-- --

IEEE P1003.0 Draft 13 – September 1991

4.6 Windowing System Services D

Responsibility: Marti Szczur and Ruth Klein

4.6.1 Overview and Rationale

Human user interface standards are considered in this clause. The user interface is a key component of all computer systems that support direct human-machine interaction. Until recently, most computer operating systems interpreted commands that were typed in from the keyboard of an alphanumeric computer terminal. Special purpose applications, such as those for CAD/CAM, have always presented user interfaces based on series of menus or pointing at visual displays with tablets and lightpens. The availability of low-cost bitmapped graphic workstations and personal computers has lead to the proliferation of graphical user interfaces (GUI s), windowing technologies, generic commands, and an assortment of selection techniques (e.g., mouse, track ball, tablets). In several of these technologies de facto standards are emerging and becoming informally accepted by the user community, and with more frequency, mandated for use in systems being developed within government agencies and private industry. The primary motivations for considering human/computer interaction standards and their relation to POSIX standards include:

— The existence and popularity of windowing systems

— The requirement for development of applications that take advantage of the windowing system environment

— The requirement of many users and manufacturers for a basic consistency in the presentation and behavior of human computer interaction across multiple graphics platforms

As the windowing system technology evolves within the graphics environment, the differences between windowing services and graphic services becomes less distinct. The distinction for purposes of this document is that graphic services are associated with providing general purpose interfaces for creating virtually any kind of two- and three-dimensional graphics (e.g., GKS for 2-D and PHIGS for 3-D). HCI services certainly utilize graphic technologies, but are limited to providing graphics related to window-based user interfaces and specifications on how human users may interact with an application within a window environment. The graphic services are addressed independently in 4.7.

Performance metrics are used in the specification of HCI components. It is intended that (eventually) a set of metrics that will be sufficient to characterize all of the services defined above.

— Screen width—effective horizontal width of screen

— Screen height—effective vertical height of screen

— Screen depth—effective depth of screen (i.e., bit width, gray scale)

— Screen resolution—size in units of 10^6

— Application display latency—time to satisfy a display request measured in units of milliseconds; the interval begins with release of the request from the (possibly remote) application. The interval ends with completion of requested service.

4.6.2 Scope

Standards and standards initiatives in the user interface area cover a wide area, ranging from keyboard layout to screen management. In this clause, the following specific standards are considered:

— Protocols for window management on a local or remote display device

— Application Program Interfaces (API) for such protocols

— HCI drivability features that define a common subset of ''look and feel''; i.e., appearance, screen positioning, and behavior of human/computer interaction objects within windows on a graphic screen

— Language-independent functional specifications and appropriate associated language bindings for the display, manipulation, and management of interaction objects within windows on a graphic screen

— Command-language interface that may be entered interactively by the human user or retrieved from a stored procedure.

— Language-independent functional specifications and appropriate associated language bindings required to support character (non-bitmapped) terminals.

— Language-independent functional specifications and appropriate associated language bindings for the translation, manipulation, and management of command statements (or messages).

Standards relating to the following are not considered:

— Graphics; see 4.7.

— Keyboard layout (out of scope for HCI Services)

— Network transport protocols; see 4.3.

— Hardware device interfaces (out of scope for HCI Services)

4.6.3 Reference Model

This subclause identifies the entities and interfaces specific to the construction of an HCI architecture. This architecture is consistent with, and extends the architecture of, Section 3. As illustrated in Figure 4-14, the interface components involved in the user interface process are divided into two groups, called the external environment interface (EEI) and the application program interface (API).

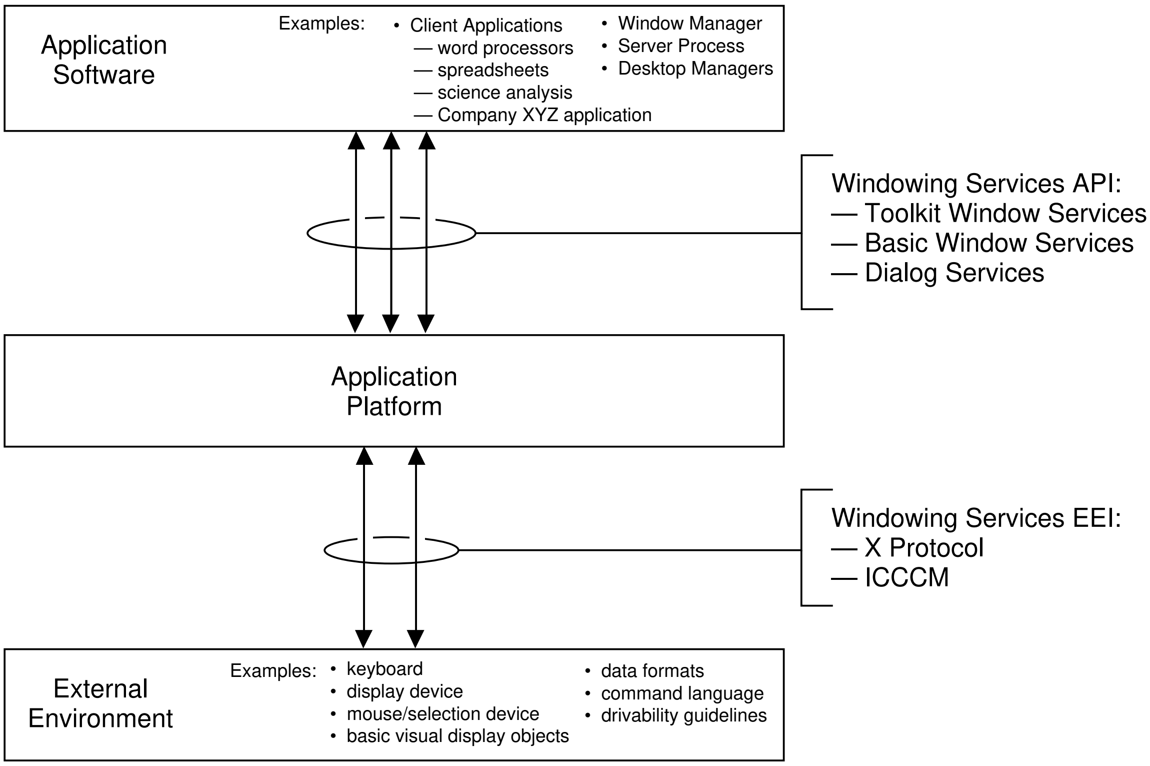

Figure 4-14 – Windowing Reference Model D

The EEI is concerned with the communication with the user via the physical HCI devices (e.g., keyboard terminal, mouse, display screen). The applicable EEI standards are driven primarily in support of user and data portability across different application platforms. standards and guidelines are intended to define a minimal set of commonality in human user interfaces, which will eliminate problem areas such as:

— Error provoking inconsistencies

— Misleading expectations about the results of user actions

— Gross inconsistencies in the high level user model or metaphor

— Incompatible motor control tendencies

The drivability concept derives its name from the concept of ‘‘driving’’ an interface. A frequently cited analogy is the automobile. Having a standard location for the clutch, brake, accelerator pedals, ignition key, and steering wheel allows a driver to move between car models with relative ease (until he/she has to roll down the window, turn on the lights or windshield wipers!) Similarly, the EEI drivability guidelines will provide standards for human user interfaces that will ensure ease of moving between application platform models. For example, which mouse click causes an interaction object (e.g., radio button) to be selected or how a scroll bar should behave would be candidates for standard EEI specification.

The API is concerned with the interface between the application semantics and the HCI services. It is the interface between the application software and the application platform and is defined primarily in support of application portability. These services provide functions for creation and manipulation of visual display objects such as menus, buttons, scrollbars, and dialog boxes. In addition, these functions allow information about user actions to flow back to the application software; for example, when the user has selected an item from a menu. This information about user actions is known as an event. Applications that require communication with the human user are inherently event-driven. That is, associated with an application's dialog window (i.e., a window in which a user response is expected) is a main event loop waiting for the user to make a selection that will trigger an operation to be performed by the application.

The API will support a specific user interface policy, which will define the application's ''look and feel.'' Although the specific look and feel need not be standard across application platforms (i.e., different implementations of the API may have unique styles) the API definition shall ensure that the application software can be ported across POSIX platforms; and the API shall support the EEI drivability guidelines, enabling human users to easily operate the application across platforms.

Elements of the HCI Architecture are Application Software elements, Application Program Interface (API) elements, and External Environment Interface (EEI) elements. These elements are linked by the use of common concepts and definitions associated with the HCI entities, interfaces, services, and standards.

4.6.3.1 Application Software Elements

Application Entity Elements include:

(1) Window System Server

The Window System Server provides a function that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Applications and other programs that use basic windowing services are called ''clients.'' Many clients may talk to the same server. All application requests to write to the screen must go through the server via the basic windowing services. The server is independent of operating system, programming languages, or network communication.

(2) Window Manager

A Window Manager provides a uniform method for manipulating windows, which includes a basic set of window management capabilities that allow for development of alternative and/or user-preferred window managers. Required human user capabilities shall include but are not limited to:

- — Resize window
- — Move window
- — Push/pop window to top/bottom
- — Shrink window to a reduced visual representation of window (i.e., frequently referred to as an icon of the window)

(3) Local and Remote Applications

These applications are clients that provide the functions required to perform the specific task(s) that the human user needs to achieve (e.g., spreadsheets, scientific analysis systems, CASE tools, process and control tasks.)

4.6.3.2 Application Program Interface (API) Elements

The API are language binding specifications that define the services available to the application programmer. API Elements are: basic window services, toolkit window services, and dialog services.

4.6.3.3 External Environment Interface (EEI) Elements

The EEI elements are specifications (and in some cases, aspects of physical objects) that define how the application platform interacts with the external world. Note that application software, as defined here, interacts with the outside world only via the application platform.

External Environment Interface Elements include:

- — Display Device Specifications
- — Data Protocol Format
- — User Drivability Guidelines (e.g., ‘‘look and feel’’ of window interface)
- — Keyboard Device Specification
- — Selection Device Specification (e.g., mouse, graphics tablet, touch screen)
- — Command-language Definition (syntax and semantics guidelines)

4.6.4 Service Requirements

Human-Computer Interaction (HCI) Services provide a controlled interface between the application-specific software and the user-interface-specific (i.e., HCI) software, allowing each to be designed and implemented separately. Users of these services include all POSIX system users and those charged with maintaining the processor and human interface communication. A common, standardized human/computer interaction for applications shall be available to users across all POSIX Open Systems Environments.

Services shall support raster (i.e., bitmapped) graphics displays. Methods for supporting vector graphics displays can be addressed, but are not mandatory.

4.6.4.1 Application Program Interface Services

Application services include those services made available to the application developer to separate the application functions from the HCI functions as much as possible. They include such areas as screen management, windowing, and user input device services.

These standard services support requirements for application portability, software commonality, application interoperability and data communications transparency.

A programmer shall have access to the following services from an application via language bindings.

4.6.4.1.1 Basic Window Services

The basic window services, callable from client applications, support a window-based user interface. They should be based on a ‘‘client-server’’ model. The server is a program that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Many clients may talk to the same server. All application requests to write to the screen must go through the server via the basic windowing services.

The major functional areas are:

- Window Management
- Presentation Management
- Event Handling
- Error Handling
- Interclient Communications
- Input Device Management: Keyboard, Pointing Device
- Screen Management
- User Preferences Management
- Server Connection Management

The following functions are available under each functional area.

Window Management

Functions available for Window Management are:

- Create a window, map a window onto the screen, delete a window (includes support for character-based emulator window)

— Manipulate a window (move, resize, change view precedence)

— Manipulate window attributes (set, get, change; attributes may be related to appearance, redraw performance, event handling, or change authority)

— Seize and relinquish control over the Server for display purposes (permits uninterrupted client output; output requests from other clients will be queued and displayed later)

Presentation Management

Functions available for Presentation Management are:

— Associate data with a window (context manager functions and association table functions)

— Manipulate the graphics context for a given object (create a graphics context, obtain current graphics context, change graphics context, etc.)

— Get and set fonts (load font, list fonts, unload font, etc.)

— Draw graphics primitives (draw arc, draw line, fill rectangle, clear rectangular window, clear entire window, etc.)

— Manipulate window cursors (create, destroy, assign, change, etc.)

— Draw text and obtain text metric information

Event Handling

The basic window services support application requirements to respond to the human user's actions, rather than forcing the human user to respond to the application in a rigid, serialized manner. This requirement necessitates that a program either (1) be capable of handling any one of a number of events at any single point in time, or (2) attach a routine to each event to be called automatically when that event occurs. There is a separate set of events for each window used by the application. An application selects the events for a particular window, maps the selected events to the window, and reads events from the event queue as they occur. There are three major types of events:

— Input device events (button press event, keypress event, etc.)

— Window management events (window exposure event, colormap event, etc.)

— Client message events (selection data transferred (by another application) event, private interclient communication event, etc.)

Functions available for Event Handling are:

— Select events

— Map events to a window

— Get information about events

— Send events

Error Handling

Functions available for Error Handling are:

— Get error message

— Get error description

— Set error event handler routine

Interclient Communication

The basic window services are required to be network transparent to an application or client. This means that an application on one host may write to the display screen connected to another host without being aware that networking is involved. The basic window services handle the network connections and follow the protocols necessary for the application to interact with the display. This convention allows redistribution of applications in a networked system with no effect on the application software. Therefore, an application client cannot assume that another client can open the same files or seize the same processing environment. Interclient communication via the server has three forms:

— Properties

Clients may associate arbitrary information with a window; generally used for communication between a client and the window manager.

— Selections

Selections are selected by the user out of one client's window, then ''sent'' to another client and displayed in the second client's window.

— Cut Buffers

Cut Buffers are a specialized form of communication. It is possible to receive notification when a cut buffer (property) is set.

Functions available for Interclient Communication are:

— Manipulate window properties (list, delete, change, get, etc.)

— Set and get selections

— Manipulate cut buffers

Input Device Management

Functions available for Input Device Management are:

— Receive keyboard input and pointing device button events

— Gain exclusive control of keyboard or pointing cursor

— Track the pointing cursor

— Change Server-wide keyboard mappings

— Set and get keyboard and pointing device preferences

Screen Management

Functions available for Screen Management are:

— Manipulate color using colormaps (copy, change, install, deinstall, get default, etc.)

— Get, display, and manipulate bitmapped screen images

— Screen saver functions (blanking screen on idle)

— Retrieve display information (default colormap, number of display planes, screen width and height, etc.)

User Preferences Management

The services and data structures used for managing user preferences are provided and collectively referred to as User Resource Management. There may be up to four sets of options that need to be read and merged:

— The human user's defaults stored in the root window's user resource manager property

— The human user's defaults stored in a user's defaults file

— The application program's defaults

— The command line arguments

Functions available for User Preferences Management are:

— Set and get preference data

Server Connection Management

Functions available for Server Connection Management are:

— Control access to the Server [add host to the access control list (ACL), list ACL, disable ACL, etc.]

— Connect and disconnect a client from a Server (and the display controlled by the Server)

— Obtain Server implementation information

— Flush output buffer to Server and wait for Server to process all events in the output buffer

4.6.4.1.2 Toolkit Window Services

The Toolkit Window services provide a mechanism for runtime access to a library of visual objects. A visual object is a graphical display object (i.e., interaction object) with associated software that receives input from human users (typically via a keyboard and a pointing device) and communicates with applications and other visual object software. The graphical representation of a visual object can be modified to reflect the results of application processing. Examples of visual objects are graphical push buttons, check boxes, and editing boxes. (Note: The term used within the X Window System community to define visual objects is ‘‘widgets.’’)

Toolkit Window services are provided for two reasons:

— To allow application software to directly utilize a visual object library

— To allow application-specific visual objects to be created and added to the widget library (Note: creating a visual object includes writing software that uses the Toolkit services)

Therefore, Toolkit services may be logically divided into two categories, with some overlap: Visual Object Interface Services, which are called by an application or dialog service, and Visual Object Programming Services, which are called by the visual object software.

An application may use Toolkit Window services to:

— Perform toolkit initialization/exit

— Set up visual object resources

— Create/delete a visual object

— Display a visual object

— Add/remove application-specific routines to be called by a visual object (event callbacks)

— Retrieve/modify the state of a visual object

— Turn control over to the toolkit for user input processing

A visual object software program may use Toolkit Window services to:

— Manage child visual objects (a child visual object is a visual object that is displayed inside of another visual object)

— Manage window events, timer events, and file input events

— Handle visual object geometry (sizing, positioning, child visual object placement)

— Handle user input

— Manage visual object resources

— Translate an event into an action

— Manipulate graphics contexts

— Manipulate pixmaps (pixel arrays—used to display a graphical object by turning pixels on and off)

— Manage memory associated with HCI

— Handle errors associated with HCI

— Allow inter-visual object communication (via the selection mechanism)

— Initiate other visual object routines (visual object event callbacks)

— Initiate application-specific routines that have been associated with the visual object by the application (application event callbacks)

4.6.4.1.3 Dialog Services

Dialog services provide functions to support high-level HCI management for applications with the primary goal of delivering human user inputs to the application program and application-driven information to the human user. The dialog services allow for a separation of the user interface specifications from the application program. For example, there are many applications that are not concerned with whether a user entry object is a pull-down menu or a scrollable list. These applications are only interested in what the user specified or selected from the user entry object (i.e., the parameter value), which will then trigger some action by the application. To support this notion, a single dialogue function might be specified for displaying a window made up of a composite of visual display objects, such as radio buttons, text key-in objects, and scrollable text lists. The application program does not need to manage or understand what the look, location, or visual feedback of any of these items will be. The dialog function has access to the presentation information required to display the specified window and it handles the display of the application specified window. Another dialog service would provide a high-level event loop that returns the user specified input as an application parameter value.

The services provide simplicity over the degree of freedom available in Basic and Toolkit Window Services. Most User Interface Management Systems (UIMS s) provide dialog services to fulfill their requirement of separation of user interface from application software.

These services are subdivided into:

— Window services: provide services used to initialize the window service, create and delete windows with predefined associated visual objects, and manipulation of the pointing cursor. They include services that allow the application to communicate directly with the human user via modal or modeless windows.

— Visual object manipulation services: provide services used to access the HCI designed by the application HCI designer, display the visual objects defined by the HCI , and associate them with application-tied inputs and outputs.

— Event control services: provide services to allow the application to define a set of events and handle triggered events in one of two ways:

(1) Wait on the occurrence of any event, processing triggered events one at a time from an input queue (event-driven method)

(2) Attach to each event a function that is automatically executed when the event is triggered (callback method)

4.6.4.2 External Environment Interface Services

These services provide support for the actual elements with which the human user physically interacts. These functions provide services in three areas:

- HCI : provides definition of the presentation and behavior of the visual display objects, command language definition (syntax and semantics), specifications related to keyboards, selection devices, audio and video input/output devices.
- Information Interfaces: provides specification of user resource data formats, containing presentation and action information pertaining to visual display objects.
- Network Interfaces: provides protocol services for data transport, which is basically the bottom six layers of the OSI model

4.6.4.3 Interapplication Entity Services

These services provide support for general conventions and specifications for interaction between HCI components. The services provide support for generalized network/multisession services, such as message handling between HCI components, intermediate language definition, and a standard definition of the format used for saving the presentation, behavior, and action information about HCI objects.

4.6.4.4 Windowing Resource Management Services

These services provide general management functions across the HCI components, which include system administration-oriented functions (i.e., management of human/computer interfaces within the scope of the administrator, such as setting up defaults and human user customization functions. For instance, it is important to allow reconfiguration and addition of terminals and displays without affecting the application interface.) These resource management services are independent from the OLTP Resource Management Services defined in 4.10.4.4.

4.6.5 Standards, Specifications, and Gaps

Standards that relate to the user reference model presented earlier are considered here. Related standards that might be relevant for one or more of the interface components will also be mentioned.

4.6.5.1 Current Standards in the POSIX OSE

No current international or national standards exist for the HCI Services, primarily due to the recent emergence of the windowing technology. However, several standard activities are underway and referenced under 4.6.5.2.

There is a de facto standard for the base window system. The X Window System windowing protocol and the Xlib functional interface to the protocol were developed at Massachusetts Institute of Technology. Development is continuing under the aegis of the X Consortium, a group of interested parties in the computer industry and computer manufacturers. Relevant documents from the X Consortium are ‘‘XWindow System Protocol, X Version 11,’’ ‘‘Xlib – C language X Interface,’’ ‘‘X Toolkit Intrinsics – C Language Interface,’’ and ‘‘Bitmap Distribution Format 2.1.’’

The X Window System protocol and functional interface are considered to be de facto standards in the base window system area because of their widespread adoption by major computer vendors and industry groups.

Within the government, the National Institute of Standards and Technology (NIST) issues Federal Information Processing Standards (FIPS) that require purchases made by the United States Government to adhere to certain standards. NIST has adopted the X Window System Version 11 Release 3’s X Window System protocol, Xlib, Xt Intrinsics, and Bitmap Distribution Format as FIPS 158. This is a noncompulsory (i.e., voluntary) standard.

Table 4-9 – Windowing Standards

Service	Specification	Subclause	
Basic Window Services	de facto: X Window System	4.6.5.?	D
	emerging: ANSI X3K13.6		D
Toolkit Window Services	de facto: X Window System	4.6.5.?	D
	emerging: ANSI X3K13.6		D
	emerging: IEEE POSIX. 2 {*ref*}		D
	emerging: IEEE POSIX .1 {2}		D
Dialog Services	Gap	4.6.5.?	D
EEI Services	emerging: ANSI X3V1.9	4.6.5.?	D
	emerging: ISO/IEC JTC 1/SC18/WG19		D
	emerging: ANSI HSF-HCI		D
	emerging: ISO TC159/SC4/WG5		D
	emerging: P1201.2		D
Interapplication Entity Services	Gap	4.6.5.?	D
Window/Character Resource Management Services	Gap	4.6.5.?	D D

4.6.5.2 Emerging Standards in the POSIX OSE

MRS : This subclause still needs work.

— ANSI X3K13.6. Currently developing an X Protocol standard.

— ANSI X3V1.9. User-System Interfaces and Symbols: Working on a ISO/IEC Standard 9995, Keyboard Layouts for Text and Office Systems. Also working on the Voice Messaging User Interface Forum (VMUIF). This is a mirror standards effort with ISO/IEC JTC 1/SC18/WG19.

— ISO/IEC JTC 1/SC18/WG19. User-System Interfaces and Symbols. Working on developing standards for user interfaces and symbols associated with text and office systems.

— ANSI HFS-HCI. Working on drafts on the design process, information presentation, forms-based dialogs, and window-based interaction.

— ISO TC159/SC4/WG5. Software Ergonomics and Man-Machine Dialog: Working on developing parts of the ISO Standards 9241, Ergonomics of Visual Display Terminals. Their areas of concentration are software ergonomics, dialogue principles, dialogue styles, methods for evaluating software usability, coding and formatting of information, and terminology

— IEEE P1201. Application and User Portability: Chartered to develop standards that facilitate application and user portability in the X Windows environment. P1201.1 is involved in defining a set of virtual toolkit services that would be independent of any windowing system. P1201.2 is involved in defining drivability guidelines.

— ANSI CODASYL. Working draft available for Forms Interface Management Systems (FIMS), which covers the interface between a programming language and any form fill-in application on a computer or terminal screen.

4.6.5.3 Gaps in Available Standards

— Object Definition File Format: There are no standards addressing the format used for describing the ‘‘look and feel’’ of HCI objects.

— Toolkit Services

— Dialog Services

— Interapplication Entity Services

— Window/Character Resource Management Services: A standard definition of the format used for saving the presentation, behavior and action information about HCI objects would provide a vehicle for exchanging HCI information between software tools, such as User Interface Management Systems (UIMSs) and Interface Design Tool (ITDs).

4.6.6 OSE Cross-Category Services

4.6.6.1 Security

This section should address the security aspects of HCI and would include:

— Authentication of person at login

— Authentication of person when a service request is made to a client application

— Provisions for visual labeling of sensitive material

— Option selections available in support of sensitive activities

— Prevention of moving data (cut/past) from a more protected security environment to a less protected environment

4.6.7 Related Standards D

Currently, the basic windowing services provide a certain level of graphics functionality, but the existing and proposed graphics standards (e.g., PHIGS , GKS) provide a much more comprehensive solution to graphic support. As the graphics and windowing technologies evolve, this distinction between the windowing and graphics services will continue to be blurred. For instance, proposals are already being developed that provide extensions to the X Window System that support 3-D graphics (i.e., PEX , PHIGS EXtensions), and implementations of GKS are currently available that use the X Window System to create the graphics.

4.6.8 Open Issues

— Audio input/output

— Video input/output

— Security

— Desktop

The Desktop, or HCI graphical shell, is a specification for the HCI work surface (i.e., the whole display screen).

The desktop provides the user with a visual interface to available computer resources. A desktop may be characterized as a visual analog of the POSIX shell. It provides access to system resources, such as devices and files, and provides methods to start applications. Desktops typically also provide a set of often used utilities such as a calendar, a notepad, etc. The desktop is an important component of the look and feel of a human user interface, but the current state of the industry is too immature for any standardization to materialize on a desktop specification in the immediate future.

NOTE: There are some valid arguments for defining some requirements for standards at this level. The goal is to enable a user to easily go between application platforms and operate common functions in a similar manner.

4.7 Graphics Services

Responsibility: John Williams

HLJ : This is a full clause replacement for Draft 13. It is not further diff marked. D

4.7.1 Overview and Rationale

Graphics Services are key components and play an important role in the POSIX Open Systems Environment as it is used today in many different areas of industry, business, government, education, entertainment, and most recently, the home. The number of applications is growing rapidly, with increasing graphics capabilities. Some of these areas are user interfaces, computer-aided drafting and design, electronic publishing, plotting, simulation, animation, scientific visualization, art, and process control. The use of pictorial graphics provides a more intuitive interface and thus facilitates man/machine interaction.

Graphics has become a routine part of most organizations today, ranging from hardcopy graphs and charts to user interfaces and complex 3-D visualizations incorporating video and sound. The graphics technology of rendering objects has become dramatically more realistic and hence is used by engineers, architects, artists etc., to enable them to see precisely what their final products, whether automobiles or buildings, will look and behave like under real-world conditions.

Graphics has allowed dramatic improvements in the ''look and feel'' of user interfaces and the trend is towards increasing use of these interfaces to interact with computers graphically, via windows and icons and this reduces the time involved in learning to use a computer.

Standardization of graphics services has many benefits for application developers, users, and systems integrators. The underlying motivations for considering graphics standards and their relation to the POSIX Open Systems Environment include:

(1) **Portability:** In order to protect investment and achieve independence from a particular technology and a particular supplier of technology, portability at both hardware and software levels is necessary. There are many aspects of portability within graphics, all of which are potential money and time savers.

- Applications portability
- Graphics package portability
- Host machine independence
- Device independence
  - input devices: dials, mouse, tablets etc.
  - output devices: plotters, raster, vector etc.
- Window system independence