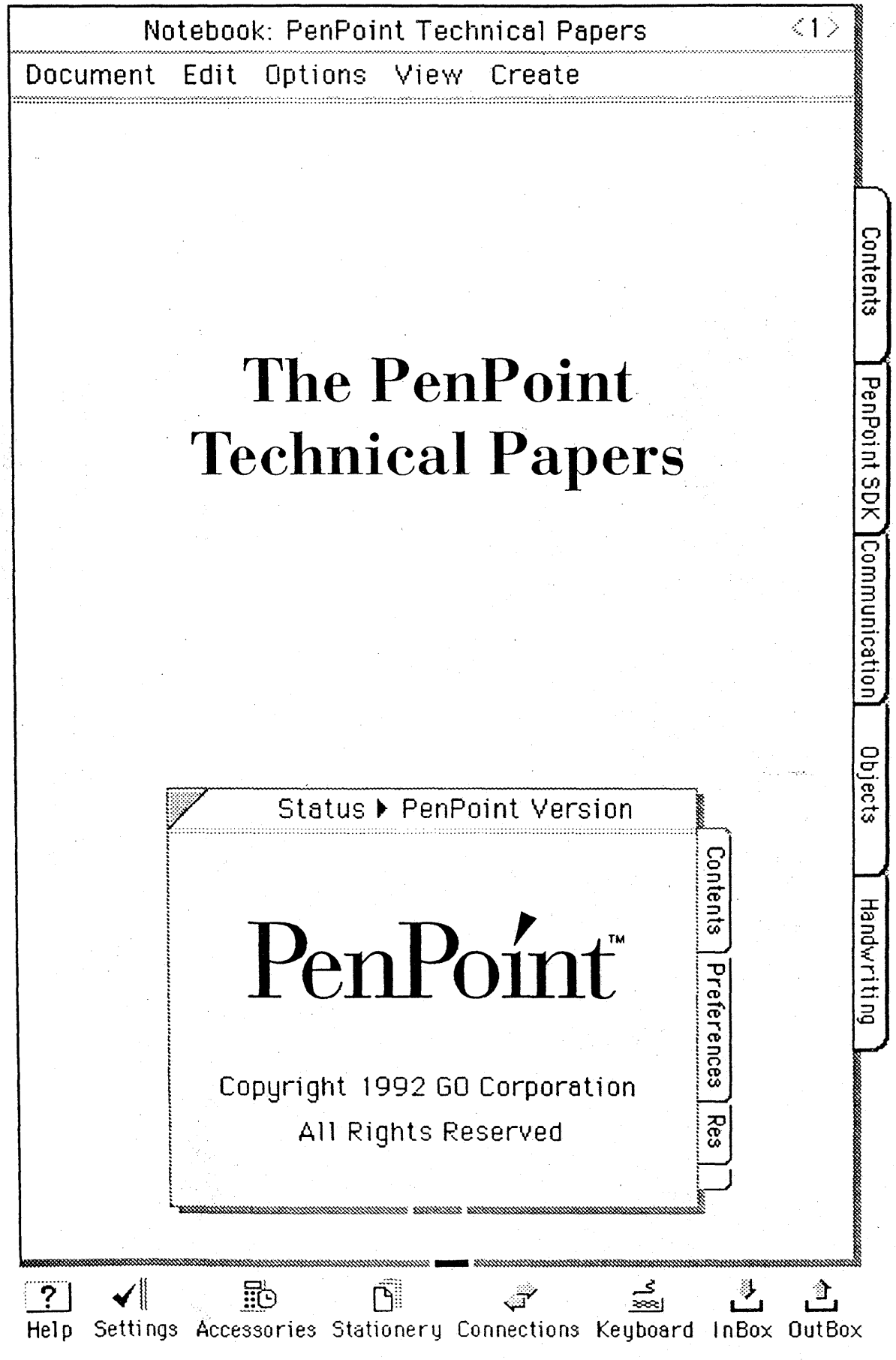

Notebook: PenPoint Technical Papers
<1>
Document Edit Options View Create
Contents
PenPoint SDK
Communication
Objects
Handwriting
The PenPoint Technical Papers
Status ▸ PenPoint Version
Contents
Preferences
Res
PenPoint™
Copyright 1992 GO Corporation
All Rights Reserved
Help
Settings
Accessories
Stationery
Connections
Keyboard
InBox
OutBox

The Vision of Mobile Computing . . .

From mainframe to mini to workstation to PC, computers have gone from remote, mysterious objects of interest only to highly trained specialists behind glass walls, to personal productivity tools available to any reasonably handy person willing to invest some time and effort. The computer industry grows in waves, rather than through consistent, smooth expansion. Each wave exploits advances in component technologies to deliver new solutions in new formats to new users, making computers more accessible.

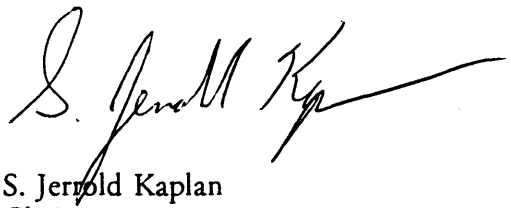
We are at the beginning of the next great wave of computing. Advances in hardware, software and communication make possible a new class of machine: the mobile, pen computer. These machines are operated with a pen rather than with a keyboard and a mouse.

Mobile pen computers combine the convenience of a notebook with the power of a computer. This new format - more akin to a paper notebook or clipboard than to today's PCs - will once again expand the computer industry by serving new users with new uses.

The pen itself promises simplicity to the new user. It is a single, familiar tool to enter data and issue commands. Good tools fit their function so well that they are transparent to their user. The user focuses only on the result; the tool becomes a natural extension of the individual. The tennis racket, the violin, and the sewing needle merge with a skilled practitioner to the point of invisibility. Some tools are more extensions of the mind than of the body; the pen is a tool for manipulating information and ideas rather than physical objects.

Future computers must be better tools. They must become extensions of ourselves, processing information and communicating when and where the need arises. To meet this challenge, computer design must break with the past and derive from the pen, rather than the typewriter. Mobile pen computers can meet the needs of business professionals, students, and even consumers, by combining the power of a computer with the convenience of a pen.

The PenPoint Technical Papers contained in this document describe GO's unique approach to serving the needs of mobile computer users. We hope that you find them enlightening.



S. Jerrold Kaplan
Chairman

Background Information
GO Corp.'s PenPoint™ Operating System
For Mobile, Pen Computers

CONTACTS:

Marcia Mason (415) 358-2000
GO Corp.
950 Tower Lane, Suite 1400
Foster City, CA 94404

Mari Mineta Clapp (415) 354-4449
Regis McKenna Inc.
1755 Embarcadero Road
Palo Alto, CA 94303

Contents

The PenPoint operating system, from GO Corporation, is the only general purpose operating system designed for mobile, pen-based computers. PenPoint delivers major new benefits to users and developers of pen-based applications. These innovations include:

- the Pen and Paper Interface
- simple, practical applications
- features to support mobile connectivity
- a flexible design
- a rich, object-oriented design for the 1990s

A New Market With Unique Requirements

When GO Corp. founders established the company in 1987, they had one goal: to create a market for mobile, pen-based computers. In exploring the needs of this market, GO quickly discovered that mobile, pen-based computing represented a new market, distinct from the existing desktop computer market. Many of the users and applications would be different from the desktop environment.

By virtue of their form factor and lack of a keyboard, mobile, pen-based computers can be used in new settings where desktop or portable computers have in the past been inappropriate or impractical. They are particularly suitable for face-to-face activities, such as meetings and customer visits, where keyboards can be disruptive or socially unacceptable. Since pen-based computers do not require typing, they can also be used while walking, standing or moving about, allowing a machine to be held in one hand and operated with a pen held in the other.

Pen-based computers should fade into the background and be as unobtrusive as possible, not just to the user, but also to those with whom the user is working. Many mobile, pen-based users will use their computers to support other tasks that demand their concentration. For example, an insurance salesperson needs to concentrate on a client, not on a computer.

Because of the way in which mobile, pen-based computers will be used, a general purpose operating system for pen-based computing must meet two key requirements: it must provide unparalleled ease of use, and it must provided support for mobile connectivity so users can move from one work site to another without restrictions. In addition, it must provide a rich development environment suitable for developing a wide range of vertical and horizontal applications.

Why a New Operating System?

In investigating operating systems for mobile, pen computers, GO's founders quickly realized that existing systems were not suitable for the kind of mobile, easy-to-use, compact computing systems its founders envisioned. GO faced three choices:

- Build a "pen compatibility layer" onto an existing operating system.
- Merge selected components of several existing operating systems to create a new mobile, pen-based system.
- Create a new operating system from the ground up.

The first approach, layering a pen interface on top of an existing system, was rejected because it was likely to significantly compromise the ease of use promised by the pen. Layering a pen on top of an existing system doesn't, by itself, make a computer easy to use, just as a mouse doesn't guarantee ease of use. In fact, the layered approach could well increase the complexity of the user interface.

With a layered approach, the pen becomes an alternate form of keyboard and mouse. Because pen events are mapped onto a keyboard and mouse, users must select a mode by pressing a button on the pen or

screen. To emulate the special control characters used by desktop systems, users must have a way of entering them by tapping with the pen on a picture of a keyboard. Furthermore, a pen interface, by itself does nothing to simplify the complexities associated with desktop operating systems such as starting applications, opening and saving files, working with directories, and so on.

Because many mobile, pen-based applications will be fundamentally different from desktop applications - for example, forms completion, electronic calendars, freehand sketching and notetaking - the benefit of using an existing operating system for the sake of application compatibility is lost. Even traditional applications such as spreadsheets and word processors will require enough rethinking for the pen that the amount of rewriting will be substantial.

Apart from these reasons, using an existing operating system poses several other hurdles. Existing systems are designed for desktop use. They assume that the user is always plugged into a power source, and that any network connections are in place. Moreover, existing systems assume that unlimited disk space is available to support large applications. Mobile, pen-based computers, on the other hand, must be lightweight, compact, power-efficient and easily moved from one place to another.

For these reasons, GO saw no clear advantage in assembling an operating system from existing components.

It was clear that if GO built an operating system from the ground up, it could be expressly tailored to meet the needs of mobile, pen users. The system could be built around the unique benefits of the pen to achieve a new level of ease of use. In addition, the system could be optimized for mobile use and connectivity to a variety of other systems.

PenPoint: Designed for the Pen and Mobility

PenPoint is designed expressly to meet the needs of the mobile, pen-based computing market. It incorporates five key features that make possible a new class of computers that are easier to use than existing systems:

Pen and Paper Interface - Familiar pen and paper concepts make it easier to use than conventional operating systems. PenPoint incorporates a simple organizing metaphor - the Notebook - combined with a document model, commands (called "gestures") issued with a pen, and powerful handwriting translation.

Simple, Practical Applications - PenPoint's simple pen and paper interface enables innovative new categories of applications and fresh approaches to existing applications. Object-oriented design guarantees that applications inherit the best of the pen and paper interface.

Mobile Connectivity - Consistent send interface, In/Out Box (deferred I/O), Instant Connect/Disconnect, compatibility with industry standards, and the Connections Notebook make possible truly portable computers for mobile users.

Flexibility - While expressly designed for small, lightweight, portable computers, PenPoint is highly hardware-independent and scales to a variety of formats, from pocket-size to wallboard-size computers.

Rich OS for the 1990s - A true, 32-bit, multitasking, flat memory model architecture that supports the Intel 386 architecture and easily portable to other processor families. PenPoint also provides a powerful and compact imaging model, ImagePoint ™.

The Pen and Paper Interface Delivers a New Level of Ease of Use

By combining intuitive organization and navigation with gestures and handwriting translation, PenPoint's Notebook User Interface, delivers unparalleled ease of use.

Simple Navigation and Intuitive Organization

PenPoint uses a notebook metaphor to organize the user's work. Users are insulated from the complexities of applications and files, and interact instead with documents.

Like the notebook it imitates, PenPoint's makes navigation simple and easy to learn. On the first page is a Table of Contents (TOC) that lists all of the documents in the Notebook. Associated with each document is a page number. Documents can be grouped into sections. Tabs can be attached to any document or section. To move from the TOC to a document, the user simply touches the appropriate page number in the TOC, the tab, or the document's icon.

The user can easily reorganize the contents of the TOC by dragging documents to different locations. The Notebook's pages are then automatically renumbered.

Below the TOC is the Bookshelf, where users can access often-used functions such as Help and Preferences.

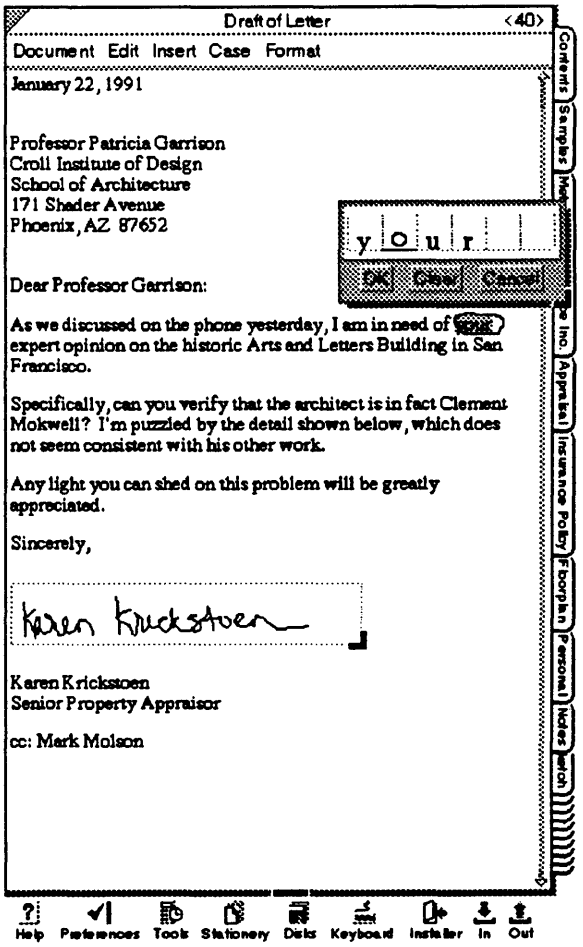
A Table of Contents Instead of File Directories and Windows

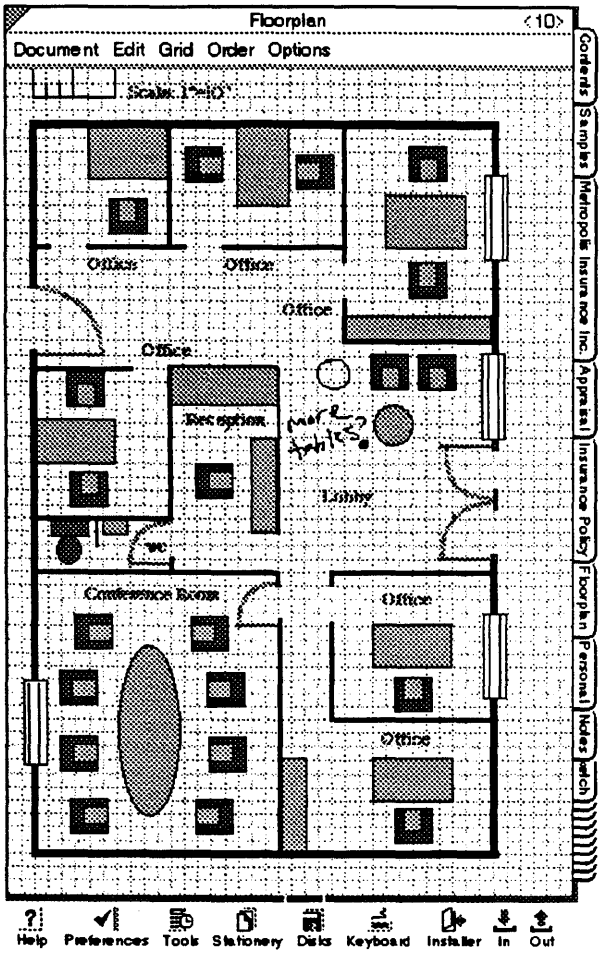
Because the Notebook User Interface and its TOC provide immediate access to every file in the computer, it is more appropriate for mobile, pen-based users than a desktop operating system's windows. With a traditional GUI, users must open every application and file they wish to use concurrently. The number of applications the system can accommodate is limited by the amount of memory available - usually only two or three.

But if the user wishes to open a new document, he or she must first decide if the appropriate application is running, and if not, select the application and load a file. Or, the user must go back to the desktop, find the correct file, attempt to load it, dismiss a different application if necessary to free up sufficient memory, and then retry to open the application. Once the user has the correct file open, it may need to be enlarged to full screen.

This process is acceptable in a desktop environment where the user's concentration is focused on the computer, but it does not meet the needs of mobile, pen-based users who require quick access to all of their data. PenPoint provides this access by allowing users to move easily from page to page.

To provide this capability, PenPoint implements the document model on top of a traditional file system.





Switching from one page to another automatically starts the application on the new page and loads the appropriate data. Then, PenPoint dismisses the previous application after filing its data. All of this activity is hidden from the user.

Gestures

Gestures are commands issued with a pen. They enable a pen to be significantly easier to use than a mouse and keyboard because they combine selection and action into a single, intuitive motion. Examples of gestures include crossing out words or graphics with an X, turning pages or scrolling with a flick, and moving text or graphics by dragging them with a pen. Users may also use the menu equivalents of gestures if they prefer.

Because PenPoint was designed for the pen from the ground up, its 11 basic gestures work consistently throughout the system and applications. The same gestures are used to edit text, control the Table of Contents or operate a drawing application. PenPoint users never fall back into a character-oriented world in which they must type onto a keyboard in order to modify a configuration or restart the environment, as they would in a desktop OS-based system.

In addition, PenPoint users don't have to press a button on the pen or the screen to differentiate between gestures and text. For example, a user can draw a circle on the screen in different applications, or even within the same application, and be assured of logically different results, depending on the context surrounding the circle.

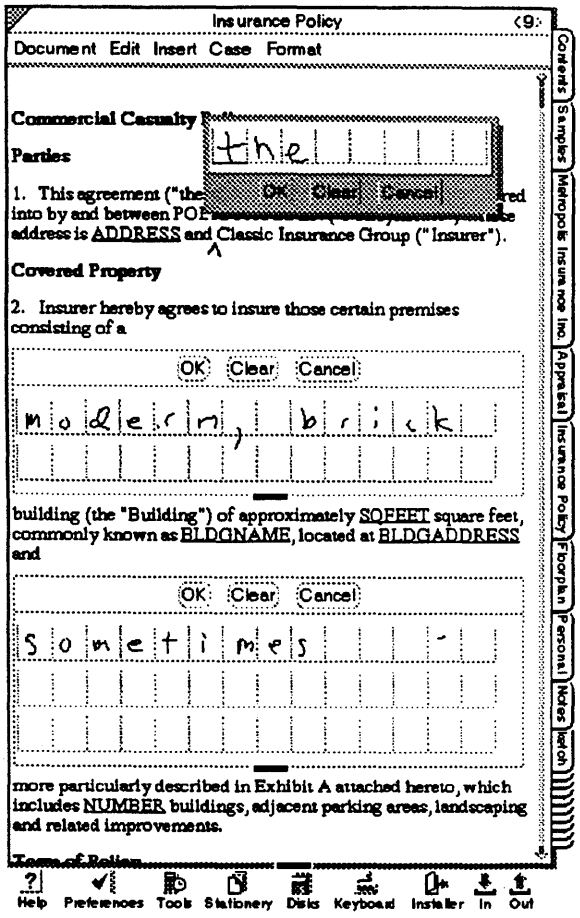
In one case, the user might write a circle that will be translated as the letter "O," while in another instance, it will be interpreted as the gesture meaning "correct this word." In a third case, the user could draw a circle that PenPoint will interpret as a shape, while in a fourth instance, the drawn circle might be part of a signature.

PenPoint understands the difference between these four circles, allowing the freedom to "write" in a manner that comes naturally. This is possible because PenPoint applications supply contextual information that helps determine the meaning of a user's pen strokes, freeing the user from selecting a mode. This capability makes PenPoint easier to use than desktop-based systems and allows users to concentrate on their surroundings instead of their computers.

The Directness of the Pen

Under the PenPoint operating system, a user can touch the pen down wherever or whenever desired, and write onto as many areas of the screen as desired, just as one would with a pad of paper. In contrast, desktop operating systems assume that input comes from one place - the keyboard - and that input is directed to a single location chosen by the mouse or cursor keys.

PenPoint applications can also preserve a user's writing context when inserting text. For example, if a user adds a new sentence in the middle of a paragraph, the paragraph splits open so the user can see the preced-



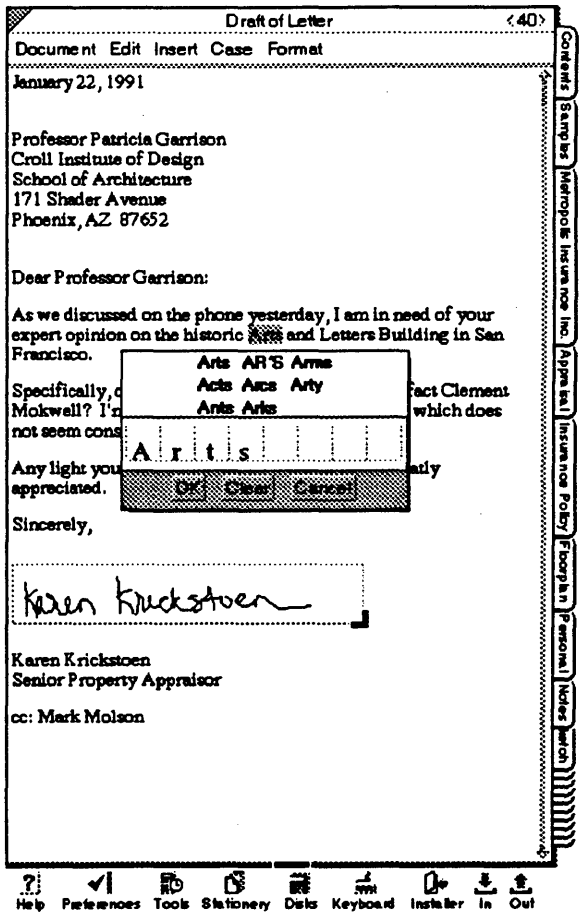
ing and subsequent sentences. Text is not obscured by input pads, nor must text always be entered into a reserved area on the screen.

PenPoint users never need to drag a cursor around the screen, nor do they have to move a selection point or tap in a field before they can write. Cursors, insertion points and field selection are mouse and keyboard complexities that PenPoint users don't have to think about. Instead, they touch the pen to the screen and work.

Handwriting Translation

PenPoint's provides a powerful handwriting translation system that accommodates a wide variety of handwriting styles. It is designed to reduce the amount of time required to input corrected text into a document. To achieve this goal, PenPoint's handwriting translation system provides:

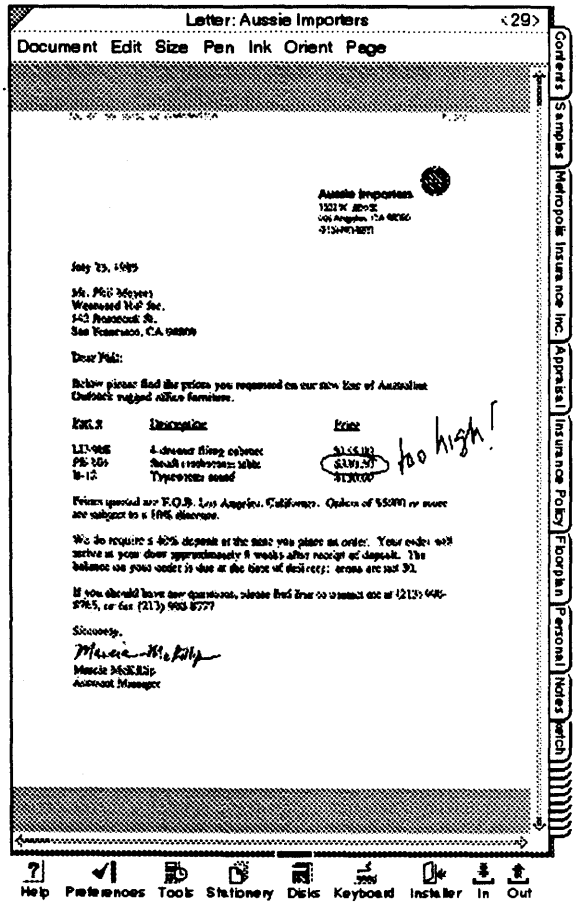
• Standard user interface components such as input pads that are tightly coupled with the translation system. After text is written into an input pad, it is translated and presented to the user in easy-to-correct editing pads. Then, when any necessary corrections are made, the user inserts the text into the document. This close coupling speeds the input of corrected text.



• The ability to accept upper and lower case characters, numerals, punctuation and some overlapping characters. The translation system is stroke order-independent, so users can dot an "i" or cross a "t" after they have finished writing a word.

• High levels of translation accuracy, measured not only by the percentage of characters translated correctly, but also by the percentage of words translated correctly. Word-level accuracy is important because it determines the number of corrections a user must make. High word-level accuracy translates into faster, more efficient text entry.

• A two-way link between applications and the translation system that allows applications to supply context to improve translation results. For example, an application can indicate whether it is expecting alpha or numeric input in a given field, or whether a circle will be interpreted as the letter "o" or an edit gesture.



• The ability to present alternative translations so that users can select another if the first is incorrect.

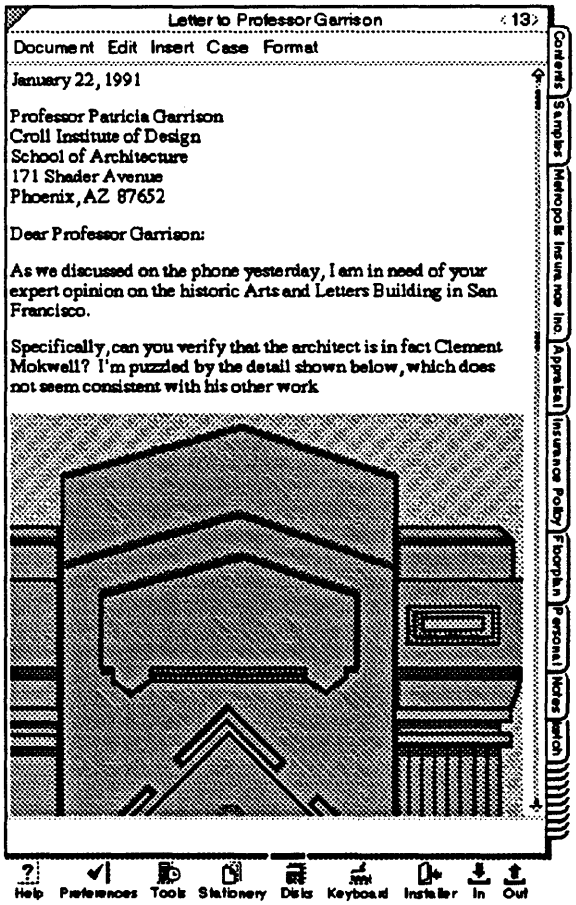
• Case heuristics that capitalize words in a sentence appropriately. These heuristics also allow the 20 to 30 percent of users who prefer to write upper case to do so, while allowing their input to be translated correctly into the appropriate combination of upper and lower case characters.

• User-definable preferences that allow users to write in character boxes of various sizes or on ruled lines of various heights.

• The ability to train the handwriting system to recognize an individual's unique style of writing. Several users can share a single machine, easily switching between different handwriting profiles without re-booting or complex reconfiguration.

• Application Programming Interfaces (APIs) that allow other vendors to incorporate their own handwriting translation systems into PenPoint. This ensures that PenPoint users will always have access to the best available handwriting translation technology.

PenPoint's also has the ability to preserve handwriting without translating it into text. This is particularly applicable where annotation or signature capture is required. For example, consider fax mark-up. A user might receive a fax on his PenPoint computer, make edits and comments in his own handwriting, and fax the marked-up document back to the sender.



PenPoint also allows users to defer the translation of their handwriting into text, allowing them to preserve the visual context of their handwritten notes.

Simple, Practical Applications

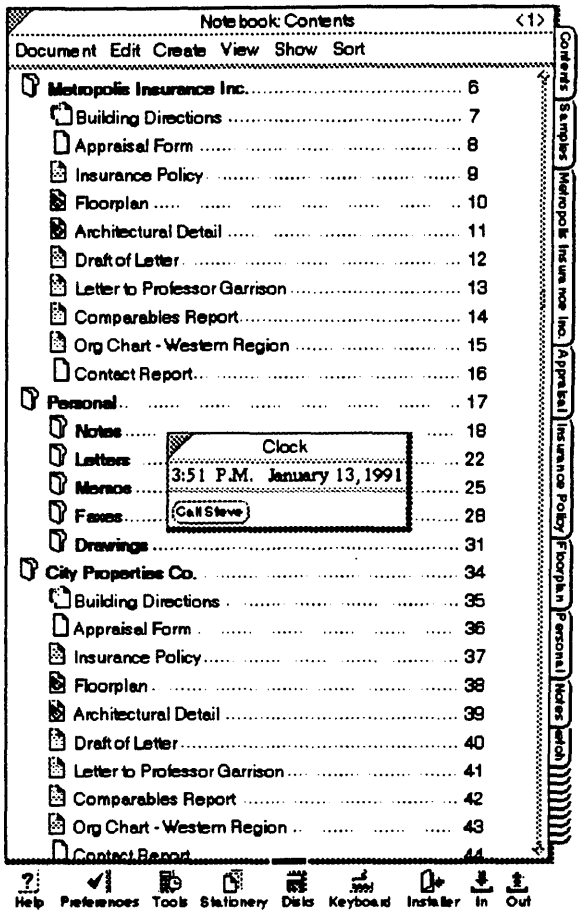
PenPoint incorporates an object-oriented development environment which enables application developers to design simple, practical applications. Design innovations like Embedded Document Architecture - EDA - that allows users to embed live documents, PenPoint's Application Framework, and rich new data types (raw ink and annotation) are all provided to the developer "for free".

Live Document Embedding

EDA allows any PenPoint application to be embedded within any other to create compound documents. For example, the user can use a word processor to create a letter and embed a drawing into it. But, unlike traditional operating systems, both applications are "live" in the same document - the user can edit the letter or the embedded drawing at the same time. The user can even drag text from the letter into the drawing and vice versa.

Reference Buttons

EDA allows users to create reference buttons that can be used to navigate between locations in the notebook. Users can even place reference buttons in accessory applications such as clocks. Developers can use this capability to create fully configured Notebooks with predefined navigation paths. For example, a customer service Notebook might present its user with a series of buttons that lead to forms, customer service information - even a special help system. Yet EDA allows developers to provide this capability without writing any code. These reference buttons work in embedded documents, too. In fact, any document - even a compound document - can be linked to any other.



PenPoint Delivers Mobility

PenPoint computers are truly mobile devices that can be used while walking, standing or kneeling. PenPoint is uniquely capable of meeting the needs of mobile users because it provides instant on, deferred input and output, detachable networking, and the ability to connect with a wide range of systems.

Instant On

Because a PenPoint computer can be consulted "on the fly," it can be used just like a physical notebook or organizer. Documents are available immediately because a PenPoint machine does not need to be re-booted when it is turned on. Booting is typically required only when a new version of the operating system is installed.

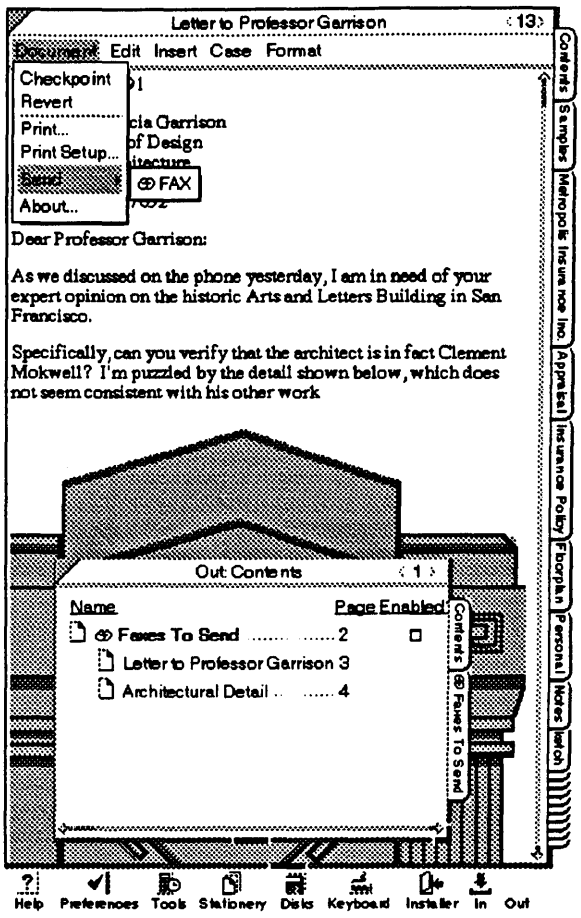
When PenPoint users turn their machine on, they are returned to the page that was being viewed when the computer was turned off. In contrast, desktop operating systems must be re-booted whenever the machine is turned on, and the applications and files must be re-loaded.

PenPoint's code is loaded into memory at boot time and stands ready to operate as soon as the processor awakens. The same is true of applications, which are loaded when they are installed. Starting and stopping PenPoint, then, is simple: The processor simply freezes, and all processes halt in place to begin again when the system "awakens."

If the PenPoint machine has backing store such as a hard disk, power-off causes the contents of memory to be copied to disk before the processor freezes. When power returns, the minimum required data is reloaded. All of this happens so quickly that the user experiences instant access.

In contrast, desktop operating systems typically require code and data to be relocated into memory whenever they are powered on. The time required to load code and data prevents these machines from delivering instant-on capability.

PenPoint's instant-on capability is closely related to its power-saving technology. A PenPoint machine enjoys a long battery life because the processor and many of the hardware subsystems spend most of their time in a dormant state.



Deferred Input and Output

Even when they are away from any physical connection, PenPoint users can issue commands for printing, faxing, filing or any other form of data transfer. PenPoint simply moves a copy of the document to be transferred to the Out box, which is accessible from the Bookshelf. When the user makes the appropriate connection, the document is transferred out of the Out box automatically - without any assistance or intervention from the user.

Similarly, a PenPoint machine can awaken to request data from a host computer or receive a fax, even if it is left unattended. The incoming data is placed in the In box, also accessible from the Bookshelf. When the user returns to the machine, PenPoint informs the user that documents have arrived in the In box, which the user can subsequently check.

This capability, called Deferred I/O, is a critical feature for mobile users because it frees them from having to remain connected to outside services. It is accomplished through a set of connectivity Application Programming Interfaces (APIs). These APIs provide communications protocols, automatic detection of connections (to networks, for example), support for direct access to files formatted for a variety of operating systems, translation of files to and from PenPoint internal formats, deferred I/O and easy access to, and management of, connected devices.

Detachable Connectivity

PenPoint users can connect and disconnect from networks without re-booting or performing special manipulations. The user simply unplugs the network cable, and network operations are suspended. Later, when the cable is reconnected, network operations are resumed and any documents waiting in the Out box can be transferred.

In addition, PenPoint users can switch between different networks by simply plugging in the relevant cables. This is possible because several protocols can be installed in a PenPoint machine simultaneously. When PenPoint senses that a network connection has been attempted, it awakens the appropriate protocol and resumes the connection.

Finally, PenPoint is ideal for use with wireless network connections because it will not lose data when sessions are interrupted by a network interference. PenPoint can simply suspend and resume the connection as a user's PenPoint machine moves in and out of the network range.

Connecting PenPoint to Other Systems

Because mobile, pen-based computers must access data stored in many places, PenPoint is designed to connect with a wide variety of systems.

- PenPoint can use a special version of TOPS networking software to move, copy, and manipulate files on PCs and Macintoshes, and any network volumes available to them.

- PenPoint allows other network protocols to be installed and configured on the fly. GO is currently working with Novell to provide a Netware client for PenPoint.

- PenPoint provides an extensible file system that can be adapted to support any operating system's file format. In its first implementation, PenPoint reads and writes MS-DOS formatted disks.

- PenPoint provides a consistent, easy-to-use import and export system. When a user attempts to copy a foreign data file into PenPoint, a dialog appears which lists the document types (applications) that will accept the file.

  For example, if a user copied a Lotus 1-2-3 file called BUDGET.WK1 into PenPoint, the dialog would list the installed applications that accept 1-2-3 files. The user might select a spreadsheet, in which case a new document would appear in the table of contents called "Budget." When the user selected that page, the Budget spreadsheet would appear.

  Export is equally simple. The user selects the desired file format, and the application performs the translation. The file is placed on the user's floppy disk or in a directory on an attached PC.

PenPoint is Flexible

The mobile, pen-based market demands different sizes of computers that can run the same applications. PenPoint's compact and scalable design supports these diverse configurations, yet its hardware-independent APIs allow the same applications to run unmodified on any PenPoint machine.

PenPoint machines can:

- Vary greatly in format, from shirt-pocket to wallboard-size. In fact, a PenPoint machine could be designed to sit directly on any overhead projector, allowing it to serve as an electronic acetate.

- Use many types of pens, tethered or untethered; proximity sensing or not.

- Provide single-or two-tier memory architectures.

- Include cellular modems, fax modems, conventional modems, etc.

Device Independence

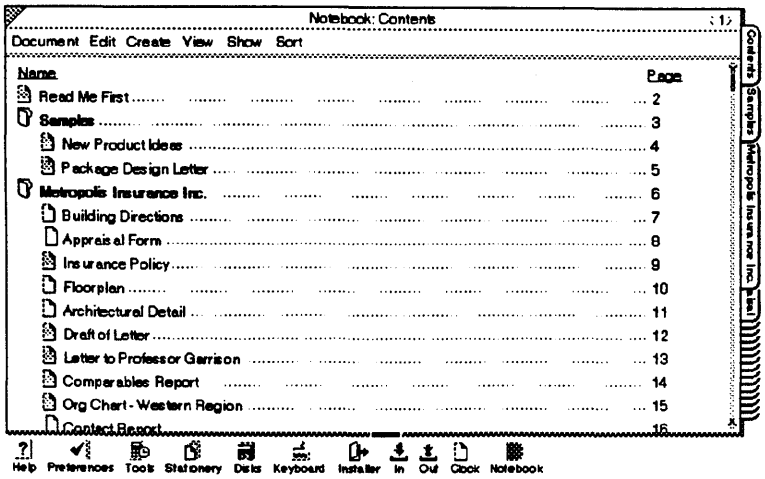
PenPoint provides this unique opportunity for hardware differentiation because its hardware dependencies are isolated in a small portion of PenPoint called the Machine Interface Layer, or MIL. The MIL connects PenPoint to hardware. When PenPoint is ported to new hardware, only the MIL must be rewritten.

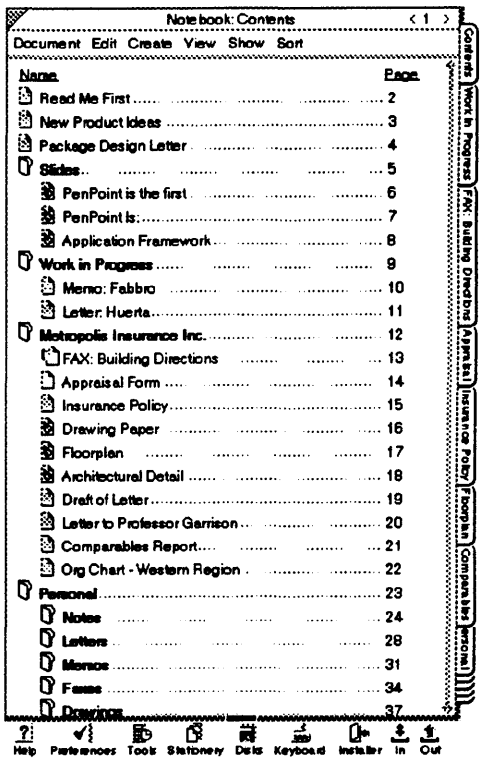
Since PenPoint applications don't communicate directly with hardware, the same applications run unmodified on any PenPoint machine.

Scalable User Interface

PenPoint's user interface components provide a device-independent coordinate system and auto-layout capability that allow applications to present themselves attractively regardless of screen size, resolution or aspect ratio.

PenPoint's auto-layout capability allows developers to set parameters for user interface components in relative terms. For example, in the table of contents, the page numbers and tabs are positioned relative to the right side of the screen, while the document titles are positioned with respect to the left. When the screen orientation is switched from landscape to portrait, the table of contents still looks logical. This same capability allows PenPoint applications to "look right" on any size screen - from shirt-pocket to wallboard-size.





No Keyboard Required

Since PenPoint machines do not require a keyboard, they can be significantly smaller than keyboard-based portables. Of course, PenPoint accepts keyboard input and even provides an on-screen, "virtual" keyboard that allows users to tap out characters if they wish.

Memory Conservation

PenPoint requires less memory than a desktop operating system with a pen because PenPoint was designed with low memory consumption in mind. In contrast, reduced memory consumption was not a key design criteria for desktop operating systems.

Reduced memory requirements meant fewer memory chips and smaller computers. In contrast to desktop GUI applications, which typically require multiple megabytes of memory, PenPoint applications typically require only 100 to 200 kilobytes.

PenPoint conserves memory by using a single copy of code instead of requiring one copy in memory and another on disk, as is required by desktop operating systems. PenPoint also allows code to be shared. Its object-oriented design allows developers to inherit most of their applications' behavior from components included in PenPoint. Because these objects are re-entrant and re-usable, multiple PenPoint applications do not require multiple copies of code.

Different Memory Configurations

PenPoint supports different physical memory architectures, from all-DRAM to a combination of dynamic random access memory (DRAM), static random access memory (SRAM), and rotating or silicon hard disks. Each of these combinations meets a particular market need.

- All DRAM. Suitable for small, inexpensive PenPoint systems where small form factor and/or low hardware costs are important

- SRAM and FLASH. Suitable for small PenPoint systems in which protection ofdata against power loss sustained periods of time is important. FLASH memory does not lose data when the system loses power.

- DRAM and backing store. Suitable for PenPoint users who require rapid access to large amounts of data.

An Operating System for the 1990s

Because PenPoint is built on a strong foundation of advanced technologies, it will serve the mobile, pen computing market through the 1990s and beyond. These technologies include:

- Object-oriented interfaces down to the kernal. Developers can make use of object-oriented interfaces of every function that PenPoint provides.

- Pre-emptive multitasking. PenPoint's multitasking capabilities are similar to OS/2's, enabling smooth user interface interactions, background communications and smooth translation of hand writing while the user is writing.

- True 32-bit architecture. PenPoint's 32-bit addresses and flat memory model take full advantage of the 386 processor and provide PenPoint with the opportunity to be ported to other processor families.

- Modular and extensible. PenPoint contains installable components: file systems, network protocols, peripheral drivers. Replaceable modules, including the imaging model, the handwriting engine.

- A compact and powerful imaging model. PenPoint's ImagePoint imaging model unifies screen display and printing. All text operations are fully integrated with graphics operations. Outlined fonts are used to render text on demand at any size. In addition, ImagePoint provides scaling, translation, rotation and sampled image rendering.

- Automatic installation of applications. Users don't have to deal with complicated installation scripts. Instead, they just insert a disk into a drive, and the PenPoint Installer handles the rest.

- Dynamic binding. Applications can leverage software components provided with other applications by binding with them at installation time.

Summary

The potential of pen-based computing is best realized with an operating system designed from the ground up to meet the special requirements of mobility and the pen. In this way, applications can fully exploit the capabilities of the pen while simplifying what the user needs to know to operate a mobile, pen computer.

PenPoint meets these requirement with the five innovations described in this paper:

- The Pen and Paper Interface, that combines a notebook metaphor with gestural control and handwriting translation;

- Simple, Practical Applications, that are developed through leveraging PenPoint's object-oriented development environment.

- Features for mobile connectivity that detachable networking, simultaneous installation and residence of multiple protocols, deferred I/O, and an easy-to-use import and export facility.

- A flexible design that allows PenPoint machines to take on a wide variety of physical characteristics; and,

- A combination of advancements including object-oriented interfaces, a highly portable, processor-independent architecture and modular and extensible components that make PenPoint an operating system for the 1990s and beyond.

With PenPoint, users have a natural way to apply the power of computing in new situations, developers have a modern operating system that they can use with existing tools and skills, and hardware companies have the opportunity to build exciting, differentiable products.

###

Glossary

Automatic constraint-based layout:	A mechanism for describing the design of graphical user interface elements in relative rather that absolute terms. This allows the user to display characteristics such as resolution, size and aspect ratio.
Bookshelf:	Found at the bottom of the PenPoint screen, the bookshelf contains icons that represent PenPoint accessories.
Deferred I/O:	Documents and files to be transmitted via E-mail or fax or to be printed are queued until the appropriate connections are made.
Detachable networking:	The ability to disconnect and reconnect to networks without the need for special commands or re-booting.
Document model:	A feature of the Notebook that overlays a document model on top of a traditional file system, allowing users to deal with documents rather than with applications and files. All documents are immediately available, and the user can easily switch betweenthem by simply turning pages in the Note book.
Editing Pad:	A pad containing text the user wants to change.
Embedded Document Architecture (EDA):	A design innovation that allows users to embed live documents inside each other and to create hyperlinks between and within documents.
Flick:	A gesture that is used to scroll documents and move between pages in the Notebook.
Handwriting profile:	A customized handwriting prototype set created and saved when a user runs the Handwriting Customization accessory.
ImagePoint:	A compact and powerful imaging model that provides scaling, translation, rotation, bezier curves, sampled image rendering and outline fonts.
In box:	A system service that stores incoming network transmissions such as E-mail, faxes and file transfers.
Input pad:	An area on the screen into which the user enters handwriting. Input pads come in various sizes and types, such as boxed or ruled.
MIL:	The Machine Interface Layer. A portion of PenPoint that isolates hardware-specific code.

Out box:	A system service that stores outgoing transmissions such as E-mail, faxes and printing tasks until the appropriate connection is established.
Notebook:	The structure containing documents, applications and information in PenPoint.
Notebook User Interface:	Provides a simple organizing concept (the Notebook meta phor), a document model, gestural commands and a powerful handwriting translation system.
Pre-emptive multitasking:	An approach to multitasking that allows the operating system to determine when and to whom system cycles are allocated. This is in contrast to "cooperative multitasking," in which the applications themselves determine when they will release the resources they are using.
Reference Button:	A button that is created by the user or programmer to allow automatic navigation to another location in the Notebook or within the same document.
Tab:	A marker for a document or section in the Notebook. The user taps a tab to go to that page.
Tap:	The user puts the pen down on the screen and lifts it away. The tap is used to open menus, choose commands, turn to pages and select some items in the Notebook.
TOPS:	A networking protocol from SITKA Corp. which, when installed on a PenPoint system, allows a user to move, copy and manipulate files on PC and Macintosh computers.

Background Information
GO Corp.'s PenPoint™ Development Environment

CONTACTS: Marcia Mason (415) 358-2000
GO Corp.
950 Tower Lane, Suite 1400
Foster City, CA 94404

Mari Mineta Clapp (415) 354-4449
Regis McKenna Inc.
1755 Embarcadero Road
Palo Alto, CA 94303

Contents

Overview: An Accessible and Modern Software Environment

The PenPoint operating system is fully object-oriented and provides programmers with a rich development environment. This environment includes much of the infrastructure necessary to develop complex applications very quickly. The system provides many functional components (objects) that make it easy to develop applications with a consistent user interface. Furthermore, PenPoint is designed for pen computing, but provides all the functionality expected from an advanced operating system.

PenPoint is written in the C programming language and provides application programming interfaces (APIs) written in C. Therefore, developers familiar with C will quickly become productive in the PenPoint environment. They will also find that substantial portions of their existing C programs can be ported to PenPoint.

GO's primary product for software developers is the PenPoint Software Development Kit (SDK). The SDK contains things necessary to develop applications and includes the following:

- ***PenPoint binaries*** that run on a PC as well as on pen based machines.
- ***Software development tools*** such as debuggers, compilers, and class browsers.
- ***Sample programs*** that illustrate the salient features of the PenPoint environment. Many of the sample programs are explained in detail in the documentation.
- ***Documentation*** that covers all aspects of programming in this environment. This includes a detailed explanation of the PenPoint object-oriented architecture, the user interface guidelines, the programming interface, and the development environment.

What Developers Need to Know

The PenPoint Kernel

The PenPoint kernel provides many advanced operating system features that are still not provided by most commercial operating systems. The PenPoint kernel supports hardware memory protection, preemptive multitasking, multi-threading, and the dynamic link library (DLL) mechanism. The PenPoint memory management system has been designed to provide optimal performance in limited memory configurations. In order to do this, it extensively shares code and data among applications.

PenPoint provides sophisticated object-oriented file system features such as memory-mapped files, extensible attributes, and installable file systems. It also supports a special in-memory file system to let applications run transparently on RAM-only hardware configurations. The PenPoint file system is particularly robust and designed to prevent information loss even if the machine is turned off unexpectedly.

The PenPoint kernel is also modular, with well-defined interfaces between its components. This makes PenPoint particularly portable across hardware architectures because its machine dependent portion is distinct from its machine independent portion.

The Class Manager

PenPoint is an extensible operating system that uses a "Class Manager" to support Object Oriented Programming (OOP).

The Class Manager provides all the object functionality that developers have come to expect from environments such as SmallTalk and Object Pascal. Developers can use the Class Manager to create classes and class hierarchies, to manipulate class instances, to inherit functionality from existing classes, and to send messages between objects.

Since the Class Manager is a subsystem of PenPoint rather than a language extension, its capabilities are available via standard C syntax. In addition, these same capabilities can be made available to other languages, such as Pascal and C++.

Virtually all PenPoint application programming interfaces are based on Class Manager messages and objects. This encourages developers to reuse and adapt system code at many levels. As a result, applications are smaller and provide a consistent user interface because they share standard functionality provided by PenPoint subsystems.

The Class Manager object model is also dynamic: It allows program objects to communicate with other objects that were undefined when the program was compiled and linked.

Application Framework

The entire PenPoint system is one large class library based on the PenPoint Class Manager. One of the most important portions of this class library is the PenPoint Application Framework.

The PenPoint Application Framework is a collection of classes that define and implement a standard application protocol. These classes define how applications behave when they are installed or removed from the system, when they are activated or embedded within another application and when some part of a document is linked to any part of another document. All these capabilities are provided as shared code, which new applications "inherit" automatically.

For example, extra code does not have to be written for users to add a hyperlink button to their documents. This capability is already in the Application Framework. To implement more application-specific behavior, such as turning a selected word into a link, the programmer need only implement a set of standard messages.

Users see only documents in the PenPoint Notebook, not system abstractions such as files and application programs, because the PenPoint Application Framework does a lot of work behind the scenes. For example, PenPoint applications automatically file data for the user, and when the user selects a page in the notebook, PenPoint automatically activates the application code associated with that document.

Embedded Document Architecture

One of the more interesting aspects of the Application Framework is its Embedded Document Architecture (EDA™).

With EDA, documents created by different applications can be embedded "live" within each other as needed by the user. For instance, EDA enables the user to embed spreadsheets and business graphics into a text file or any other PenPoint application without any additional code.

The result is a true compound document capability in which users can mix and match applications seamlessly within a single document. Users do not need to be aware of which application is active — the correct application responds automatically to the portion of the document on which the user is working.

UI Toolkit

PenPoint's user interface is a graphical one that supports the pen while employing a familiar "notebook" organizing concept to help users learn and manage their system and data.

The UI Toolkit is a PenPoint class library that supports graphical user interface features. It operates in a memory-constrained environment, automatically scales to various screen sizes, integrates pen input, and operates modelessly.

In addition to menus, windows, and familiar controls such as buttons and scroll bars, the Toolkit has a hierarchy of specialized classes that know how to respond to pen input according to the application context. For example, one class knows how to respond to the standard gestural interface, while another can capture data input from form-field uses.

The UI Toolkit allows applications to scale automatically to available screen sizes in device-independent fashion. This is because GO designed the PenPoint operating system to run on a wide range of computers from pocket sized to drafting-table sized. GO engineers built a constraint-based automatic layout mechanism into all user interface toolkit objects, ensuring that programs using the Toolkit classes will work well on any size display.

Handwriting Recognition Subsystem

GO's Handwriting Recognition Subsystem is an integral part of the PenPoint operating system. Developers can easily and quickly use the standard services, or they can add specialized recognition capabilities.

With the subsystem, a developer can optimize recognition algorithms to an application's needs and improve its recognition rates, including word-level (as compared with character) recognition rates, which are key in text applications. The developer can also use the system to recognize other symbols and even shapes for drawing.

The Handwriting Recognition Subsystem also interprets gestures, an important part of the user interface. The developer can either use the standard recognition of gestures or adapt gestural interpretation for the specialized needs of an application.

Mobile Connectivity

PenPoint supports networking and communications in a mobile environment. Users can connect and disconnect from networks without rebooting or performing special functions, and they can switch back and forth between different networks simply by plugging in the relevant cables. To terminate a session, the user need only unplug the cable. Later, when the cable is reconnected, the session is re-established exactly where the user left off.

PenPoint provides facilities for deferred network transmission — an essential capability for a mobile environment. The In box and Out box facilities are available throughout the system, allowing applications to batch print, fax or mail transmissions until a network connection has been established. When the system senses that a connection has been made, the Out box automatically sends the data without additional program intervention.

Components and System Services

PenPoint must operate well in memory-constrained hardware. Therefore, PenPoint designers maximized code sharing by using object-oriented programming, and also by sharing functionality through the Application Component System and the System Service Architecture. For example, an installation subsystem — built into PenPoint — is used by applications, the Application Component System and the System Service Architecture.

The **Application Component System** allows developers to package functionality in a dynamic link library (DLL). PenPoint automatically checks for the presence of this DLL when the application is installed, ensuring that extra copies are not installed if more than one application uses it.

GO provides a bundled component to handle standard text editing behavior and licenses an object-oriented drawing component for use and redistribution by third-party developers.

The **System Service Architecture** provides globally shared facilities, such as file transfer/translation mechanisms, additional networking protocols or databases. In this way, programmers have a single, well-defined way to provide systemwide services within their applications.

GO provides a number of system services, including the In box and the Out box and some networking protocols. Third parties will provide additional services.

The Development Environment

Currently, PenPoint applications can be developed in two configurations. Developers can use either a PC-only development environment or a cross-development environment consisting of a PC and a tablet computer.

For the PC-only configuration, the programmer compiles and links code under MS-DOS and then runs a version of PenPoint on the PC, where the program can be tested. Either a mouse or digitizer tablet can be used to emulate the pen. GO provides the PC drivers to support both. GO recommends programmers use a PC configured with at least 8MB of RAM.

In the dual-machine environment, the programmer still uses the PC and MS-DOS to compile and link the PenPoint program, but then transfers the program via a fast communications link (GO initially supports TOPS Flashcard) to the PenPoint computer.

There, the program can be tested and debugged in its native hardware environment. GO provides a remote debugging capability as part of the SDK, allowing the programmer to debug a program running on the PenPoint tablet from the PC. This frees the programmer from having to boot PenPoint after each compilation, and it allows the developer to test the program in its final environment.

Current Tools

The SDK provides several programming tools that aid the development of PenPoint applications. Many of them are those typical of any software development environment, but some are particularly useful in PenPoint's object oriented environment.

- ***Compiler*** - An optimized version of the WATCOM C 8.5 80386 compiler is available to developers from GO Corporation. This compiler contains several necessary optimizations for PenPoint, but retains the efficiency and speed of the standard WATCOM C8.5 compiler.

- ***Debuggers*** - A source-level symbolic debugger that runs under MS-DOS and PenPoint. The symbolic debugger can be used to debug PenPoint running on the PC, or as a remote debugger with PenPoint running on a tablet machine and the debugger running on both the PC and the tablet. It allows the programmer to examine Class Manager objects and messages, set break points at the source and message level, manage multiple tasks, and execute sophisticated debugging scripts. A complete profiling capability lets developers use the debugger for performance tuning.

- *Class Browser* - A static class browser that enables one to traverse the class hierarchy and obtain information on each class. This is particularly useful for searching information about classes in the system.

- *Resource management tool* - PenPoint uses resource files to store frequently modified objects. The SDK includes tools allowing developers to create and compile resource files and associate these resources with applications.

- *Bitmap Editor* - A PenPoint based development tool that allows the developer to modify and create bitmaps for use in PenPoint icons. GO also supplies a program to create and modify outline fonts for use with PenPoint.

- *Document preparation tool* - Several utilities are provided to help developers create effective documentation. These include facilities that allow developers to take snap shots of the screen and to develop documentation that has the look of PenPoint documentation.

Documentation and Samples

GO understands that an invaluable part of any successful software product is the support documentation and sample programs that go along with it. This is even more important for a system like PenPoint which introduces many new concepts. The SDK includes documentation that explains the various parts of the system in detail and uses the sample programs to illustrate concepts with concrete examples. Some important pieces of the documentation set are given below.

- *User Interface Guidelines* - A guide to the PenPoint Notebook User Interface (NUI) and its use in designing application user interfaces.

- *Application Development Guide* - Introductory material about PenPoint programming concepts, the configuration of the PenPoint development environment and how to use the SDK tools.

- *Architectural Reference* - Detailed explanations for using PenPoint subsystems and programming interfaces.

- *Application Programming Interface Reference* - A complete, structured reference to the programming interfaces of PenPoint, showing the correct use and options of all PenPoint objects and APIs.

Future Development Environments

Although the PenPoint Development Environment provides many facilities for efficient program development, there is tremendous scope for further improvement. GO is already in the process of planning its next generation of products in this area, and is eager to work with other companies who are interested in developing such products.

Summary

PenPoint is the state-of-the-art, 32-bit, object-oriented operating system designed specifically for mobile pen computers. It is accessible today to programmers who can work in a familiar development environment using the C language, familiar programming tools, and existing hardware. The PenPoint Software Developers Kit provides PenPoint software, developer tools, object-oriented programming interfaces, sample code, and the documentation programmers need to develop new and compact applications for the emerging pen computing market.

###

Background Information
GO Corp.'s PenPoint™ Operating System
Communications Capabilities

CONTACTS: Marcia Mason (415) 358-2000
GO Corp.
950 Tower Lane, Suite 1400
Foster City, CA 94404

Mari Mineta Clapp (415) 354-4449
Regis McKenna Inc.
1755 Embarcadero Road
Palo Alto, CA 94303

Contents

Overview: A Connectivity Architecture for People on the GO

The PenPoint operating system, from GO Corporation, is the new general purpose operating system designed to meet the unique needs of people on the go. PenPoint meets the needs of this market by providing the key user and developer benefits listed below:

- Pen and paper interface
- Simple, practical applications
- Mobile connectivity
- Flexibility
- Rich OS for the '90s.

GO used the following criteria to define the needs of the target market, and to derive PenPoint's communications capabilities:

- Adapt PenPoint to the way people on the go work today:
  - Users move from site to site, network to network.
  - Constant connection and disconnection to peripherals and networks.
  - All peripherals and networks can be disconnected at any time—*disconnection should not be an error condition.*
- Address ease of use:
  - Manual configuration should not be required to connect to different networks and peripherals.
  - Should be possible to utilize remote resources and networks *even if the system is disconnected.*
- Address wireless network and communications requirements
  - Independence between applications, protocols and network
  - Automatic detection of connection and disconnection.

PenPoint achieves these goals through a number of enabling technologies built into PenPoint and into every PenPoint application. A description of each of these technologies follows.

PenPoint Services: "The PenPoint Connectivity Framework"

Services are PenPoint system extensions that can be installed and removed by the user. Network protocols, handwriting engines, database systems, and local and remote file systems are all implemented as Services in PenPoint. Each PenPoint Service has the following characteristics:

- Each Service is organized beneath an associated Service Manager for its category. There are Service Managers for link and transport protocols, for local and remote file systems, and other system extensions. The Service Manager's job is to find and enumerate the Services it controls, observe changes in the state of the Services, bind to the Service (allowing a client to receive notification messages from the Service) and open the Service.

- Taking advantage of PenPoint's object-oriented technology, the Service inherits its default behavior automatically from PenPoint, specifically from clsServices, eliminating the need for the developer to duplicate functions available to all Services. Services send, observe and act upon *messages* from PenPoint or other Services, allowing Services to communicate with PenPoint, other Services and applications.

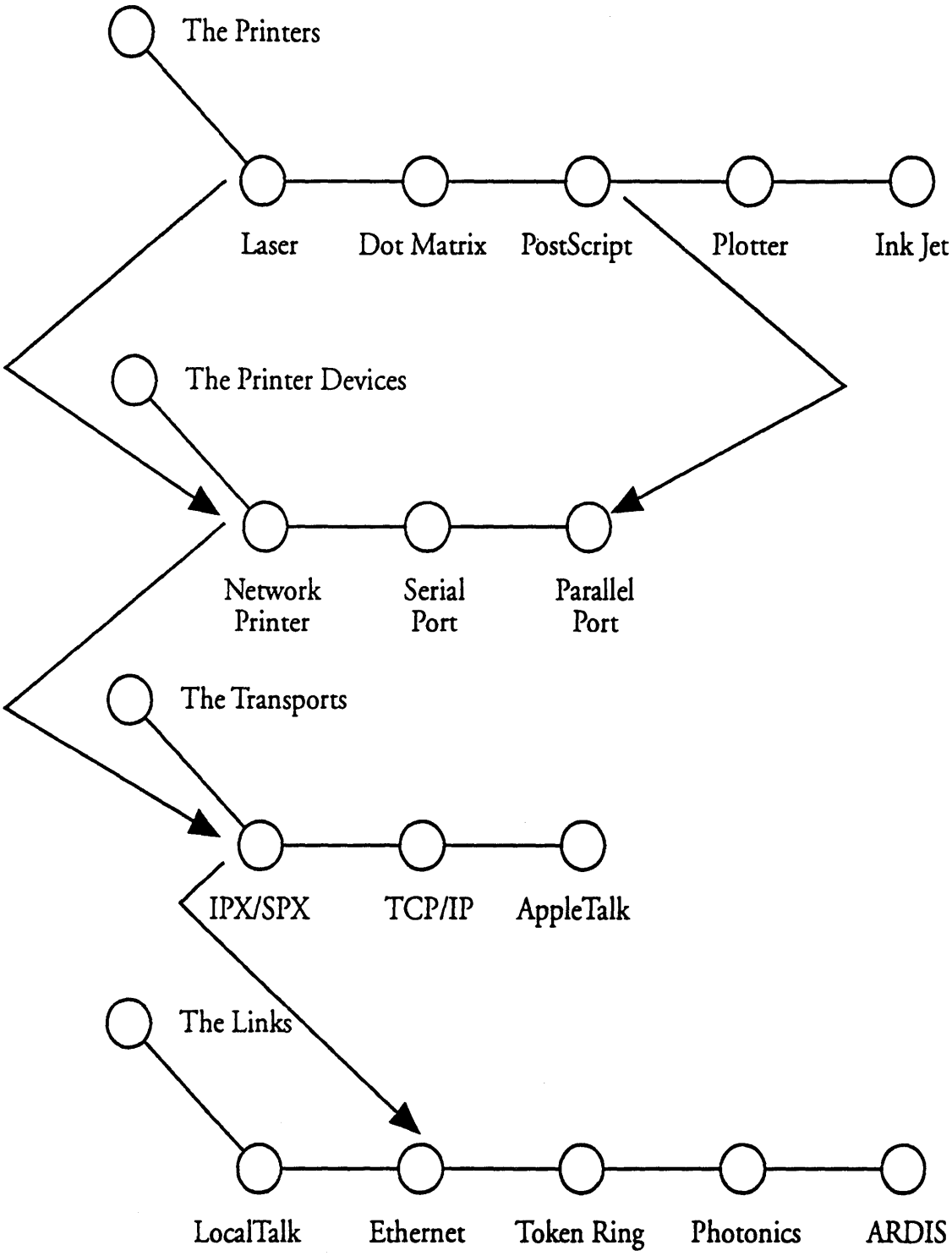
- Services that reference a given hardware device are called MIL device Services. These Services inform higher levels of Services of connections that are made or broken with the associated hardware. Other Services that perform a higher layer function, such as a link protocol service, reference the MIL device Service. In this example, the link protocol Service *targets* the MIL device Service. In turn, other Services may target the link protocol Service. The resulting targeting relationships between Services may extend to any depth.

  Each Service that targets another Service adds a layer of functionality and abstraction on top of the targeted Service. An application at the top of the chain of Services can open the upper layer Service without consideration of the lower layers. This capability would allow a terminal emulator to be written to operate over a serial port, a

modem, or a LAN, ignoring all but the upper layer service for transport protocols.

- Using the PenPoint MIL, PenPoint automatically detects connections of any type. The PenPoint MIL—Machine Interface Layer—is a collection of routines that control the individual hardware devices. MIL device Services provide an interface between these routines and PenPoint. The routine either polls a given hardware port to check for physical connection and disconnection, or is alerted to the change through a hardware interrupt. The MIL device Service is informed of connection state changes by the routine, and changes its state to reflect the new connection status, and sends messages containing the status change to other observers (that have previously bound themselves to the Service) that target the particular MIL device Service.

  This chain of state change – message – Service connection and higher level Service state change continues until the chain of Services is complete (or broken). The diagram on the following page depicts a series of services establishing a connection between a printer on a network and PenPoint which results in printing the documents.


The Printers
Laser
Dot Matrix
PostScript
Plotter
Ink Jet
The Printer Devices
Network Printer
Serial Port
Parallel Port
The Transports
IPX/SPX
TCP/IP
AppleTalk
The Links
LocalTalk
Ethernet
Token Ring
Photonics
ARDIS

- Multiple Services of the same type can co-exist on the machine simultaneously. In other words, Services for the AppleTalk Filing Protocol, TOPS, and NetWare can exist on the same PenPoint computer at the same time. Once the PenPoint computer connection to a given physical network is complete, PenPoint, not the user, completes the selection of the proper Services, e.g., protocols, etc.
- Services can be installed and removed dynamically by the user at will without booting the machine. The user can add and delete Services to conserve memory or add functionality when needed, and remove the Service when it is no longer required. The Services can be installed in any order; i.e., transports before link protocols. PenPoint supports the concept of delayed binding in order to allow arbitrary installation order.

Consistent Interface

PenPoint serves the needs of people on the go, whether they are old hands at using computers, or are new to the computer as a business tool. In either case, PenPoint does not require a long conceptual leap from familiar concepts like pen and paper and notebooks.

PenPoint institutes consistent use of familiar gestures to navigate through networks to establish connections, transfer documents, and import and export data to and from PenPoint applications. Once the user learns how to use the PenPoint notebook, the user knows how to manipulate documents in the PenPoint In box, Out box, and the Connections Notebook, each of which are also notebooks.

PenPoint offers user interface standards and elements such as menu items and gestures that work consistently regardless of the application type. Applications inherit these user interface elements and standard behaviors from the PenPoint Applications Framework. Every PenPoint application contains a *Document* menu, which in turn contains a *Send* selection. The *Send* selection reveals a submenu which allows the user to choose any deferred connectivity Service, known as a "sendable Service", that can send documents. Examples of documents sent by the "sendable" Service are faxes and electronic mail.

Building the Send command into every application allows the user to transmit a document without using another application. The user can continue to focus on the task at hand. The "sendable Services" appear to the user as a part of the application and to PenPoint itself.

Deferred Connectivity

Deferred connectivity is perhaps the most powerful aspect of PenPoint's connectivity architecture. Users can print documents, send fax or electronic mail ***without being connected to the destination device or network.*** This capability is available to any PenPoint application.

The Service places a copy of the document in the PenPoint Out box. The copy appears below the corresponding Section for the particular "sendable Service" — printer, fax, etc. — as shown in Figure 9. The user can view and revise the document while it is in the Out box. The user can reorder the

documents within the particular section as well as delete documents from the Section using gestures identical to those used in the PenPoint Notebook.

The user can control Out box operations through the use of the PenPoint Service "Enabled" option. This option can be set for each PenPoint Service shown in the Out box; e.g., for each printer that the user creates for the machine, for the fax application, and for electronic mail. Processing of the documents for a given PenPoint Service —printing or transmission—will not begin, even though the physical connection exists, if "Enabled" is off.

If a connection breaks while a communications operation that makes use of the PenPoint Out box is in process, PenPoint will display a message asking the user to reconnect. The user can cancel the operation as well. PenPoint will not remove the document(s) that was transmitting at the time the connection broke. They will transmit entirely when the user re-establishes the connection.

Incoming electronic mail (or fax) documents come into the machine via the PenPoint In box. Similar to the Out box, the user manipulates the documents in the notebook using the same set of gestures used in the PenPoint notebook.

Automatic Detection of Connections

PenPoint detects connections to physical devices automatically, and can initiate an operation after the connection is complete. For example, connecting the PenPoint machine to a network causes PenPoint to send a message to all Services that utilize network connections. Documents waiting to be printed to a network printer would begin transmission in the order shown in the Out box. Electronic Mail message processing is identical.

Software developers can create other applications that make use of PenPoint's ability to detect connections. A developer of a shared calendar application would program the application to "observe" messages from the PenPoint file system about remote disk connections. Once the volume is available, the file on the volume that contains the shared calendar database is available. The series of connections, from the physical network connection, to the transport protocol, and finally to the file system, is automatic. The PenPoint Service Managers for each type of PenPoint Service—clsLink,

clsTransport, and clsFile System—interact with each other without user involvement.

Connections Notebook

The PenPoint Connections Notebook provides the single interface point for connection management between PenPoint and peripheral devices and networks. Users can perform the following functions through the Connections Notebook:

- Browse disks—floppy disks, hard drives, and network drives—and perform disk management functions (move, copy, delete and rename files)
- Eject or dismount disks.
- Format disks.
- Browse networks, and enable network resources—disks, printers, mail servers, etc.—that are available for use.
- Create and edit instances of printers.
- Initiate the PenPoint import sequence to create a PenPoint document from a non-PenPoint file on disk.
- Initiate PenPoint document export to create a non-PenPoint format version on disk of a PenPoint document.
- Database and other connectivity categories can be added to the Connections Notebook by GO and third party developers.

In addition to the more basic functions listed above, the Connections Notebook allows the user to view peripherals and networks currently connected to the PenPoint computer. The user can also view those available to the machine on the current network connection, if any.

The figures shown below illustrate the two views of the Connections Notebook.

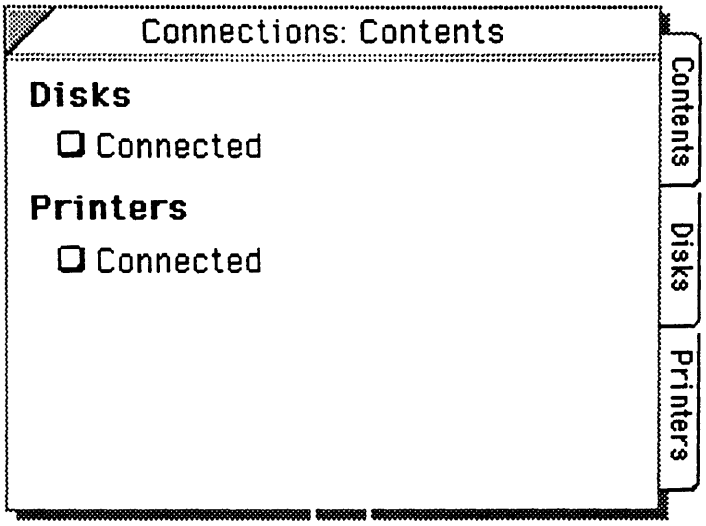


Figure 1: Connections Table of Contents

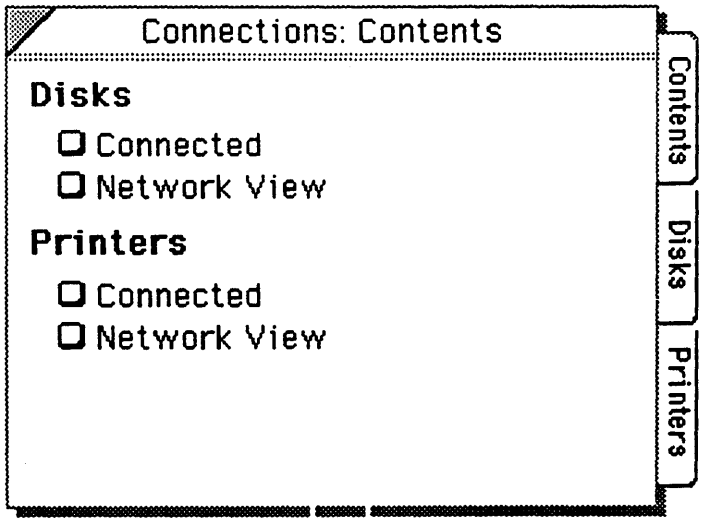


Figure 2: Connections Table of Contents (Network Connected)

When the user turns the Connections Notebook to the Disks page, he or she will see a display similar to the figure below:

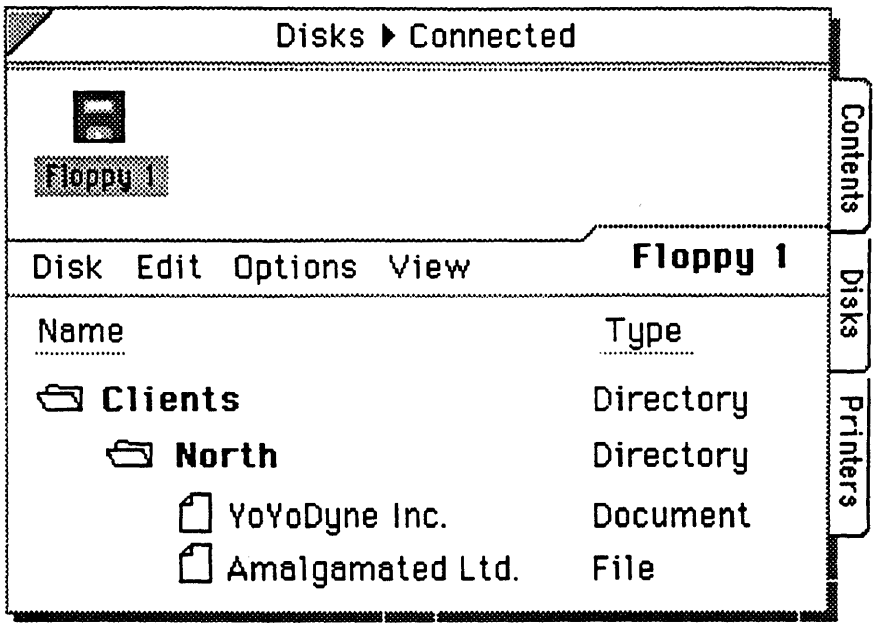


Figure 3: Available Disks Page of Connections Notebook

The page displays all currently connected disks. An icon for each disk appears at the top. When the user taps the icon, the display reflects the contents of the disk, as shown.

The View menu allows the user to choose various views of the contents of the disk. The view categories are directory, installable software, and bookshelf.

- **Directory View.** This view, shown in figure 3, provides a traditional layout of directories and files.

- **Installable Software View.** The user can pick one of several categories of software that can be installed, including applications, handwriting engines, PenPoint Services, fonts, etc. An example depicting applications appears below:

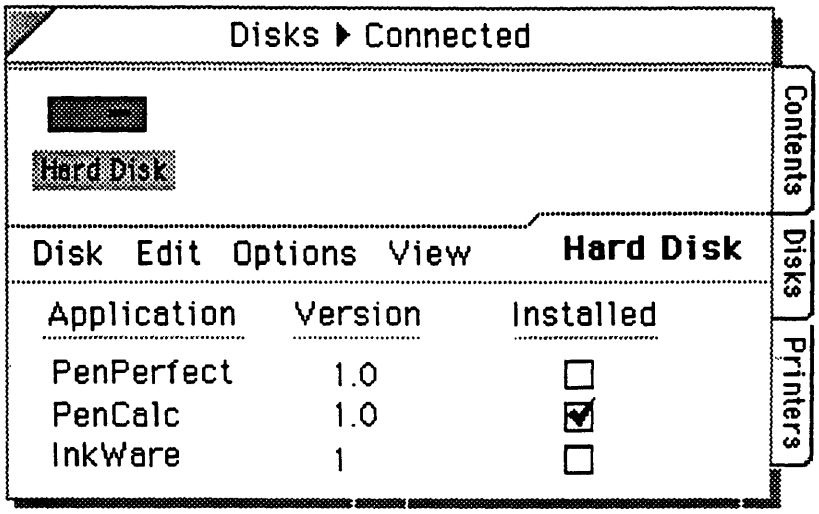


Figure 4: Applications View

Note the indication of software already installed.

- **Bookshelf View.** This view shows all PenPoint Notebooks available on the disk. The user displays the notebook(s) contents by tapping on the notebook icon(s).

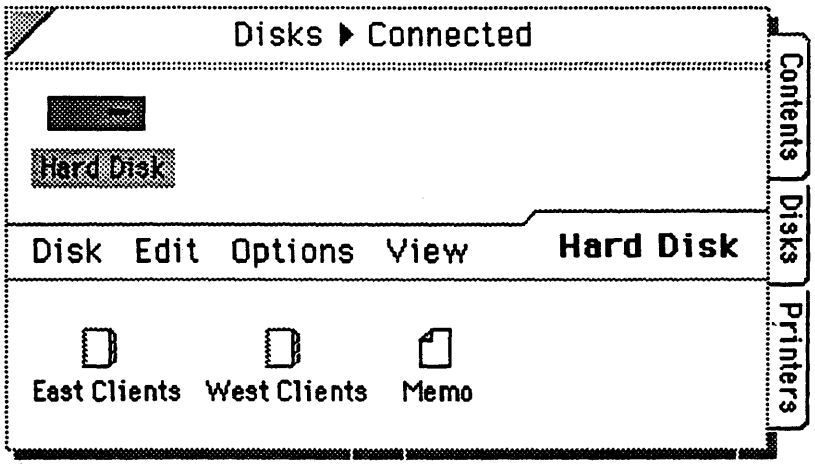


Figure 5: Bookshelf View

PenPoint's capability to use disk-based notebooks provides a great deal of flexibility to individual users as well as work groups. For example, a group administrator could maintain a standard notebook on a file server that all users could access to share documents or retrieve the latest versions of forms, price lists or spreadsheets. PenPoint's disk-based notebook capability allows an individual to access a notebook or a document within a notebook that is larger than their machine's available storage.

When working with network disks, the Connections Notebook displays connected and available disks on the network.

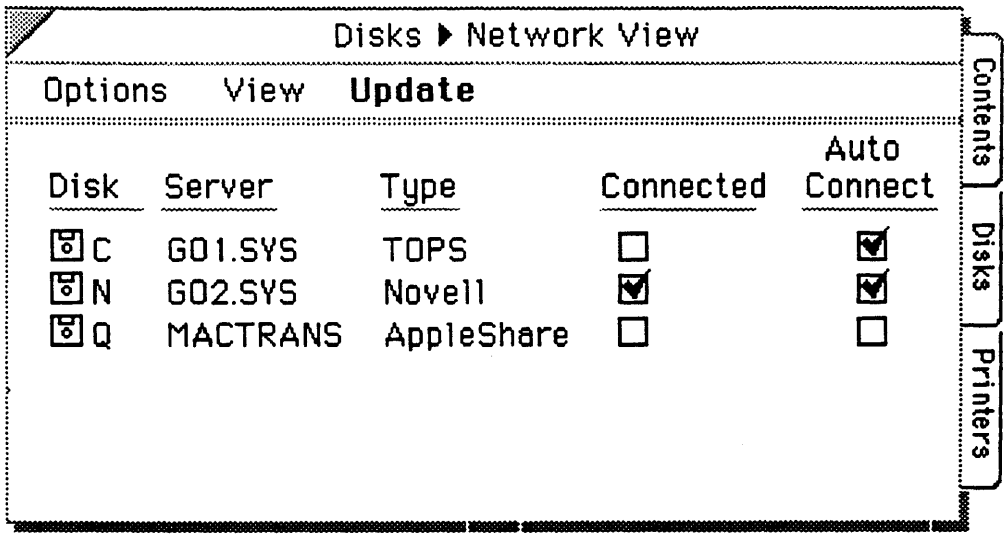


Figure 6: Disks — Network View

Tapping the *Update* button, allows the user to refresh the display to reflect the current state of the network. Tapping the *Connected* check box initiates the connection to a remote disk, database, or other server, and may cause a network-specific dialog box to authenticate the user for the particular server. Tapping *Auto Connect* for a given disk will cause automatic attachment of the disk once the physical connection to the disk is complete.

PenPoint Printing

PenPoint's printing architecture provides several key benefits for mobile users.

- Printing is a standard behavior that all applications inherit from the PenPoint Application framework. As a result, the user interface for printing is consistent throughout PenPoint and applications.
- PenPoint will include a variety of printer drivers for dot matrix and PCL printers.
- Definition of printers. Either the user or an administrator can define printers. Multiple printers can co-exist on the machine at the same time. The user defines the printer through the Connections Notebook for local printers or printers accessed through a network connection. The user uses the Connections Notebook to view the connected printers or to access network printers

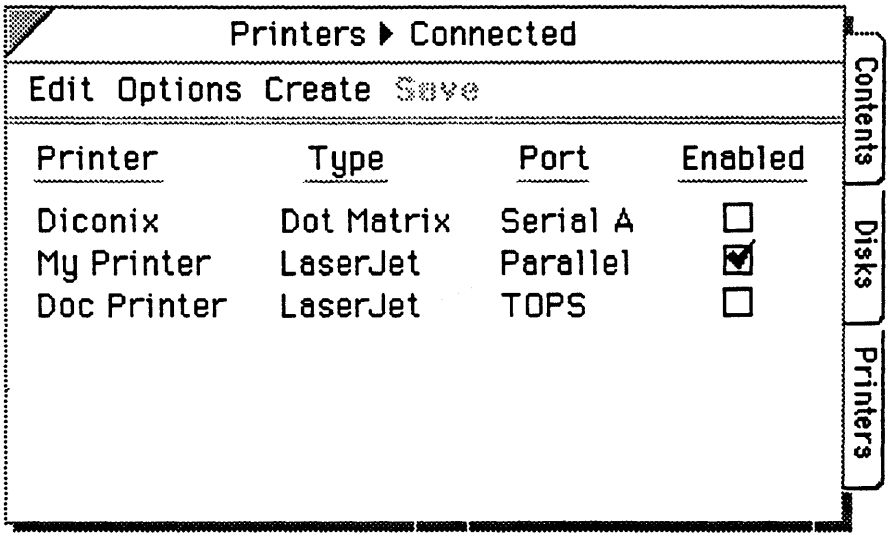


Figure 7: Connected Printers Page

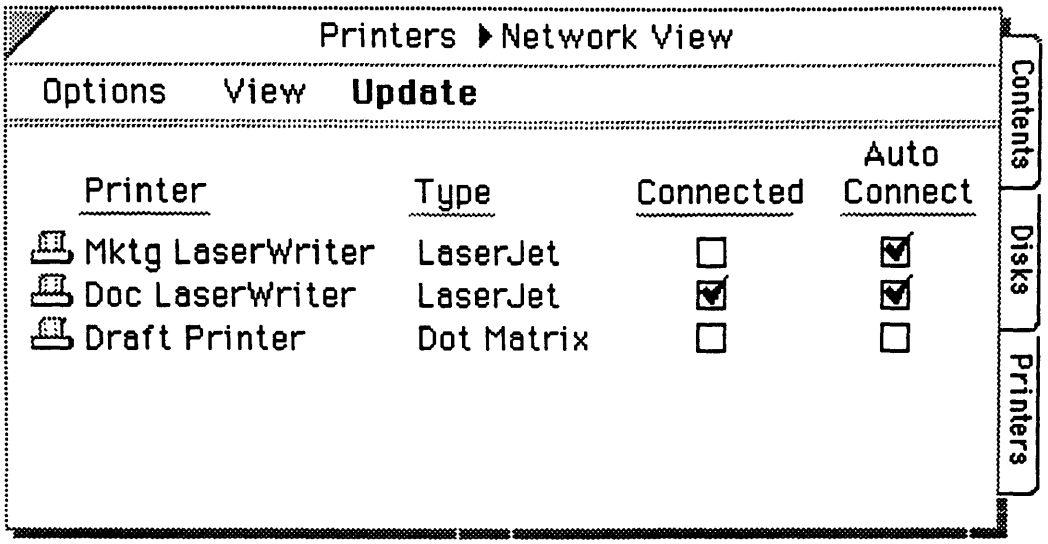


Figure 8: Printers on Network

- PenPoint supports print servers that require user authentication by displaying the appropriate dialog box to log on to the print server.
- PenPoint places documents in the PenPoint Out box beneath the particular printer name that the user selected in the printing dialog box, as shown below.

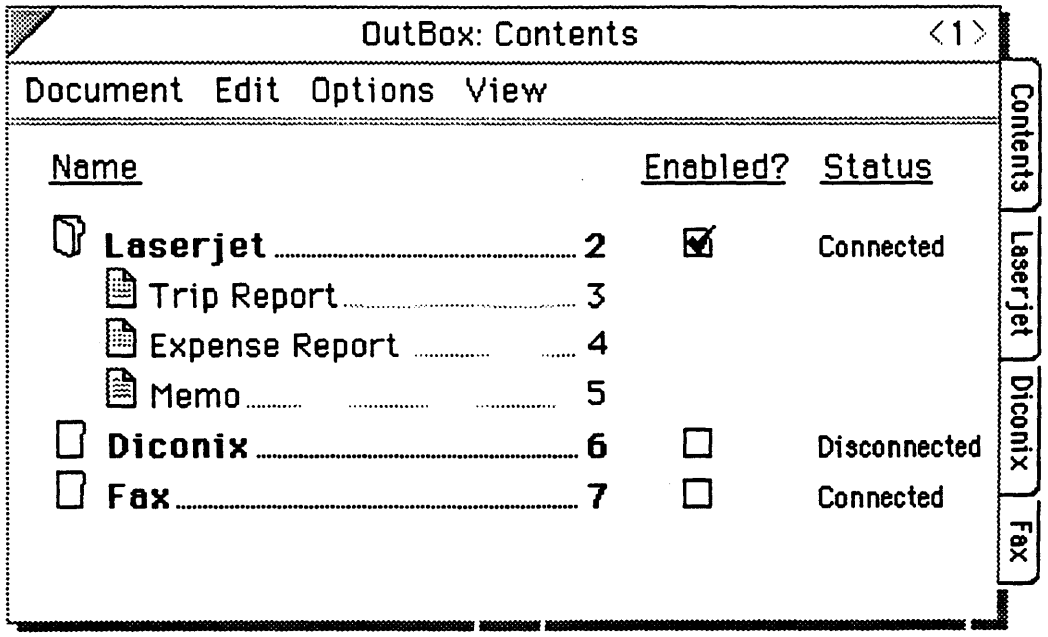


Figure 9: Out box

The user can view and edit the document in the PenPoint Out box prior to the actual printing of the document, as with other PenPoint "Sendable Services."

- Through the PenPoint Out box, the user changes the order of the documents queued for a specific printer instance with the same gestures used in other PenPoint notebooks to reorder documents.
- Printing begins automatically upon detection of printer connection, as soon as the user connects and enables (or enables and connects) the printer.

Compatibility with Industry Standards

GO approaches networking and communications with two goals in mind. First, to make the user interface easily understood by new and experienced users alike. Second, to conform to existing industry standards rather than requiring special versions of server software to be written to support PenPoint.

GO is working with a number of partners to create implementations of LAN requester software, terminal emulators, file format conversion software and other communications applications for PenPoint. While these applications and Services will use the Connections Notebook, the In box and Out box, and other unique PenPoint capabilities, the administrator of the network will not have to change the networking and communications environment to accommodate PenPoint machines.

The PenPoint Applications Framework provides support for data compatibility for applications developers. PenPoint automatically detects whether an incoming file is an actual PenPoint document or not. If not, PenPoint polls all installed applications to determine whether the file format is readable by any application. A dialog box displays those applications that can read the format.

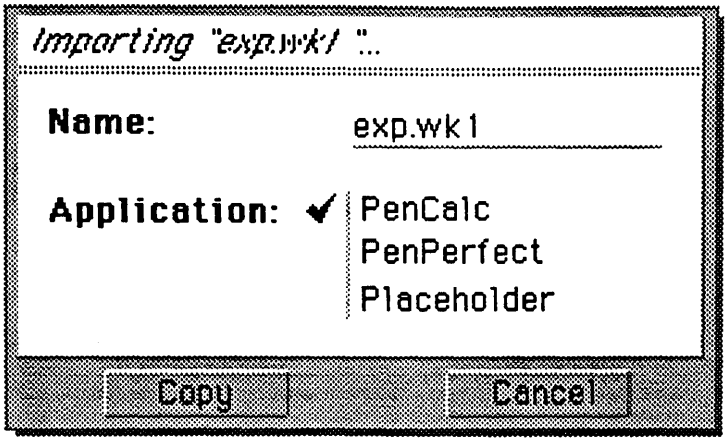


Figure 10: Import Note

The user taps on the application that they wish to use, and then taps the *Copy* button. The application then reads the file in, and writes it in the PenPoint Notebook in the application's internal format.

Export works in a similar fashion. The user selects the document in the notebook, and drags and drops the document to an external drive. PenPoint displays a dialog box asking the user whether they intend to make a copy of the document in its PenPoint format, or if they wish to export the document to an external file format

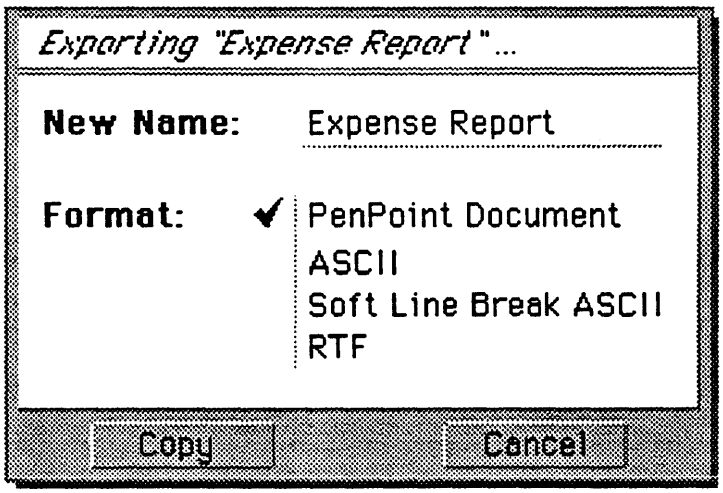


Figure 11: Export Note for Copy

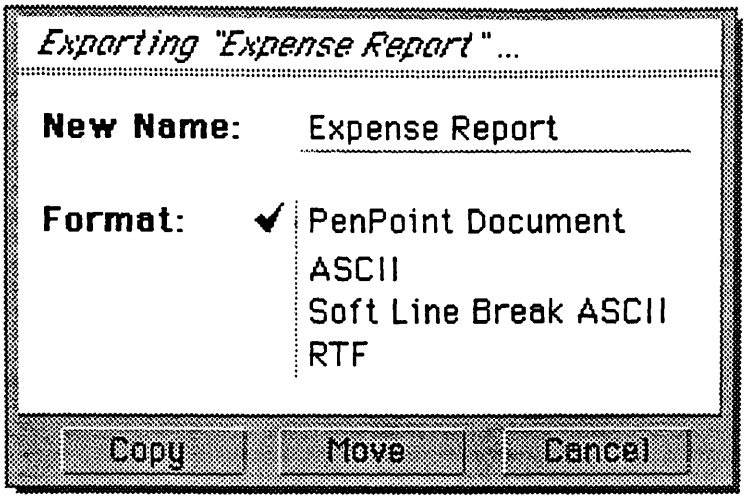


Figure 12: Export Note for Move

If the user indicates an external format, the application writes a file to the disk in the format selected.

The PenPoint file system runs atop multiple local and remote file systems. PenPoint applications interface with the PenPoint file system. This separation allows the implementation of the local file system, which is a PenPoint Service, to handle specific implementation aspects of the file system. The application is unaware of any particular local or remote file system, as is the user.

PenPoint Information Architecture

The PenPoint Information Architecture is a set of PenPoint Services for registration of and access to local and remote database systems. P.I.A. provides benefits for application developers and developers of database Services, as well as end users. Some of the benefits are:

- P.I.A. is a common API for local and remote data base access.
- P.I.A. will insulate applications developers from the underlying database, and allow many applications to work with a given DBMS back end Service.

- P.I.A. does not prevent access to DBMS features not provided through the P.I.A. standard API. For example, a developer that developed their application expressly to work with SQL Server could make SQL Server calls in addition to P.I.A. calls.
- Allows development of data base management system back ends by third parties.
- Uses PenPoint Connectivity Architecture, including the Connections Notebook. DBMS back ends can register themselves with P.I.A.. The user accesses the DBMS through a page in the Connections Notebook as has been described earlier for disks and printers.

Summary: Mobile Connectivity for Real People

The PenPoint operating system provides an extremely simple user interface that is better than pen and paper. PenPoint's architecture for mobile connectivity provides access to a wide variety of peripherals, network services and database systems through the same familiar gestures and navigation concepts used elsewhere in PenPoint. PenPoint users can access information in very large and complex networks simply and directly. GO is working with industry leaders to make PenPoint connectivity as broad and rich as any other platform in the industry.

###

Position Paper
GO Corp.'s PenPoint™
System-Wide Object-Oriented Programming

CONTACTS: Marcia Mason (415) 358-2000
GO Corp.
950 Tower Lane, Suite 1400
Foster City, CA 94404

Mari Mineta Clapp (415) 354-4449
Regis McKenna Inc.
1755 Embarcadero Road
Palo Alto, CA 94303

Contents

Introduction: The Need for a New Operating System

When introducing a paradigm shift in the way we do computing, history shows that one needs to introduce a new operating system designed expressly for that new paradigm. Whether it be CP/M and DOS, which ushered in character-based personal computing, or the Macintosh OS, which ushered in the era of graphical user interfaces, it has always taken a new operating system to enable application developers to leverage the revolutionary characteristics of the new paradigm.

Four years ago, GO Corporation set out to deliver a new kind of computing to allow mobile individuals to perform tasks which were not possible with the current generation of personal computers. In particular, we wanted to allow a user to use a pen for the input of handwriting, gestures, commands, drawings, etc. directly on the screen of an extremely light, mobile tablet-like device with integrated communications capabilities. We believed that the convergence of a number of technologies — cordless digitizers, handwriting recognition, compact imaging, object-oriented programming models, wireless networking, low-power microprocessors and peripherals, etc. — would allow our vision to be fulfilled, and after much deliberation, we decided to build PenPoint, a new operating system for the new paradigm of mobile pen computing.

There are three major constituencies that must be served by an operating system: software developers, end users and hardware manufacturers.When we looked at previous OS design efforts, we noticed that each effort emphasized the needs of one of those constituencies at the expense of the others. Some efforts provided great graphical user interfaces, but had primitive tools for the software developer. Others had unpleasant user interfaces but made it easy to port the operating system to disparate hardware platforms. Still others had great software tools for application development but ran on limited hardware platforms. From the beginning, PenPoint has been designed to provide the best possible experience for the end-user, the most profitable experience for the hardware manufacturer, and the richest development environment for the software developer.

The PenPoint operating system incorporates the following development tools to provide the developer with a rich and flexible operating system for application development:

- Classes and subclasses
- System-wide object-oriented programming
- Rich class substrate
- Application framework.

Classes and Subclasses

GO's software development environment is based on object-oriented programming (OOP) techniques. More specifically, PenPoint defines a set of well-known ***classes,*** and allows developers to define additional classes. A class describes an aggregate of data and the behaviors that can be applied to that data. PenPoint, for example, describes classes for applications (clsApp), buttons (clsButton), windows (clsWin), labels (clsLabel), etc.

These classes serve as templates from which instances called ***objects*** are created that hold data and are the focus of operations when the PenPoint system and PenPoint applications are running. For example, for a dialog box that has two choice buttons for "yes" or "no," the application would typically create one "yes" object as an instance of the clsButton class and another "no" object as an instance of the clsButton class. When the application would like an object to perform a certain behavior, it sends the object a ***message*** that indicates the desired behavior. The object determines how its class implements that behavior and calls the appropriate ***method*** that the class has supplied for that behavior.

Using the process called ***subclassing,*** a programmer can create a new class by refining an existing class. For instance, a developer can create a subclass of clsLabel called clsDangerLabel, which has an additional data field to hold an emergency message and in which the Drawing ***method*** has been ***overridden*** to draw in red instead of black.

System-Wide Object-Oriented Programming

In most object-oriented systems, the objects typically live within a single application and exist only while that application is running. One application is forbidden from accessing another application's objects. This is extremely restrictive, as often two applications might like to share the same data.

With PenPoint's system-wide object-oriented programming, objects are managed not by an individual application but by PenPoint's Class Manager. Multiple applications can send messages to the same object with a completely transparent interface.

For example one might have a clsStockerTicker class, and might have three objects of that class for the New York Stock Exchange (NYSE), American Stock Exchange (AMEX), and NASDAQ. The NASDAQ object might have first been created in an application that was reading the ticker off of the FM airwaves, and might implement messages like "msgCurrentPrice(companyName)." A separate application, like a PortfolioManager, a StockGrapher, or a Spreadsheet would be able get a handle to the well-known instance of the NASDAQ object and send it the "msgCurrentPrice(companyName)" message as if the NASDAQ object were actually in objSend that application's address space.

Where other systems have very difficult and idiosyncratic interprocess communication mechanisms, in PenPoint, communicating with another process is virtually identical to communicating with objects in your own process. Since one of the promises of pen computing is to enhance cooperating and communications amongst users and applications, interprocess communication becomes an important function.

Rich Class Substrate

In fact, PenPoint defines hundreds of important classes that together provide a rich substrate that software developers can leverage in building their applications. Rather than forcing implementation from scratch, PenPoint provides the developer with, among others:

- ImagePoint™ Graphics Classes
- Windowing Classes
- User Interface Classes
- Input Handling Classes
- Text Editing Building Block Classes
- Table Classes
- Resource File Classes.

For example, developers who need a full-functioning handwriting-based word processing module with their application can simply directly use or create a subclass of the text edit building block classes. This ability to leverage existing system classes, and subclass them with refinements, is fundamental to object-oriented programming. It is also fundamental to software development for small pen-based systems, because the use and refinement of system classes assures that the applications that are developed remain small, since much of the code is shared. In other systems, the absence of a class structure and any system-provided way to share basic building blocks makes the size of each application explode. With PenPoint, the application developer doesn't have to cut down his or her own tree to get some wood; the PenPoint class substrate is the lumber yard and easily provides the pre-cut building blocks.

Application Framework

While our class-based building blocks are much easier to use and refine than traditional procedural toolkits, it is but one half of the story. The flaw with most other graphical user interfaces (GUIs) systems, has been to provide a low-level toolbox, provide a set of written user-interface guidelines for how an application should look and operate, and then let each application developer put the toolbox pieces together to get a working application. Since most applications share much commonality in areas such as printing, searching, scrolling, saving, restoring, handling input, handling menus, etc., it is a waste of programming effort to require each application developer to re-invent this large base of common code, not to mention requiring them to suffer through the

painstaking debugging of every last detail, so they will be fully compliant with the look-and-feel standard of the operating system.

PenPoint provides an object-oriented ***application framework***, which dispenses with the need for application developers to focus on anything but the implementation of their particular application's code. The application framework provides a set of classes that manages the notebook user interface, the application life cycle (starting, terminating, activation, deactivation, saving, restoring, installation, deinstallation), the input handling (directing pen input to appropriate windows, menus, tabs, and identifying gestures, etc.), printing, pattern searching, spell checking, etc. These classes are designed in such a way that they work very closely together. In fact, a program of less than 50 lines which initializes and instantiates an object from the *clsApp*class is all that is needed to get a fully functioning application. With only 50 lines of code, this PenPoint *universal application* exhibits all of the behaviors of a full PenPoint application — standard application menus, saving, restoring, printing, scrolling, etc. The only thing different about the universal application from a commercial application is that the universal application doesn't do anything inside of its page.

In fact, that is the whole point of the application framework. PenPoint provides a fully-working universal application, and the application developer, through subclassing of the universal application classes and through the overriding of the class behaviors, refines the universal application to perform the specific functions that are within a notebook page. The application developer does not need to be concerned with pen tracking, menu tracking, window resizing, etc. The developer only concentrates on the data structure handling, data structure rendering, and interaction that goes on within a page.

For example, the application developer might subclass clsApp to create a clsSpreadsheetApp. At appropriate times, clsApp will receive messages from other parts of the application framework, typically when user input requires clsApp to perform some functions. The class clsApp will first try to handle the function generically. If it is a function that only clsSpreadsheetApp would know how to handle, then the developer of clsSpreadsheetApp would be required to override that behavior so that when the message was received, the spreadsheet-specific functionality would be invoked.

The ability to concentrate on building specific application functionality, rather than reinventing all of the code to implement a system-wide look-and-feel is one of the major contributions of PenPoint's application framework and universal application. However, there are additional benefits. Because the standard look-and-feel is only implemented once, PenPoint applications show an extremely high-level of consistency. Because every application is based on the app framework, new building blocks that fit into the app framework paradigm are immediately useable by a large number of application developers. As well, because all PenPoint applications are derived from the rich application framework, they are typically much smaller than those found in traditional operating systems. Small applications are extremely important for pen computers, which typically have limited amounts of memory compared to desktop machines and may not even contain a hard disk. This paradigm is very different than getting the source code for a skeleton application and editing it for your specific use. With that strategy, each developer modifies the original skeleton, and each derivation tends to diverge from the original and incorporate more errors. Because there is only one application framework, PenPoint guarantees that the universal application that you start with will always be most current.

Additional PenPoint Features

Just as it has an application framework for end-user applications, PenPoint provides a service framework for embedded services such as networking and connectivity services, device drivers, handwriting recognition services, etc. Again, rather than writing a device driver or a network protocol from scratch, PenPoint provides complete working classes that can be subclasses for your particular application. Many of the nitty-gritty elements of device driver writing, for instance, are not specific to a given device, but are highly-generalizable; these are incorporated in the service framework, so that the device driver writer can simply use them, rather than invent them from scratch.

Summary

Pen computing needs a new operating system because it is trying to first and foremost make computing much, much easier for the end-user. Pen computing requires more consistency, smaller size, more directness, and more complicated processing (handwriting and gesture recognition for instance) than desktop computing, since one of the largest markets will be people for whom desktop computing was too complicated and arcane.

At GO, we took the opportunity to help the end user as our first mission, and in doing so, discovered that one of the best ways we could help that end user was to make life as easy as possible for the application developer. Only if application developers have the tools from which to build familiar, accessible, and highly functional natural applications, will GO truly be able to meet our goal for the end user. PenPoint's goal is to set a new standard, for both the use of and the development of truly pen-centric applications by providing system-wide objects, a rich class substrate, an object-oriented application framework, a universal application, and a service framework.

###

Position Paper
GO Corp.'s PenPoint™
Handwriting Recognition Technology

CONTACTS: Marcia Mason (415) 358-2000
GO Corp.
950 Tower Lane, Suite 1400
Foster City, CA 94404

Mari Mineta Clapp (415) 354-4449
Regis McKenna Inc.
1755 Embarcadero Road
Palo Alto, CA 94303

Contents

Introduction

Now that pen-based products are becoming a reality outside of research settings, the features and capabilities of handwriting recognition systems have moved from the laboratory to practical application.

GO Corporation has invested heavily in the development of a powerful general-purpose handwriting recognition system. GO also has developed a research program chartered with developing "real-world" metrics and methods to evaluate the performance of the system for a wide variety of tasks.

The following sections will discuss GO's handwriting recognition system and describe the metrics and measurements involved in assessing handwriting recognition technology. Also examined is the impact that an effective dialog between applications and a handwriting recognition system has on the usability of applications.

Finally, this paper will address the larger issue of how handwriting recognition technology relates to GO's strategy for pen computing which seeks to open a new method of computing to a new class of users.

Evaluating the Capabilities of Handwriting Recognition Systems

The following are criteria that should be considered when evaluating handwriting recognition technology.

Symbols recognized. Does the system recognize only upper case characters and numbers (36 symbols), or does it also recognize lower case characters (a total of 62 symbols)? How many punctuation characters does it recognize?

GO's handwriting recognition system recognizes 25 punctuation symbols in addition to upper and lower case letters and numbers for a total of 87 symbols.

Segmented versus unsegmented input. Is the system capable of computing breaks between letters or are users required to write in boxed or combed fields to indicate letter spacing? Is the system capable of computing spaces between words, or is the user required to write a special space character or skip a space in a boxed or combed field to indicate word spacing?

GO's handwriting recognition system can compute both letter and word spacing; users do not need to write a special space character or write in boxes or combs to indicate letter and word spacing. However, users may write in boxed or combed fields if they want to, to help maintain consistent spacing.

Flexibility of writing style. Can the system recognize a wide variety of printed forms for each character, or must users print characters in one or more standard ways? Are script forms of characters recognized? Can characters overlap or connect to each other without the pen being lifted? Can script handwriting be recognized (all characters connected)? Can a mixture of disconnected and connected characters (script and printed characters) within a word be recognized?

GO's handwriting recognition system recognizes a wide variety of printed forms for each character. Users can also train the system to recognize new shapes, see below. The system also recognizes script forms of characters. GO's system tolerates some overlapping as well as connected characters, although better recognition is obtained if characters are clearly separated. GO's current system does not recognize continuous script or mixed script and printed forms of writing.

Customization. Can users customize the handwriting system to recognize their particular idiosyncratic shapes or methods of forming characters? If the system can be customized (or trained), is this required before the system can be used, or is it optional? What is the user interface for this customization process? Can more than one user store his or her individually customized recognition engine on the system at the same time?

GO has more than one handwriting recognition engine, and it is important to note that although not all engines may be customizable, at least one engine is available which users can customize to recognize their idiosyncratic shapes or forms. This is optional, not required, as the system is

usable for most people prior to using the customization application. The user interface for customization prompts users to write specified sentences, and learns how the user writes individual characters in the context of words and sentences. In addition, users can choose to focus on customizing only specific individual characters. More than one customized recognition engine can reside on each computer, and users can easily switch among them.

Gestures. Can the handwriting recognition system recognize a wide variety of symbols and shapes as gestures, or command accelerators?

GO's handwriting recognition system recognizes over 50 gestures. Eleven of these gestures are "core" gestures which deliver the most important functionality and which have the same basic meaning across all applications (similar to the standard application menus featured in other GUIs). The additional gestures can be thought of as optional accelerators similar to keyboard accelerators used in many applications.

Flexibility in writing size. Can users write in different sizes, or is the system limited to characters written within a narrow range of heights and widths?

GO's handwriting recognition system is capable of recognizing writing over a broad range of height and width.

Stroke order independence. Is the system capable of recognizing strokes out of sequence? For example, can users dot their "i"s and cross their "t"s at the end of a word, or on a previous word, or must they complete each character before writing the next?

GO's handwriting recognition system recognizes strokes that are added to characters out of sequence, either at the end of a word, or even on any previous word in the entire input line.

Flexibility of handwriting user interface. Does the system allow users to vary the features of the input pads that they write into? Can users choose between ruled lines and boxed or combed fields? Can users vary the line height or the height and width of boxes or combs? Can users write in multiple lines? Can users vary the number of lines or rows in the writing pads?

GO's handwriting recognition system allows users to vary all of these features of writing pads.

Gesture recognition in writing pads. Is the system capable of recognizing and responding to gestures in writing pads (in the same space and at the same time that characters are also being recognized)?

GO's system recognizes certain gestures in writing pads and distinguishes them from text. For example, users may "scratch-out" handwriting before initiating recognition in order to avoid having the system attempt to translate certain letters or words. In addition, characters in boxed or combed fields may be deleted, or extra boxes or combs can be added with gestures.

Flexibility of user interface for error correction. Does the system allow users to determine when to recognize or translate handwriting? Can users defer the recognition process indefinitely? Can multiple writing pads or input fields be simultaneously left in such a deferred translation mode? When the user does initiate recognition, does the system provide choices for how the results of the recognition are presented for verification and correction? Or is there only one user interface that all users must use regardless of the accuracy rate that they achieve? Does the system enable users to choose the symbol that will be displayed when it is unable to recognize a particular character?

GO's handwriting recognition system provides a flexible user interface for error correction in each of these respects.

Support for upper case-only writers. Does the handwriting recognition system enable users to write in all upper case letters and automatically convert the characters to lower case after recognition? If so, does it do so intelligently, taking into account which characters should remain upper case, or does it force all characters to lower case?

GO's handwriting recognition system uses heuristics to enable users to write all upper case characters and have them intelligently converted to lower case, preserving upper case when appropriate.

Speed. How fast does the handwriting recognition system recognize characters? Does the recognition proceed in the background, or does a majority of the computation take place after the user initiates the recognition process?

GO's handwriting recognition system recognizes characters at the rate of about 3 chararacters per second. The recognition proceeds in the background in multiple line writing pads, so users rarely experience delays longer than the time required to recognize one line (2-3 seconds), regardless of the amount of writing in the pad. This is an advantage of PenPoint's true multitasking architecture.

GO User Research: Metrics and Methods

GO's research has discovered that there are four topics that are very important to end-users of pen computers that have not been systematically measured or reported by vendors of handwriting recognition systems. The User Research Group at GO has worked extensively with the engineering team for over two years (since 1989) to design and implement improvements to the performance of the system in each of the following areas.

First, users are concerned about the acceptability of the handwriting recognition system's performance when writing sentences and paragraphs of text (as opposed to just short strings of characters in form fields).

Second, users want the methods used when gathering system performance data to be realistic and representative of typical (not ideal) performance.

Third, users are concerned about the acceptability of the total time (including error correction) it takes to complete a writing task.

Fourth, developers of applications making heavy use of data entry fields (like those found in forms) need to know how the handwriting recognition system will perform when users are given various constraints on valid input (e.g., fields that accept numbers only, or fields that accept only certain upper case characters).

Metrics for writing sentences and paragraphs of text

Users' acceptance of handwriting recognition accuracy in this context is related much more to word-level accuracy (the percentage of words translated correctly) than it is to the percentage of characters translated correctly. sample 1 shows a paragraph of text with some recognition errors. When asked to rate whether this level of accuracy would be acceptable, no users say that it would be, and most guess the character accuracy rate to be between 60% and 75%.

Sample 1. Handwriting Recognition

> Mg bvss at Wilcox Resegrch spoke rery kighly of yoyr cofpany and hecomkended that I aet in tuuch with yau. I am cuvrently looktng for a senior marlceting position in a grouth ofiented hign teoh somqany. I dould be very tnterested in kearpng your pefspeytive on the ibdustry tnd which negmehts are vffering good carefr oploortunities.

Compare this to Sample 2 on the following page, which almost all users rate as acceptable, and most guess that the character accuracy is between 90% and 95%. In fact the two paragraphs have exactly the same character accuracy rate (300/333 = 90% correct) but widely different word-level accuracy. Figure 1 has a 50% word accuracy rate (28/56 words correct) because the individual character errors are distributed almost evenly over many words. Figure 2 has a much higher 91% word accuracy rate (51/56 words correct) because the errors tend to cluster together in words.

The difference between character-level and word-level accuracy is important when evaluating handwriting recognition systems because two systems that have similar levels of character accuracy can have widely different levels of word accuracy.

Sample 2. Handwriting Recognition

My boss at Wilcox Rcseaneh spoke very highly of your company and recommended that I get in touch with you. I am cnenerfig looking for a senior marketing position in a growth oriented high tech company. I would be very ehfoncofca in hearing your perspective on the industry and ynlan segments are offering good ednoon opportunities.

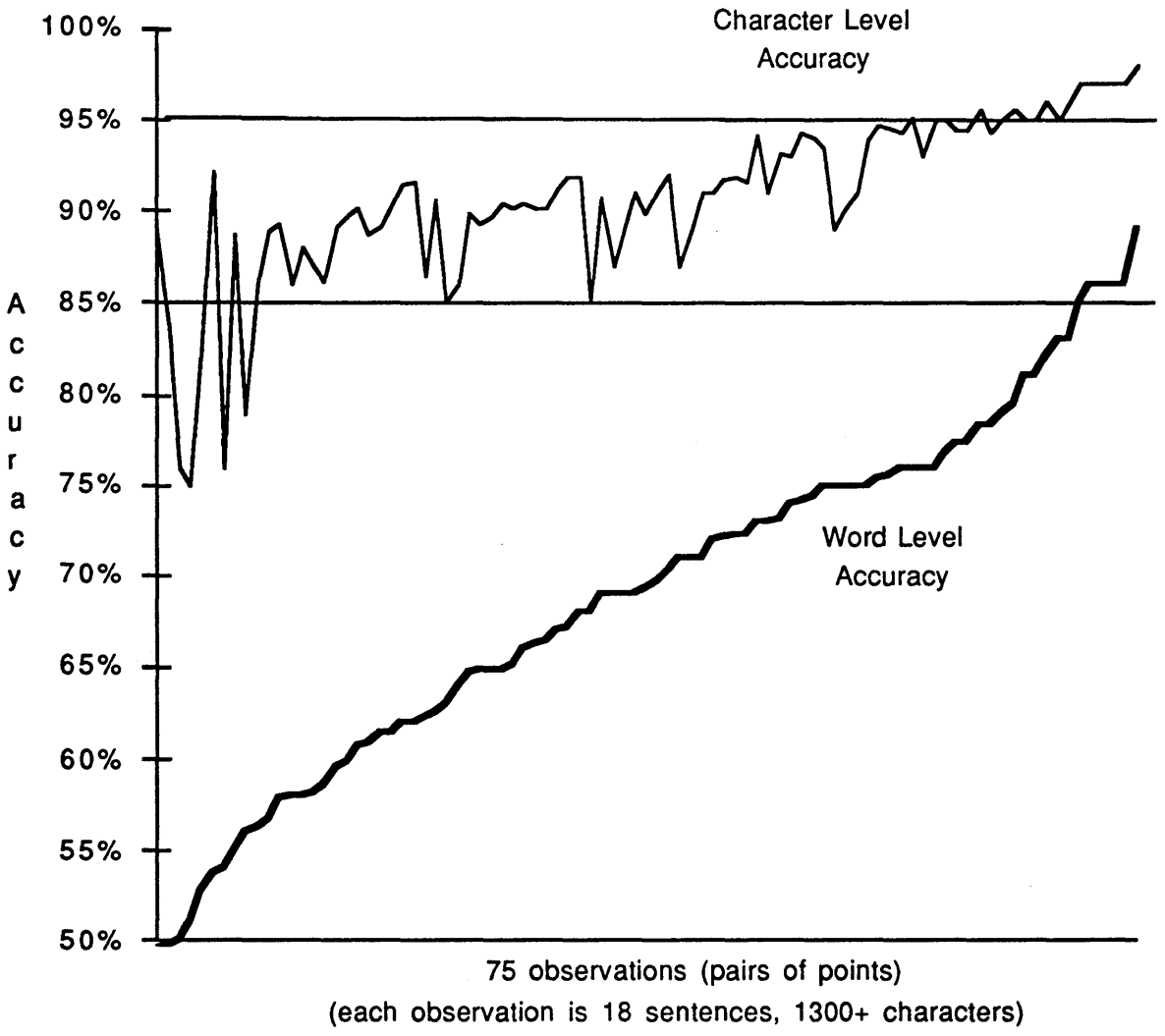
Figure 3 on the following page, illustrates the relationship between word accuracy and character accuracy for 75 samples of handwriting recognition that GO has observed in measuring early versions of its handwriting system and other systems currently available (each of the 75 samples represents a different user writing over 1300 characters). It is clear that high character accuracy rates do not guarantee high word accuracy rates. Many times, systems with approximately 90% character accuracy rates produced relatively low word accuracy rates (between 50-70%).

This level of performance is typical of handwriting recognition systems that focus primarily on the recognition of individual isolated characters and do not make effective use of dictionaries and/or other information about the relative frequencies of combinations of letters.

GO's User Research Group has published a detailed technical report describing some of our research in this area. This report, titled "*Evaluation of Handwriting Recognition Technology: Word-level vs. Character-level Accuracy*" was presented at the 35th annual meeting of the Human Factors Society (1991) and is available from GO upon request.

Figure 3.

The Relationship between word-level accuracy and character-level accuracy in 75 samples of handwriting recognition.



GO's User Research program has also determined that users' acceptance of handwriting recognition systems is strongly affected by how easy it is to correct words. GO's recognition system provides users with a list of alternatives for each word, so many words can be corrected simply by tapping on the correct word in a list. Other words can be corrected simply by writing over one or two characters. When users are faced with more than one or two errors in a word, and can't choose the correct word from a list, they typically choose to rewrite the entire word. Another important metric therefore is the percentage of words that users will rewrite versus the percentage of words that can be corrected with a simple edit.

Finally, it is important to measure and report the variance in users' accuracy rates. How much better than average do some users do, and how much worse than average do others do? It is useful to know the standard deviation of the mean (a statistical measure of variance) or to know what the average of the top 50% of users is as well as the average of the bottom 50%.

Methods for measuring typical performance

Users' acceptance is determined by their own experience with the system. In order to predict the level of acceptance a system will have, it is important to test it and obtain metrics that are representative of typical users' experiences. The three most common ways in which handwriting recognition systems are measured with inappropriate methods are by using test subjects that are not representative of typical users, by using test materials (samples of writing) that are not representative of the material that typical users will usually be writing, and by using levels of practice before measurement that are not representative of typical use.

Samples of subjects used to test handwriting systems are often biased towards younger and more highly educated users. When interpreting test results, it is important to know the ages and educational backgrounds of the subjects. Sometimes systems are measured with samples of writing that contain only dictionary words, or that contain little if any punctuation. However, normal business writing includes punctuation and many words not usually in a dictionary, and this should be taken into account. Sometimes the accuracy of systems is measured after subjects have been practicing for weeks or

months with the system, or only after they have extensively adapted their handwriting style. When evaluating results, it is important to understand exactly how much practice and/or training subjects in a test have had.

Metrics for total time to write

Users' acceptance is affected greatly by the total amount of time it takes to write a given amount of text. Thus, research data should include the time to write, the time for the system to translate the handwriting, and the time it takes the user to correct any recognition errors. The user interface for writing pads (areas of the screen that accept handwriting) has a large impact on writing time. For example, pads that require writing characters in segmented boxes typically result in increasing writing time by about 25%-30%.

Similarly, the user interface for error correction has a large impact on the time required to correct errors. One system may have a lower accuracy rate than another yet be strongly preferred by users if the total amount of time to enter, translate, and correct text is lower. A complete suite of metrics should include these time measurements in characters per second and words per minute.

Metrics for writing in fields with constrained input

Character-level accuracy data should always be reported in conjunction with the size of the symbol set the recognition was being constrained to. These metrics are important to understand when designing applications involving large amounts of data entry into fields. What is the accuracy rate for numbers in a field that accepts numbers as well as upper case and lower case letters (62 symbols)? What is the rate in a field that accepts only numbers (10 symbols)? Similarly, what are the accuracy rates for upper and lower case letters in fields where 62 symbols are valid, and how does it compare to fields in which only 26 symbols are valid?

Accuracy of GO's Current Handwriting Recognition System

The following information was obtained from a series of studies done on internal releases of PenPoint and GO's handwriting recognition system in the Spring and Summer of 1991. The data reflects improvements made to the system after the Developer Release of PenPoint (February 1991). The data does not reflect improvements that will be made to the system prior to the release of PenPoint 1.0.

Not all metrics are collected in each study, therefore some of the metrics below are expected to improve more than others as they are measured with the latest versions of GO's handwriting recognition system. An update of all the metrics will be presented upon the release of PenPoint 1.0 (scheduled for first quarter 1992). In the meantime, the latest test data and analyses are available from GO upon request.

Methods

Subjects. 11-14 subjects were recruited for each of these studies by a temporary employment agency in the San Francisco area. Roughly half were male and half were female; about one third were between the ages of 20-25, one third between 25-35, and one third between 35-45; about one third had 0-1 years post secondary education, one third had 2-3 years, and one third had 4 or more years.

Materials. For word-level metrics, subjects wrote sentences of text that had been randomly selected from letters to the editor of a major business daily newspaper. The sentences contained punctuation, and names of people, companies, and products that are not in the system dictionary. For field data entry metrics, subjects wrote name, address, and part number fields to obtain metrics for various levels of constraint (size of symbol set).

Practice/Training. Subjects spent two consecutive half-days in a group training session (1 instructor, 5-7 students) that was modelled after a typical corporate training session. About 1 hour was spent gather pre-test or "walk-up" accuracy data, about 2-3 hours were spent practicing and/or customizing (depending upon which engine was being tested), about 1 hour was spent

gathering post-test or final data, and 3-4 hours were spent on other non-writing tasks to provide rest breaks for subjects.

Results

Results: Word-Level Accuracy.
[Note: these metrics are expected to improve prior to the release of PenPoint 1.0]

In a test sample of 12 sentences (144 words) the average user achieved a word-level accuracy rate of 82.1% (sd = 7.6%). The average character accuracy rate was 96.2% (sd= 3.1%). Of the 17.9% incorrect words, most could be corrected with a simple edit (16.8%, sd=6.2%) by either choosing an alternative word from the proof pad, or by overwriting 1 or 2 characters.

An average of only 1.1% (sd=1.9%) of words were categorized as needing to be rewritten (this is an overestimate because it counts errors in one and two character words as rewrites, and because some users will in fact only rewrite incorrect characters and not the entire word even when there are more than two errors).

The average of the top 50% of users was 87.1% word accuracy, 98.2% character accuracy. 14.1% of words could be corrected with simple edits, and less than 1% were categorized as needing to be rewritten.

Results: Time to enter/correct. [Note: these metrics are not expected to improve prior to the release of PenPoint 1.0] GO's handwriting recognition system accepts handprinted characters on ruled-line input pads (boxed input is optional). As a result, the initial text entry rate averaged about 17 words per minute, or about 1.5 characters per second. GO's unique user interface for error correction enables words to be corrected in an average of under 5 seconds per incorrect word. This resulted in an average total gross throughput including writing time, translating time, and editing time, of about 13 words per minute, or about .1.1 characters per second.

Results: Field Data Entry Accuracy.

[Note: GO's new recognition technology developed since PenPoint DR performs much better on these metrics. The data reported below is substantially higher than we previously reported. Most importantly, the range of accuracies reported below includes "walk-up" measurements taken prior to any training or practice with the system.]

Because of the relatively small sample sizes of our studies, the average character accuracy rate under various constraints can easily vary up or down by several percentages from one study to the next. For this reason, and because a great deal of fine tuning is currently under way prior to completing PenPoint 1.0, we will report the range of accuracy achieved in our last several studies to provide a sense of the variance we encounter.

The average accuracy rate for numbers in fields constrained to numbers ranged from 94% - 97%; the accuracy rate ranged from 90% - 93% when numbers were written in fields that could accept any of upper case characters, lower case characters, numbers, or punctuation (87 symbols).

The average accuracy rate for upper case characters in fields constrained to upper case characters ranged from 92% - 97%; the accuracy ranged from 90% - 95% when upper case characters were written in fields that could accept any of upper case characters, lower case characters, numbers, or punctuation (87 symbols).

The average accuracy rate for lower case characters in fields constrained to lower case characters ranged from 93% - 95%, the accuracy ranged from 88% - 92% when lower case characters were written in fields that could accept any of upper case characters, lower case characters, numbers, or punctuation (87 symbols).

Level of acceptability for various tasks

[Note: these metrics are not expected to improve prior to the release of PenPoint 1.0]

Subjects were asked whether the level of accuracy and the effort required to correct errors that they experienced were acceptable for the task of filling out several forms a day. Typically 90% -100% of subjects rated the system as acceptable for these tasks. Subjects were also asked whether the level of accuracy and the effort required to correct errors that they experienced were

acceptable for the task of writing several memos, letters, or notes a day. Typically 75% - 85% of subjects rated the system as acceptable for these tasks. Subjects rating the system as unacceptable were not simply those subjects experiencing the lowest handwriting recognition accuracy, although accuracy was an issue for some of them.

The majority of users experiencing lower than average recognition accuracy still rate the system as acceptable for these tasks. Reasons cited by those occasional subjects who experience good recognition accuracy yet rate the system unacceptable seem to be idiosyncratic. For example, we have recorded such diverse concerns as the speed of handwriting compared to typing for skilled typists, the physical strain of printing large amounts of text by hand, relative unfamiliarity of printing compared to script, legibility of the screen, etc.

To summarize, although 10% or more users rate their experience of the recognition system as unacceptable for certain tasks, it is important to understand that further improvements in the recognition accuracy will not guarantee that the system is acceptable by 100% of users for 100% of tasks.

To place these findings in perspective, we should ask what percentage of the population would rate today's computers, user interfaces, and input devices acceptable for a wide variety of tasks. We are not aware of any research indicating that it is as high as 80% - 90%. Of course the best way to do this type of research is to have users directly compare the ease of performing a series of tasks on a PenPoint computer to the ease of performing the same tasks on keyboard/mouse computers.

GO's User Research Group has only recently begun to conduct this type of comparative research. Preliminary data from studies focusing on the user interface indicate that for a wide variety of basic operations, most users rate PenPoint computers as easier to use than keyboard/mouse computers. Further studies are being conducted as more applications become available.

Many Applications Do Not Rely On Handwriting Recognition

Many applications with broad horizontal appeal do not require users to enter a significant amount of handwriting that needs to be recognized (or "translated") by the system.

For example, communications applications such as receiving, marking up, and sending faxes, or receiving, reviewing, and sending structured replies to electronic mail messages require little if any handwriting translation. A wide range of applications can be characterized as primarily information access and retrieval programs (query, sort, search, and display data from databases and large text documents or manuals). Drawing, painting, drafting, and layout applications do not require handwriting recognition, nor do applications designed to give computer-based interactive presentations.

Also under development are applications designed for notetaking, which do not require recognition but instead focus on storing handwriting in a raw form that allows it to be organized, rearranged, moved, copied, edited, and hyperlinked to other data. This type of application also lends itself to group brainstorming and meeting facilitation, especially when combined with screen projection and shared screen networking.

Another key set of applications with broad appeal allow users to review and edit files created on other systems (with round-trip data integrity), thus combining the power of the gesture commands for editing with the convenience of the mobility and portability of pen computers.

Finally, one should consider that many additional horizontal computer applications may only require a limited amount of handwriting recognition. For example, some applications can make heavy use of highly constrained fields (like numbers-only fields, fields with templates, or word lists) where recognition rates are very high, or multiple-choice fields or pick-lists for choosing frequently used input. Examples of applications of this type include spreadsheets, calendaring and scheduling (personal information management, or PIM), contact management, and a wide variety of forms-based applications.

Support for Dialog Between Applications and the Handwriting Recognition System

When evaluating a pen-based operating system and its handwriting recognition system, it is important to understand what level of communication between applications and the handwriting recognition system is supported. Are applications in control of the raw input (strokes)? Can they perform their own analysis or recognition of strokes? When applications pass the raw input to the handwriting recognition system, can they pass along additional information that the system can use to aid in the recognition process? If so, what information can be passed to the handwriting system, what can the handwriting system do with this information, and what impact does it have on the recognition accuracy?

Looking at the subject of communication from the other direction, it is also important to know what information the handwriting recognition system can make available to applications in addition to the best guess at recognizing the input. Finally, it is important to understand whether the functionality to support this dialog between applications and the handwriting recognition system is a feature of the operating system, or is a feature of the particular handwriting recognition system. If the handwriting recognition system is replaceable, must every system provide this functionality, or can a new shape matching engine be integrated and inherit this functionality from the operating system?

PenPoint applications are in control of the raw input; they can process it themselves, or they can utilize a very rich set of APIs that support a dialog with the handwriting recognition system. These APIs are a feature of PenPoint and its Context Management Subsystem — this means that the same dialog will be supported even if a new shape matching engine is integrated into the system.

The following is a brief overview of what information applications can provide to PenPoint and the handwriting recognition system to aid in the recognition process:

- Choice of input UI (Boxed versus ruled input pads, size of boxes, line height, etc.).
- Choice of editing UI (direct to application's client area with gestures to bring up edit pads, choice lists, etc., or direct to edit pads, choice lists).
- Choice of context aids or rules which facilitate the recognition process (e.g., spelling dictionary, personal dictionary, lists of acceptable characters, lists of acceptable words, templates similar to those used in database applications, case heuristics, punctuation rules, spacing rules).
- Level of influence that context aids and rules should have in the recognition process (four levels: enable, propose, veto, and coerce).
- Choice of post-processing aids to the recognition process (e.g., spelling correction, case correction, space correction).
- Lists of acceptable gestures to aid in gesture recognition.
- Choice of where to send strokes (gesture engine, text engine, both, or neither).
- Choice of when to process strokes or send them to a recognition engine (applications can store raw input and process them at any time).

In addition, PenPoint applications can manipulate strokes independently of the handwriting recognition system. They can: 1) filter strokes before sending them to any recognition engine, 2) perform their own analysis or recognition of the strokes, or 3) perform their own post-processing on the output of the recognition system. They can do any or all of the above in any combination. It is this flexibility which enables an application to determine whether a circle should represent a circle, the edit gesture, or the letter "o".

PenPoint's handwriting recognition system is also capable of providing a great deal of information to applications that they can use to help interpret handwriting input. The recognition system can provide to applications:

- Lists of possible characters for single character input, not just a best guess character.
- Lists of possible words for word input, not just a best guess word.
- Weightings or probabilities when lists of multiple characters or words are provided.
- Size and boundary information per character or word (this information can be used by applications to determine what to do with the input, where to place the result in its client area, or how large the result should be — especially important for "free-form" applications like drawing, notetaking, or outlining).
- Size, boundary information, and hot point of gestures.

PenPoint also provides several components in the UI Toolkit that exploit this rich communication functionality with the handwriting recognition system by providing a higher level set of APIs that support specific application functionality. For example, one component greatly facilitates the process of developing forms fields with a set of APIs that tie together functionality of the input system, the handwriting recognition system, and the windowing system. Other UI Toolkit components facilitate the common uses of gestures and ruled-line input pads. PenPoint's object-oriented architecture encourages the development of such higher level components, and more will undoubtedly be developed over time.

Handwriting Recognition System Replaceability

Some pen computers have integrated handwriting recognition technology so tightly that it is not feasible to replace the system with one from another independent or third-party developer. PenPoint is designed with a clean application programming interface (API) that enables the shape matching engine in GO's handwriting recognition system to be easily replaced with engines from other vendors.

Since the shape matching engine is implemented below the level of the Context Management Subsystem, any engine that replaces it will still benefit from the powerful functionality that the Context Management Subsystem provides to applications. In much the same way that some desktop computer operating systems today can support different manufacturers' printers by installing printer drivers, PenPoint will allow handwriting recognition systems from independent third-party developers to be installed.

This replaceability is important because handwriting recognition technology is still evolving and improving. Systems currently under development are based on radically different technologies ranging from neural networks to fuzzy logic algorithms. It is possible that one of these technologies may be significantly better than others at any given point in time over the coming years. The marketplace that PenPoint computers will create for handwriting recognition systems will stimulate competition among developers, which will inevitably lead to the development of more powerful systems.

The capability of replacing GO's handwriting recognition system with other systems will also enable computer manufactures to bring PenPoint computers to market in countries that require different character sets (for example, systems that support various European or Asian language character sets). It is also possible to add support for different symbol sets, enabling applications to be developed that recognize handwritten mathematical symbols or musical notation.

Users of PenPoint computers need not worry that they will be locked in to obsolete handwriting recognition technology. GO has an active program to recruit and support the efforts of independent third party developers of

handwriting recognition systems. Several developers are currently in the process of modifying the interfaces of their systems to be installable in PenPoint. For example, Nestor, OCRSystems, and Paragraph have all announced their support of PenPoint. As we go to press with this document in September of 1991, several vendors expect to demonstrate their recognition technology, running under PenPoint at Fall Comdex.

Summary

PenPoint is currently available with GO's handwriting recognition system. This system is highly accurate as measured by a wide array of real-world metrics and methods, and it is rated as acceptable for a wide range of tasks by most users tested. GO has had an ongoing serious research effort to focus on real-world use of handwriting recognition systems since 1989.

GO's User Research group will continue performing research to improve the effectiveness of GO's handwriting recognition system over time, as well as evaluating systems from other independent developers. PenPoint users, ISVs, and licensees can be assured that the best handwriting recognition system for their application will be available to them because it is a feature of PenPoint that handwriting recognition systems can be replaced. In addition, GO has an active program of supporting third party developers' efforts to port their recognition systems to PenPoint.

Finally, when evaluating an operating system for pen computers, it is important to keep the significance of the handwriting system in the proper context. Many applications require little (or very limited) handwriting recognition. PenPoint is a new operating system that offers developers and users a great deal of valuable, innovative functionality and significant benefits in addition to handwriting recognition.

###