THE INTERACTIVE PLANNING WORK STATION: A GRAPHICS-BASED *UNIX*™ TOOL FOR APPLICATION USERS AND DEVELOPERS

Richard Bournique, AT&T Bell Laboratories, Holmdel, New Jersey 07733
Ronald Candrea, AT&T Bell Laboratories, Holmdel, New Jersey 07733
Don Hartman, MLC Inc., 150 E.Riverside Drive, #400, Austin, Texas 78704

ABSTRACT

The Interactive Planning Work Station (IPWS) is a *UNIX*-based system intended to support planning and other complex decision-making tasks. IPWS addresses the needs of at least two classes of users: end users of an application (including both basic and sophisticated) and application developers. IPWS software provides for a menu-oriented user interface, interactive database management capabilities, and graphical input and output. The work station capabilities, how they support end users, and how they simplify the development of new applications, is the topic of this paper.

RÉSUMÉ

L'Interactive Planning Work Station (IPWS - Poste de travail à planification interactif) est un système basé sur UNIX, conçu pour soutenir la planification et autres taches complexes de prise de décision. IPWS répond aux besoins des deux classes d'utilisateurs au moins: les utilisateurs finals d'une application (fondamentale et compliquée inclus) et les dévelopeurs d'application. Le logiciel IPWS fait provision pour une interface d'utilisateur liée au menu, des capacités pour la gestion des bases de donnécs interactive, et pour l'entrée et la sortie graphiques. Le sujet de ca travail comprend les capacitiés du poste de travail, la manière dans laquelle elles soutiennent les utilisateurs finals, et comment elles simplifient le développement des nouvelles applications.

KEYWORDS: graphics workstations, user interface, Core, GKS, network planning

™ Trademark of AT&T Bell Laboratories.

I. INTRODUCTION

The Interactive Planning Work Station (IPWS) is a *UNIX*™-based system intended to support planning and other complex decision-making tasks. Network planning at AT&T has always been supported by a wide variety of edp systems. A typical planning system includes a collection of algorithms that produce large amounts of data to produce a network plan. The primary user interface was, for many years, a set of voluminous printouts. If the system contained an interactive component it was usually in the form of a single package for editing data files or submitting batch runs.

As part of AT&T's Bell Laboratories responsibility to design planning methods and tools to support facility planning within AT&T Communications (formerly Long Lines), a study was undertaken in 1978 to review the software architecture of the existing generation of AT&T facility planning systems and propose an architecture for next generation systems. One of the principal conclusions of the study was that next generation systems require a much better user interface than the current ones, and the only reasonable way to provide the required interface is to build a separate system that can be adapted to meet the needs of a variety of applications. It was out of this study that work on IPWS grew. IPWS design began in early 1979; at that time the intent was to demonstrate the feasibility of the concept and provide a testbed for further development.

As IPWS has evolved, more users and applications have been identified and the need for a complete set of underlying tools, reaching beyond those needed simply for network planning, has become even greater. A next generation low-cost work station has since evolved in both hardware and software with potential applications extending into other areas including marketing, operations research studies, and statistical analysis.

™ Trademark of AT&T Bell Laboratories.

II. WHAT IS IPWS? WHO USES IT?

IPWS can best be viewed as a collection of software capabilities along with a control structure that ties these capabilities together. The IPWS philosophy is distinguished from other work station philosophies in its emphasis upon software. That is, IPWS should not be viewed as a specific hardware device but, rather, as a software architecture that can be implemented on a variety of hardware devices. A typical work station "terminal" consists of an alphanumeric terminal for displaying menus and entering text, a color raster graphics monitor for displaying pictures, and a pointing device such as a tablet or mouse for direct interaction with the graphics screen. One configuration is an AT&T 6300 PC with a high resolution color board and monitor and a Microsoft mouse (Fig. 1).

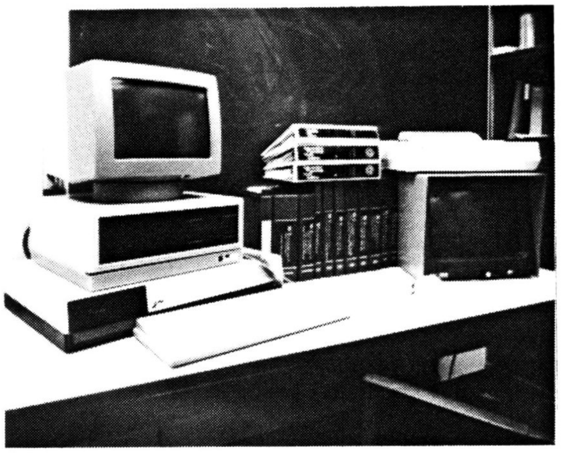


Fig. 1-The Interactive Planning Work Station hardware.

IPWS addresses the needs of at least two classes of users: *end users of an application* (both basic and sophisticated) and *application developers* [1]. Both classes of users have different but overlapping requirements.

2.1 End users of an application

The basic users of an application are typically non-programmers who have little interest in learning about software or an underlying database structure. Normally they work in a highly structured and tightly controlled environment. To assist these users, IPWS provides the ability to display information graphically and add, delete, or modify the display by pointing at various items on the screen. Typically, these users want to interact through a set of menus which define alternatives available at each step of the application process.

More sophisticated users are those who use the system on a regular basis to manipulate data to, for example, perform special studies and ask one-time "what-if" questions. The tasks to be performed change frequently and the exact functions required are often not known prior to performing the task. These users typically have some programming experience and are frequently motivated to understand an underlying database. The tools needed by sophisticated users are those that generate graphics displays to aid in understanding the data (and to help locate data errors) as well as tools that help users edit, synthesize, and manipulate the data.

2.2 Application developers

Application developers design algorithms and software packages for other users. They require the same capabilities as sophisticated users plus tools to assist in building application packages. Software provided by IPWS here includes a langauge for building application menus, a database management system, an interactive graphics package, and some general purpose software development packages.

2.3 IPWS features

Both end users and application developers see IPWS as a system with three major features:

(i) A menu-oriented user interface
(ii) Interactive database management capabilities
(iii) Graphical input and output.

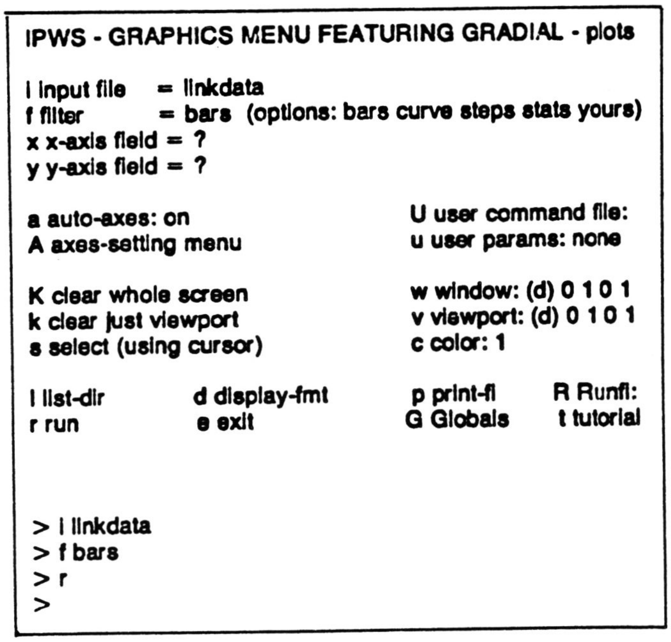
These three features are expanded upon in the sections below.

III. THE MENU-ORIENTED USER INTERFACE

The question of whether it is better to provide users with a command-oriented or menu-oriented user interface to their applications is a difficult one. That issue is resolved in the IPWS environment by essentially providing both.

3.1 IPWS menus

Fig. 2 is an example of a work station menu. The menu is displayed at the top of the alphanumeric screen in an applicatior programmer-defined format. (See Section 3.2 below.) The remaining bottom portion of the screen is used as a scrolling region.



```
IPWS - GRAPHICS MENU FEATURING GRADIAL - plots

I input file    = linkdata
f filter        = bars  (options: bars curve steps stats yours)
x x-axis field = ?
y y-axis field = ?

a auto-axes: on                          U user command file:
A axes-setting menu                      u user params: none

K clear whole screen                     w window: (d) 0 1 0 1
k clear just viewport                    v viewport: (d) 0 1 0 1
s select (using cursor)                  c color: 1

l list-dir        d display-fmt          p print-fl     R Runfl:
r run             e exit                 G Globals      t tutorial



> I linkdata
> f bars
> r
>
```

Fig. 2-Plots is one of the basic menus on the Interactive Planning Work Station. Users type the desired option in the scrolling region below the menu. Online help information and tutorials are available for all the basic work station menus.

Whenever the prompt character, ">", is displayed, the user may select either an option from the current menu (a one-character entry) or any built-in or *UNIX* command (a multiple-character entry). To *UNIX* system users, this appears in practice to be a menu-oriented shell. Menu options may spawn application programs, update entries in the current menu, or invoke other menu pages.

IPWS provides a tree structure of basic work station menus [2]. These menus include basic capabilities many users want, including database manipulation and graphical display. (See Sections IV and V.) The basic menus may be run stand alone or may be linked with application-specific menus.

3.2 The menu language

To facilitate application developers in the creation of menu interfaces for their applications, a menu language [3] was developed for IPWS. In many ways the Menu Language looks like a subset of C along with some additional functions. Fig. 3 is an example of a menu language program.

```
page sortmenu()
{

        static char inf, outf;

        display(0, "Sorting Menu")
                {

                /* "I" menu option at row 1, col 1 */
                :I, 1, 1; "Input file = ", inf;
                        {
                        inf = getfld();
                        update;
                        }

                /* "O" menu option at row 2, col 1 */
                :O, 2, 1; "Output file = ", outf;
                        {
                        outf = getfld();
                        update;
                        }

                /* "R" menu option at row 3, col 1 */
                :R, 3, 1; "Run sort";
                        {
                        if ((inf != "") && (outf != ""))
                                exec(BIN, "sort", inf, outf);
                        else
                                print("Missing file name(s)");
                        }
                }
}
```

Fig. 3-A Menu Language program.

The most noticeable addition to the Menu Language is the ***display*** statement which lets users create their own menus. To define a menu entry, the programmer provides in the ***display*** statement:

(i) The pick character, i.e., the character typed to choose this menu option

(ii) The location of the entry using a row-column format (the top of the screen is row zero; the left-hand side of the screen is column zero)

(iii) The explanatory text to be displayed beside the menu entry

(iv) The action(s) to be performed when the menu option is chosen.

In addition to *display*, other built-in functions, like *getfld* and *update* in Fig. 3, are provided as part of the language for inputting data and updating the menu information on the screen.

IV. INTERACTIVE DATABASE MANAGEMENT

The interactive database management capabilities on IPWS is a relational database management system (DBMS) [4]. Much of the design of the DBMS borrows from other database systems, notably IBM's System R.

IPWS database software has been designed in two levels (Fig. 4). On the bottom lie the low-level database access routines and the C language interface. This level includes record and page managers and an indexed access (B-tree) manager. On top are numerous higher-level tools. They include database menus, some basic database management utilities, a database editor, and a graphical database language. An interactive query interpreter is also in the works.

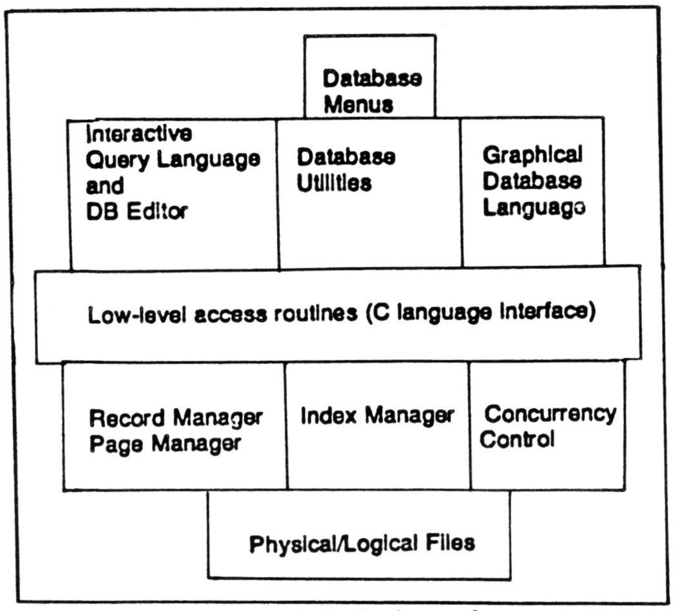

Database Menus
Interactive Query Language and DB Editor
Database Utilities
Graphical Database Language
Low-level access routines (C language interface)
Record Manager Page Manager
Index Manager
Concurrency Control
Physical/Logical Files

Fig. 4-Levels of IPWS database software.

4.1 Low-level database management

IPWS supports both logical *UNIX* databases and physical *UNIX* databases. Logical *UNIX* databases, in which relations are stored as plain *UNIX* files, are intended primarily for personal use where concurrent access and crash recovery are not required. Physical *UNIX* databases, in which relations are stored in a *UNIX* physical file system, are intended for applications that require large shared databases.

Users may process databases sequentially or through indexed access provided through the use of B-trees. Indices may be on a single field or on the concatenation of several fields. Users can also create databases made up of a collection of interdependent relations.

The C language interface consists of a set of low-level C subroutines that are used for processing the database on a record-at-a-time basis. Record-at-a-time access is useful for operations that are inherently procedural. In addition to accessing the database, subroutines have also been provided that support transaction control.

4.2 Higher level database tools

4.2.1 Database utilities

Many database functions are so frequently needed that a set of database utilities has been provided for quick and easy access. These utilities include the ability to:

(i) Convert an ASCII file to a database file
(ii) Select and/or join database files
(iii) Print a database file
(iv) Concatenate, sort, and/or summarize database files
(v) Index a database file.

Most of these utilities are accessible either through an IPWS command or through one of the basic work station menus.

4.2.2 Interactive query language and editor

A database query language is in the works. The fourth generation interactive query language will allow users to query a database and browse through the results using the database editor.

The database editor,qde (query, display, and edit) is a screen-oriented interactive program that allows a user to browse through a relation in a database. The editor appears much like the *vi* text editor in that a user can easily move around in a relation and add, delete, or modify records. Format files can optionally be provided that define the screen layout for records being displayed. The format file can aiso make certain fields invisible or protected.

4.2.3 Graphical database language

GRADIAL (Graphics and Database Interpreter and Language) is a high-level procedural language that marries database access and graphical display. It has been mentioned here for completeness, since it is both a database as well as graphics language. More is said about GRADIAL in Section 5.3.2 after a more complete discussion of graphical input and output is presented.

V. GRAPHICAL INPUT AND OUTPUT

Like the database management software, there are distinct levels of graphical functionality [6] to consider (Fig. 5).

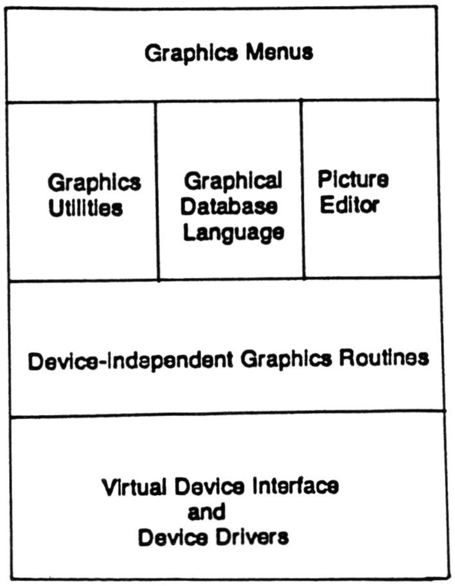

Graphics Menus
Graphics Utilities
Graphical Database Language
Picture Editor
Device-Independent Graphics Routines
Virtual Device Interface and Device Drivers

Fig. 5-Layers of IPWS graphics software.

At the lowest level are the hardware device drivers and a virtual device interface (VDI) that controls and invokes those low-level functions. The next level of software is a set of device-independent routines that permits programmers to develop their own graphical applications in C. Higher-level software, built on top of the device-independent routines, resides at the topmost level. A basic set of graphics utilities are available as well as a graphical database language and an interactive picture editor.

5.1 Device drivers and the VDI

From the beginning, the intent was to allow users to configure their IPWS work stations with the hardware that their particular application dictated. Consequently, many different graphics drivers running under IPWS were anticipated. In the authors' work environment alone, output drivers have been written for the AT&T Teletype 5620 bit-mapped terminal, the Ramtek 9351, the Ramtek 6211, the Printacolor ink jet color printer, the Hewlett-Packard 7221S plotter, the Bausch and Lomb DMP HIPLOT-29 and DMP-9 plotters, the Epson FX-80 black and white printer, and the Matrox, NEC, and Heurikon graphics boards; input drivers exist for the Summagraphics bitpad, the GTCO and CalComp Wedge tablets, and the Microsoft and Logitech mice.

The highly repetitive and generally well understood task of writing a device driver led to the definition and implementation of a standard set of routines and data structures that must be provided to drive the device. The result is a device-independent/device-dependent (DI/DD) interface [7] or VDI. The concept of a DI/DD interface is nothing new; but without the prospect of an industry standard soon, it was thought necessary to go ahead and define a standard IPWS interface.

The DI/DD interface allows the inclusion of not only *hardware* devices but so called *pseudo* devices as well. In particular, a device driver was written that stores the graphics calls in an application-independent metafile, thus allowing for the saving and retrieving of images in a picture library.

5.2 The device-independent graphics routines

The device-independent graphics routines comprise the mid-level graphics software of IPWS. In actuality there are two packages. One package is a C implementation of Core, the ACM SIGGRAPH

Graphics Standards Planning Committee's industry standard for graphics functions [8]. The second package is a C implementation of the newer, internationally popular Graphical Kernel System (GKS) standard [9]. Newer applications tend to use the more complete GKS package.

Both packages consist of hundreds of user-callable functions [10] and have, presently, the ability to:

(i) Draw two-dimensional line, marker, text, or polygon primitives

(ii) Perform clipping, viewing, and image transformations

(iii) Group the primitives into higher-level nested segments

(iv) Set and inquire about various graphical attributes

(v) Read input from different virtual input devices.

5.3 Higher-level graphics software

5.3.1 Graphics utilities

Many graphics functions are rather basic and are regularly used in an interactive session, regardless of the particular application. Some of these basic functions include:

(i) Automatic scaling and drawing of axes for charts and graphs

(ii) Definition of colors to be used in drawing the picture elements

(iii) Definition of the special markers (icons) to be used as picture symbols

(iv) Interactive windowing and viewporting

(v) Production of panels (iconic menus) for the graphics screen.

A set of about a dozen graphics utilities have been provided on IPWS for easy access to these frequently used operations. From the user's viewpoint, these utilities appear to be *UNIX*-like commands that can be invoked directly or through basic menu options.

5.3.2 Graphical database language

The IPWS graphical database language, GRADIAL (Graphics and Database Interpreter and Language) [11] is a high-level language for describing a picture to be drawn, using the information in the records of a database file. The GRADIAL interpreter translates these higher-level constructs into the appropriate lower-level graphics function calls.

Fig. 6 is a simple GRADIAL program that displays LATAs (Local Access and Transport Areas).

```
strtpick                    " save LATAs for picking
ue                          " continue Until End of file
  read                      " read the next database record
  if lata lt 800            " AT&T LATAs are < 800
       color lcolor         " set LATA color
     marker x y 14          " place marker #14 on point
  endif                     " end of if
endue                       " end of loop
```

Fig. 6-A GRADIAL program.

The database file from which the picture was constructed includes the fields *lata* (LATA number), *lcolor* (LATA color), and (x, y) (location of LATA). With the appropriate database file and GRADIAL program, the interpreter will generate the graphics that, equivalently, would require a C program about fifty lines long. Fig. 7 is a black and white rendition of the color display generated by this program.

In addition to producing the picture, the interpreter can also store the indices to the database records from which the items in the picture were generated. A later application process can let a user point to an item on the screen causing the database record on that item to be displayed on the alphanumeric screen. The *strtpick* statement in Fig. 6 tells the interpreter to store the database records for later picking.

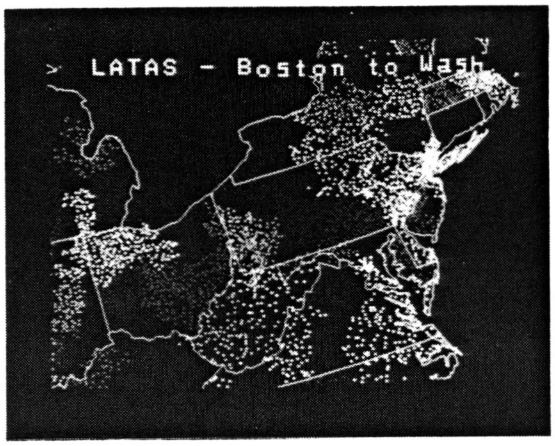


Fig. 7-The display produced by the program in Fig. 6.

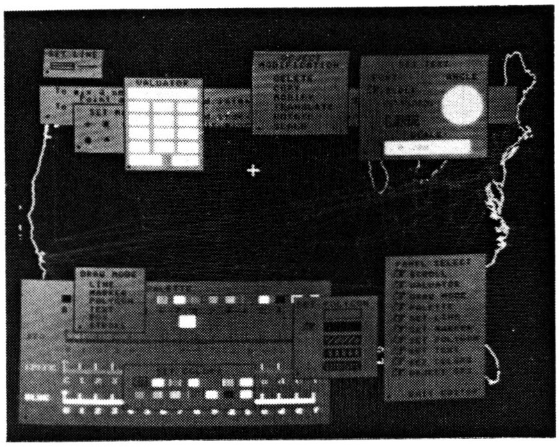


Fig. 8-Picture editor panels are superimposed on top of the picture to be edited. Users can move the panels and make them visible or invisible as needed.

6.3.3 Interactive picture editor

In the same way that GRADIAL allows post-inquiry of the picture elements, it was also thought important to provide users with the ability to post-edit a picture, independent of the particular application that produced it. An interactive picture editor [12] has been provided within the work station for that purpose.

Most of the interaction takes place on the graphics screen, using a tablet and cursor (or mouse). The windows of control information, superimposed on top of the picture, can be shuffled around like sheets of paper on a desk (Fig. 8).

The picture editor allows a user to perform a wide variety of tasks interactively that normally would require some kind of programming effort. Users can, for example, mix new colors, create new marker symbols, stylize networks, and design flowcharts, all interactively.

VI. SUMMARY

The Interactive Planning Work Station is a *UNIX*-based software architecture addressing the needs many different users. Both end users and application developers see IPWS as a system with a menu-oriented user interface, interactive database management capabilities, and graphical input and output. The numerous applications developed and running on IPWS have demonstrated its potential for becoming a valuable decision-maker's assistant in the workplace.

Future IPWS plans include the evolution of the next generation of hardware, movement toward a single-screen multiple-window environment, and the development of other higher-level software tools such as a graphical database editor. Work on IPWS should continue to be challenging and exciting for some time to come.

REFERENCES

1. D. Hartman, S.J. Russo, and S. Udovic, "The Interactive Planning Work Station- an Introduction," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.
2. H. Witting, "A Design Tutorial Using Basic IPWS Menus," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.

3. D. Hartman, "Menu Language for the Interactive Planning Work Station," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.

4. D. Hartman and S. Kashdan, "The IPWS Database Management System," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.

5. S. Kashdan, "The Database Editor, qde (Version 1.1)," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.

6. R. Bournique, "Graphics Software Tools on the Interactive Planning Work Station," *Proceedings of the Application Development Systems Symposium* (April 1983), pp. 89-95.

7. R. Bournique and N. Mowatt, "IPWS General Driver Interface," *Interactive Planning Work Station User's Manual- Volume 3*, Version.

8. "Status Report of the Graphics Standards Planning Committee," Computer Graphics, *13*, No. 2 (August 1979).

9. "Graphics Kernel System (GKS) - Functional Description," *ISO Draft Proposal ISO/TC 97/SC 5 N 728* (December 1982).

10. R. Bournique, "The Definitive Guide to the IPWS Graphics Package," *Interactive Planning Work Station User's Manual- Volume 2*, Version 2).

11. K. Kretsch, "GRADIAL Language Specification", *Interactive Planning Work Station User's Manual- Volume 2*, Version 2.

12. R. Bournique, "Picture Editing Made Easier: A Guide to the IPWS Picture Editor," *Interactive Planning Work Station User's Manual-Volume 2*, Version 2 .