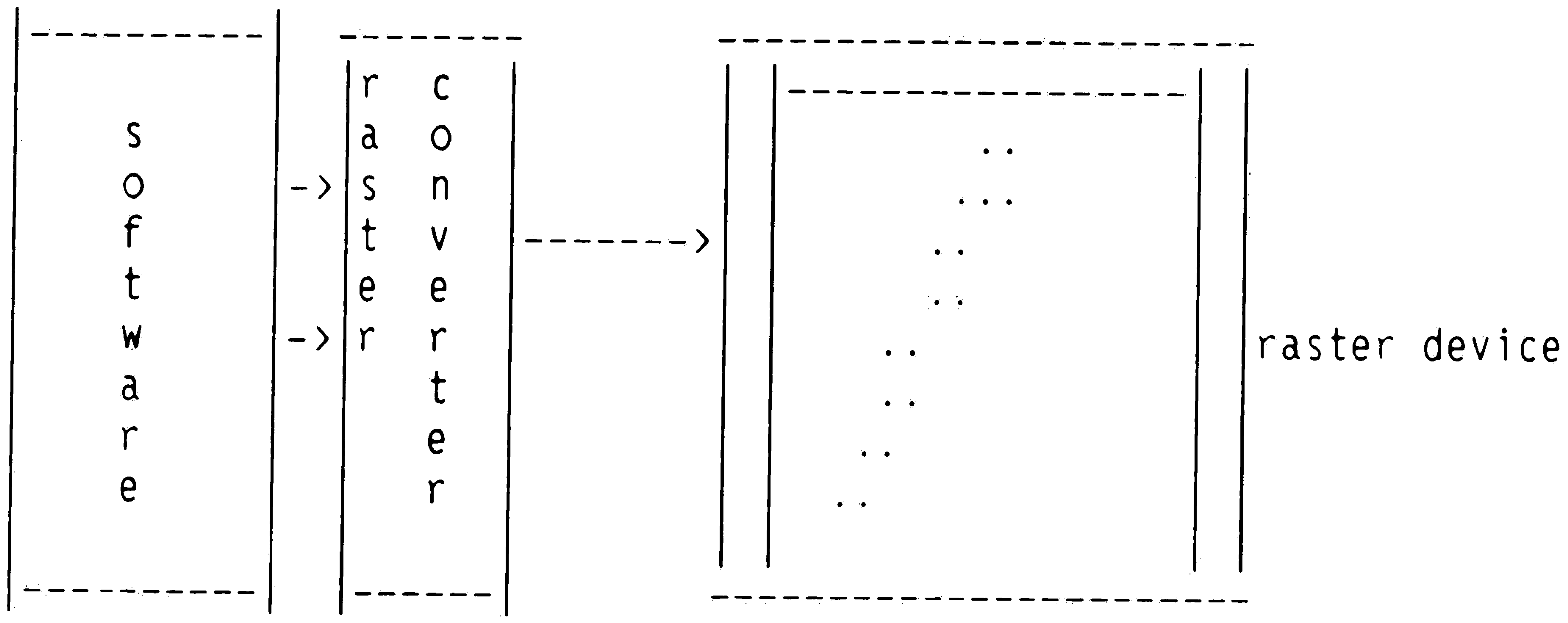


Figure 4.1
Raster Device

4.1.1 Character Cell Graphics

In character cell graphics, characters are stored as patterns in the memory of the terminal. Character cell graphics is still the technique employed in the low-end personal computer market. The limitations inherent in this technique are the reason why personal computer output has not been sufficient for presentation purposes. "Jaggies" or bumpy output results from the smallest addressable unit being a block of pixels. Output consists of a predefined set of graphics characters. Some representations become difficult. To store a large number of cells requires additional memory within the terminal increasing the cost of the device.

Graphical Data Display Manager (GDDM), an IBM program product, uses programmed symbol cells which are generated, usually on the host, and passed to the terminal. Because the cells are created as required, CPU utilization can be intensive. On devices such as the IBM 3279, symbol overflow can occur because the quantity of symbols being passed to the terminal is more than the device can accept. IBM's newer devices are more intelligent and do not use programmed symbol cells. Character cell graphics are useful for limited applications such as forms generation.

4.1.2 Pixel Graphics

In pixel graphics every pixel on the CRT is addressable either individually or by groupings. For simple traditional business graphics, this is cumbersome and no real gain is achieved. Imaging applications, however, gain considerably by being able to address individual pixels. Examples include medical imaging where the brightness of the picture changes pixel by pixel. Popular uses of pixel graphics are paint systems and pop-up menus. An example of a paint system using this technique is Apple's MacPaint for the Macintosh. Typically a device such as a mouse is used in conjunction with a software package allowing the user to create a picture on the display. A group of predefined icons or symbols can be chosen, manipulated and filled with color. Freehand objects can be created as well. Additional functions are provided with pop-up menus without having to return to a different level. Images that appear almost as photographs are excellent examples of the use of pixel graphics.

A picture is displayed when combinations of pixels are turned on at the same time. If a terminal is intelligent, its firmware interprets computations required to tell it which pixels to turn on. For an illustration of an intelligent device, refer to Figure 4.2.

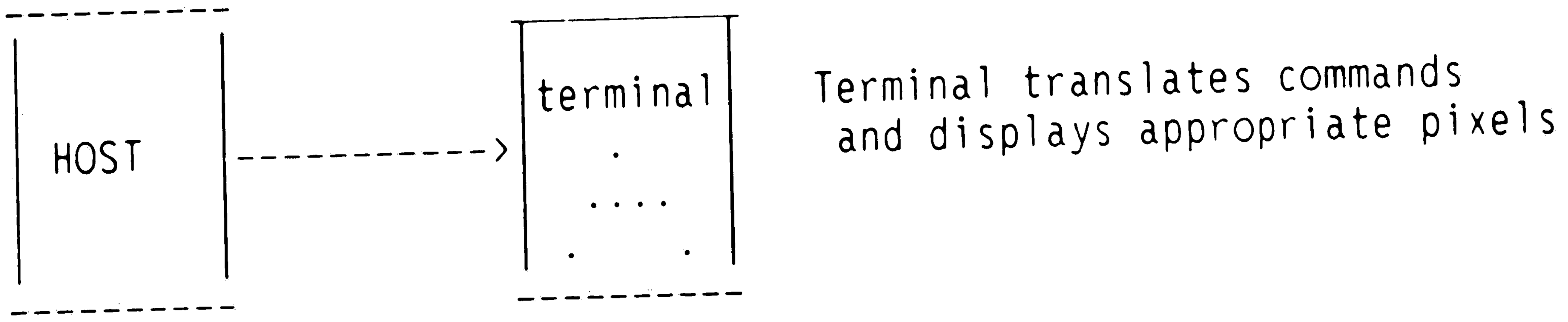


Figure 4.2
Intelligent Terminals

4.1.3 Vector Graphics

For analysis, presentation and modeling, the use of vector or straight line graphics is superior. Vector graphics use commands to draw basic or primitive graphics. Examples are polygons, panels, vectors, graphtext, markers and symbols. Refer to Figure 4.3.

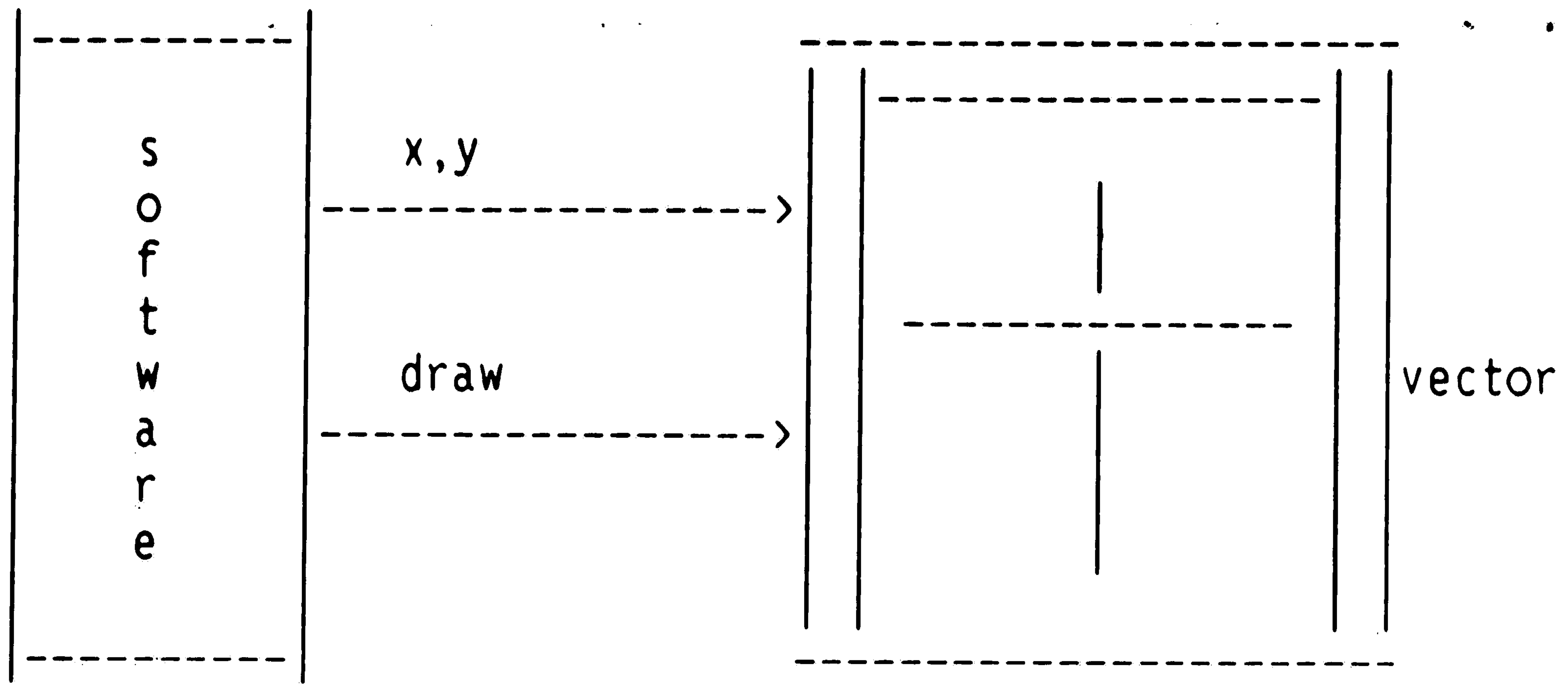


Figure 4.3
Vector Device

4.2 Stroke Technology

In stroke storage technology, a beam of light draws a curve at defined (x,y) coordinates. Storage tubes typically have a higher resolution image. A major problem, however, is that to change or delete a portion of the image, the screen must be erased and the image rewritten. Considering this, storage tubes are best suited for high resolution, somewhat stable applications where the image does not change frequently. For data analysis, modeling, and design applications, the best approach is the use of vector graphics. The advantage of pixel graphics is achieved without having to address individual pixels. A vector graphics command includes commands to draw graphics primitives such as vectors, graphtext and symbols. Generally, raster technology is best for traditional business applications.

CHAPTER 5

USING EXPERT SYSTEMS

5.1 Why is an Expert System Useful for Choosing Graphics Hardware?

When should expert systems be the approach used in solving a problem? First, the solution of the problem must have a significant payoff in order to warrant the development of the system. Perhaps other techniques have been tried but have proven impractical. The problem is to be solved by using an expert's knowledge and not a simple algorithm. The expert system can be developed and continually updated. Still, it cannot evaluate whether a correct result is reached. How can this problem domain be characterized as a reasonable example of when an expert system should be attempted?

Characteristics of a suitable problem domain are (12):

- o experts exist
- o experts are provably better than the amateurs
- o expert takes up to a few hours to solve the problem
- o the problem is a cognitive task
- o a novice needs to learn about the problem
- o high payoff possible
- o the problem requires no common sense.

5.2 Using Natural Language Interfaces

In most instances it has been necessary to communicate with a computer by learning a programming language such as FORTRAN or COBOL.

This is somewhat cumbersome to the average computer user, hence other languages and aids have been developed. Fourth generation languages (such as FOCUS and ADSO) have allowed a nonprofessional programmer to develop applications. The use of menus, graphic icons and online help facilities have also aided in moving away from syntactically unforgiving third generation languages. Successes with Fourth Generation languages have provided impetus for research in natural language communication. How can input and output occu and how can the computer learn to manipulate this language whether it be French, English or any other natural language?

Natural language communication can take two forms: text and sound. Only text will be considered here since well developed tools such as keyboards and printers are familiar mechanisms.

What does it mean for a computer to understand language? It means that the computer is capable of processing the users' language, executing the request and providing output. Natural language examples include interfaces to databases, operating systems and robots. There are two types of analysis that need to be performed by a natural language system. One concentrates on the syntax or how the phrase is structured while the other concentrates on the semantics or what the words mean within the context of the expression.

Syntax analysis occurs through parsing which identifies the function of each word. Semantic analysis associates words and the role the word serves with information stored in the data or knowledge base. The knowledge base contains expectations against which to interpret the input. Descriptions of objects and domain relationships can be stored in the knowledge base. The system can store the new input increasing the amount of knowledge the user has to access. Information from semantic and syntactic analysis are combined to produce representations of input to answer a database query.

Natural languages have been recognized as successful as advisory interfaces. Unfortunately, it is difficult to get beyond simple argument templates which tend to be very domain specific. This is particularly true of advisors written in third generation languages and in the many of the inexpensive expert shells in today's market. Chin(23) studied a simulation of the Unix Consultant where he found that 25 percent of the queries submitted used contextual syntactic constructs, among them ellipsis, anaphora, indirect speech acts and grammatically incomplete clauses. Interestingly enough, he also found that if the query was for a human consultant, the incidence of such contextual syntactic constructs increased two fold. Apparently, the people using a computerized consultant knowingly restricted the contextual contructs, whereas for humans, the human was expected to understand the query.

Interfaces can vary in how much users must keep to the structure of the system. Some interfaces are more natural than others. Some systems, for example, tell the user its interpretation of the query with the option to continue or perhaps to change an entry or spelling after which the query is reinterpreted.

Another issue is that a natural language system assumes that its knowledge is complete. An answer of "no" to a query may really mean that the system does not know what the answer is, that is, the knowledge base is not adequate to answer the query.

Machine translation of natural language reveals that human language cognition is a complex ability where meaning of words, pattern recognition, sentence structure and context play important roles.

5.3 Knowledge Bounds

How much knowledge needs to be in this knowledge base to make it useful? Although it is recognized that more knowledge about a particular domain, as well as more natural language capability would make an advisor more effective, this adds considerably to the cost of development. It is necessary to constrain the knowledge representations to provide a system that the average user can interact with and perhaps modify. If a potential user is expected to enhance the knowledge base and the words that are understood, the system is different from a sophisticated one built on an artificial intelligence machine where typically only a knowledge engineer has the required skills to add to knowledge and vocabulary.

To answer a query, a natural language system must:

- o identify what field of the data base is required
- o identify records in the data base that match criteria
- o determine retrieval and display process that is required to answer the request
- o invoke processes in a proper order

The natural language system must contain a lexicon, or list of words, that the program understands. Rules tranform user input into system output. Clarification dialogs are provided to get around ambiguous requests. Rules parse or pick apart an entry to determine its meaning.

5.4 Beginning the Graphics Assistant

One of the difficulties in beginning the design of this particular expert system was getting started. The old adage of keeping the project initially small and well contained applied to this task. The first concern was to define the problem domain. Initially the problem needed to be assessed for its applicability to an expert system. More so, did a real requirement exist for even looking at this particular problem? Was the problem a good candidate for a knowledge based system? A feasibility study should be conducted to assess the problem domain potential. A very small and restricted prototype would aide in deciding the feasibility. During the course of the development, it was accepted that attempted protyping might show that the project could not continue. As a second phase, an evolutionary path would be followed if the prototyping proves to be positive.

Beginning with an assistant level system, continuing with a colleague equivalent one and finishing with an expert system was determined to be the least risk project plan. At any time after the initial prototyping, the project could be stopped while still retaining some payback for developing the system.

Several areas of concern related to what need to be accomplished or understood at the beginning. The purpose of the system is to:

- create a natural language interface
- convert a typical user's way of thinking about the problem to the way the data is stored
- be able to answer the user's level of question
- understand the navigational problem because large systems tend to have knowledge in many files.

CHAPTER 6

CHOOSING GRAPHICS SOFTWARE AND HARDWARE

6.1 Steps in Choosing Software and Hardware

For the uninitiated, choosing a new graphics software package, softcopy output or hardcopy output device can be an overwhelming task. It is very easy to become caught up in the market hype and overestimate the required characteristics of a graphics device or software package. How does a user describe what he really needs? Decisions have to be based on many concepts.

The flowchart in Figure 6.1 and following discussions pertain to steps that one needs to review when choosing a system or device.

6.2 Cost

Devices referred to as workstations typically have a higher initial cost than most terminals. An area to consider is whether a multi-user capability is required. Will more than one user be running the same application? The important issue when comparing costs is the real cost per user. If an adequate host computer exists, the addition of lower cost terminals may be more appropriate. While a host version of the software costs more, serving multiple users may make it more cost effective than single copy, single user software. Many times softcopy or hardcopy devices are not chosen to satisfy the application for which the device is required. More expensive software is chosen, but low cost devices that cannot take advantage of the output the software is capable of are purchased. The naive user of the

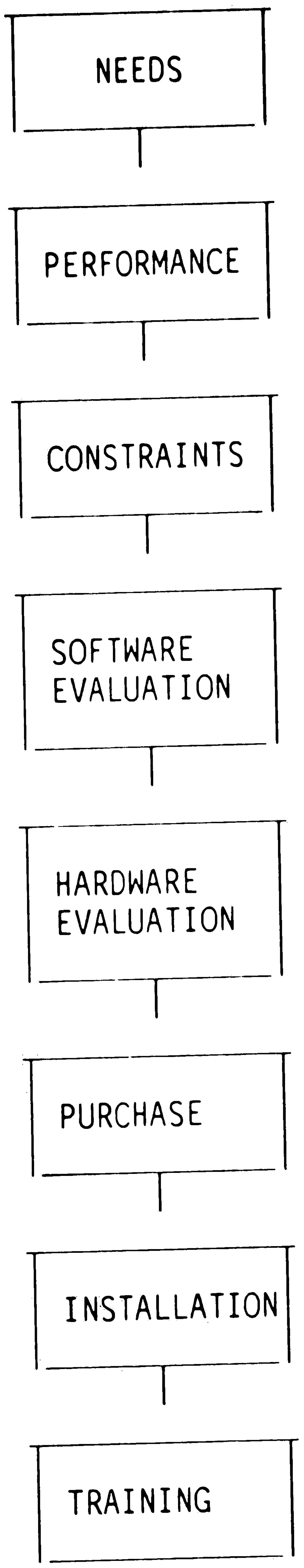


Figure 6.1
Steps in Choosing a System

software becomes frustrated, and generally does not differentiate between the capabilities of the software and hardware.

Higher resolution, pan and zoom, graphics input usually increase cost per device. The amount of memory a device contains also is reflected in the cost. Memory is affected, as an example, by the number of colors a device can display at one time. Compare a 640 X 480 device with a 1280 X 1024 device. The second device requires four times as much memory as the first. If the first has four colors at one time (2**2) and the second 256 (2**8) then the second requires four times as much memory. Along with the resolution difference requiring additional memory, the total requirement is sixteen times the memory for the second device. In addition to the memory chips, a higher rated power supply is also needed.

CAD terminals are high-end devices that have at least 1024 X 780 resolution, color and monochrome, local graphic support and local primitive support. Most such devices are priced well over $5000.

6.3 Features

What features should be available in a graphics device? The device's screen should be capable of a bright, stable output. A flicker free display is important especially if the device is well used. Such devices should have a minimum of a 60 Hertz refresh rate. A 60 Hertz refresh rate indicates that the display is reenergized 60 times per second. Lower Hertz rates will cause the screen to flicker. Pixel resolution or resolution refers to the number of pixels in both x and y directions. As an example, a device with 640 X 480 pixel resolution has 640 pixels in the x direction and 480 pixels in the y direction. A 640 X 480 resolution would be considered medium resolution. Low resolution would be approximately 200 pixels on any one axes while high resolution is about 1000 pixels in any one direction. A trade off to higher resolution devices is local pan and zoom which allows a picture to be enlarged to see more detail. This allows large drawings to be reviewed a small area at a time. Dialog areas are useful in multipurpose devices so that a mix of graphics area application and text processing can be achieved. Multi-colored dialog areas can enhance user interaction.

6.4 Color Capability

Many times the color requirement of an application is overestimated or underestimated. Depending on the device, the number of colors can range from four to eight through several million. A more important concern is how many colors can be displayed at any one time? How many colors can be displayed at one time in the dialog area? Is the choice of colors for the graphics area independent of dialog area?

6.5 Screen Size

Screen size varies between 13" and 25". Generally most users are comfortable viewing a screen of this size. Under 13" is too small and over 25" is too large to sit very close to. Higher resolution usually requires having a larger screen.

6.6 Choosing a Graphics System

How can a graphics system be choosen? How can it fit the needs of an application? First, the analyst must develop the application and system requirements together. Then, he must determine what will be accomplished and how it will be accomplished. After that, he develops hardware and software requirements.

How is a terminal or workstation choosen? Several of the qualities that should be reviewed follow. Whatever can be configured to lower the cost per user is a preferable goal. A workstation environment with multi users may accomplish this, or it may be achieved by adding software and hardware to a host environment, especially if the host environment already exists. Personal computers and workstations are well suited to small to medium sized applications. Additionally, these environments also can be special purpose, that is, configured to meet the needs of a specific application. Other issues in choosing an environment that need consideration include database access, the need for extensive numerical computations, and intensive input/output applications. High volume input and output can be cumbersome when connected in a slow speed communications environment, such as 9600 baud.

6.7 Characteristics of Quality Color Graphics Terminals

Because of the wide selection of terminals available on the market today, the acquisition of a device can be confusing. What are the most important characteristics of a device that need consideration? A bright and stable display that is flicker free is important.

Color convergence is needed at all screen areas. Electron beam guns must converge or come together to create crisp lines. Red, green and blue guns must converge in the outer edges of the screen as well as in the middle. Being able to control intensities of the colors is also important since devices are not always able to be placed in the best environments for their use.

The resolution must match the requirements of the application. For example, if local segmentation as well as pan and zoom are available with which to enlarge a drawing, perhaps high resolution is not a requirement. Business applications generally do not require high resolution, whereas, engineering applications such as stress analysis do.

The following provide guidelines to differentiate pixel resolution:

o low resolution	200 X 200
o medium resolution	640 X 480
o high resolution	1000 X 1000

6.8 Pixel Resolution

A pixel resolution of 4(X-axis) :3(Y-axis) will uniformly distribute the pixels for displaying both text and graphics. Fonts capable of displaying both upper and lower case such that the characters are readable are a must. Some devices do well at graphical display or text display but not at both. The user has to be aware of this potential if the applications displayed on the device are more oriented either toward text or graphics. A multipurpose terminal used for both text and graphics should have separate graphics and text areas with a dialog area so that the interaction with the computer does not damage the graphics. This allows menus and prompts to be displayed independently of the graphics. Hardware vendors provide emulation support of a device to provide competitive and widespread support of applications. Emulators (refer to Glossary for definition of emulators) preferred and generally accepted as standard are:

- Tektronix emulation - 41xx series
- DEC VT series
- DEC Remote Graphics Instruction Set (ReGIS)

Tektronix 41xx series emulation is supported by a minimum of thirteen terminal manufacturers. The TEK41xx series generally communicates via RS-232C command stream, although coaxially connected 41xxCX series is available for IBM environments. Vector graphics primitives should be available. Primitives are line segments, arcs, circles, line color, dashed and dotted lines, panels, panel fill, markers and graphtext.

Tools for graphics input should be available for creating and editing graphics. Pick, locate and stroke functions for selecting menu items are important features.

Local segmentation reduces CPU utilization by offloading the host. International keyboard support and Pi and Special character sets are particularly important if working within a scientific or engineering environment. High quality imaging such as in medical applications requires control of each pixel to provide realistic graphics. Color may not be important. If it is, how many colors need to be displayed in one application? Some devices with four colors are sufficient for small scale applications.

Some devices have the capability to have the color changed from within the device. Being able to split the screen into graphics and dialog areas, support for ANSI X3.64 text editing and IBM 3279 alphanumeric support ar important alphanumeric considerations.

6.9 Communications Support

The most common types of communications support are RS-232 (asychronous or ASCII) and COAX (synchronous or EBCDIC). Most devices are capable of one or the other communication method although some manufacturers such as Tektronix offer terminals that are capable of switching between the two environments.

6.10 Hardcopy Devices

Areas of concern in looking at hardcopy units include what devices are potential selections that are of good quality and speed. How does what you see on the screen get to the hardcopy unit?

6.11 Software support issues include:

- software compatibility with device
- available device drivers or software that translates software to a form that the terminal can respond to
- does software have growth potential?

6.12 What else needs to be decided when looking for a graphics device?

- Does the device offer local intelligence?
  Fewer CPU cycles are required.
- Human ergonomics design; brightness, contrast, size, portability, screen tilt, glare
- Who supplies the software for the device?
- Resolution
- Number of colors
- Video signal: RGB, composite
- Hardcopy availability
- Maintenance and support
- Cost

CHAPTER 7

COMMUNICATIONS

7.1 Communication Methods

Any device must be capable of conforming to the host environment. Examples are handshaking protocols and data transfer rates. Standards such as RS-232 are implemented differently by different vendors. Detailed information such as pin diagrams should be reviewed. What CPU resources are limited? What options are available to optimize these resources? Can device buffers be increased in size? Should a vector to raster converter be considered? Is the device being used an ASCII device in an EBCDIC world? What other communications hardware does the hardware device or device driver need to communicate (ex. IBM 3708, 3274)?

RS-232 is an asynchronous or ASCII communication method. In simple terms, the difference is in how the transmission is timed. Asynchronous communication sends the signals at any time while in synchronous communications, signals are sent in a well defined timing scheme.

7.2 Responsiveness

Responsiveness is a critical consideration in choosing a communications method. Communication speed is dependent on how quickly data are generated from a host source, converted to a modem and transferred through a channel. This is stated in baud which equals one bit of data per second. Coax or direct cable is the most efficient method of communicating between host and terminal.

Related to hardcopy and softcopy are other important factors. The volume of the output generated by an application is key. System throughput which includes host response time, I/O rate, graphics protocols, communications modes, device synchronization modes, graphics device throughput rates are variables that affect the total elapse time between entering requests and obtaining the results. Hardcopy turnaround is determined by the devices plotting speed, I/O rate and buffer size and also the devices ability to stack output jobs.

The device's I/O speed defines baud rate. Devices daisy chained (See Figure 7.2) on an RS-232 serial line can only work at the maximum speed of the slowest devices.

The volume of data required to create a graphic vary dependent on protocols. An example is the difference between ASCII and Binary protocols. In ASCII an additional twelve bytes could be required to determine a single point. A compressed binary that has been optimized might take only two or three bytes.

7.3 Protocols

Handshaking methods affect I/O rates. FORTRAN I/O handshaking allows a task to be swapped out while waiting causing a delay. Other handshaking methods are XON/XOFF and Data Terminal Ready (DTR). IBM offers two protocols as standards:

3270 (Bisynchronous)

SDLC (Synchronous Data Link Control)

In the 3270/Bisychronous method, a central host communicates to devices such as IBM 3179G through a cluster controller such as the IBM 3274. SDLC, a hardware communications protocol, is part of SNA (Systems Network Architecture). Interfacing ASCII RS-232 devices in a IBM environment is sometimes difficult. Generally devices RS-232 connectable tend to lower costs and have more functionality. Also cables can be effectively run longer distances. To interface an ASCII device with an IBM would require protocol conversion (Figure 7.1).

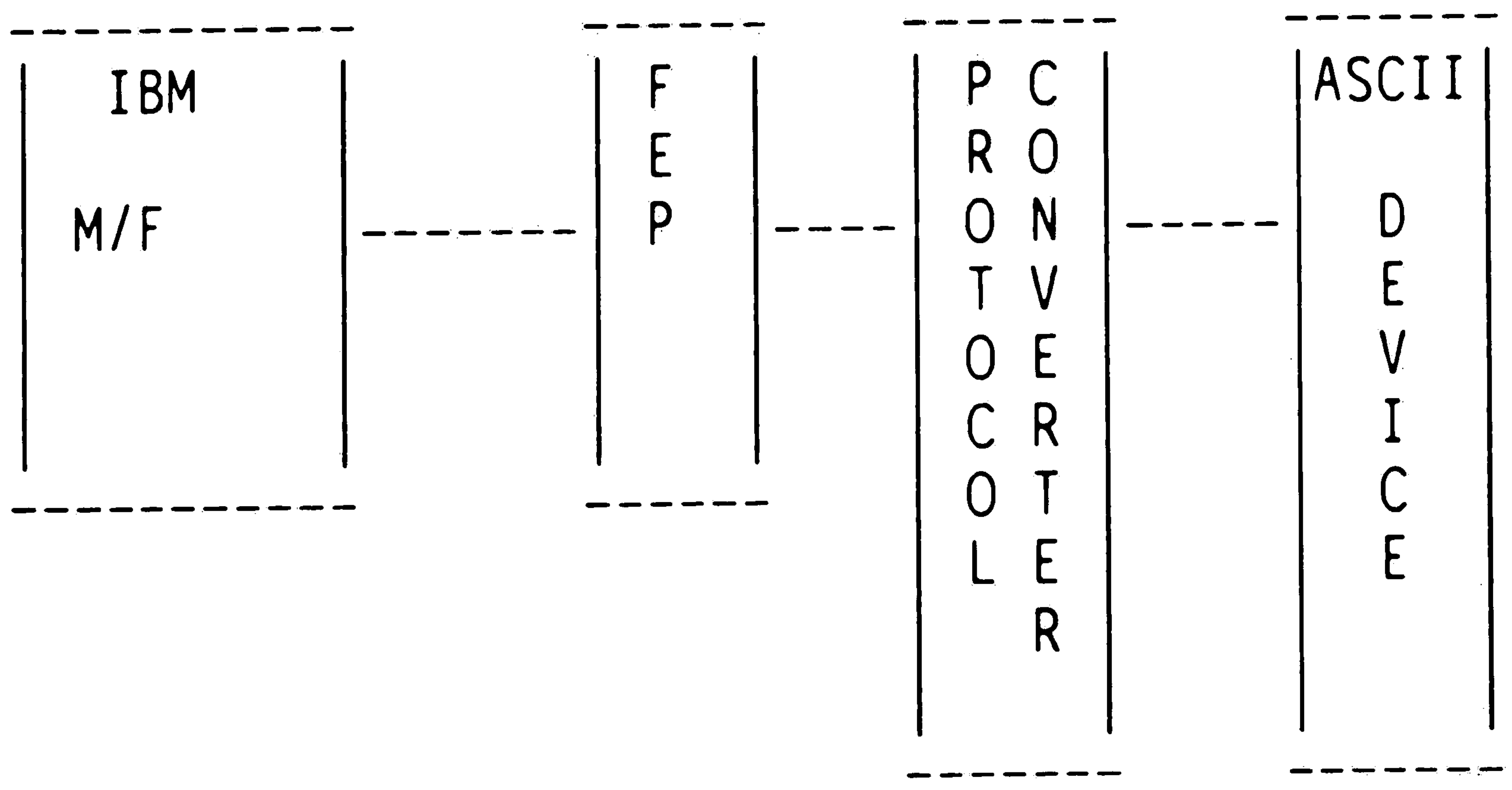


Figure 7.1
Protocol Conversion

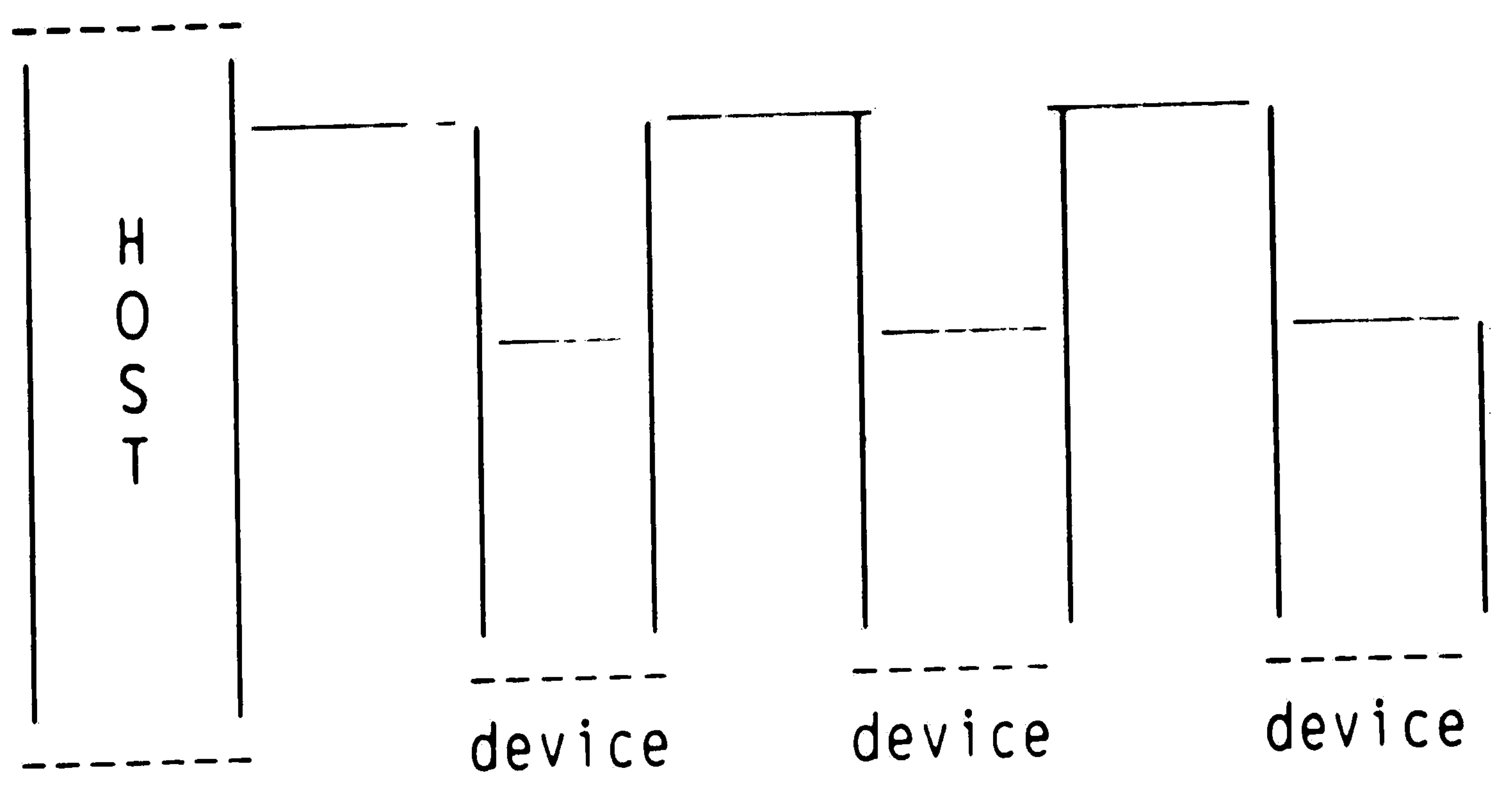


Figure 7.2
Example of Daisy Chain

This common arrangement provides flexibility for devices that allow passthrough of data from the host to devices. Communications are typically RS-232.

Communication between host and device include many aspects. Transparent I/O allows the host to transmit buffers of data intact. Typically, however, buffering characters are added (DC1, DC3, CR, DEL, NUL are a few). Some devices will ignore this additions while others will terminate functioning. Spooling is a technique where files are stacked or queued waiting for a multi-user device. An example is a coaxially connected Zeta 887 pen plotter which can be controlled by IBM Job Entry System (JES) and VPS.

Data Communications can be confusing and compatibility is a necessity. Hardware interfaces of types RS-232, SNA, Sychronous, Bisychronous must be compatible with the access method supported by the host environment.

Every hardcopy device must be compatible with the controlling terminal. When using RS-232 this means such concepts as baud rates of the devices being identical.

CHAPTER 8

THE GRAPHICS ASSISTANT

8.1 Purpose

The purpose of the prototype graphics device selection expert assistant is to provide a tool to persons interested in acquiring a graphics device. The user does not need to be knowledgeable about all the requirements needing definition before choosing a device.

8.2 Choosing a Tool for the Assistant

Several tools were explored for their potential use in the Graphics Assistant expert system. Among the more difficult transitions to make in using an expert system shell, particularly those available for use on a personal computer, is forgetting any experience with third generation languages where every step must be programmed. Also, although numerous shells were examined, few seemed appropriate for use with this assistant. Several wanted to make use of a probabalistic methodology in assigning a probability to each possible response. Although workable in many shells, this method made it difficult to answer that the answer is unknown. Since the purpose in using such a system is to learn a topic, it would be difficult not to respond that the solution is not known.

After several false starts, Prolog, created in 1972 at the University of Marseilles by Alain Colmeraurer, became a clearer choice. Although Prolog, the choice of Japan's Fifth Generation Project, tends to be memory hungry while searching to match string against string, it is quite flexible in representing knowledge. Another reason for its extensive memory usage is its technique of backward chaining in moving from top down and from left to right. Turbo Prolog Version 1.1 is an affordable, fast, easy to use, and well documented implementation of Prolog. Although not a Prolog that adheres to the Clocksin and Mellish de facto standard, the new dialect was designed to appeal to Pascal programmers. It contains a comprehensive development environment which includes full screen editing, interactive source tracing, run-time color graphics support and windowing. An extensive library of built-in predicates is provided.

8.3 Database Queries

The graphics assistant demonstrates a natural language interface to a database on graphics hardware. A user can query the system using a limited English. Once a level of understanding of how Prolog operates is achieved, other English strings can be added as well as additional common words to ignore or words that are synonyms to already understood words.

8.4 The Graphics Data Base Scope

There are many areas that can be addressed when choosing a graphics system. For the purposes of this example, the potential field was limited for the prototype. At some later time, it would be possible to add new features deemed important as well as new devices made available in a rapidly changing environment. A commentary, however, on the acquisition of new devices made available by new vendors is is order. Most large organizations try to limit the number of vendors from which devices are purchased for use on a network. Conservativism is more the norm while allowing many newer, less expensive and, perhaps, more functional devices to be passed over. A user of this system would be wise to understand his organizational environment as well as technical one prior to becoming too enthusuastic about a new device.

8.5 Graphics Assistant Database

8.5.1 Records

The assistant currently has a simplified set of records. This was done to allow for an end-user to add new records not only for the existing devices but also to be able to add new database predicates without significant effort. The structure of the system is designed to allow people that use the system to modify the data base when new devices become available, as well as when they become obsolete.

8.5.2 Basic Device Information

An initial record containing basic information about a device provides the name, the device type, host communications, the availability of a rasterizer and the list price of the device. Volume discounts are relevant to vendor and customer and are not discussed here.

```
device(NAME,TYPE_DEVICE,HOST_COMMUNICATIONS,RASTERIZER,COST).
```

name = the name of the device(ex. tek4105).

type = type of device.(ex. CRT, laser printer, ink-jet plotter, 35mm camera)

communication method = type of communication standard (ex. coax or direct connect, RS-232).

rasterizer = yes/no

cost = price of the device in U.S. dollars

8.5.3 Resolution

The terminal display or pixel resolution is determined by the number of pixels that a device can display along each axis. As an example, a device with a 640 X 480 pixel resolution which is a medium resolution device can display 640 pixels along the horizontal axis and 480 pixels along the vertical axis.

```
resolution(NAME,HORIZONTAL_RESOLUTION,VERTICAL_RESOLUTION).
```

8.5.4 Graphic Primitives

Graphics information and text information are displayed independently on the device screen and are stored in different areas of terminal memory. Pictures and other graphics information are displayed in the graphic area. The dialog area displays such information as the user's interaction with the host computer, any commands to the device, messages that are informative or errors as well as and text that is being edited. Typically, the dialog area is displayed in front of the graphics area. In any case, the graphics and dialog areas are treated as separate surfaces.

Many times in choosing a device best suited to a specific applications, the fundamental graphics display units must be examined. These basic building blocks indicate how the graphics display is contructed.

A database record describes the graphics primitives that are available in each device.

```
graphic(NAME,VECTORS,ARCS,PANELS,MARKERS,MATH_CHARS,PIXEL_OP,SEGMENT,
        PAN_ZOOM,SURFACES,VIEWS,VIEWPORTS).
```

The first entry after the device name is the availability of vectors. A vector is the most predominantly used graphics feature.

The quality or smoothness of the curve or arc is determined by the pixel resolution. A curve is typically simulated by connecting a series of tiny vectors.

A panel is a closed region bounded by vectors or arcs. The arcs are outlined or filled with colors or patterns as defined by the terminal. Markers are used to identify or enlarge a point in a picture. An example of their use is the highlighting of data points. A segment which is stored in the terminal's memory, is a group of graphics objects. This reduces data editing, repetitive graphics and examples where the graphic is moved around the screen.

Pan and zoom play important roles in applications where a graphic might be redrawn at a different scale.

Surfaces require that graphics memory be separated into several parts. A simple example of this is a transparency overlay in which some information is on two transparencies and the whole picture is perceived when the transparencies are overlayed.

Views and viewports are collections of display information. Examples include the ability to view different graphs simultaneouly, show different rotations of the same object or showing a process as it progresses.

8.5.5 Copiers

Hardcopy units can provide support for a number of different softcopy units.

```
   colorcopy(COPIER,DEVICE_LIST).
```

8.5.6 Emulators

```
   emulator(EMULATOR, DEVICE_LIST).
```

To emulate another device means to assume its identity. Drawing commands intended for another manufacturer's device are accepted and interpreted. A few devices are accepted as standard so that most other devices will emulate one of the standards. The record provides a list of devices that emulate standard devices.

8.5.7 Screen Characteristics

```
screen(NAME,SCREEN_SIZE,TYPE,WINDOWING,MAX_COLOR_PALLETTE,
       DIALOG_AREA_COLORS,FILL_PATTERNS, RGB_VIDEO, GRAPHICS_COLORS).
```

Screen_size is given in inches along the diagonal. Max_color_pallette, dialog_area_colors, fill_patterns and graphics_colors are given as number possible. Windowing and rgb_video are yes or no responses.

8.5.8 Mouse

Interaction among mouse, menus and graphical icons is oriented toward novice or intermittent users who forget the syntax and semantics of a command driven environment. A mouse is connected to a computer by a serial port or adapter card. A trackball on the undersurface moves the cursor in the required direction as the user pushes the mouse in the appropriate direction. A button in pushed when the mouse reaches the particular location activating the selected function. Several different mouse designs exist in the current market.

8.5.9 Maintenance costs

```
maintenance(DEVICE,COST).
```

Maintenance cost are sometimes not considered initially. The relationship between reliability and cost is not always understood. Somes devices have high maintenance costs because they are sensitive devices that require continual fine tuning. Others have high cost but are very stable devices that almost never need servicing. Most organizations, both large and small, pay the costs rather than suffer the down time of the device.

8.6 Using the System

8.6.1 Starting the System

After Graphics is compiled and executed, an initial menu will appear as in Figure 8.1. Two selections are general purpose interfaces into an editor and the DOS environment where DOS commands can be entered. Figure 8.2 reflects the tutorial that is available as a selection from the main menu. The tutorial discusses what information is available in the database, examples of queries that might be entered as well as a review of language contained within the database.

Loading the database is necessary to provide information to the Assistant. Certain text editors place an EOF (end of file) at the end of the file causing an error to be reported when loading the database. Actually the database is loading correctly.

The error is being detected during the compilation of the database. Any concern about this can be alleviated by saving the database for review because Prolog will save only that which it thinks is correct.

The novice or occasional user of the system can familiarize himself quickly with the syntax and semantics by reviewing each of the sub topics presented in Figure 8.3 for "View the Language".

Figure 8.4, "Schema of Relations", indicates the actual database with the information associated with each database record. For example, the software record provides a relationship between a device and the software available to drive the device.

In Figure 8.5 associations between entities can be reviewed. This can assist the user of the system in formulating queries to the database.

All possible entities can be reviewed by accessing "Names of Entities". Figure 8.6 lists the names of entities.

Figure 8.7 shows the screen for the synonyms provided for with the assistant. It is possible to add to this list.

Associations are provided in Figure 8.8. These words provide Prolog word associations with which to parse the language.

Words that provide no added meaning that are used in conversation can be ignored by the system. The words which can be added to are listed in Figure 8.9.

Measurement units are associated with entities. An example is the cost of a device is measured in dollar units. Figure 8.10 shows the units associated with entities that relate to measure.

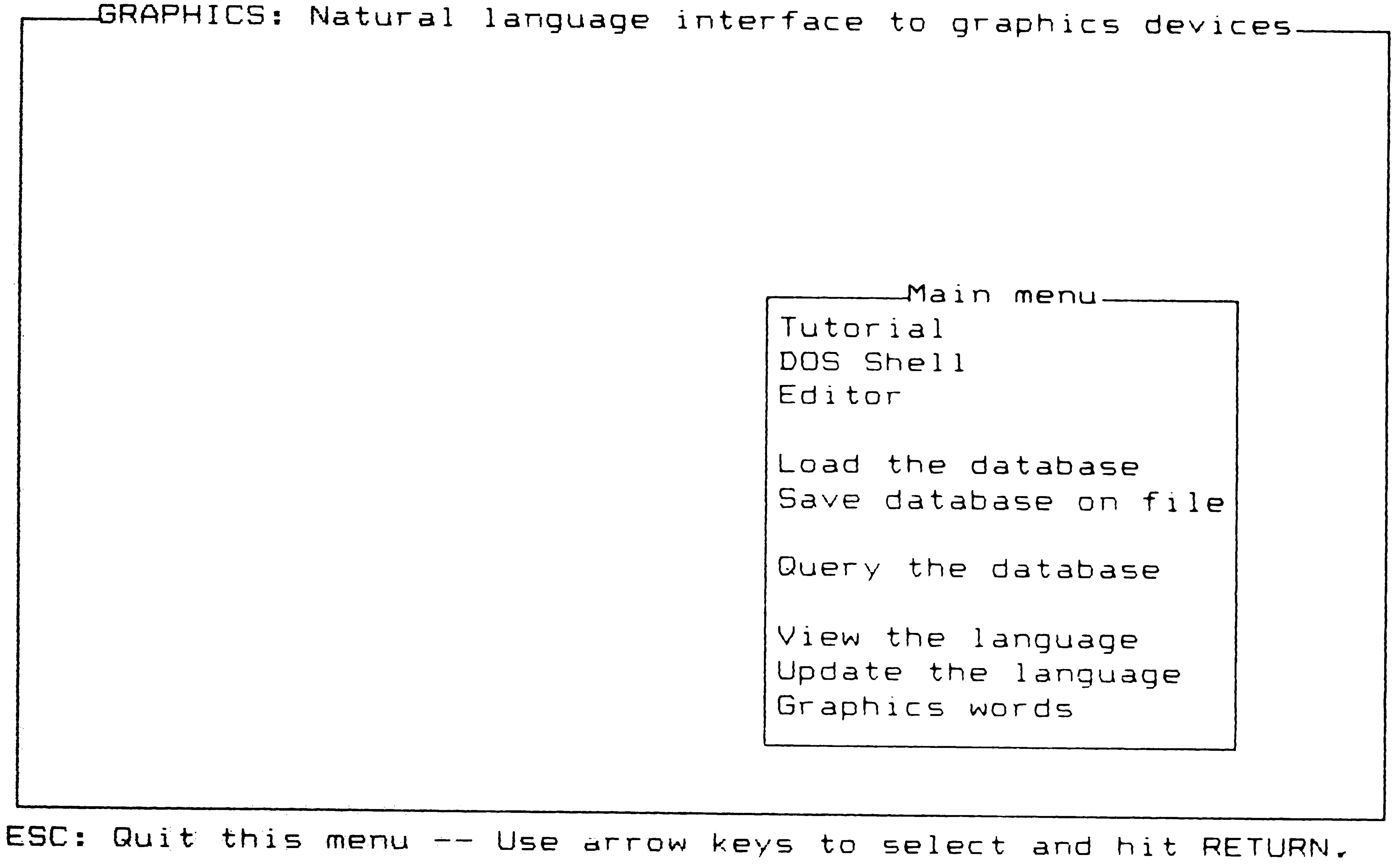
Figure 8.11 lists names of relational operators. Various words for the concept of greater than, for example, are given. Minimums and maximums are provided in Figure 8.12.

Definitions associated with graphics are provided from selecting "Graphics Words". A list of words is given in Figure 8.13.

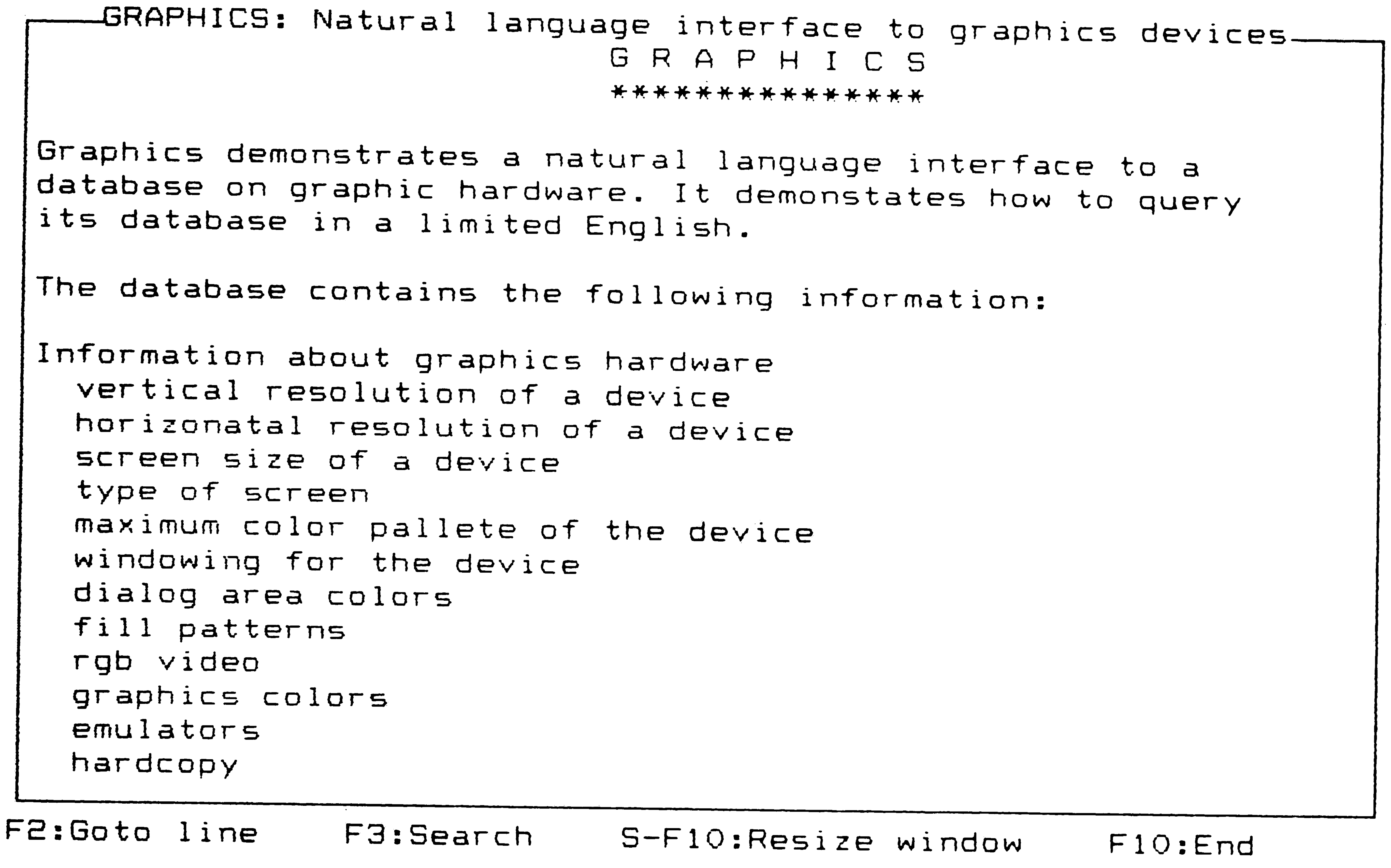
Figure 8.14 shows the screen provided for updating the language. New sysnonyms, words to ignore and alternative associations can be added.

A sample session can be reviewed in Figure 8.15.

Figure 8.1
Graphics Assistant Initial Menu

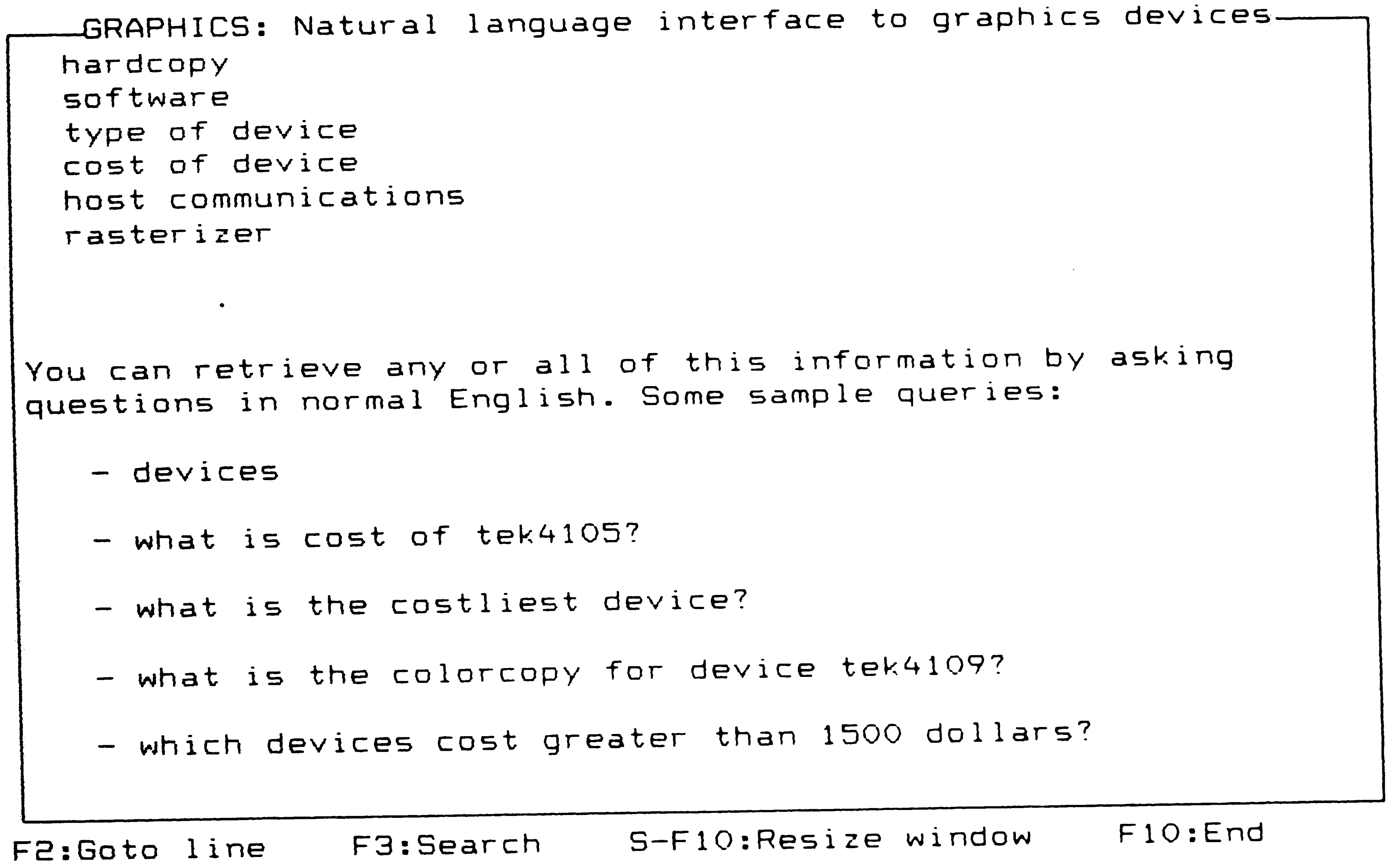


```
GRAPHICS: Natural language interface to graphics devices

                                     Main menu
                                   Tutorial
                                   DOS Shell
                                   Editor

                                   Load the database
                                   Save database on file

                                   Query the database

                                   View the language
                                   Update the language
                                   Graphics words

ESC: Quit this menu -- Use arrow keys to select and hit RETURN.
```



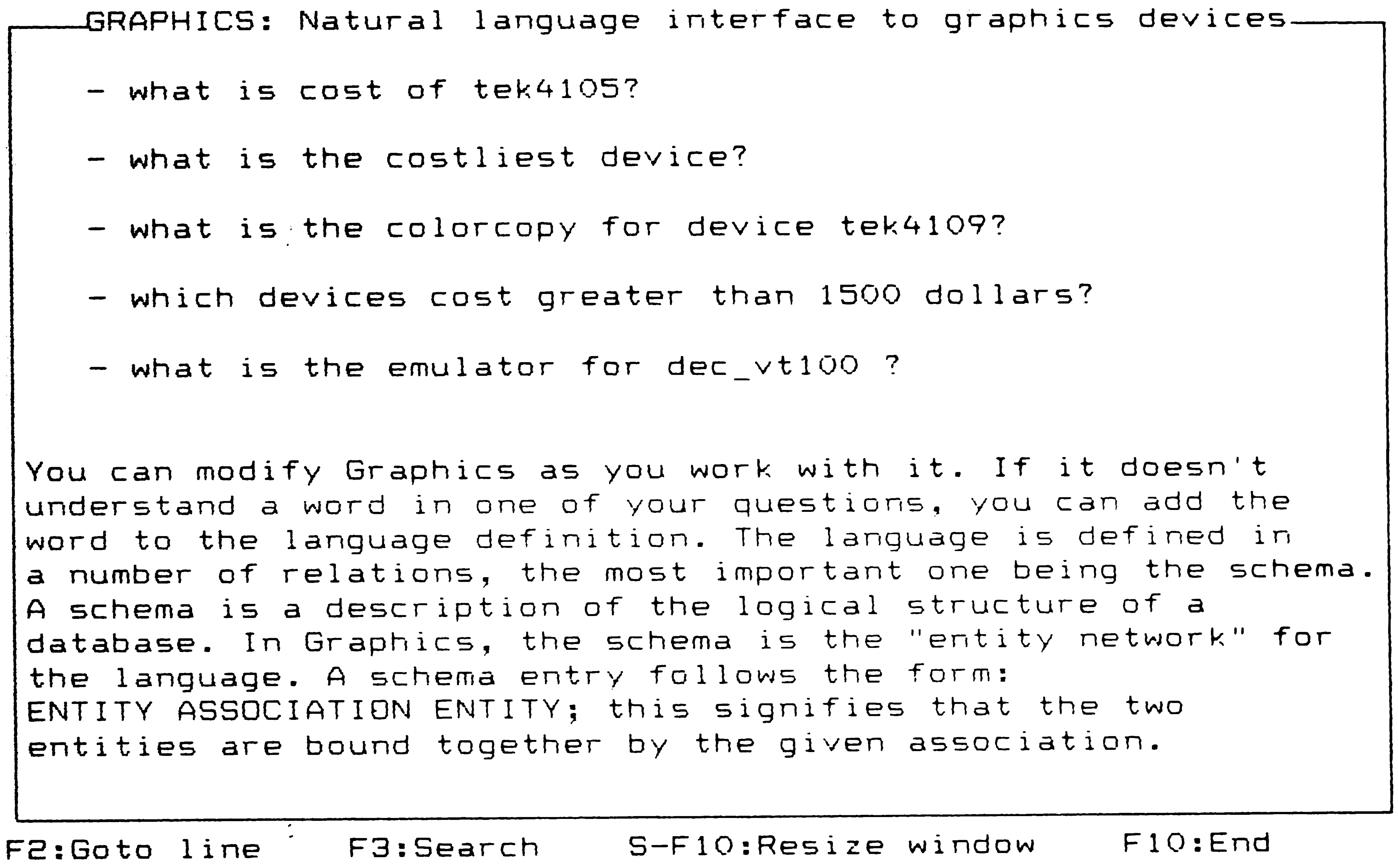
```
GRAPHICS: Natural language interface to graphics devices
                         GRAPHICS
                      ***************

Graphics demonstrates a natural language interface to a
database on graphic hardware. It demonstates how to query
its database in a limited English.

The database contains the following information:

Information about graphics hardware
  vertical resolution of a device
  horizonatal resolution of a device
  screen size of a device
  type of screen
  maximum color pallete of the device
  windowing for the device
  dialog area colors
  fill patterns
  rgb video
  graphics colors
  emulators
  hardcopy

F2:Goto line    F3:Search    S-F10:Resize window    F10:End
```

Figure 8.2
Tutorial



```
GRAPHICS: Natural language interface to graphics devices
  hardcopy
  software
  type of device
  cost of device
  host communications
  rasterizer

            .

You can retrieve any or all of this information by asking
questions in normal English. Some sample queries:

   - devices

   - what is cost of tek4105?

   - what is the costliest device?

   - what is the colorcopy for device tek4109?

   - which devices cost greater than 1500 dollars?

F2:Goto line     F3:Search     S-F10:Resize window     F10:End
```

Figure 8.2 cont.
Tutorial

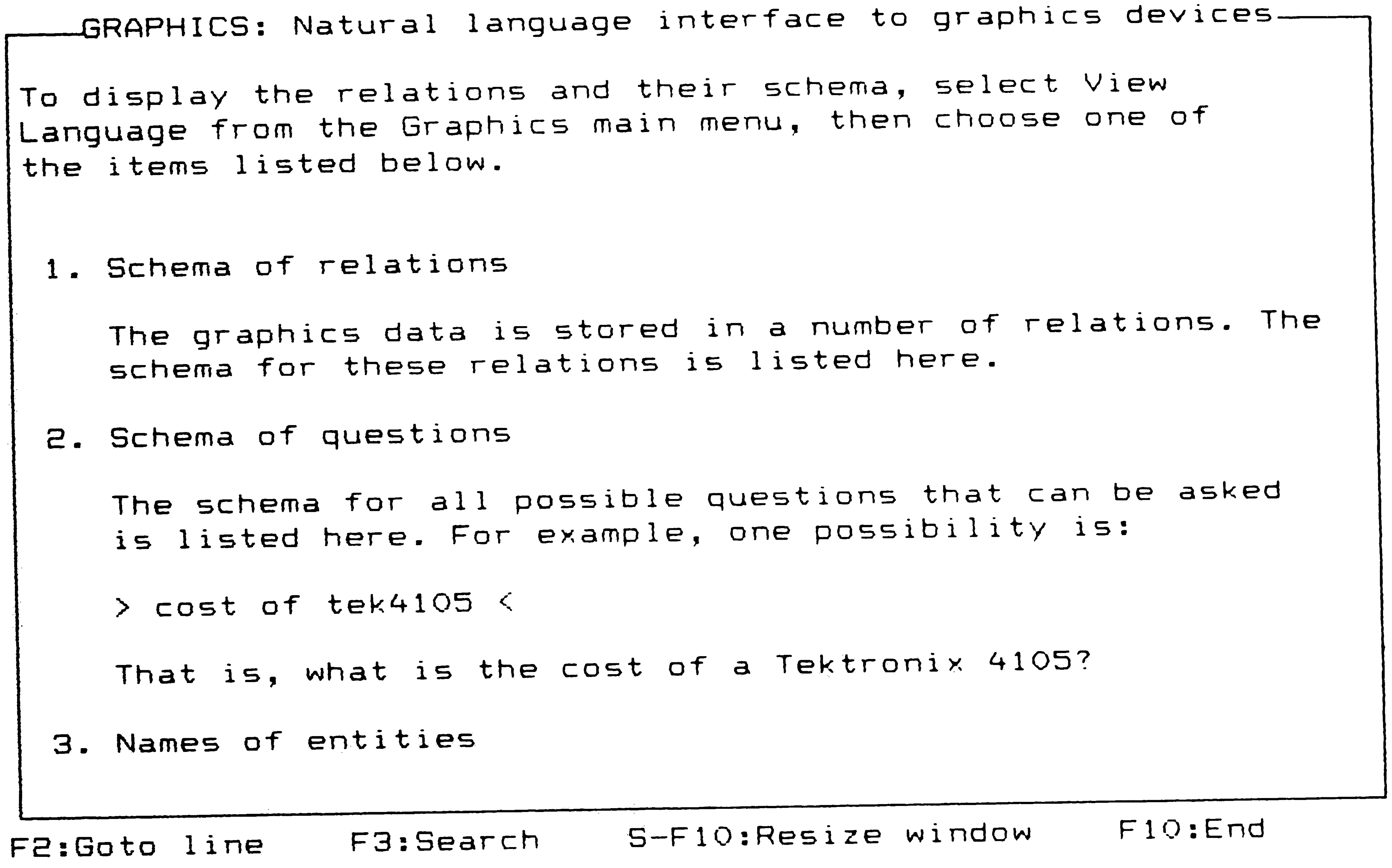


GRAPHICS: Natural language interface to graphics devices

- what is cost of tek4105?
- what is the costliest device?
- what is the colorcopy for device tek4109?
- which devices cost greater than 1500 dollars?
- what is the emulator for dec_vt100 ?

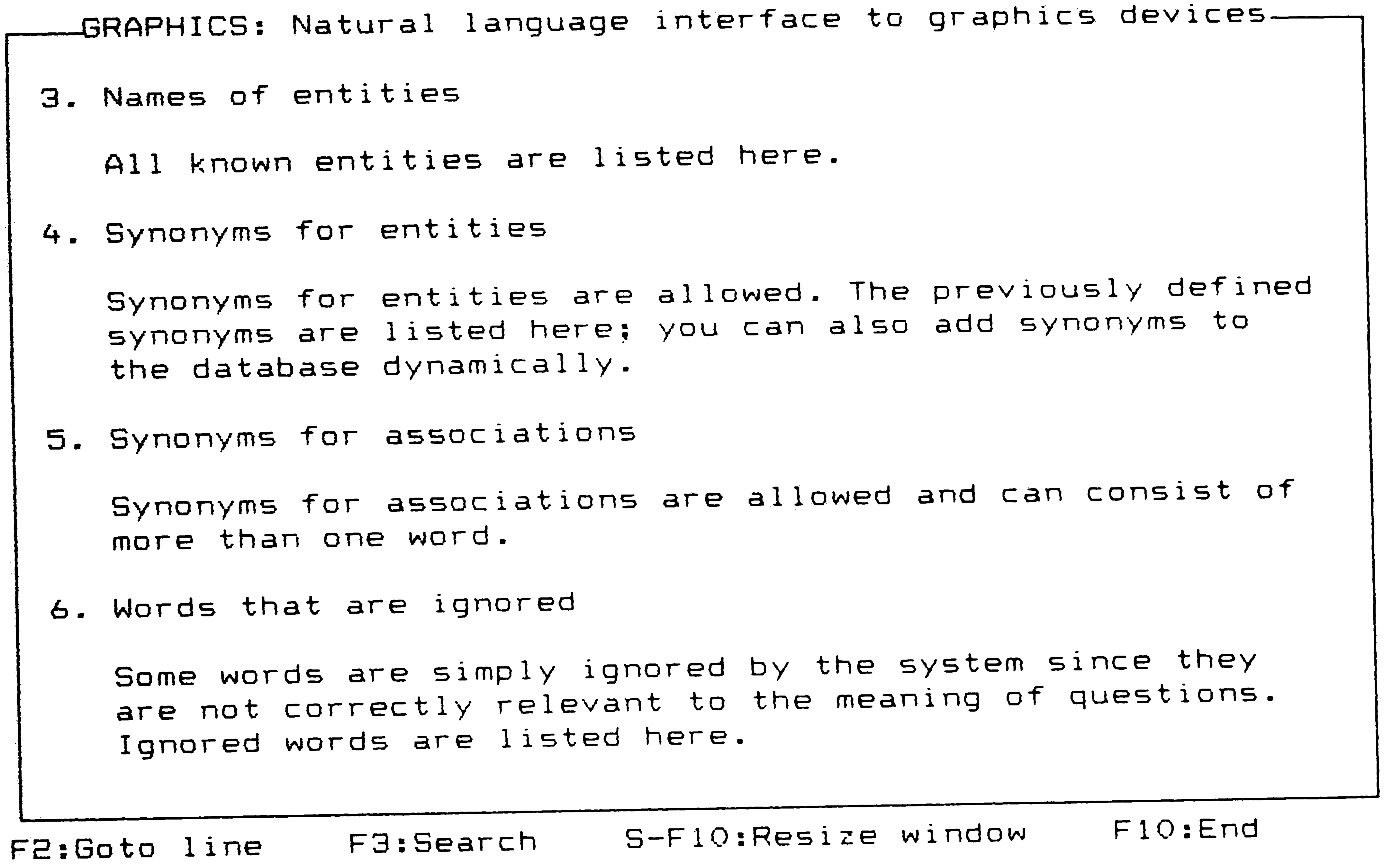
You can modify Graphics as you work with it. If it doesn't understand a word in one of your questions, you can add the word to the language definition. The language is defined in a number of relations, the most important one being the schema. A schema is a description of the logical structure of a database. In Graphics, the schema is the "entity network" for the language. A schema entry follows the form:
ENTITY ASSOCIATION ENTITY; this signifies that the two entities are bound together by the given association.

Figure 8.2 cont.
Tutorial



```
GRAPHICS: Natural language interface to graphics devices

To display the relations and their schema, select View
Language from the Graphics main menu, then choose one of
the items listed below.

 1. Schema of relations

    The graphics data is stored in a number of relations. The
    schema for these relations is listed here.

 2. Schema of questions

    The schema for all possible questions that can be asked
    is listed here. For example, one possibility is:

    > cost of tek4105 <

    That is, what is the cost of a Tektronix 4105?

 3. Names of entities

F2:Goto line    F3:Search    S-F10:Resize window    F10:End
```

Figure 8.2 cont.
Tutorial



GRAPHICS: Natural language interface to graphics devices

3. Names of entities

   All known entities are listed here.

4. Synonyms for entities

   Synonyms for entities are allowed. The previously defined synonyms are listed here; you can also add synonyms to the database dynamically.

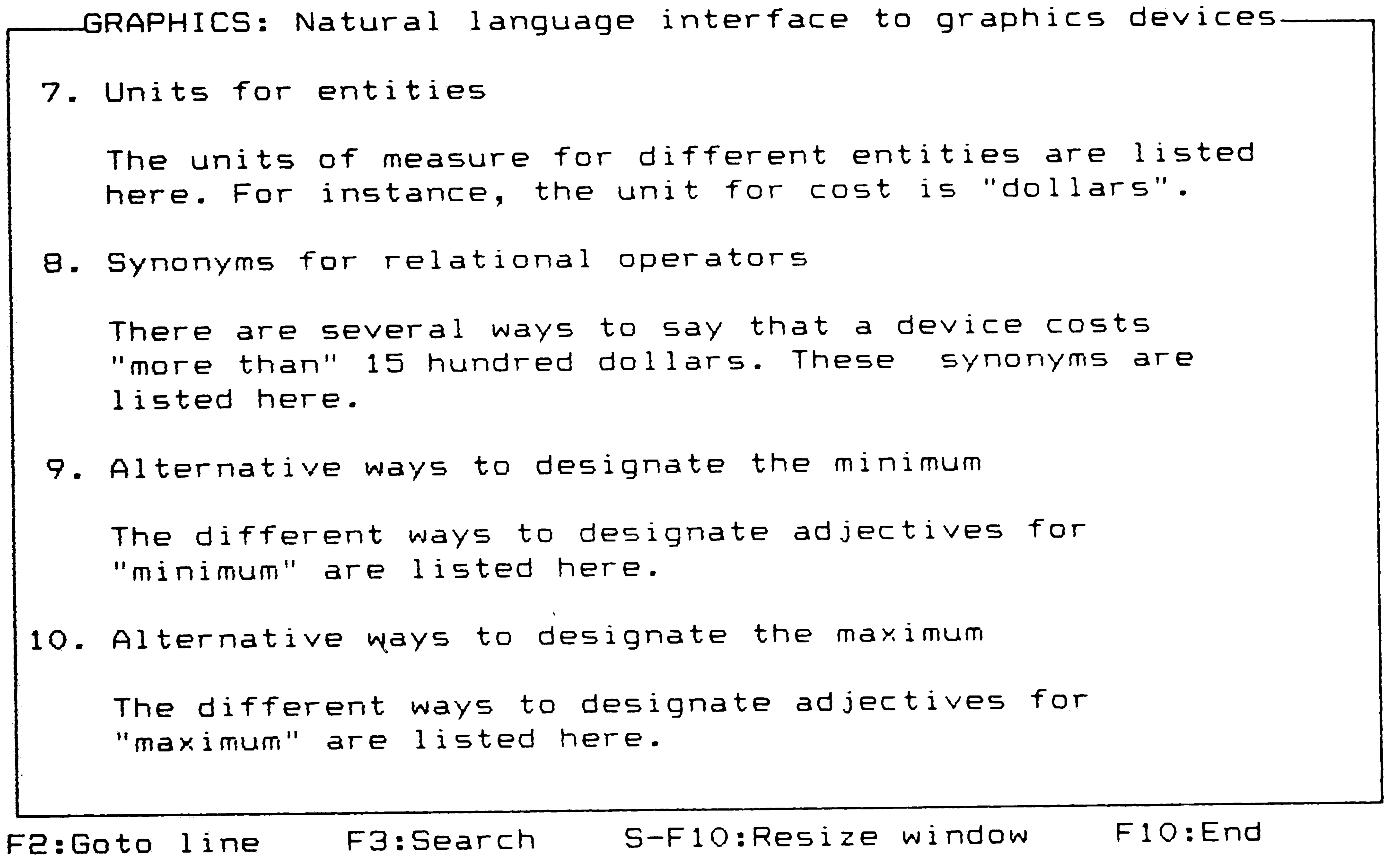
5. Synonyms for associations

   Synonyms for associations are allowed and can consist of more than one word.

6. Words that are ignored

   Some words are simply ignored by the system since they are not correctly relevant to the meaning of questions. Ignored words are listed here.

Figure 8.2 cont.
Tutorial



GRAPHICS: Natural language interface to graphics devices

7. Units for entities

The units of measure for different entities are listed here. For instance, the unit for cost is "dollars".

8. Synonyms for relational operators

There are several ways to say that a device costs "more than" 15 hundred dollars. These synonyms are listed here.

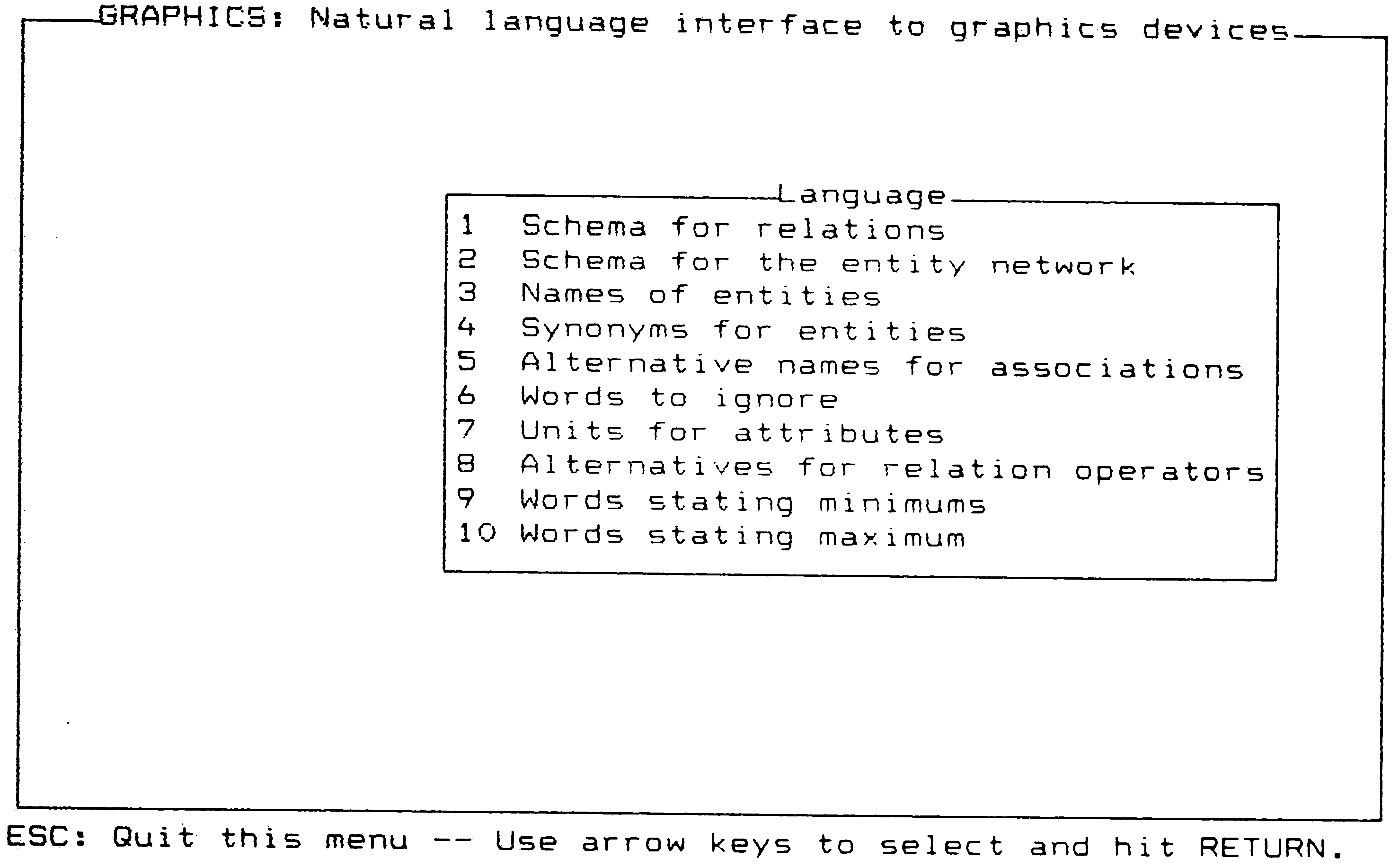
9. Alternative ways to designate the minimum

The different ways to designate adjectives for "minimum" are listed here.

10. Alternative ways to designate the maximum

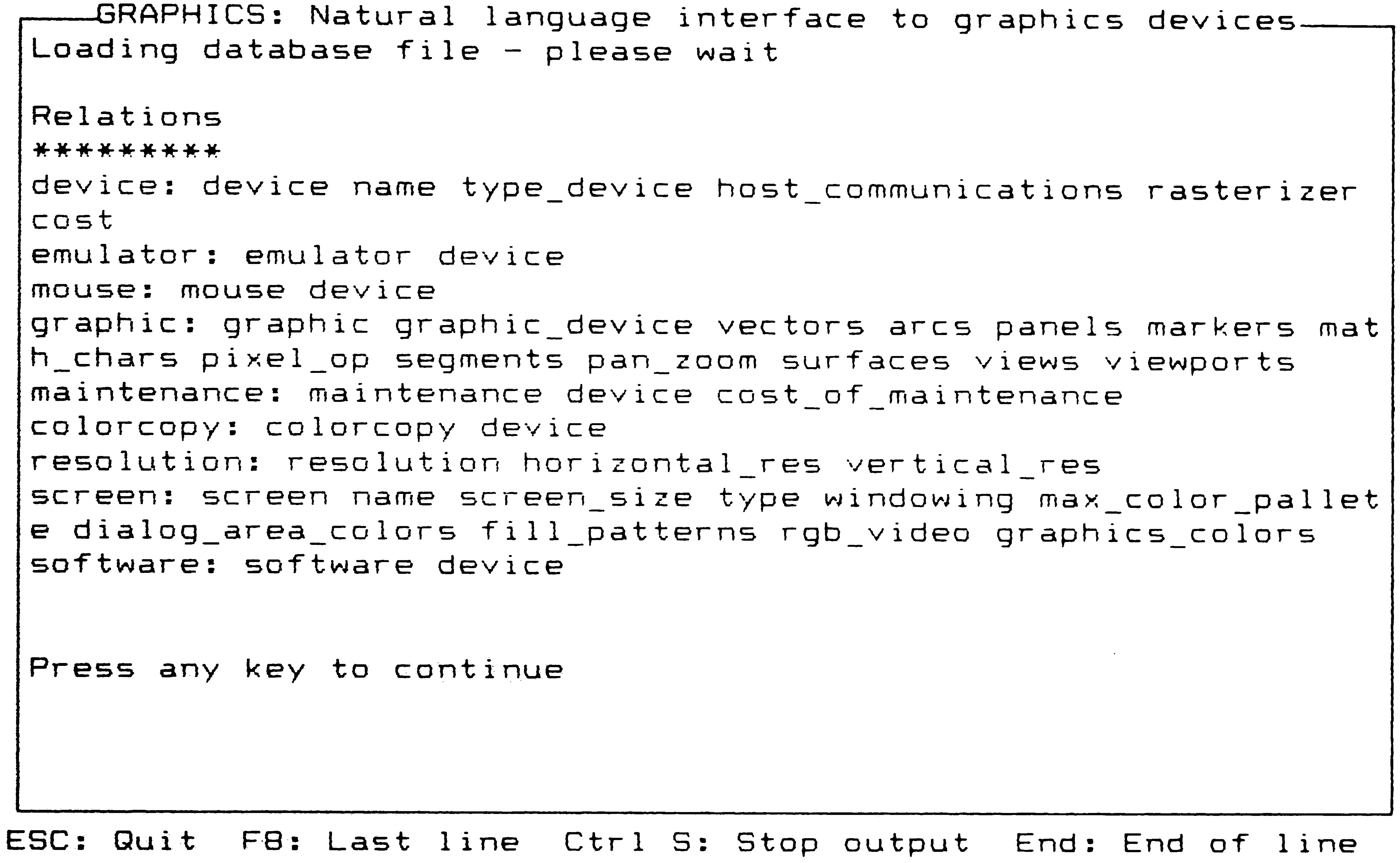
The different ways to designate adjectives for "maximum" are listed here.

Figure 8.2 cont.
Tutorial



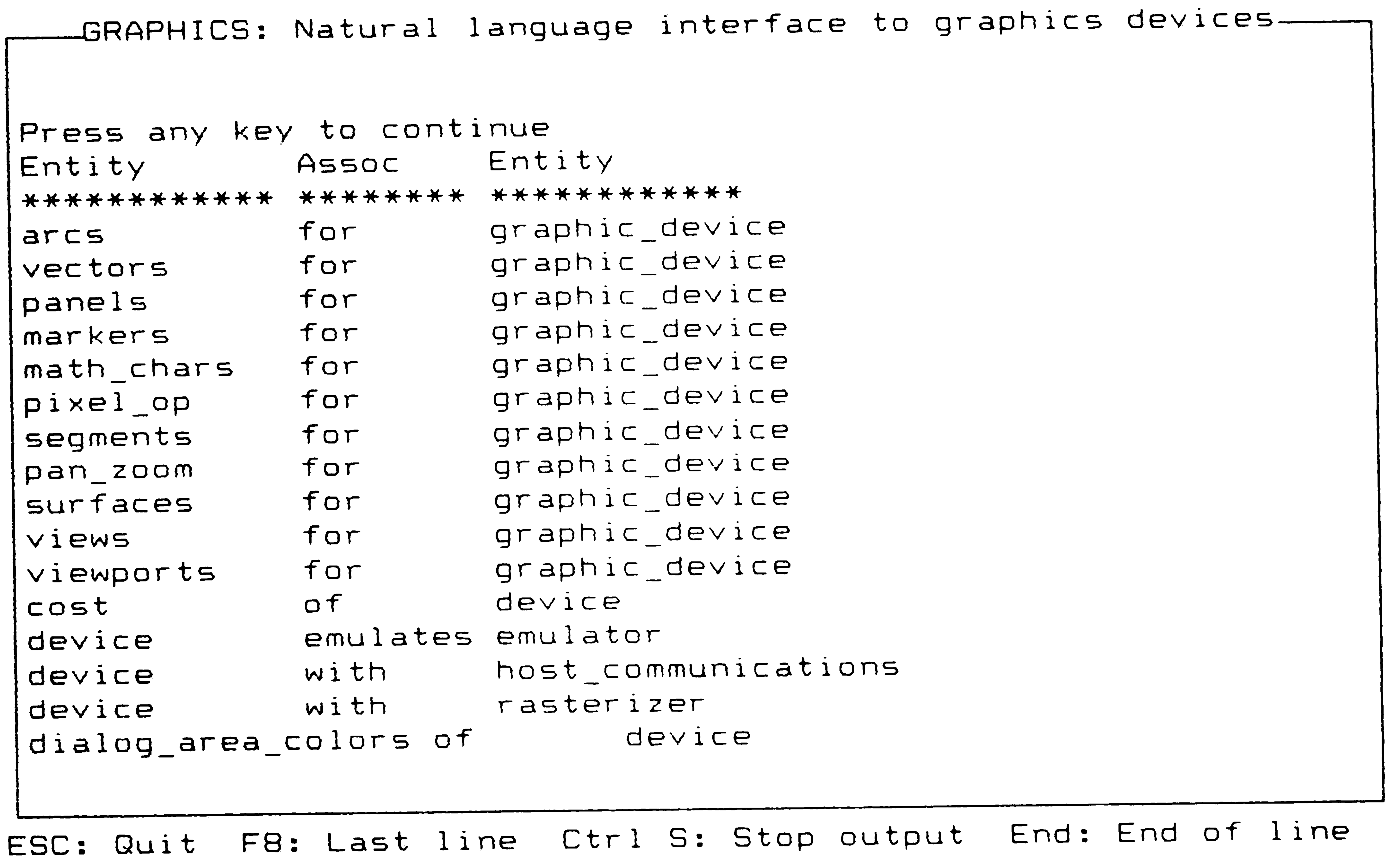
```
GRAPHICS: Natural language interface to graphics devices

                         Language
                1  Schema for relations
                2  Schema for the entity network
                3  Names of entities
                4  Synonyms for entities
                5  Alternative names for associations
                6  Words to ignore
                7  Units for attributes
                8  Alternatives for relation operators
                9  Words stating minimums
                10 Words stating maximum

ESC: Quit this menu -- Use arrow keys to select and hit RETURN.
```

Figure 8.3
View the Language



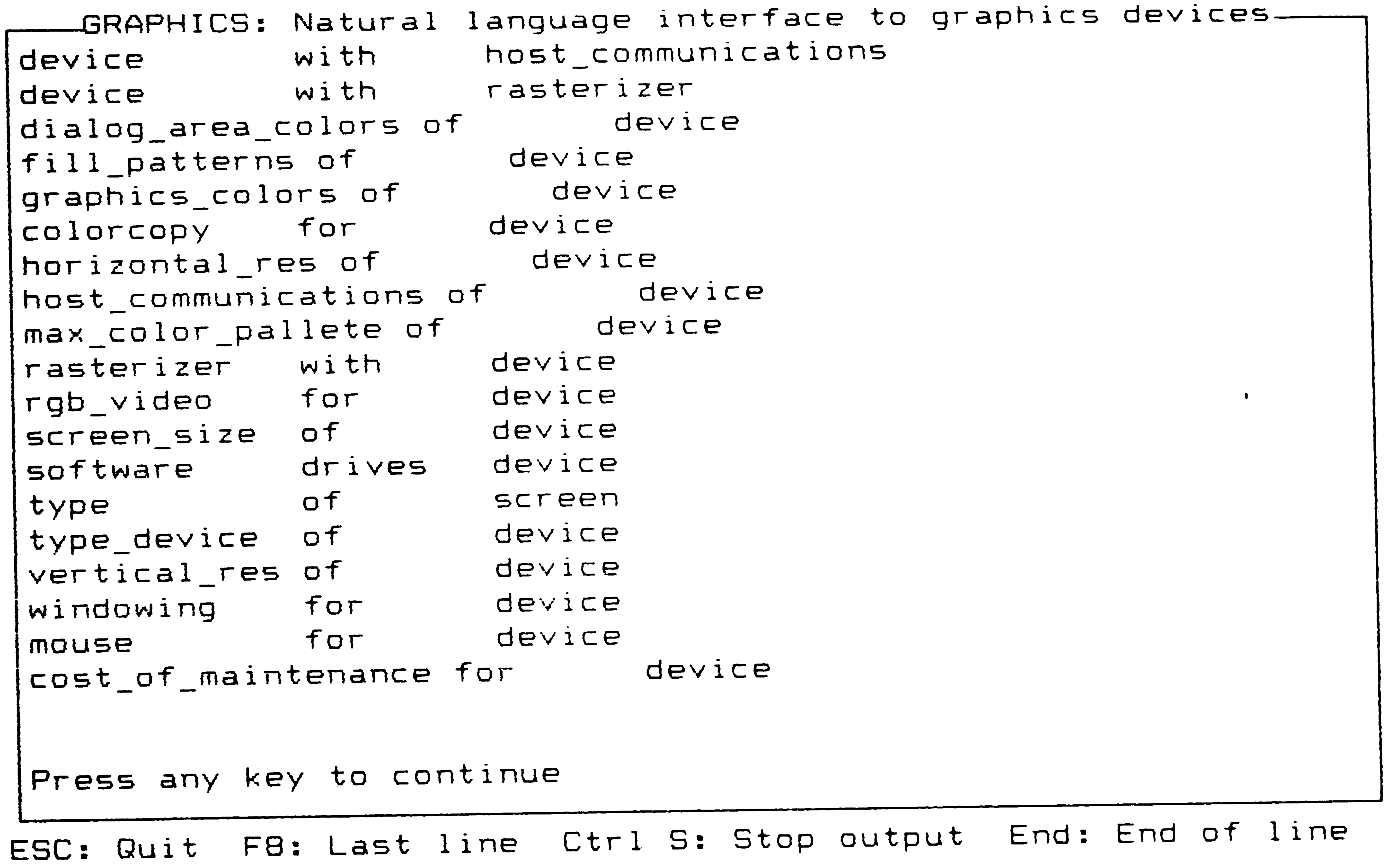
```
GRAPHICS: Natural language interface to graphics devices
Loading database file - please wait

Relations
*********
device: device name type_device host_communications rasterizer
cost
emulator: emulator device
mouse: mouse device
graphic: graphic graphic_device vectors arcs panels markers mat
h_chars pixel_op segments pan_zoom surfaces views viewports
maintenance: maintenance device cost_of_maintenance
colorcopy: colorcopy device
resolution: resolution horizontal_res vertical_res
screen: screen name screen_size type windowing max_color_pallet
e dialog_area_colors fill_patterns rgb_video graphics_colors
software: software device


Press any key to continue




ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

Figure 8.4
Schema for Relations



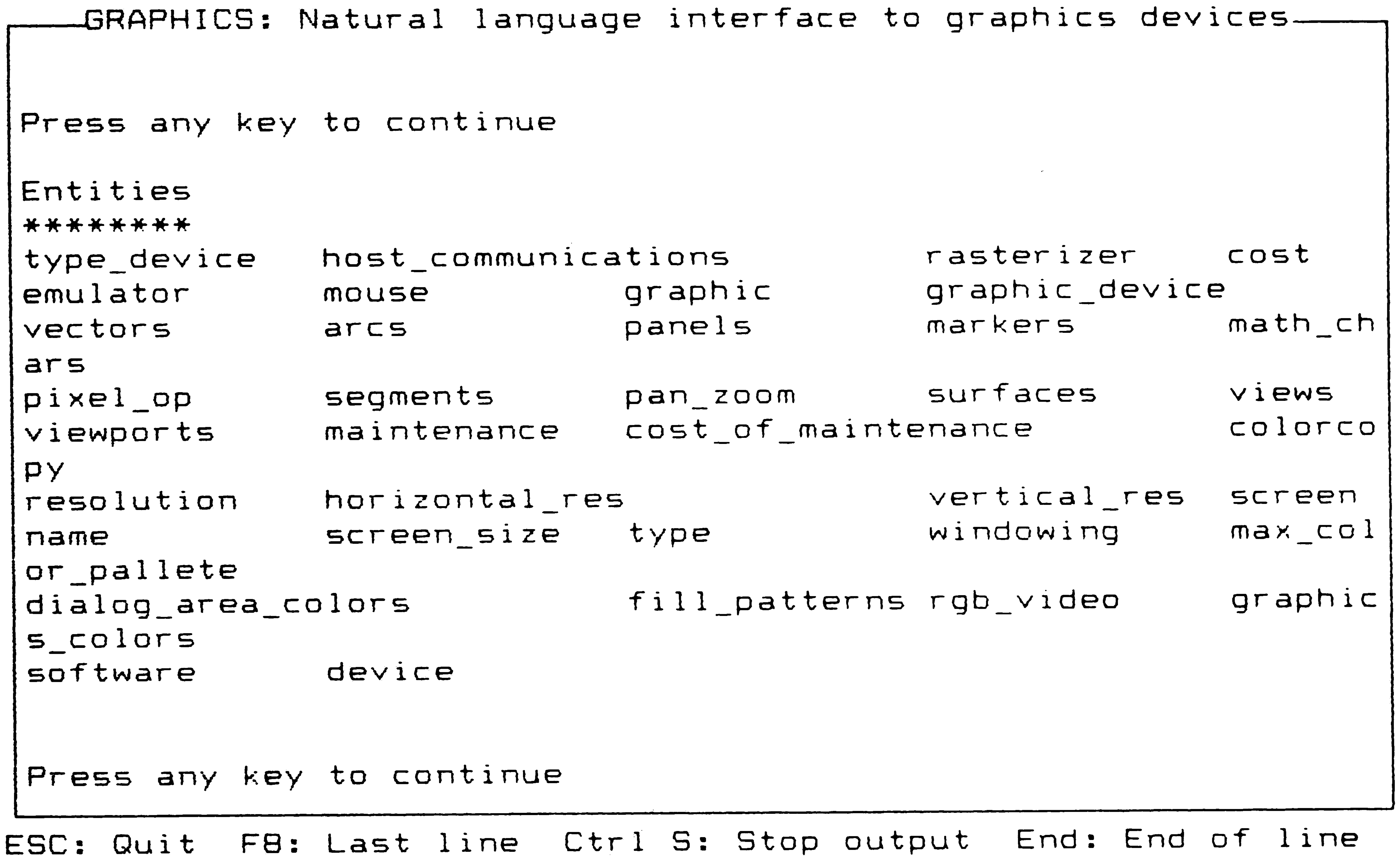
```
GRAPHICS: Natural language interface to graphics devices

Press any key to continue
Entity       Assoc    Entity
************ ******** ************
arcs         for      graphic_device
vectors      for      graphic_device
panels       for      graphic_device
markers      for      graphic_device
math_chars   for      graphic_device
pixel_op     for      graphic_device
segments     for      graphic_device
pan_zoom     for      graphic_device
surfaces     for      graphic_device
views        for      graphic_device
viewports    for      graphic_device
cost         of       device
device       emulates emulator
device       with     host_communications
device       with     rasterizer
dialog_area_colors of       device

ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

Figure 8.5
Schema for Entity Network



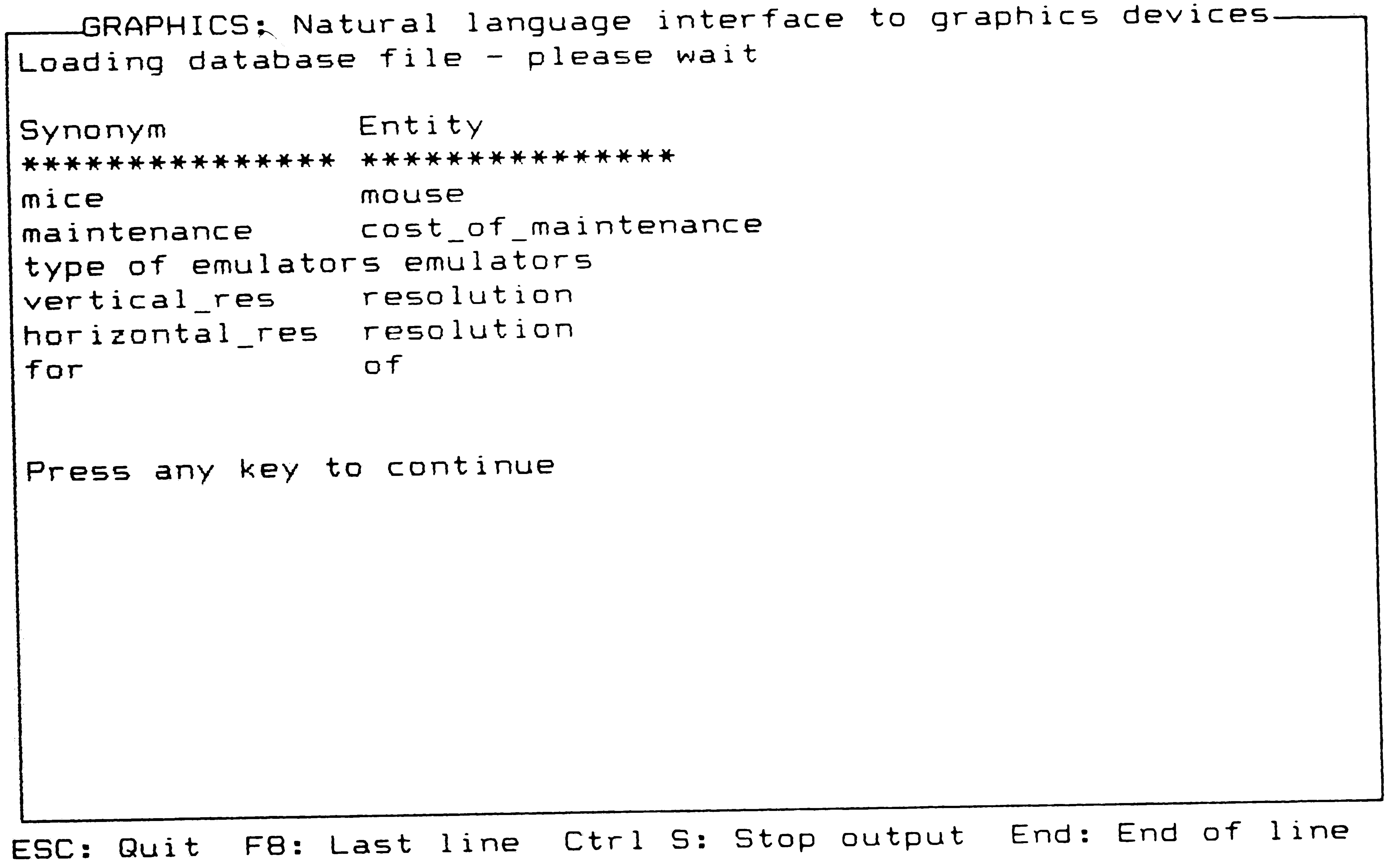
```
GRAPHICS: Natural language interface to graphics devices
device          with     host_communications
device          with     rasterizer
dialog_area_colors of         device
fill_patterns of         device
graphics_colors of         device
colorcopy      for       device
horizontal_res of         device
host_communications of         device
max_color_pallete of         device
rasterizer     with      device
rgb_video      for       device
screen_size    of        device
software       drives    device
type           of        screen
type_device    of        device
vertical_res   of        device
windowing      for       device
mouse          for       device
cost_of_maintenance for        device


Press any key to continue
ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

Figure 8.5
Schema for Entity Network



```
GRAPHICS: Natural language interface to graphics devices

Press any key to continue

Entities
********
type_device   host_communications           rasterizer    cost
emulator      mouse           graphic       graphic_device
vectors       arcs            panels        markers       math_ch
ars
pixel_op      segments        pan_zoom      surfaces      views
viewports     maintenance     cost_of_maintenance         colorco
py
resolution    horizontal_res                vertical_res  screen
name          screen_size     type          windowing     max_col
or_pallete
dialog_area_colors            fill_patterns rgb_video     graphic
s_colors
software      device


Press any key to continue
ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

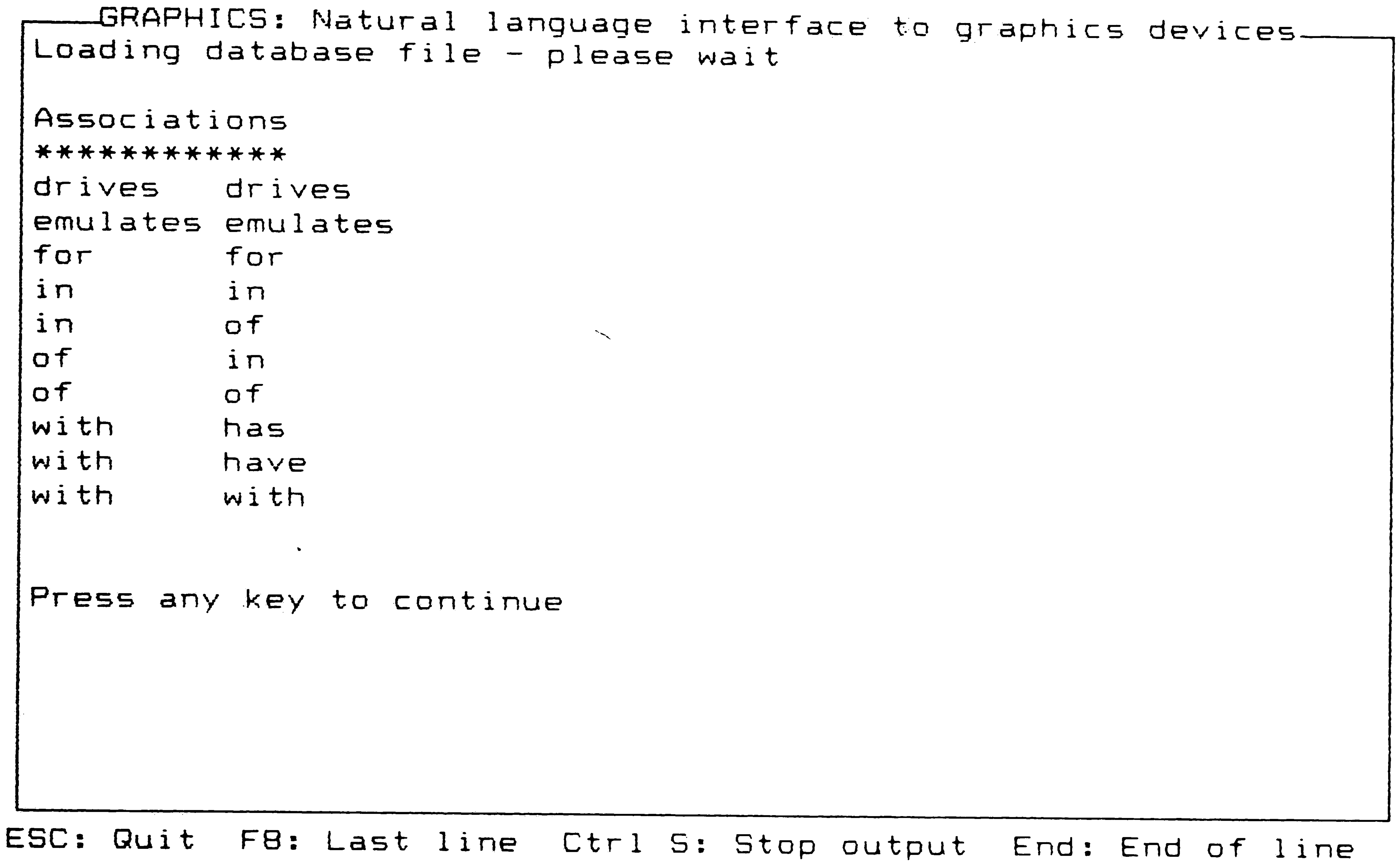
Figure 8.6
Names of Entities



```
      GRAPHICS: Natural language interface to graphics devices
Loading database file - please wait

Synonym          Entity
*************** ***************
mice             mouse
maintenance      cost_of_maintenance
type of emulators emulators
vertical_res     resolution
horizontal_res   resolution
for              of


Press any key to continue
```

ESC: Quit F8: Last line Ctrl S: Stop output End: End of line

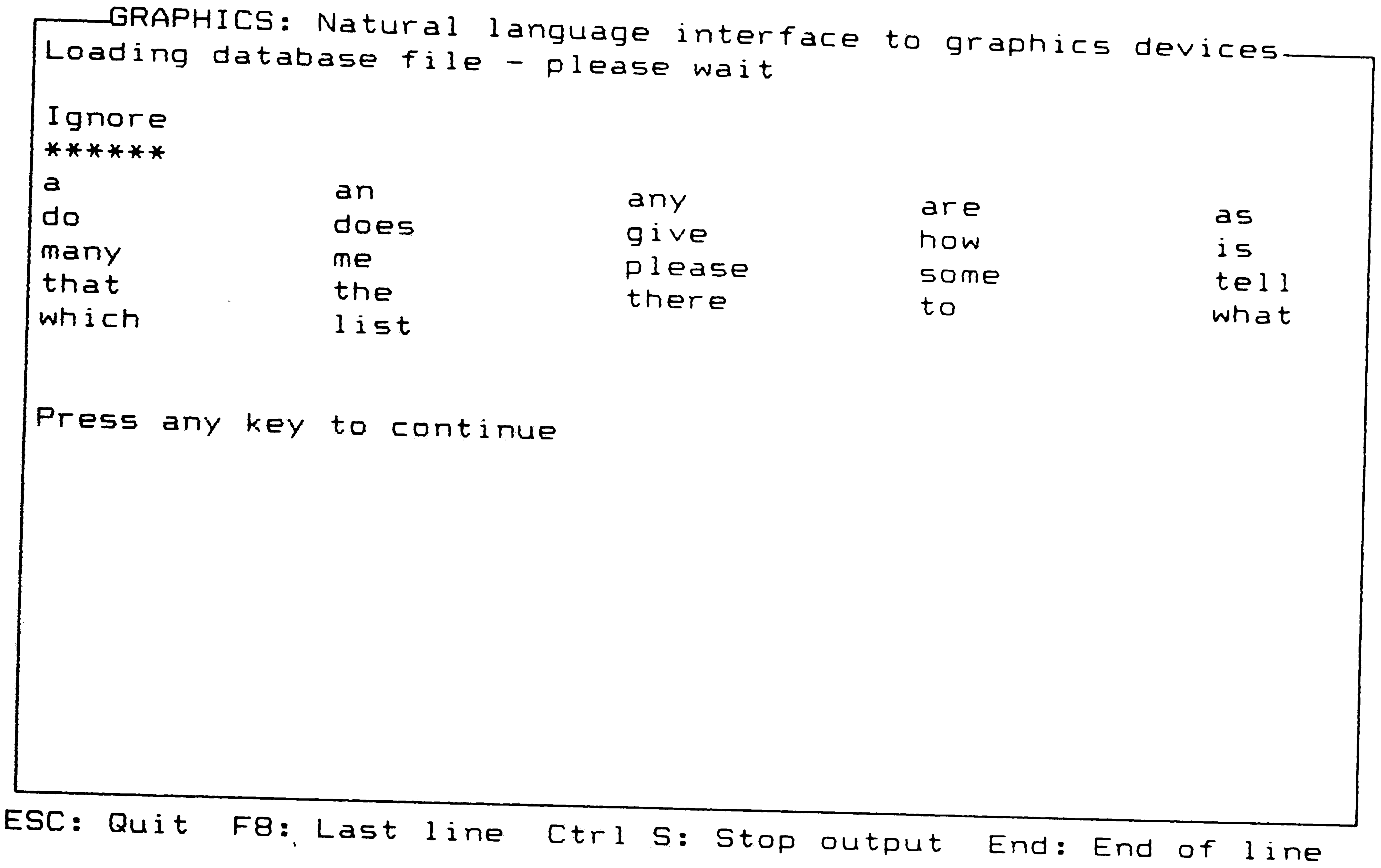
Figure 8.7
Synonyms for Entities



```
GRAPHICS: Natural language interface to graphics devices
Loading database file - please wait

Associations
************
drives   drives
emulates emulates
for      for
in       in
in       of
of       in
of       of
with     has
with     have
with     with


Press any key to continue
```

```
ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

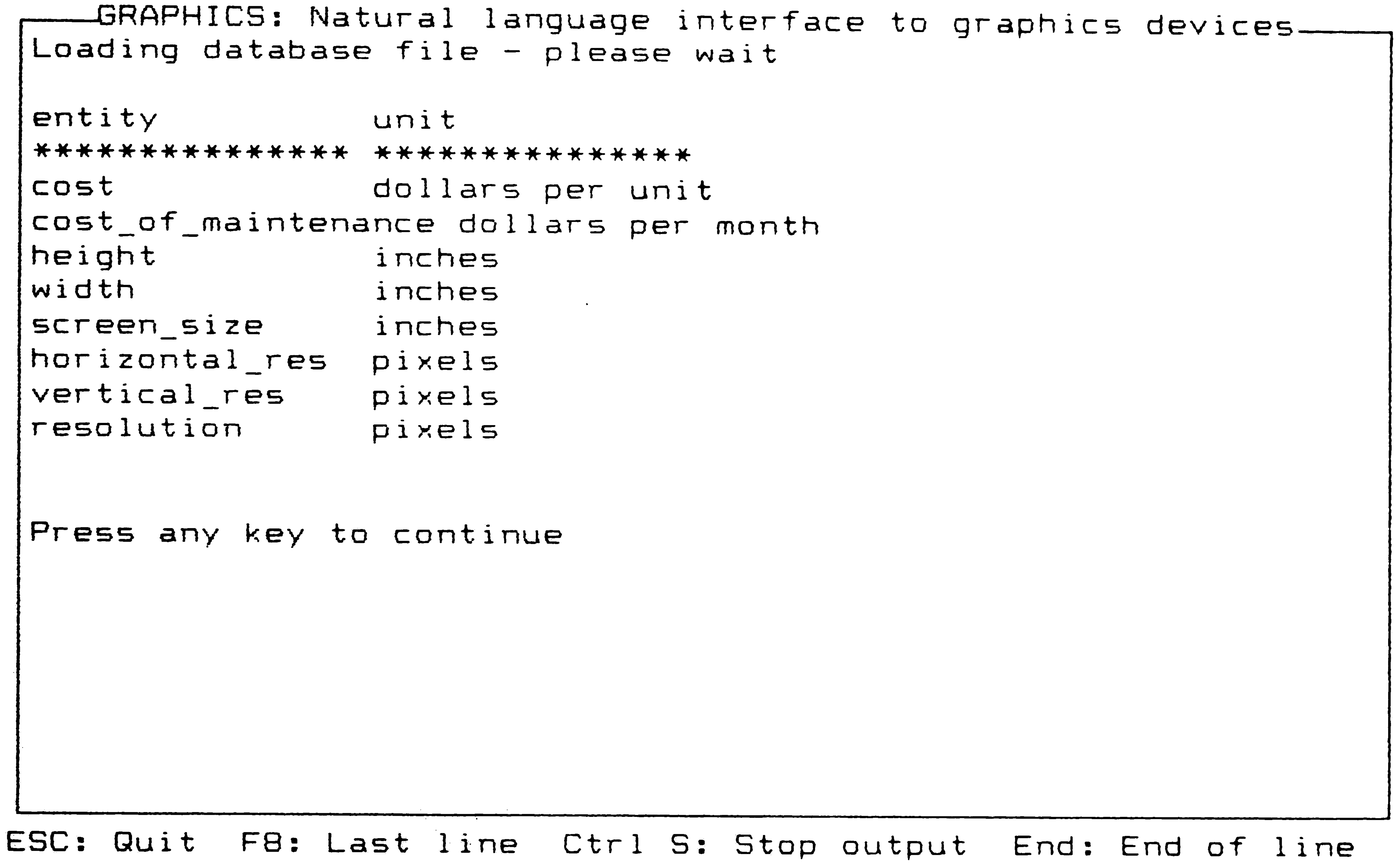
Figure 8.8
Alternative Names for Associations



```
GRAPHICS: Natural language interface to graphics devices
Loading database file - please wait

Ignore
******
a           an          any         are         as
do          does        give        how         is
many        me          please      some        tell
that        the         there       to          what
which       list


Press any key to continue
```

ESC: Quit   F8: Last line   Ctrl S: Stop output   End: End of line

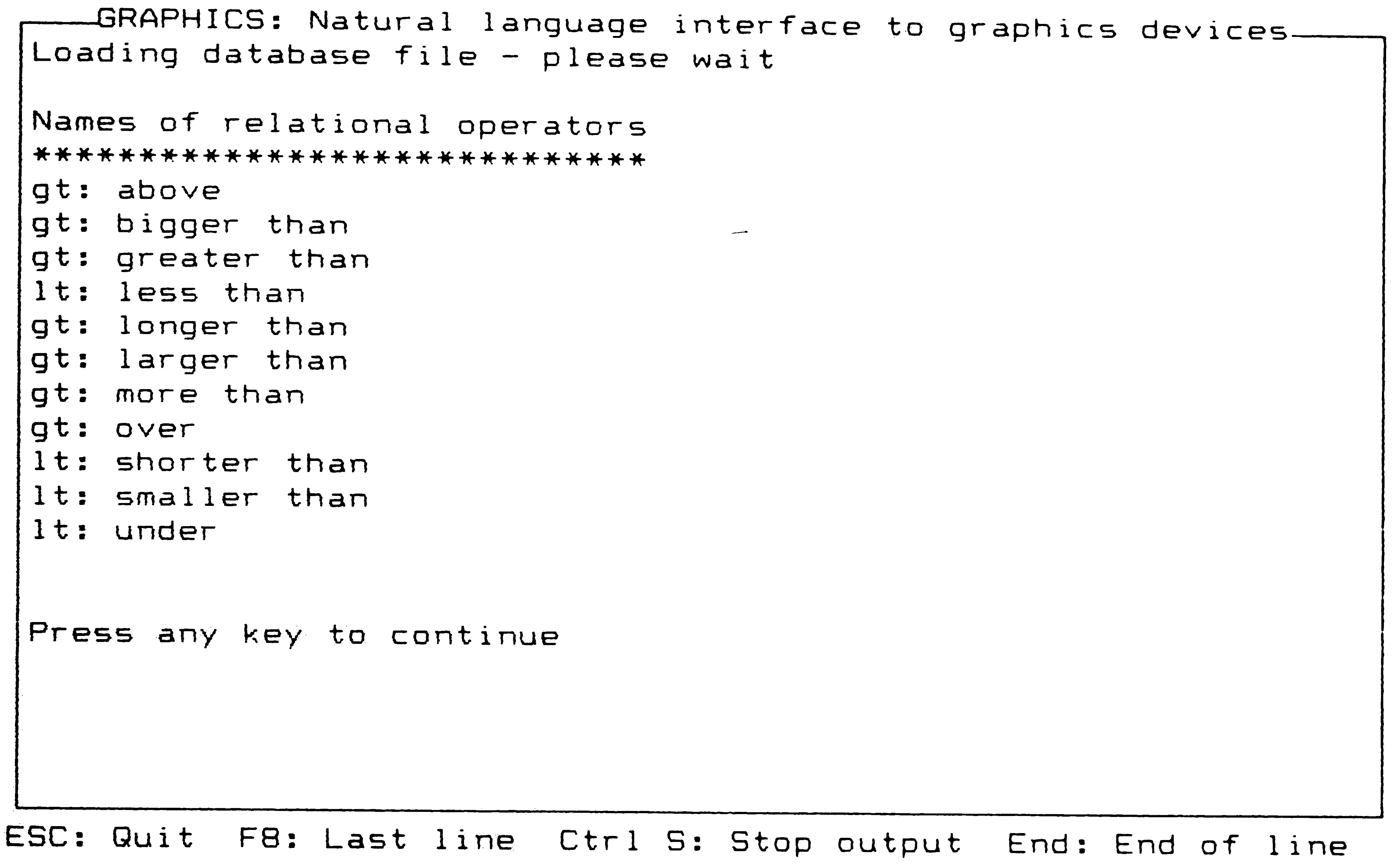
Figure 8.9
Words to Ignore



```
——GRAPHICS: Natural language interface to graphics devices——
Loading database file - please wait

entity              unit
*************** ***************
cost                dollars per unit
cost_of_maintenance dollars per month
height              inches
width               inches
screen_size         inches
horizontal_res      pixels
vertical_res        pixels
resolution          pixels


Press any key to continue
```

```
ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

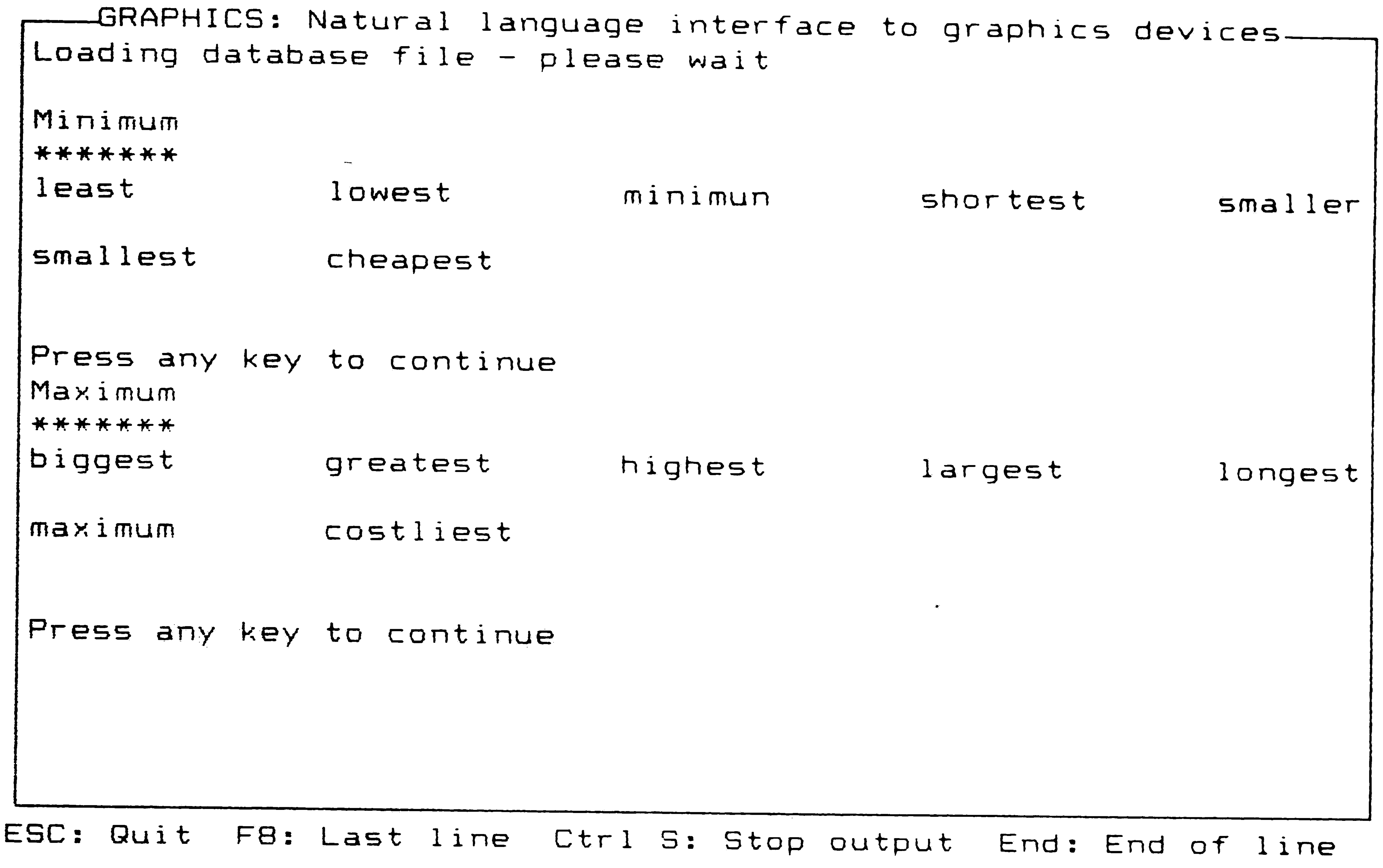
Figure 8.10
Units for Attributes



```
GRAPHICS: Natural language interface to graphics devices
Loading database file - please wait

Names of relational operators
*****************************
gt: above
gt: bigger than
gt: greater than
lt: less than
gt: longer than
gt: larger than
gt: more than
gt: over
lt: shorter than
lt: smaller than
lt: under


Press any key to continue
```

ESC: Quit F8: Last line Ctrl S: Stop output End: End of line

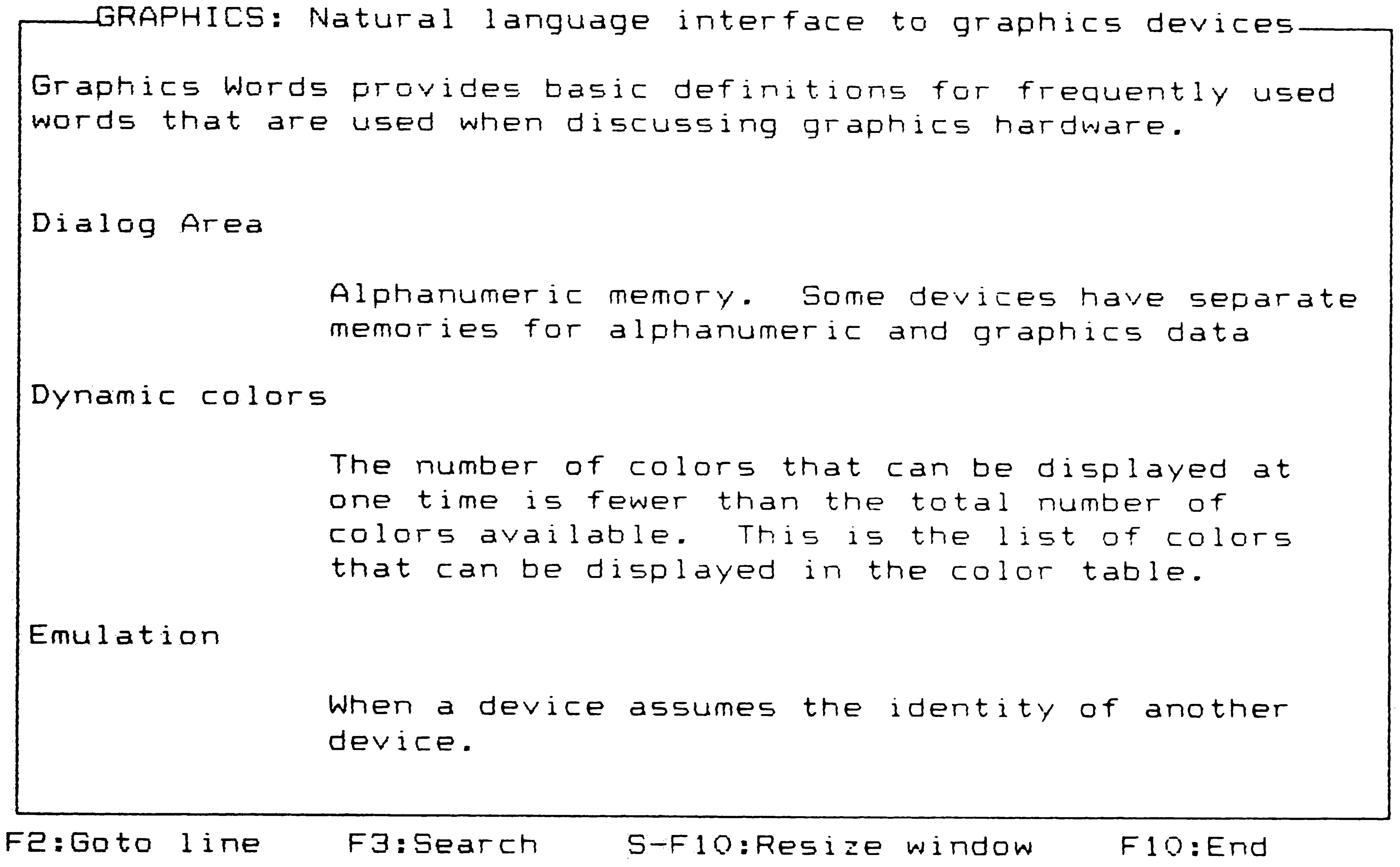
Figure 8.11
Alternatives for Relation Operators



```
——GRAPHICS: Natural language interface to graphics devices——
Loading database file - please wait

Minimum
*******
least          lowest         minimun        shortest       smaller

smallest       cheapest


Press any key to continue
Maximum
*******
biggest        greatest       highest        largest        longest

maximum        costliest


Press any key to continue




ESC: Quit   F8: Last line   Ctrl S: Stop output   End: End of line
```

Figure 8.12

Words Stating Maximums/Minimums



GRAPHICS: Natural language interface to graphics devices

Graphics Words provides basic definitions for frequently used words that are used when discussing graphics hardware.

Dialog Area

Alphanumeric memory. Some devices have separate memories for alphanumeric and graphics data

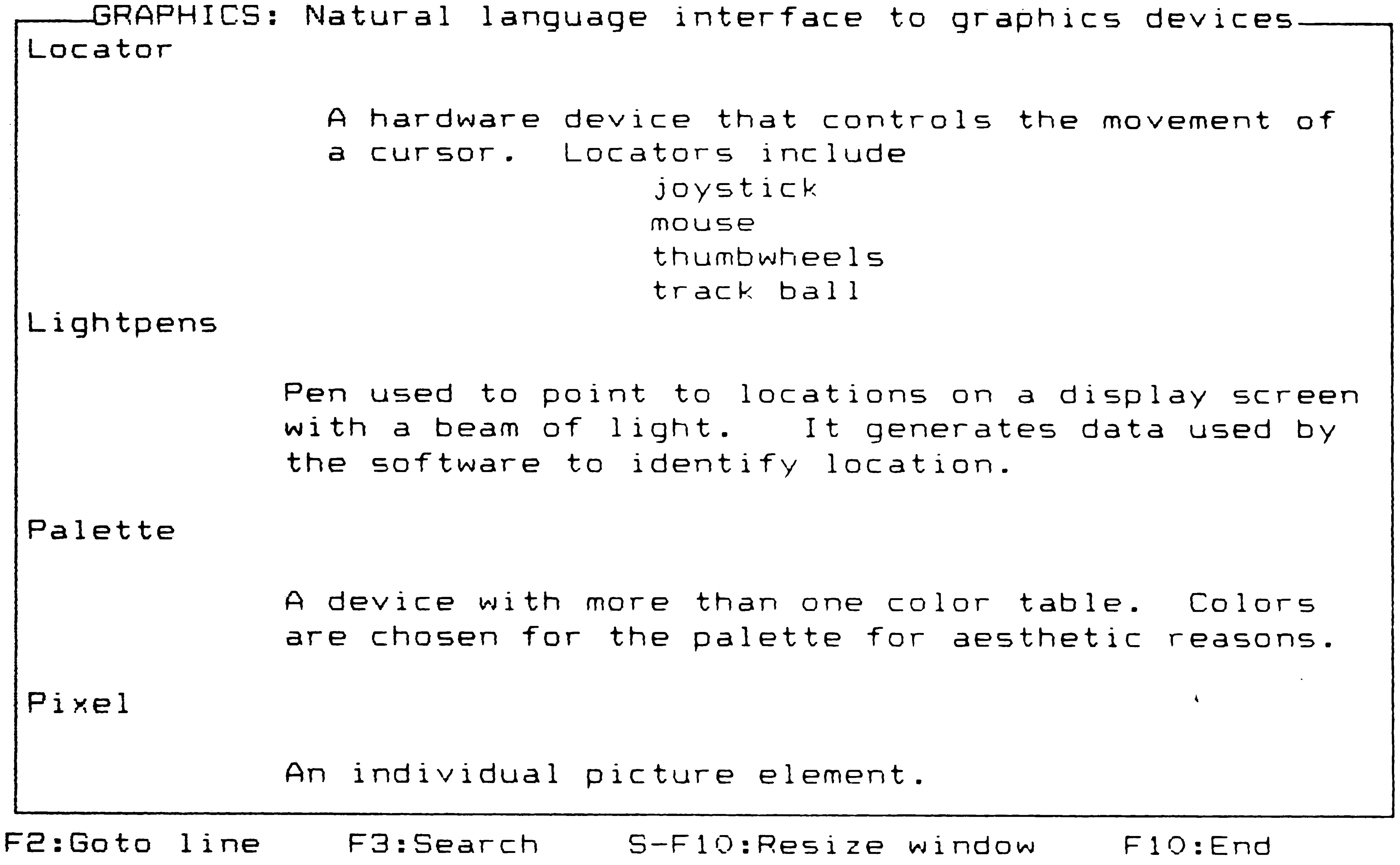
Dynamic colors

The number of colors that can be displayed at one time is fewer than the total number of colors available. This is the list of colors that can be displayed in the color table.

Emulation

When a device assumes the identity of another device.

Figure 8.13
Graphics Words



GRAPHICS: Natural language interface to graphics devices

Locator

A hardware device that controls the movement of a cursor. Locators include

joystick
mouse
thumbwheels
track ball

Lightpens

Pen used to point to locations on a display screen with a beam of light. It generates data used by the software to identify location.

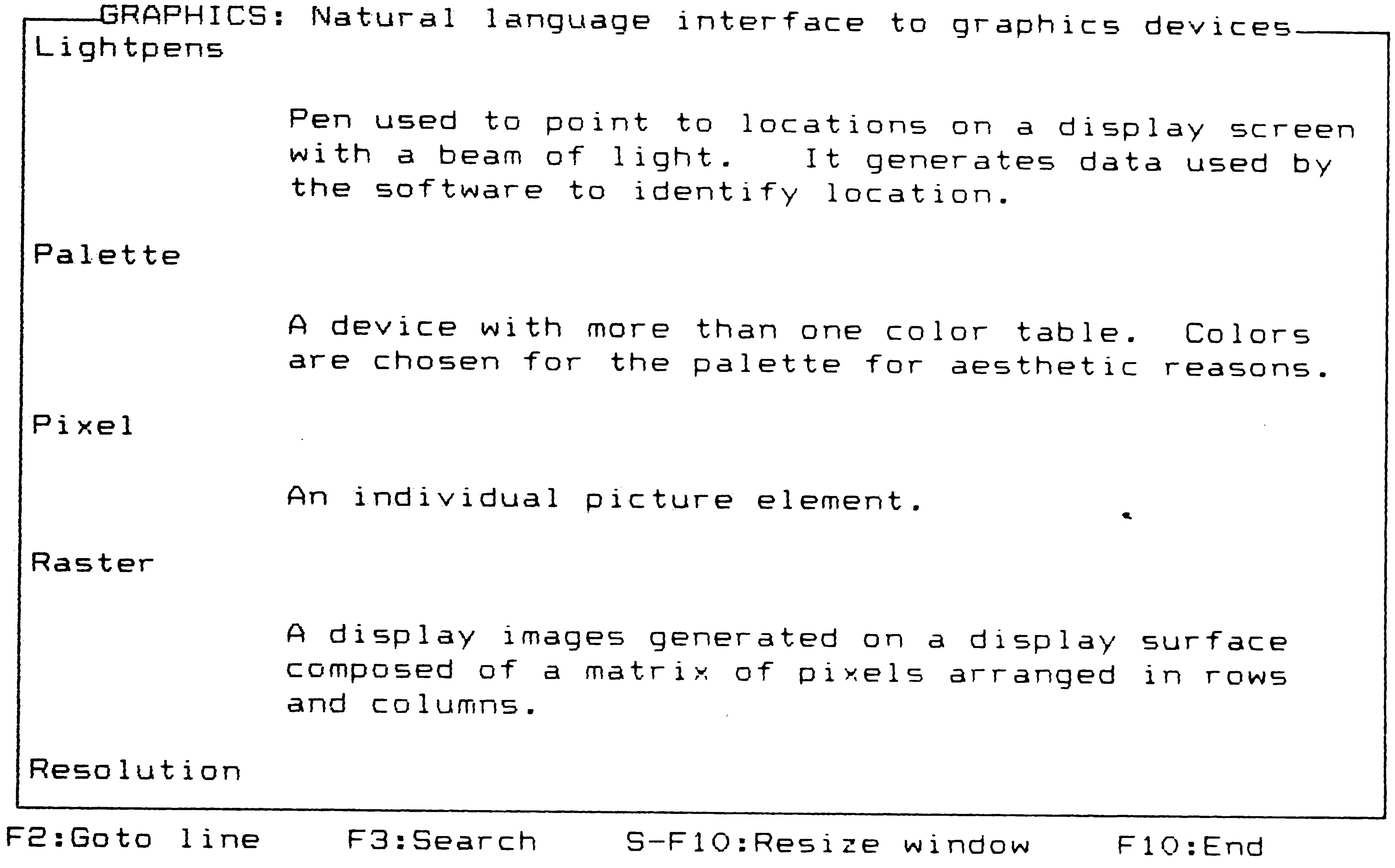
Palette

A device with more than one color table. Colors are chosen for the palette for aesthetic reasons.

Pixel

An individual picture element.

Figure 8.13
Graphics Words cont.



GRAPHICS: Natural language interface to graphics devices

Lightpens

Pen used to point to locations on a display screen with a beam of light. It generates data used by the software to identify location.

Palette

A device with more than one color table. Colors are chosen for the palette for aesthetic reasons.

Pixel

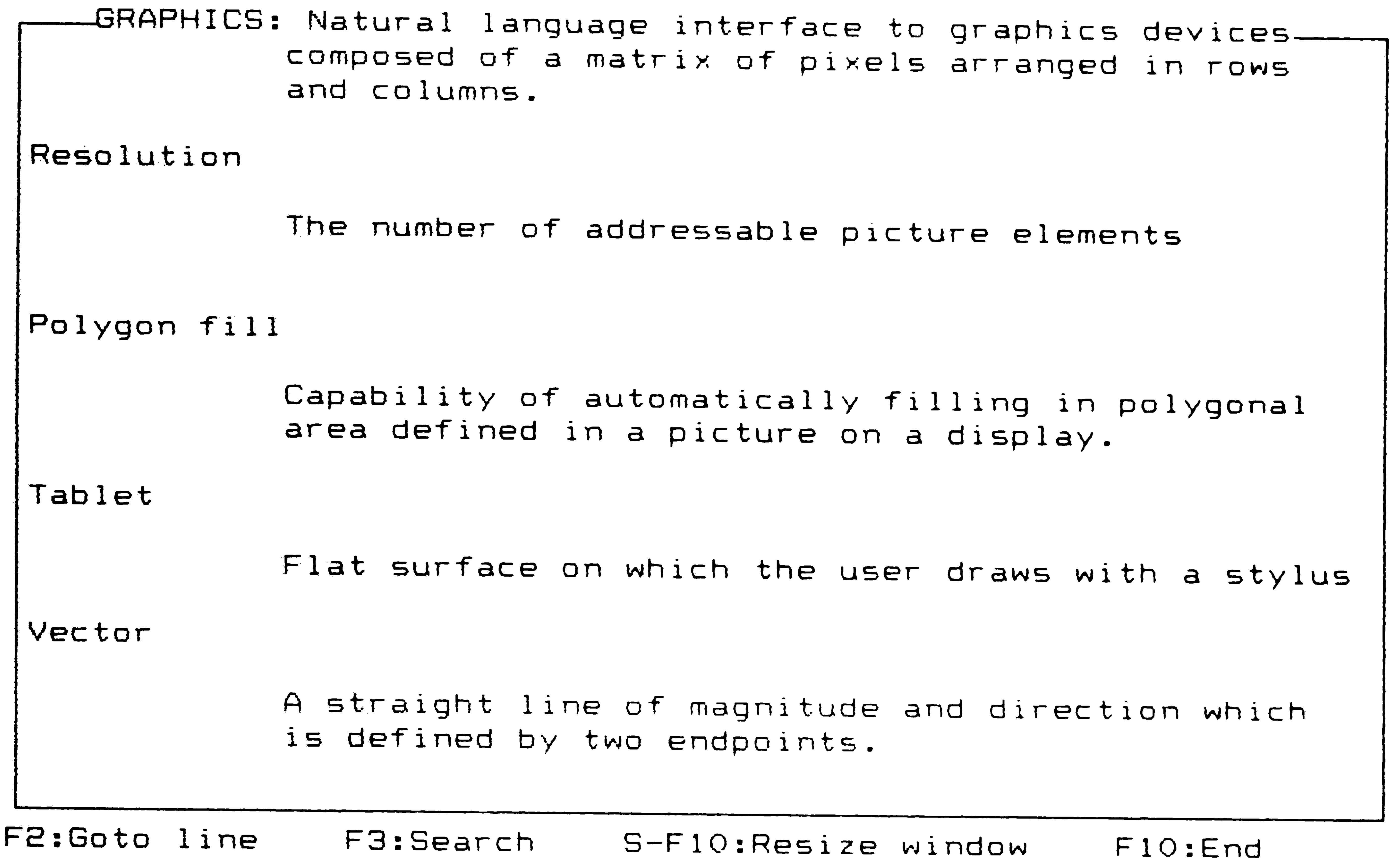
An individual picture element.

Raster

A display images generated on a display surface composed of a matrix of pixels arranged in rows and columns.

Resolution

Figure 8.13
Graphics Words cont.



GRAPHICS: Natural language interface to graphics devices composed of a matrix of pixels arranged in rows and columns.

Resolution

The number of addressable picture elements

Polygon fill

Capability of automatically filling in polygonal area defined in a picture on a display.

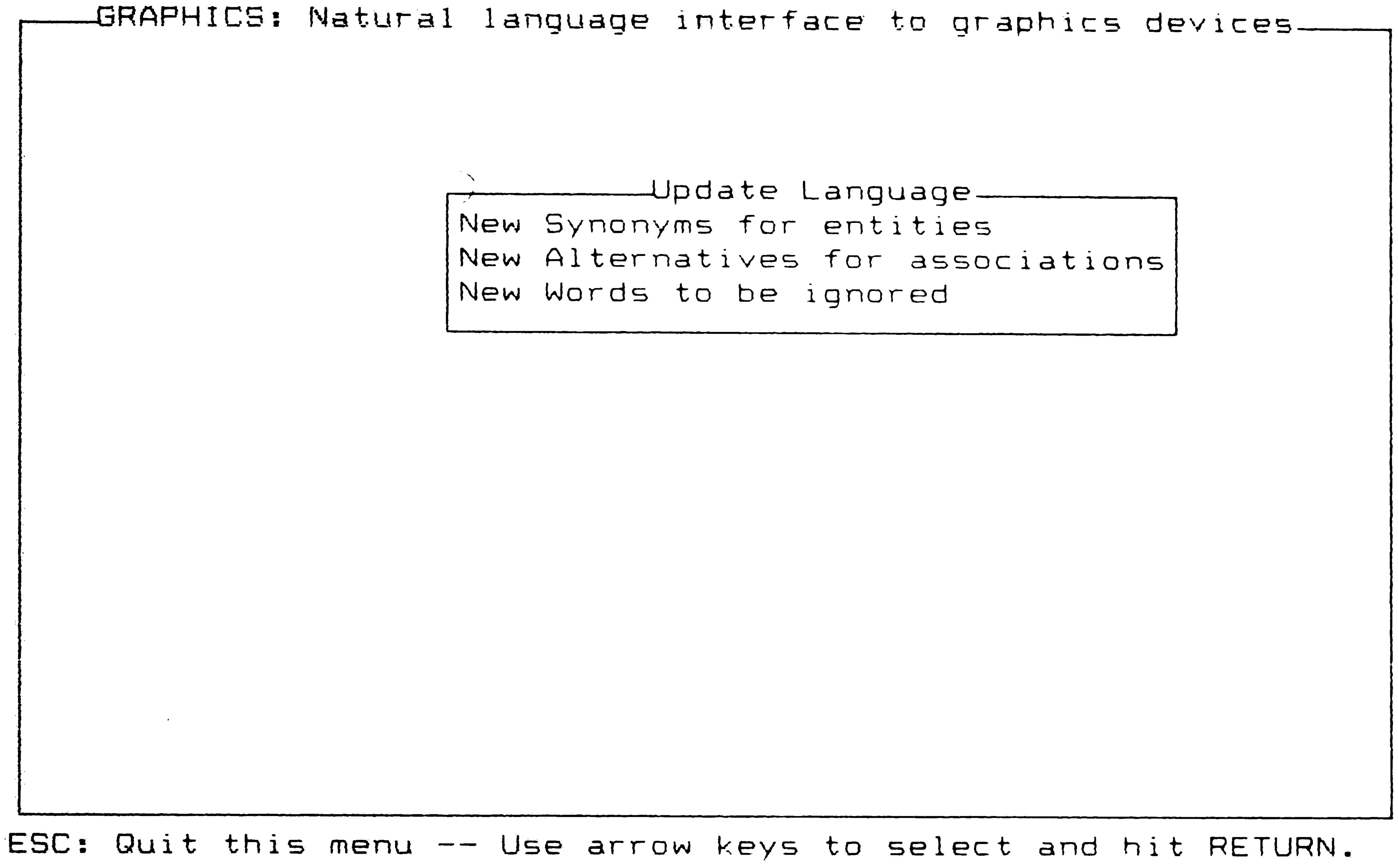
Tablet

Flat surface on which the user draws with a stylus

Vector

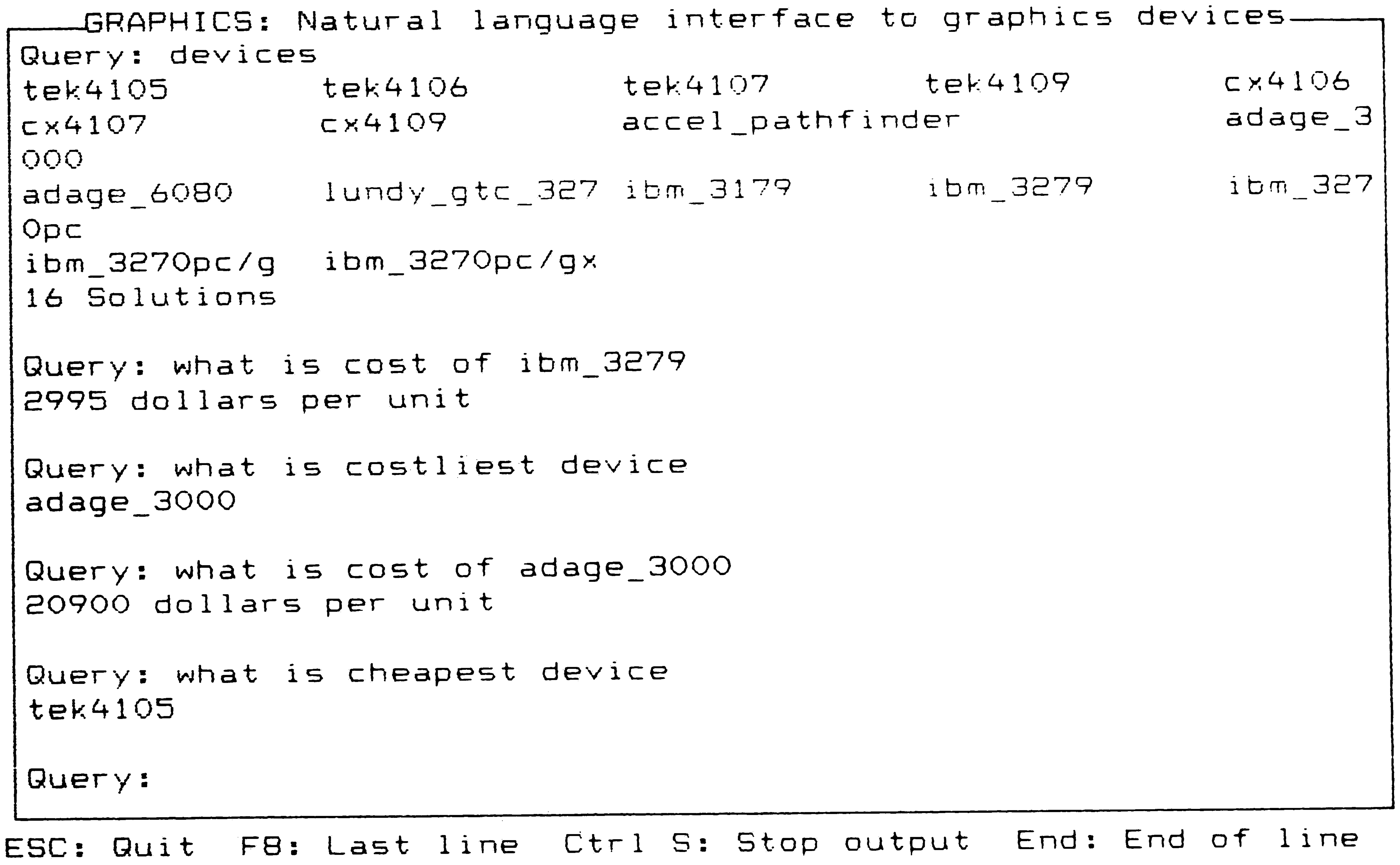
A straight line of magnitude and direction which is defined by two endpoints.

Figure 8.13
Graphics Words cont.



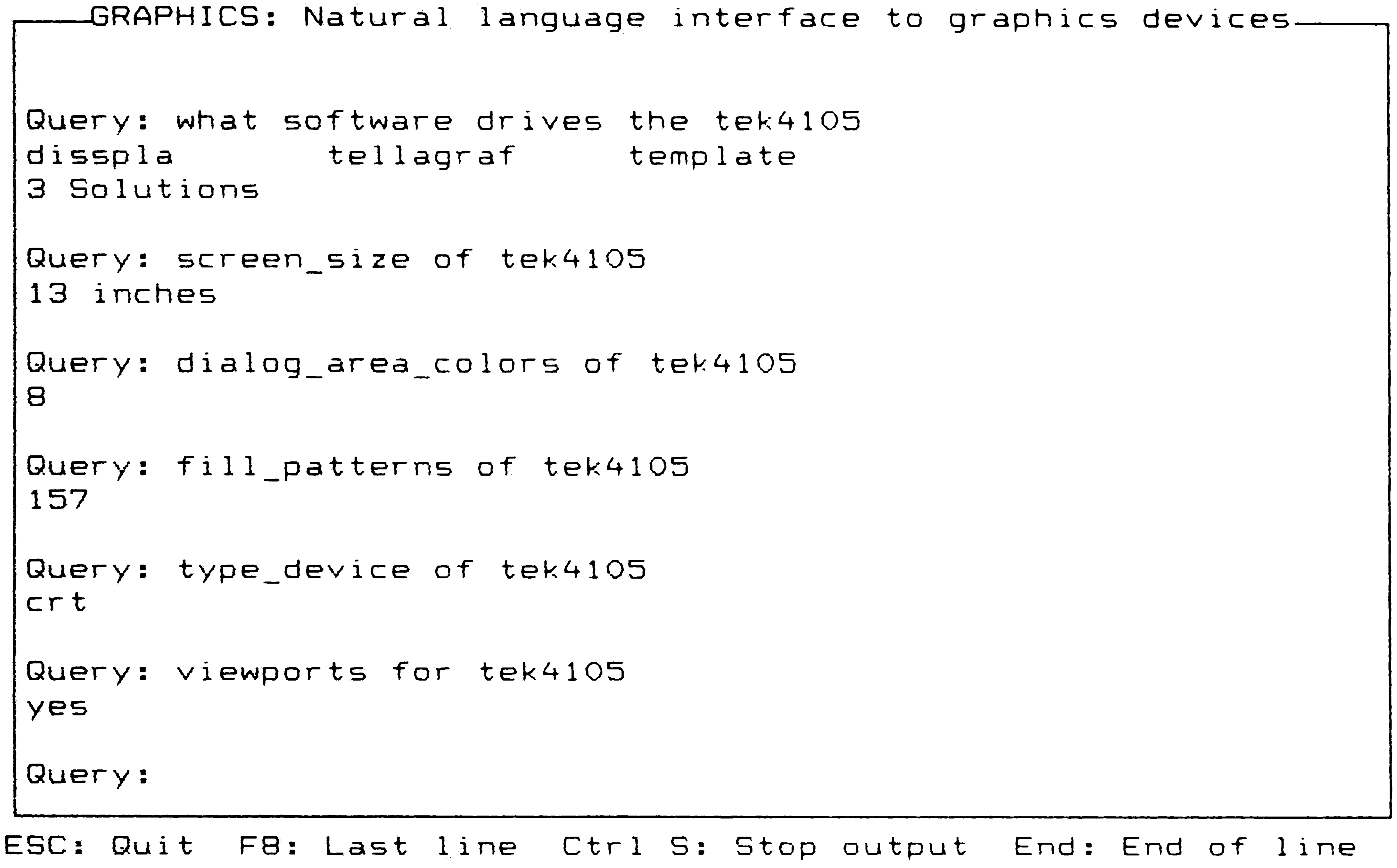
```
GRAPHICS: Natural language interface to graphics devices

                 Update Language
           New Synonyms for entities
           New Alternatives for associations
           New Words to be ignored

ESC: Quit this menu -- Use arrow keys to select and hit RETURN.
```

Figure 8.14
Update the Language



```
GRAPHICS: Natural language interface to graphics devices
Query: devices
tek4105       tek4106       tek4107       tek4109       cx4106
cx4107        cx4109        accel_pathfinder            adage_3
000
adage_6080    lundy_gtc_327 ibm_3179      ibm_3279      ibm_327
0pc
ibm_3270pc/g  ibm_3270pc/gx
16 Solutions

Query: what is cost of ibm_3279
2995 dollars per unit

Query: what is costliest device
adage_3000

Query: what is cost of adage_3000
20900 dollars per unit

Query: what is cheapest device
tek4105

Query:
ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line
```

Figure 8.15
Sample Session



```
GRAPHICS: Natural language interface to graphics devices

Query: what software drives the tek4105
disspla       tellagraf     template
3 Solutions

Query: screen_size of tek4105
13 inches

Query: dialog_area_colors of tek4105
8

Query: fill_patterns of tek4105
157

Query: type_device of tek4105
crt

Query: viewports for tek4105
yes

Query:
```

ESC: Quit F8: Last line Ctrl S: Stop output End: End of line

Figure 8.15
Sample Session cont.

CHAPTER 9

FUTURE RESEARCH

9.1 Future Database Enhancements

With the increasing growth of graphic devices becoming available from manufacturers and the increased movement toward end-user control of the computing environment, continued investigation of new devices and methods to provide connectivity is a high priority. With the cost of devices decreasing, more users will require graphics devices. Continued additions of graphic device characteristics as well as new entries to the existing database will provide the means of maintaining the Assistant. Because there is still considerable growth possible within the existing system, a personal computer can continue to be the environment used to create and maintain such a database.

9.2 Future Software Enhancements

As more users access the system, additions to the language that the system can understand can be completed. This includes phrases that are commonly used as well as synonyms, words to ignore, minimums and maximums. New records can be created without much disruption to the existing database. The tutorial and helps can be expanded readily to explain the newer characteristics. The system is modular in nature so that the database records, tutorials and help files can be maintained separately. The prototype can be expanded with a front-end menu system which could limit the search direction. For example, maybe the user wants to restrict the search to coaxially connected devices. A menu selection could include coax or RS-232. Another environment could include a prompting sytem which would help the system to direct its search to records suggested from the responses. Do you want coax or RS-232 devices? Both types or front-ends would provide techniques to migrate the asssistant toward an actual expert sytem.

CHAPTER 10

CONCLUSIONS

Based on the initial objectives of this prototype, several conclusions can be made with regards to using a personal computer with Prolog as the language for the expert assistant and, with regard to the effectiveness of an expert system for the acquisition of graphics devices. The personal computer expert assistant has shown that graphics device acquisition decisions can be successfully supported using a personal computer. The system is an excellent tool to perform the ad hoc inquiries so often needed in deciding what device is appropriate.

Turbo Prolog was found to be an effective tool for the development of the system. Its many valuable characteristics aided in the development. Among the more useful characteristics are Prolog's flexibility in language. Once an understanding of its capabilities were achieved, the addition of new language structures was easily attained. The ability to separate the various parts of the system into modules was helpful while developing the system and should provide an appropriate environment in which to change the system. An example of this is that the actual database is a separate entity and is also the most likely to be modified.

REFERENCES

1. Barr, Avron and Feigenbaum, The Handbook of Artificial Intelligence. Vol I, California: Heuristech Press-William Kaufman, Inc., 1982.

2. Barr, Avron and Feigenbaum, The Handbook of Artificial Intelligence. Vol II, California: Heuristech Press-WIlliam Kaufman, Inc., 1982.

3. Barr, Avron and Feigenbaum, The Handbook of Artificial Intelligence. Vol III, California: Heuristech Press-WIlliam Kaufman, Inc., 1982.

4. Carbonell, Jaime G., Michalski, Ryszard S., Mitchell, Tom M., "Machine Learning:A Historical and Methodological Analysis", AI Magazine. Vol. 4 No. 3 (1983) pages 56-66.

5. Clocksin, W. F. and Mellish, C. S., Programming in Prolog. Berlin: Springer-Verlag, 1984.

6. Epstein, David A., "Symbolic Structure", Digital Review, Vol. 2, No. 8 (May 1985), pp. 37-49.

7. Epstein, Jonathan A., "Natural Phenomenon", Digital Review, Vol. 2, No. 8 (1985), pp. 54-67.

8. Feigenbaum, Edward A. and McCorduck, Pamela, The Fifth Generation. Massachusetts: Addison-Wesley, 1983.

9. Fisher, Edward L., Knowledge-Based Facilities Design, Ann Arbor, Michigan: University Microfilms International, 1984.

10. Gonzalez, J. Comacho, M. H. Aitchison, "Evaluation of the Effectiveness of Prolog for a CAD Application" , IEEE-CG&A, pp. 67-75.

11. Guertin, Elizabeth, "Overwhelming Evidence", Digital Review, Vol. 2, No. 8 (1985), pp. 60-70.

12. Harmon, Paul and King,David, Expert Systems. New York: Wiley and Sons, 1985.

13. Harvey, Robert, "Artificial Intelligence - Now It's Machines That Think", Iron Age, Vol 3 No 4 (1984), pp.30-42.

14. Hayes-Roth, Frederick, Waterman, Donald A., Lenat, Douglas B., Building Expert Systems. Massachusetts: Addison-Wesley, 1983.

15. Hofstadter, Douglas R.,Godel,Escher,Bach: An Eternal Braid. New York: Basic Books, Inc., 1979.

16. Laroff, Gary P., Choosing Hard Copy Devices for Visual Information Systems. San Diego: ISSCO, 1984.

17. Lenat, Douglas B., "Computer Software For Intelligent Systems", Scientific American Vol. 251, No.3, (1984) pp. 204-213.

18. Paller, Alan, "Million Dollar Application", San Diego, Integrated Software Systems, 1986.

19. Rauch-Hindin, "Artificial Intelligence-A Solution Whose Time Has Come, Part 1", Systems and Software

, December 1983, pp. 150-177.

20. Schlosberg, Jeremy, May 1985, "Almost Human", Digital Review, Vol. 2, No. 8 (1985), pp. 17-24.

21. Scott, Joan E., Introduction to Interactive Computer Graphics. New York: John Wiley and Sons, 1982.

22. Scrown, Susan J., The Artificial Intelligence Experience: An Introduction. Massachusetts: Digital Equipment Corporation, 1985.

23. Stefik, Mark, and Bobrow, Daniel G., and Mittal, Sanjay and Conway, Lynn, "Knowledge Programming in Loops: Report on an Experimental Course", Menlo Park, California, AI Magazine Vol. 4, No.3 (1983), pp. 35-51.

24. 3M Center, 1983, "How to Present More Effectively- and Win More Favorable Responses from More People in Less Time", Audio Visual Division/3M, St. Paul, Minnesota.

25. Von Limbach, Geoffrey and Taylor, Michael B., "Expert System Rules Read Natural Language", Systems and Software, August (1984), pp. 119-125.

26. Walz, David L., No. 4, "Artificial Intelligence", Scientific American, Vol. 247, No. 4 (1982),pp. 118-133.

27. Waterman, Donald A., A Guide to Expert Systems. Massachusetts: Addison-Wesley, 1986.

28. Winston, Patrick H. and Prendergast, Karen A., The AI Business-Commercial Uses of Artificial Intelligence. Massachusetts: MIT, 1984.

APPENDIX A

GRAPHICS CHARACTERISTICS GUIDELINES

o minimum of 60 Hertz refresh rate

Lower Hertz rates will cause the display to flicker. At 60 Hertz the display is refreshed at a rate of 60 times per second.

o color convergence

Electron beam guns must converge or come together to create crisp lines

o adjustable brightness

o resolution

Match the resolution to the requirements of the application. Also relevant are some other aspects of the device. If local segmentation as well as pan and zoom are available with which to enlarge a drawing perhaps high resolution is not a requirement.

The following provide guidelines to differentiate pixel resolution

o low resolution	200 X 200
o medium resolution	640 X 480
o high resolution	1000 X 1000

A pixel resolution of 4(X-axis) :3(Y-axis) will uniformly distribute the pixels for displaying both text and graphics

- readable fonts

  capable of displaying both upper and lower case

- separate graphics and text areas
  - multi-use device
  - interaction with the computer does not damage the graphics
  - menus, prompts do not disturb graphics

- emulates commonly used devices
- vector primitives are available
- arc, panels possible
- line variations such as dashed,dotted or solid are available
- use of panels and panel fill
- markers
- graphtext
  - draw as series of vectors
  - can make use of user defined fonts
- Graphics input
  - interactive tools for edition and creating graphs
  - pick, locate and stroke functions
- Use of local segments
  - offload host
  - lower communication needs
  - position, scaling and rotation functions
  - editing

- o Pan and zoom
  - o enlarging
  - o making smaller
  - o can lower resolution needs
  - o can review large drawings a piece at a time
- o Viewports and windows
  - o multiple graphics input area
- o Surfaces
- o Character cell graphics
  - o special fonts such as math characters
  - o international keyboard
- o Pixel operations
- o Color Capability
  - o number of possible colors that are displayable
  - o number of colors that can be displayed at one time
  - o can the colors be altered by the hardware

- o Alphanumeric Considerations
  - o splitting of graphics and dialog areas
  - o ANSI X3.64 systax support which allows common text editors
  - o IBM 3279 alphanumic support (coax connected)

- o Communications
  - o RS-232 (asychronous or ASCII)
  - o COAX (synchronous or EBCDIC)

Most devices are capable of one or the other communication method although some manufacturers such as Tektronix offer terminals that are capable of switching between the two environments.

Hardware Capabilities

- o What devices are potential selections?
- o good quality
- o speed
- o How does what you see on the screen get to the hardcopy unit?

Software support

- o software compatibility with device
- o available device drivers or software that translates software to a form that the terminal can respond to
- o does software have growth potential?

APPENDIX B

EXAMPLE GRAPHICS TERMINALS

Vendor	Model	Resolution	Screen Size
Hewlett-Packard	2397A	512 X 390	12 "
Tektronix	4105	480 X 360	13"
Envision	220	640 X 480	13"
Envision	230	640 X 480	13"
Tektronix	4107	640 X 480	13"
Tektronix	CX4107	640 X 480	13"
Digital	241	800 X 240	13"
IBM	3179/G	720 X 384	14"
IBM	3270-PC/G	720 X 512	14"
Seiko	1104	720 X 512	14"
Tektronix	4109	640 X 480	19"
IBM	3270-PC/GX	960 X 1000	19"
Tektronix	4111	1024 X 780	19"
AED	1024	1024 X 767	19"
Tektronix	4125	1280 X 1024	19"
Seiko	2414	1280 X 1024	20"

APPENDIX C

GRAPHICS ASSISTANT PROGRAM LISTINGS

```
/* G R A P H I C S */
/*
  GRAPHICS-The database with a natural language
             interface

*/
/*diagnostics*/
code=4000
DOMAINS
/*
  The domain LIST is a list of words
*/
  LIST = SYMBOL*

DATABASE
/*
 The Language Database - These are the database predicates
 which define the language we will use to query Graphics
*/


  relat(SYMBOL,LIST)             /* Relation name and a list of attributes */
  schema(SYMBOL,SYMBOL,SYMBOL)   /* Entity network: entity-assoc-entity */

  entitysize(SYMBOL,SYMBOL)      /* This attribute gives the size of an entity */
  relop(LIST,SYMBOL)             /* Example: relop([greater,than],gt] */
  assoc(SYMBOL,LIST)             /* Alternative assoc names */
  synonym(SYMBOL,SYMBOL)         /* Synonyms for entities */
  ignore(SYMBOL)                 /* Words to be ignored */
  min(SYMBOL)                    /* Words stating minimum */
  max(SYMBOL)                    /* Words stating maximum */
  big(SYMBOL,SYMBOL)             /* big, long, high .... */
  unit(SYMBOL,SYMBOL)            /* Unit for length, cost ... */

/*
  The real database - These are the database predicates which
  actually  maintain the information we will access.
*/

/*software(SOFTWARE,DEVICELIST) */
  software(SYMBOL,LIST)

/*colorcopy(COPIER,DEVICELIST)   */
  colorcopy(SYMBOL,LIST)
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
/*device(NAME,TYPE_DEVICE,HOST_COMMUNICATION,RASTERIZER,COST)
*/

  device(SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL)

/*emulator(emulator,devicelist) */
  emulator(SYMBOL,LIST)

/*mouse(mouse,devicelist)   */
  mouse(SYMBOL,LIST)

/*maintenance(device,cost) */
  maintenance(SYMBOL,SYMBOL)

/*resolution(NAME,HORIZONTAL_RES,VERTICAL_RES) */
  resolution(SYMBOL,SYMBOL,SYMBOL)

/*screen(NAME,SCREEN_SIZE,TYPE,WINDOWING,MAX_COLOR_PALLETTE,
        DIALOG_AREA_COLORS,FILL_PATTERNS,RGB_VIDEO,GRAPHICS_COLORS)
*/

  screen(SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,
        SYMBOL,SYMBOL)

/*graphic(name,vectors,arcs,panels,markers,math_chars,pixel_op,
         segments,pan_zoom,surfaces,views,viewports)
*/
   graphic(SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL,
          SYMBOL,SYMBOL,SYMBOL,SYMBOL)

/* Database Access */

PREDICATES
  member(SYMBOL,LIST)           /* membership of a list */

CLAUSES
  member(X,[X¦_]).
  member(X,[_¦L]):-member(X,L).

PREDICATES
/* Access to database */
  db(SYMBOL,SYMBOL,SYMBOL,SYMBOL,SYMBOL)
  ent(SYMBOL,SYMBOL)
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
CLAUSES

/*
  ent returns values for a given entity name. Ex. if called by
  ent(city,X)  X  is instantiated to cities.
*/

  ent(device,DEVICE):-           device(DEVICE,_,_,_,_).
  ent(screen,NAME):-             screen(NAME,_,_,_,_,_,_,_,_).
  ent(graphic_device,NAME):-     graphic(NAME,_,_,_,_,_,_,_,_,_,_,_).
  ent(emulator,NAME):-           emulator(NAME,_).
  ent(software,NAME):-           software(NAME,_).
  ent(colorcopy,NAME):-          colorcopy(NAME,_).
  ent(mouse,NAME):-              mouse(NAME,_).
/*
  The db predicate is used to establish relationships between
  entities. The first three parameters should always be instantiated
  to entityname-assocname-entityname. The last two parameters
  return the values corresponding to the two entity names.
*/

/* Relationships about resolution */

   db(vertical_res,of,device,VERTICAL_RES,DEVICE):-
      resolution(DEVICE,_,VERTICAL_RES).

   db(horizontal_res,of,device,HORIZONTAL_RES,DEVICE):-
      resolution(DEVICE,HORIZONTAL_RES,_).

/* Relationships about screen */

   db(screen_size,of,device,SCREEN_SIZE,DEVICE):-
      screen(DEVICE,SCREEN_SIZE,_,_,_,_,_,_,_).

   db(type,of,screen,TYPE,DEVICE):-
      screen(DEVICE,_,TYPE,_,_,_,_,_,_).

   db(max_color_pallete,of,device,MAX_COLOR_PALLETE,DEVICE):-
      screen(DEVICE,_,_,_,MAX_COLOR_PALLETE,_,_,_,_).

   db(windowing,for,device,WINDOWING,DEVICE):-
      screen(DEVICE,_,_,WINDOWING,_,_,_,_,_).

   db(dialog_area_colors,of,device,DIALOG_AREA_COLORS,DEVICE):-
      screen(DEVICE,_,_,_,_,DIALOG_AREA_COLORS,_,_,_).
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
   db(fill_patterns,of,device,FILL_PATTERNS,DEVICE):-
      screen(DEVICE,_,_,_,_,_,FILL_PATTERNS,_,_).

   db(rgb_video,for,device,RGB_VIDEO,DEVICE):-
      screen(DEVICE,_,_,_,_,_,_,RGB_VIDEO,_).

   db(graphics_colors,of,device,GRAPHICS_COLORS,DEVICE):-
      screen(DEVICE,_,_,_,_,_,_,_,GRAPHICS_COLORS).

/* Relationships aboout emulators   */

   db(device,emulates,emulator,DEVICE,EMULATOR):-
      emulator(EMULATOR,LIST),member(DEVICE,LIST).

/* Relationships about colorcopy   */

   db(colorcopy,for,device,COLORCOPY,DEVICE):-
      colorcopy(COLORCOPY,LIST),member(DEVICE,LIST).

   db(mouse,for,device,MOUSE,DEVICE):-
      mouse(MOUSE,LIST),member(DEVICE,LIST).

/* Relationships about software   */

   db(software,drives,device,SOFTWARE,DEVICE):-
      software(SOFTWARE,LIST),member(DEVICE,LIST).

/* Relationships aboout devices     */

   db(type_device,of,device,TYPE_DEVICE,DEVICE):-
      device(DEVICE,TYPE_DEVICE,_,_,_).

   db(cost,of,device,COST,DEVICE):-
      device(DEVICE,_,_,_,COST).

   db(device,with,host_communications,HOST_COMMUNICATIONS,DEVICE):-
      device(DEVICE,_,HOST_COMMUNICATIONS,_,_).

   db(host_communications,of,device,HOST_COMMUNICATIONS,DEVICE):-
      device(DEVICE,_,HOST_COMMUNICATIONS,_,_).

   db(device,with,rasterizer,DEVICE,RASTERIZER):-
      device(DEVICE,_,_,RASTERIZER,_).
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
  db(rasterizer,with,device,RASTERIZER,DEVICE):-
     device(DEVICE,_,_,RASTERIZER,_).

  db(cost_of_maintenance,for,device,COST_OF_MAINTENANCE,DEVICE):-
     maintenance(DEVICE,COST_OF_MAINTENANCE).

/* Relationships about graphics features*/

   db(vectors,for,graphic_device,VECTORS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,VECTORS,_,_,_,_,_,_,_,_,_,_).

   db(arcs,for,graphic_device,ARCS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,ARCS,_,_,_,_,_,_,_,_,_).

   db(panels,for,graphic_device,PANELS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,PANELS,_,_,_,_,_,_,_,_).

   db(markers,for,graphic_device,MARKERS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,MARKERS,_,_,_,_,_,_,_).

   db(math_chars,for,graphic_device,MATH_CHARS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,MATH_CHARS,_,_,_,_,_,_).

   db(pixel_op,for,graphic_device,PIXEL_OP,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,PIXEL_OP,_,_,_,_,_).

   db(segments,for,graphic_device,SEGMENTS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,_,SEGMENTS,_,_,_,_).

   db(pan_zoom,for,graphic_device,PAN_ZOOM,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,_,_,PAN_ZOOM,_,_,_).

   db(surfaces,for,graphic_device,SURFACES,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,_,_,_,SURFACES,_,_).

   db(views,for,graphic_device,VIEWS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,_,_,_,_,VIEWS,_).

   db(viewports,for,graphic_device,VIEWPORTS,GRAPHIC_DEVICE):-
      graphic(GRAPHIC_DEVICE,_,_,_,_,_,_,_,_,_,_,VIEWPORTS).

 INCLUDE "GRAPHICS.INC" /* Include parser + scanner + eval + Menusystem */
```

```
/*
  SUPPORT PREDICATES - These are the clauses which support the
  general system, including the parser and the menu system. Most of
  the clauses involve list processing and are general enough to be
  used in any system.
*/

PREDICATES
  maxlen(LIST,INTEGER,INTEGER)     /* Find the length of the longest string */
  listlen(LIST,INTEGER)            /* Find the length of a list */
  writelist(INTEGER,INTEGER,LIST)  /* Used by the menu predicate */
  write_list(INTEGER,LIST)         /* Write the list separated by spaces */
  write_list2(LIST)                /* Display an answer */
  append(LIST,LIST,LIST)           /* Append two lists */
  unik(LIST,LIST)                  /* Eliminate duplicates in a list */
  repeat                           /* Make backtrack points */
  index(LIST,INTEGER,SYMBOL)       /* Select an element from a list */

CLAUSES
  index([X|_],1,X):- !.
  index([_|L],N,X):- N>1,N1=N-1,index(L,N1,X).

  unik([],[]).
  unik([H|T],L):-member(H,T),!,unik(T,L).
  unik([H|T],[H|L]):-unik(T,L).

  append([],L,L).
  append([Ah|At],B,[Ah|C]):-append(At,B,C).


   maxlen([H|T],MAX,MAX1):-
    str_len(H,LEN),
    LEN>MAX,!,
    maxlen(T,LEN,MAX1).
  maxlen([_|T],MAX,MAX1):-maxlen(T,MAX,MAX1).
  maxlen([],LEN,LEN).

  listlen([],0).
  listlen([_|T],N):-
    listlen(T,X),
    N=X+1.

  writelist(_,_,[]).
  writelist(LI,ANTKOL,[H|T]):-field_str(LI,0,ANTKOL,H),
      LI1=LI+1,writelist(LI1,ANTKOL,T).

    repeat. repeat:-repeat.

    write_list(_,[]).
    write_list(_,[X]):-!,write(X).
    write_list(4,[H|T]):-!,write(H),nl,write_list(0,T).
    write_list(3,[H|T]):-str_len(H,LEN),LEN>13,!,write(H),nl,write_list(0,T).
    write_list(N,[H|T]):-str_len(H,LEN),LEN>13,!,N1=N+2,writef("%-27 ",H),
                        write_list(N1,T).
    write_list(N,[H|T]):-N1=N+1,writef("%-13 ",H),write_list(N1,T).

    write_list2([]).
    write_list2([H|T]):-write(H,' '),write_list2(T).


/*
  Evaluating queries - This is the mechanism which reads a query, scans
  the string and removes punctuation, parses the query and evaluates
  it.  The number of solutions are also reported here.
*/

DOMAINS
/* Seven types of questions are recognized */
  QUERY =       q_e(SYMBOL) ;
                q_eaec(SYMBOL,SYMBOL,SYMBOL,SYMBOL) ;
                q_eaq(SYMBOL,SYMBOL,SYMBOL,QUERY) ;
                q_sel(SYMBOL,SYMBOL,SYMBOL,REAL);
                q_min(SYMBOL,QUERY);
                q_max(SYMBOL,QUERY);
                q_not(SYMBOL,QUERY) ;
                q_or(QUERY,QUERY) ;
                q_and(QUERY,QUERY)

PREDICATES
/* Input-output */
  loop(SYMBOL)                  /* Main loop */
  readquery(SYMBOL)
  write_unit(SYMBOL)            /* Write the unit for an entity */
  write_solutions(INTEGER)      /* Write the number of solutions */

/* Scanner */
  scan(SYMBOL,LIST)             /* Convert a string to a list of words */
  filter(LIST,LIST)             /* Eliminate commas and periods */
```

```
/* Parser */
  pars(LIST,SYMBOL,QUERY)

/* Evaluation */
  eval(QUERY,SYMBOL)


CLAUSES
  loop(STR):-   STR >< "",
                scan(STR,LIST),      /* Returns a list of words(symbols)  */
                filter(LIST,LIST1),  /* Removes punctuation and words ignored*/
                pars(LIST1,E,Q),     /* Parses queries                    */
                findall(A,eval(Q,A),L),
                unik(L,L1),
                write_list(0,L1),
                write_unit(E),
                listlen(L1,N),
                write_solutions(N),
                fail.

  loop(STR):-   STR >< "",readquery(L),loop(L).

  readquery(QUERY):-nl,nl,write("Query: "),readln(QUERY).

  scan(STR,[TOK|LIST]):-
                fronttoken(STR,SYMB,STR1),!,
                upper_lower(SYMB,TOK),
                scan(STR1,LIST).
  scan(_,[]).

  filter(["."|T],L):-   !,filter(T,L).
  filter([","|T],L):-   !,filter(T,L).
  filter(["?"|T],L):-   !,filter(T,L).
  filter([H|T],L):-     ignore(H),!,filter(T,L).
  filter([H|T],[H|L]):- filter(T,L).
  filter([],[]).

  write_unit(E):-unit(E,UNIT),!,write(' ',UNIT).
  write_unit(_).

  write_solutions(0):-!,write("\nNo solutions").
  write_solutions(1):-!.
  write_solutions(N):-!,writef("\n% Solutions",N).

/*
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
  ENTITY NAMES
*/

PREDICATES
  entn(SYMBOL,SYMBOL)           /* Convert an entity to singular form */
  entity(SYMBOL)                /* Get all entities */
  ent_synonym(SYMBOL,SYMBOL)    /* Synonyms for entities */
  ent_name(SYMBOL,SYMBOL)       /* Convert between an entity
                                   name and an internal entity name */

CLAUSES
  ent_synonym(E,ENT):-synonym(E,ENT).
  ent_synonym(E,E).

  ent_name(ENT,NAVN):-entn(E,NAVN),ent_synonym(E,ENT),entity(ENT).

  entn(E,N):-concat(E,"s",N).
  entn(E,N):-free(E),bound(N),concat(X,"ies",N),concat(X,"y",E).
  entn(E,E).

  entity(screen).
  entity(device).
  entity(X):-relat(_,ATTL),member(X,ATTL).

/*
  ERROR DETECTION -
  Once the string has been converted to a list of words, the word
  list can be checked against the language database to see if it
  is a known word. Words which are not known are collected into a
  list which the system reports on.
*/

PREDICATES
  error(LIST)
  known_word(SYMBOL)

CLAUSES
  error(LIST):- write(">> "),member(Y,LIST),not(known_word(Y)),!,
                write("Unknown word: ",Y),nl.

  error(_):-    write("Sorry, the sentence can't be recognized").

  known_word(X):-str_real(X,_),!.  /*  Check for special case words   */
  known_word("and"):-!.
  known_word("or"):-!.
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
  known_word("not"):-!.
  known_word("all"):-!.
  known_word(X):-min(X),!.     /* If not a special case word, check the */
  known_word(X):-max(X),!.     /* dynamic database for known words      */
  known_word(X):-big(_,X),!.   /* additional words.                     */
  known_word(X):-ignore(X),!.
  known_word(X):-unit(_,X),!.
  known_word(X):-assoc(_,AL),member(X,AL),!.
  known_word(X):-ent_name(_,X),!.
  known_word(X):-entity(X),!.
  known_word(X):-relop(L,_),member(X,L),!.
  known_word(X):-entity(E),not(unit(E,_)),ent(E,X).

/*
   PARSER SUPPORT -  Compound entities:
   This is used by the parser to handle a compound entity
*/

PREDICATES
  check(LIST)                             /* Check that the list is empty */
  get_ent(LIST,LIST,SYMBOL)               /* Get the compound entity      */
  get_cmpent(LIST,LIST,STRING,STRING)     /* Get the first component      */
  ent_end(LIST)                           /* Get the rest of the entity   */

CLAUSES
  check([]).

  get_ent([E|S],S,E):-ent_end(S),!.
  get_ent(S1,S2,ENT):-get_cmpent(S1,S2," ",E1),frontchar(E1,_,E),ENT=E.

  get_cmpent([E|S],S,IND,ENT):-ent_end(S),concat(IND,E,ENT).
  get_cmpent([E|S1],S2,IND,ENT):-
                concat(IND,E,II),concat(II," ",III),
                get_cmpent(S1,S2,III,ENT).

  ent_end([]).
  ent_end(["and"|_]).
  ent_end(["or"|_]).

/*
  Here begins the parser. The first two parameters for the parsing
  predicates are the inputlist and what remains of the list
  after a part of a query is stripped off. In the last parameter, a
  structure for the query is built up.
```

```
   This method is called "parsing by difference lists." Once you
   understand how it works, you can easily add new sentence
   constructions to the language.
 */


PREDICATES
  s_rel(LIST,LIST,SYMBOL)
  s_unit(LIST,LIST,SYMBOL)
  s_val(LIST,LIST,REAL)

CLAUSES
  s_rel(S1,S2,REL):-relop(RLIST,REL),append(RLIST,S2,S1).

  s_unit([UNIT|S],S,UNIT).
  s_val([X,thousand|S],S,VAL):- !,str_real(X,XX),VAL=1000*XX.
  s_val([X,million|S],S,VAL):-  !,str_real(X,XX),VAL=1000000*XX.
  s_val([X|S],S,VAL):-          str_real(X,VAL).


PREDICATES
  s_attr(LIST,LIST,SYMBOL,QUERY)
  s_minmax(LIST,LIST,SYMBOL,QUERY)
  s_rest(LIST,LIST,SYMBOL,QUERY)
  s_or(LIST,LIST,SYMBOL,QUERY)
  s_or1(LIST,LIST,SYMBOL,QUERY,QUERY)
  s_and(LIST,LIST,SYMBOL,QUERY)
  s_and1(LIST,LIST,SYMBOL,QUERY,QUERY)
  s_elem(LIST,LIST,SYMBOL,QUERY)
  s_assoc(LIST,LIST,SYMBOL,QUERY)
  s_assoc1(LIST,LIST,SYMBOL,SYMBOL,QUERY)
  s_nest(LIST,LIST,SYMBOL,QUERY)
  get_assoc(LIST,LIST,SYMBOL)

CLAUSES
  pars(LIST,E,Q):-s_attr(LIST,OL,E,Q),check(OL),!.
  pars(LIST,_,_):-error(LIST),fail.

/*
   Currently, the parser can only recognize simple English sentences
   in a specific format. The formats below can be extended to add new
   structures. The handling of compound sentence structures would be
   more difficult, but the techniques used here can provide the
   building blocks for such a parser.
*/
```

```
  /* What cost is the device tek4105-- BIG ENTITY CONSTANT */
  s_attr([BIG,ENAME|S1],S2,E1,q_eaec(E1,A,E2,X)):-
                 big(E2,BIG),ent_name(E2,ENAME),
                 entitysize(E2,E1),schema(E1,A,E2),
                 get_ent(S1,S2,X),!.

  /* What cost  is tek4109 -- BIG CONSTANT */
  s_attr([BIG|S1],S2,E1,q_eaec(E1,A,E2,X)):-
                 get_ent(S1,S2,X),
                 big(E2,BIG),entitysize(E2,E1),
                 schema(E1,A,E2),        ent(E2,X),!.

  /* How big is the biggest device -- BIG QUERY */
  s_attr([BIG|S1],S2,E1,q_eaq(E1,A,E2,Q)):-
                 big(_,BIG),s_minmax(S1,S2,E2,Q),
                 big(E2,BIG),entitysize(E2,E1),
                 schema(E1,A,E2),!.

  s_attr(S1,S2,E,Q):-s_minmax(S1,S2,E,Q).

/* The cheapest device-- MIN QUERY */
  s_minmax([MIN|S1],S2,E,q_min(E,Q)):-min(MIN),!,s_rest(S1,S2,E,Q).

/* The costliest device -- MAX QUERY */
  s_minmax([MAX|S1],S2,E,q_max(E,Q)):-max(MAX),!,s_rest(S1,S2,E,Q).

  s_minmax(S1,S2,E,Q):-s_rest(S1,S2,E,Q).


/* give me devices -- ENTITY */
  s_rest([ENAME],[],E,q_e(E)):-!,ent_name(E,ENAME).

  s_rest([ENAME|S1],S2,E,Q):-ent_name(E,ENAME),s_or(S1,S2,E,Q).


/* And has a higher priority than or */
  s_or(S1,S2,E,Q):-s_and(S1,S3,E,Q1),s_or1(S3,S2,E,Q1,Q).
  s_or1(["or",ENT|S1],S2,E,Q1,q_or(Q1,Q2)):-ent_name(E,ENT),!,
                                             s_or(S1,S2,E,Q2).
  s_or1(["or"|S1],S2,E,Q1,q_or(Q1,Q2)):-!,s_or(S1,S2,E,Q2).
  s_or1(S,S,_,Q,Q).

  s_and(S1,S2,E,Q):-s_elem(S1,S3,E,Q1),s_and1(S3,S2,E,Q1,Q).
  s_and1(["and",ENT|S1],S2,E,Q1,q_and(Q1,Q2)):-ent_name(E,ENT),!,
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
                                           s_elem(S1,S2,E,Q2).
  s_and1(["and"|S1],S2,E,Q1,q_and(Q1,Q2)):-!,s_elem(S1,S2,E,Q2).
  s_and1(S,S,_,Q,Q).


/* not QUERY */
  s_elem(["not"|S1],S2,E,q_not(E,Q)):-!,s_assoc(S1,S2,E,Q).
  s_elem(S1,S2,E,Q):-s_assoc(S1,S2,E,Q).


/* ... larger than 15 inches in diameter- REL VAL UNIT */
  s_assoc(S1,S4,E,q_sel(E,REL,ATTR,VAL)):-
               s_rel(S1,S2,REL),s_val(S2,S3,VAL),
               s_unit(S3,S4,UNIT),!,unit(ATTR,UNIT).

/* ... larger than 15 inches  -- REL VAL */
  s_assoc(S1,S3,E,q_sel(E,REL,ATTR,VAL)):-
               s_rel(S1,S2,REL),s_val(S2,S3,VAL),!,
               entitysize(E,ATTR).

  s_assoc(S1,S3,E,Q):-
               get_assoc(S1,S2,A),s_assoc1(S2,S3,E,A,Q).


/* Before s_assoc1 is called ENT ASSOC is met */

/* ... the smallest screensize     -- MIN QUERY */
  s_assoc1([MIN|S1],S2,E1,A,q_eaq(E1,A,E2,q_min(E2,Q))):-min(MIN),!,
               s_nest(S1,S2,E2,Q),schema(E1,A,E2).

/* ... the largest screensize     -- MAX QUERY */
  s_assoc1([MAX|S1],S2,E1,A,q_eaq(E1,A,E2,q_max(E2,Q))):-max(MAX),!,
               s_nest(S1,S2,E2,Q),schema(E1,A,E2).

/* ...
                                                 ENT REL VAL UNIT */
  s_assoc1([ATTR|S1],S4,E,A,q_sel(E,REL,ATTR,VAL)):-
       s_rel(S1,S2,REL),s_val(S2,S3,VAL),s_unit(S3,S4,UNIT1),!,
       ent_name(E2,ATTR),schema(E,A,E2),unit(E2,UNIT),
       UNIT=UNIT1,!.

/* .                                                 ENT REL VAL */
  s_assoc1([ATTR|S1],S3,E,A,q_sel(E,REL,ATTR,VAL)):-
       s_rel(S1,S2,REL),s_val(S2,S3,VAL),!,
       ent_name(E2,ATTR),schema(E,A,E2),unit(E2,_).
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
/* ...                                          -- REL VAL UNIT */
  s_assocl(S1,S4,E,A,q_sel(E,REL,E2,VAL)):-
        s_rel(S1,S2,REL),s_val(S2,S3,VAL),s_unit(S3,S4,UNIT1),!,
        schema(E,A,E2),unit(E2,UNIT),
        UNIT=UNIT1,!.

/* ...                                  -- REL VAL */
  s_assocl(S1,S3,E,A,q_sel(E,REL,E2,VAL)):-
        s_rel(S1,S2,REL),s_val(S2,S3,VAL),!,
        schema(E,A,E2),unit(E2,_).

/* ...                                          -- ENT VAL UNIT */
  s_assocl([ATTR|S1],S3,E,A,q_sel(E,eq,ATTR,VAL)):-
        s_val(S1,S2,VAL),s_unit(S2,S3,UNIT1),!,
        ent_name(E2,ATTR),schema(E,A,E2),unit(E2,UNIT2),UNIT1=UNIT2,!.

/* ...                                  -- ENT VAL */
  s_assocl([ATTR|S1],S2,E,A,q_sel(E,eq,ATTR,VAL)):-
        s_val(S1,S2,VAL),
        ent_name(E2,ATTR),schema(E,A,E2),unit(E2,_),!.

/* .. the device tek4105 -- ENT CONST */
  s_assocl([ENAME|S1],S2,E1,A,q_eaec(E1,A,E2,X)):-
                get_ent(S1,S2,X),ent_name(E2,ENAME),
                not(unit(E2,_)),
                schema(E1,A,E2),
                ent(E2,X),!.

  s_assocl(S1,S2,E1,A,q_eaq(E1,A,E2,Q)):-
                s_nest(S1,S2,E2,Q),schema(E1,A,E2),!.

/* .. tek4105  -- CONST */
  s_assocl(S1,S2,E1,A,q_eaec(E1,A,E2,X)):-
                get_ent(S1,S2,X),schema(E1,A,E2),ent(E2,X),!.

/* Parse a nested query */
  s_nest([ENAME|S1],S2,E,Q):-ent_name(E,ENAME),s_elem(S1,S2,E,Q).
  s_nest([ENAME|S],S,E,q_e(E)):-ent_name(E,ENAME).

/* ...                              -- ASSOC REST */
  get_assoc(IL,OL,A):-append(ASL,OL,IL),assoc(A,ASL).

/*
  EVALUATION OF QUESTIONS
```

```
*/

PREDICATES  /* Support predicates for the parser */
  sel_min(SYMBOL,SYMBOL,REAL,SYMBOL,SYMBOL,LIST)
  sel_max(SYMBOL,SYMBOL,REAL,SYMBOL,SYMBOL,LIST)

CLAUSES
  eval(q_min(ENT,TREE),ANS):-
               findall(X,eval(TREE,X),L),
               entitysize(ENT,ATTR),
               sel_min(ENT,ATTR,99e99,"",ANS,L).

  eval(q_max(ENT,TREE),ANS):-
               findall(X,eval(TREE,X),L),
               entitysize(ENT,ATTR),
               sel_max(ENT,ATTR,-1,"",ANS,L).

  eval(q_sel(E,gt,ATTR,VAL),ANS):-
               schema(ATTR,ASSOC,E),
               db(ATTR,ASSOC,E,SVAL2,ANS),
               str_real(SVAL2,VAL2),
               VAL2>VAL.

  eval(q_sel(E,lt,ATTR,VAL),ANS):-
               schema(ATTR,ASSOC,E),
               db(ATTR,ASSOC,E,SVAL2,ANS),
               str_real(SVAL2,VAL2),
               VAL2<VAL.

  eval(q_sel(E,eq,ATTR,VAL),ANS):-
               schema(ATTR,ASSOC,E),
               db(ATTR,ASSOC,E,SVAL,ANS),
               str_real(SVAL,VAL).

  eval(q_not(E,TREE),ANS):-
               findall(X,eval(TREE,X),L),
               ent(E,ANS),
               not(member(ANS,L)).

  eval(q_eaq(E1,A,E2,TREE),ANS):-
               eval(TREE,VAL),db(E1,A,E2,ANS,VAL).

  eval(q_eaec(E1,A,E2,C),ANS):-db(E1,A,E2,ANS,C).

  eval(q_e(E),ANS):-    ent(E,ANS).
  eval(q_or(TREE,_),ANS):- eval(TREE,ANS).

  eval(q_or(_,TREE),ANS):- eval(TREE,ANS).

  eval(q_and(T1,T2),ANS):- eval(T1,ANS1),eval(T2,ANS),ANS=ANS1.


  sel_min(_,_,_,RES,RES,[]).
  sel_min(ENT,ATTR,MIN,_,RES,[H|T]):-schema(ATTR,ASSOC,ENT),
       db(ATTR,ASSOC,ENT,VAL,H),
       str_real(VAL,HH),MIN>HH,!,
       sel_min(ENT,ATTR,HH,H,RES,T).
  sel_min(ENT,ATTR,MIN,NAME,RES,[_|T]):-sel_min(ENT,ATTR,MIN,NAME,RES,T).


  sel_max(_,_,_,RES,RES,[]).
  sel_max(ENT,ATTR,MAX,_,RES,[H|T]):-
       schema(ATTR,ASSOC,ENT),
       db(ATTR,ASSOC,ENT,VAL,H),
       str_real(VAL,HH),MAX<HH,!,
       sel_max(ENT,ATTR,HH,H,RES,T).
  sel_max(ENT,ATTR,MAX,NAME,RES,[_|T]):-sel_max(ENT,ATTR,MAX,NAME,RES,T).

/*
 READING THE KEYBORD - Not all of the keys defined here are used
 by the program. It has been generalized to show what keyboard
 routines one might implement, and how to go about it.
*/

DOMAINS
 ROW,COL,LEN = INTEGER

 KEY = cr ; esc ; break ; tab ; btab ; del ; bdel ; ins ;
    end ; home ; ftast(INTEGER) ; up ; down ; left ; right ;
    ctrlleft; ctrlright; ctrlend; ctrlhome; pgup; pgdn;
    chr(CHAR) ; otherspec

PREDICATES
 readkey(KEY)
 readkey1(KEY,CHAR,INTEGER)
 readkey2(KEY,INTEGER)

CLAUSES
 readkey(KEY):-readchar(T),char_int(T,VAL),readkey1(KEY,T,VAL).
```

```
  readkey1(KEY,_,0):-!,readchar(T),char_int(T,VAL),readkey2(KEY,VAL).
  readkey1(cr,_,13):-!.
  readkey1(esc,_,27):-!.
  readkey1(chr(T),T,_) .

  readkey2(up,72):-!.
  readkey2(down,80):-!.
  readkey2(ftast(N),VAL):-VAL>58,VAL<70,N=VAL-58,!.
  readkey2(otherspec,_).


/*
  POPUP MENU SYSTEM -
  This implements a popup-menu which is self adjusting based on
  the number of choices and the string length of the longest
  choice. It is called by the main menu routine as well as the
  sub-menus and sizes itself accordingly.

  The following keys can be used:
     arrows up and down: select choice
     cr and F10: activate choice
     Esc: abort

   Menu starts here
*/

PREDICATES
  menu(ROW,COL,STRING,LIST,INTEGER)
  menu1(ROW,LIST,ROW,INTEGER,INTEGER)
  menu2(ROW,LIST,ROW,INTEGER,INTEGER,KEY)

CLAUSES
  menu(LI,KOL,TXT,LIST,CHOICE):-
               shiftwindow(21),
               maxlen(LIST,0,ANTKOL),
               listlen(LIST,LEN),ANTLI=LEN,LEN>0,
               HH1=ANTLI+2,HH2=ANTKOL+2,
               makewindow(3,7,7,TXT,LI,KOL,HH1,HH2),
               HH3=ANTKOL,
               writelist(0,HH3,LIST),cursor(0,0),
               menu1(0,LIST,ANTLI,ANTKOL,CH),
               CHOICE=1+CH,
               removewindow,
               shiftwindow(22),
                shiftwindow(2).

  menu1(LI,LIST,MAXLI,ANTKOL,CHOICE):-
                field_attr(LI,0,ANTKOL,112),
                cursor(LI,0),
                readkey(KEY),
                menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,KEY).

  menu2(_,_,_,_,-1,esc):-!.
  menu2(LI,_,_,_,CH,ftast(10)):-!,CH=LI.
  menu2(LI,_,_,_,CH,cr):-!,CH=LI.
  menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,up):-
                LI>0,!,
                field_attr(LI,0,ANTKOL,7),
                LI1=LI-1,
                menu1(LI1,LIST,MAXLI,ANTKOL,CHOICE).

  menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,down):-
                LI<MAXLI-1,!,
                field_attr(LI,0,ANTKOL,7),
                LI1=LI+1,
                menu1(LI1,LIST,MAXLI,ANTKOL,CHOICE).

  menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,_):-
                menu1(LI,LIST,MAXLI,ANTKOL,CHOICE).

/*
  MAIN MENU - Here begins the user interface which demonstrates
  how to process an action from a list of choices.
*/

PREDICATES
/* Main loop */
  natlang
  mainmenu
  proces(INTEGER)

/* View and update the language */
  viewlang viewlang1(INTEGER)
  updatelang updatelang1(INTEGER)
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
GOAL natlang.

CLAUSES
  natlang:-
    makewindow(21,112,0,"",24,0,1,65),
    write("ESC: Quit this menu -- Use arrow keys to select and hit RETURN."),
    makewindow(22,112,0,"",24,0,1,65),
    write("ESC: Quit  F8: Last line  Ctrl S: Stop output  End: End of line"),
    makewindow(2,7,7,"GRAPHICS: Natural language interface to graphics devices"
                ,0,0,24,65),
    mainmenu.

  mainmenu:-    repeat,
                menu(8,35,"Main menu",
                  [ "Tutorial",
                    "DOS Shell",
                    "Editor",
                    "",
                    "Load the database",
                    "Save database on file",
                    "",
                    "Query the database",
                    "",
                    "View the language",
                    "Update the language",
                    "Graphics words"],CHOICE),
                proces(CHOICE),
                CHOICE=0,!,
                removewindow,removewindow.

  proces(0):-write("\nAre you sure you want to quit? (y/n): "),
           readchar(T),T='y'.
  proces(1):-file_str("graphics.hlp",TXT),display(TXT),clearwindow,!.
  proces(1):-write(">> graphics.hlp not in default directory\n").
  proces(2):-makewindow(3,7,0,"",0,0,25,65),write("Type EXIT to return\n\n"),
           system(""),!,removewindow.
  proces(2):-write(">> command.com not accessible. press any key"),
           readchar(_),removewindow.
  proces(3):-makewindow(3,7,112,"",9,5,15,60),edit("",_),removewindow.
  proces(4).
  proces(5):-write("Loading database file - please wait\n"),
           consult("graphics.dba"),!.
 proces(5):-write(">> graphics.dba not successfully consulted\n").
 proces(6):-deletefile("graphic.dbb"),
           renamefile("graphics.dba","graphics.dbb"),
           save("graphics.dba").
 proces(7).
 proces(8):-readquery(L),loop(L).
 proces(9).
 proces(10):-viewlang.
 proces(11):-updatelang.
 proces(12):-file_str("graph.hlp",TXT),display(TXT),clearwindow,!.


/*
   View and the language
*/

 viewlang:-    repeat,
               menu(5,20,"Language",
                 [ "1  Schema for relations",
                   "2  Schema for the entity network",
                   "3  Names of entities",
                   "4  Synonyms for entities",
                   "5  Alternative names for associations",
                   "6  Words to ignore",
                   "7  Units for attributes",
                   "8  Alternatives for relation operators",
                   "9  Words stating minimums",
                   "10 Words stating maximum"
                 ],CHOICE),
               nl,viewlang1(CHOICE),CHOICE=0,!.


 viewlang1(0).

 viewlang1(1):-
      write("Relations\n*********\n"),
      relat(REL,ATTL),
      write(REL,": "),write_list2(ATTL),nl,fail.

 viewlang1(1):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlang1(2):-
```

```
      writef("%-12 %-8 %-12\n","Entity","Assoc","Entity"),
      write("************ ******** ************\n"),
      schema(E1,A,E2),writef("%-12 %-8 %-12\n",E1,A,E2),fail.

 viewlangl(2):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlangl(3):-
      write("Entities\n********\n"),
      findall(X,entity(X),L),unik(L,L1),write_list(0,L1),nl.

 viewlangl(3):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlangl(4):-
      writef("%-15 %-15\n","Synonym","Entity"),
      write("*************** ***************\n"),
      synonym(E,S),writef("%-15 %-15\n",E,S),fail.

 viewlangl(4):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlangl(5):-
      write("Associations\n************\n"),
      assoc(X,L),
      writef("%-8 ",X),write_list2(L),nl,fail.

 viewlangl(5):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlangl(6):-
      write("Ignore\n******\n"),
      findall(X,ignore(X),L),write_list(0,L),nl.

 viewlangl(6):-
   write("\n\nPress any key to continue"),
   readchar(_).

 viewlangl(7):-
      writef("%-15 %-15\n","entity","unit"),
      write("*************** ***************\n"),
        unit(E,U),writef("%-15 %-15\n",E,U),fail.

  viewlangl(7):-
    write("\n\nPress any key to continue"),
    readchar(_).

  viewlangl(8):-
    write("Names of relational operators\n*****************************\n"),
        relop(LIST,REL),write(REL,": "),write_list2(LIST),nl,fail.

  viewlangl(8):-
    write("\n\nPress any key to continue"),
    readchar(_).

  viewlangl(9):-
        write("Minimum\n*******\n"),
        findall(X,min(X),L),write_list(0,L),nl.

  viewlangl(9):-
    write("\n\nPress any key to continue"),
    readchar(_).

  viewlangl(10):-
        write("Maximum\n*******\n"),
        findall(X,max(X),L),write_list(0,L),nl.

  viewlangl(10):-
    write("\n\nPress any key to continue"),
    readchar(_).

/*
   Update the language
*/

PREDICATES
  newignore
  newsynonym
  newassoc
  getent(SYMBOL)
  getassoc(SYMBOL)

CLAUSES
  updatelang:-  repeat,
                menu(5,20,"Update Language",
                  [ "New Synonyms for entities",
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
                "New Alternatives for associations",
                "New Words to be ignored"
               ],CHOICE),
             nl,updatelangl(CHOICE),CHOICE=0,!.

updatelangl(0).
updatelangl(1):-newsynonym.
updatelangl(2):-newassoc.
updatelangl(3):-newignore.


newsynonym:-  getent(E),write("Synonym: "),
              readln(SYNONYM),SYNONYM><"",
              assert(synonym(SYNONYM,E)),newsynonym.

newignore:-   write("Ignore:"),readln(IGNORE),IGNORE><"",
              assert(ignore(IGNORE)),newignore.

newassoc:-
              getassoc(ASSOC),
              write("New form: "),
              readln(FORM),FORM >< "",
              scan(FORM,LIST),
              assert(assoc(ASSOC,LIST)),
              newassoc.

getassoc(A):-
              findall(X,assoc(X,_),L),
              unik(L,L1),
              menu(11,30,"Assoc",L1,C),
              index(L1,C,A).

getent(E):-
              findall(X,entity(X),L),
              unik(L,L1),
              menu(2,49,"Entity",L1,C),
              index(L1,C,E).
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
relat("device",["device","name","type_device","host_communications",
    "rasterizer","cost"])
relat("emulator",["emulator","device"])
relat("mouse",["mouse","device"])
relat("graphic",["graphic","graphic_device","vectors","arcs","panels",
    "markers","math_chars","pixel_op","segments","pan_zoom","surfaces",
    "views","viewports"])
relat("maintenance",["maintenance","device","cost_of_maintenance"])
relat("colorcopy",["colorcopy","device"])
relat("resolution",["resolution","horizontal_res","vertical_res"])
relat("screen",["screen","name","screen_size","type","windowing",
    "max_color_pallete","dialog_area_colors","fill_patterns","rgb_video",
    "graphics_colors"])
relat("software",["software","device"])
schema("arcs","for","graphic_device")
schema("vectors","for","graphic_device")
schema("panels","for","graphic_device")
schema("markers","for","graphic_device")
schema("math_chars","for","graphic_device")
schema("pixel_op","for","graphic_device")
schema("segments","for","graphic_device")
schema("pan_zoom","for","graphic_device")
schema("surfaces","for","graphic_device")
schema("views","for","graphic_device")
schema("viewports","for","graphic_device")
schema("cost","of","device")
schema("device","emulates","emulator")
schema("device","with","host_communications")
schema("device","with","rasterizer")
schema("dialog_area_colors","of","device")
schema("fill_patterns","of","device")
schema("graphics_colors","of","device")
schema("colorcopy","for","device")
schema("horizontal_res","of","device")
schema("host_communications","of","device")
schema("max_color_pallete","of","device")
schema("rasterizer","with","device")
schema("rgb_video","for","device")
schema("screen_size","of","device")
schema("software","drives","device")
schema("type","of","screen")
schema("type_device","of","device")
schema("vertical_res","of","device")
schema("windowing","for","device")
schema("mouse","for","device")
schema("cost_of_maintenance","for","device")
entitysize("device","cost")
entitysize("screen","screen_size")
```

```
relop(["above"],"gt")
relop(["bigger","than"],"gt")
relop(["greater","than"],"gt")
relop(["less","than"],"lt")
relop(["longer","than"],"gt")
relop(["larger","than"],"gt")
relop(["more","than"],"gt")
relop(["over"],"gt")
relop(["shorter","than"],"lt")
relop(["smaller","than"],"lt")
relop(["under"],"lt")
assoc("drives",["drives"])
assoc("emulates",["emulates"])
assoc("for",["for"])
assoc("in",["in"])
assoc("in",["of"])
assoc("of",["in"])
assoc("of",["of"])
assoc("with",["has"])
assoc("with",["have"])
assoc("with",["with"])
synonym("mice","mouse")
synonym("maintenance","cost_of_maintenance")
synonym("type of emulators","emulators")
synonym("vertical_res","resolution")
synonym("horizontal_res","resolution")
synonym("for","of")
ignore("a")
ignore("an")
ignore("any")
ignore("are")
ignore("as")
ignore("do")
ignore("does")
ignore("give")
ignore("how")
ignore("is")
ignore("many")
ignore("me")
ignore("please")
ignore("some")
ignore("tell")
ignore("that")
ignore("the")
ignore("there")
ignore("to")
ignore("what")
ignore("which")
ignore("list")
```

```
min("least")
min("lowest")
min("minimun")
min("shortest")
min("smaller")
min("smallest")
min("cheapest")
max("biggest")
max("greatest")
max("highest")
max("largest")
max("longest")
max("maximum")
max("costliest")
big("cost","big")
big("screen_size","big")
big("screen","big")
big("device","big")
unit("cost","dollars per unit")
unit("cost_of_maintenance","dollars per month")
unit("height","inches")
unit("width","inches")
unit("screen_size","inches")
unit("horizontal_res","pixels")
unit("vertical_res","pixels")
unit("resolution","pixels")
software("disspla",["tek4105","tek4106"])
software("tellagraf",["tek4105","tek4106"])
software("template",["tek4105","tek4106"])
colorcopy("tek4691",["tek4105","tek4106"])
colorcopy("tek4692",["tek4105","tek4106"])
colorcopy("tek4695",["tek4105","tek4106"])
device("tek4105","crt","rs-232","no","2500")
device("tek4106","crt","rs-232","yes","3500")
device("tek4107","crt","rs-232","yes","4500")
device("tek4109","crt","rs-232","yes","5500")
device("cx4106","crt","rs-232_or_coax","yes","5500")
device("cx4107","crt","rs-232_or_coax","yes","5500")
device("cx4109","crt","rs-232_or_coax","yes","5500")
device("accel_pathfinder","crt","rs-232","no","4490")
device("adage_3000","crt","rs-232","no","20900")
device("adage_6080","crt","rs-232","no","18000")
device("lundy_gtc_327","crt","rs-232","no","4100")
device("ibm_3179","crt","coax","yes","2995")
device("ibm_3279","crt","coax","yes","2995")
device("ibm_3270pc","crt","coax","yes","2995")
device("ibm_3270pc/g","crt","coax","yes","2995")
device("ibm_3270pc/gx","crt","coax","yes","2995")
device("ibm_3270pc/gx","crt","coax","yes","2995")
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
emulator("dec_vt100",["tek4105","tek4106","accel_pathfinder"])
emulator("dec_vt100",["tek4107","tek4109","cx4106","cx4107","cx4109"])
emulator("dec_vt52",["tek4105","tek4106"])
emulator("tek4115",["accel_pathfinder"])
emulator("tek4027",["lundy_gtc_327"])
emulator("ibm_5080",["adage_6080"])
emulator("ibm_3279",["cx_4106","cx_4107","cx_4109"])
mouse("mouse1",["tek4105","accel_pathfinder"])
mouse("mouse2",["tek4107","adage_3000","adage_6080"])
maintenance("tek4105","120")
maintenance("tek4106","195")
maintenance("tek4107","225")
maintenance("tek4109","300")
resolution("accel_pathfinder","1024","800")
resolution("tek4105","480","360")
resolution("tek4106","640","480")
resolution("tek4107","640","480")
resolution("tek4109","640","480")
resolution("cx4106","640","480")
resolution("cx4107","640","480")
resolution("cx4109","640","480")
resolution("adage_3000","1024","1024")
resolution("adage_6080","1024","1024")
resolution("lundy_gtc_327","640","480")
resolution("ibm_3179","720","384")
screen("tek4105","13","vector","yes","64","8","157","yes","8")
screen("tek4106","13","vector","yes","64","8","157","yes","16")
screen("tek4107","13","vector","yes","64","8","157","yes","16")
screen("tek4109","19","vector","yes","4096","8","157","yes","16")
screen("cx4106","13","vector","yes","64","8","157","yes","16")
screen("cx4107","13","vector","yes","64","8","157","yes","16")
screen("cx4109","19","vector","yes","64","8","157","yes","16")
screen("accel_pathfinder","19","vector","yes","256","4096","256","yes","?")
screen("adage_3000","19","vector","yes","256","16.7million","256","yes","?")
screen("adage_6080","19","vector","yes","256","4096","256","yes","?")
graphic("tek4105","yes","no","yes","yes","yes","no","no","n/a",
     "n/a","n/a","yes")
graphic("tek4106","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
graphic("tek4107","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
graphic("tek4109","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
graphic("cx_4106","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
graphic("cx_4107","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
graphic("cx_4109","yes","yes","yes","yes","yes","yes","yes","n/a",
     "4","64","yes")
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

Graphics Words provides basic definitions for frequently used words that are used when discussing graphics hardware.

Dialog Area

Alphanumeric memory. Some devices have separate memories for alphanumeric and graphics data

Dynamic colors

The number of colors that can be displayed at one time is fewer than the total number of colors available. This is the list of colors that can be displayed in the color table.

Emulation

When a device assumes the identity of another device.

Locator

A hardware device that controls the movement of a cursor. Locators include

joystick
mouse
thumbwheels
track ball

Lightpens

Pen used to point to locations on a display screen with a beam of light. It generates data used by the software to identify location.

Palette

A device with more than one color table. Colors are chosen for the palette for aesthetic reasons.

Pixel

An individual picture element.

Raster

A display images generated on a display surface composed of a matrix of pixels arranged in rows and columns.

Resolution

The number of addressable picture elements

Polygon fill

Capability of automatically filling in polygonal area defined in a picture on a display.

Tablet

Flat surface on which the user draws with a stylus

Vector

A straight line of magnitude and direction which is defined by two endpoints.

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

```
                 G R A P H I C S
                 ***************

Graphics demonstrates a natural language interface to a
database on graphic hardware. It demonstates how to query
its database in a limited English.

The database contains the following information:

Information about graphics hardware
  vertical resolution of a device
  horizonatal resolution of a device
  screen size of a device
  type of screen
  maximum color pallete of the device
  windowing for the device
  dialog area colors
  fill patterns
  rgb video
  graphics colors
  emulators
  hardcopy
  software
  type of device
  cost of device
  host communications
  rasterizer
```

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

You can retrieve any or all of this information by asking questions in normal English. Some sample queries:

- devices
- what is cost of tek4105?
- what is the costliest device?
- what is the colorcopy for device tek4109?
- which devices cost greater than 1500 dollars?
- what is the emulator for dec_vt100 ?

You can modify Graphics as you work with it. If it doesn't understand a word in one of your questions, you can add the word to the language definition. The language is defined in a number of relations, the most important one being the schema. A schema is a description of the logical structure of a database. In Graphics, the schema is the "entity network" for the language. A schema entry follows the form: ENTITY ASSOCIATION ENTITY; this signifies that the two entities are bound together by the given association.

To display the relations and their schema, select View Language from the Graphics main menu, then choose one of the items listed below.

1. Schema of relations

   The graphics data is stored in a number of relations. The schema for these relations is listed here.

2. Schema of questions

   The schema for all possible questions that can be asked is listed here. For example, one possibility is:

   > cost of tek4105 <

   That is, what is the cost of a Tektronix 4105?

GRAPHICS ASSISTANT PROGRAM LISTINGS cont.

3. Names of entities

All known entities are listed here.

4. Synonyms for entities

Synonyms for entities are allowed. The previously defined synonyms are listed here; you can also add synonyms to the database dynamically.

5. Synonyms for associations

Synonyms for associations are allowed and can consist of more than one word.

6. Words that are ignored

Some words are simply ignored by the system since they are not correctly relevant to the meaning of questions. Ignored words are listed here.

7. Units for entities

The units of measure for different entities are listed here. For instance, the unit for cost is "dollars".

8. Synonyms for relational operators

There are several ways to say that a device costs "more than" 15 hundred dollars. These synonyms are listed here.

9. Alternative ways to designate the minimum

The different ways to designate adjectives for "minimum" are listed here.

10. Alternative ways to designate the maximum

The different ways to designate adjectives for "maximum" are listed here.

BIOGRAPHY

The author was born in Hazelton, Pennsylvania on March 8,1948 to Thomas and Dorothy McLaughlin. She attended Newark State College and graduated with a Bachelor of Arts degree in Mathematics in January, 1970. She was employed by Bell Telephone Laboratories until 1972. She is a member of Kappa Delta Pi, an Honor Society in Education, and the National Computer Graphics Association. She has coauthored a paper entitled "EZFLASH: An Interactive System for Phase Equilibrium and Thermophysical Properties" which was presented at the Spring 1985 American Institute of Chemical Engineers Conference and the 1985 Speakeasy Users' Group Conference. Currently, she is a Senior Technical Support Analyst with Air Products and Chemicals, Inc. in Trexlertown , Pennsylvania. Responsibilities include the evaluation and support of graphics hardware and software as well as scientific and engineering software support.