-- --

IEEE P1003.0 Draft 13 – September 1991

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

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

— Programming language independence

— Programmer portability

— User portability

(2) **Interoperability/Distributed Graphics:** In order to allow applications to execute on one machine and display graphics on remote display servers, standard graphics protocols are necessary. This allows for display of graphics on machines that are incapable of executing particular types of applications and it also facilitates graphics conferencing.

(3) **Graphics Data Exchange:** In order to share or exchange graphical information between diverse applications, standard graphics data exchange mechanisms are necessary.

This clause presents a reference model for this component and describes the services provided to application programmers and users. It also describes the current national/international standards, emerging standards, de facto standards, and any existing gaps that need new standardization efforts.

4.7.2 Scope

Included within this component are standards in the graphics area that address the following topics :

— Application Program Interface (API) Standards

— Language Bindings Standards

— Metafile and Archive Standards

— Device Independent Interface/Protocol Standards

— Computer Graphics Reference Model

— Conformance Testing of Implementations of Graphics Standards

— Distributed Graphics Standards

— Imaging Standards

— Performance Metrics Standards

The standards not addressed here are:

— Data Exchange Standards

— Graphical User Interface Standards

— Window Management System Standards

4.7.3 Reference Model

Over the past decade many computer graphics standards have been developed. While they are similar in concepts, their underlying reference models are different. This restricts the degree to which the standards are compatible. By producing a reference model to which all future graphics standards are to adhere, compatibility of graphics standards is assured.

Formal work on the Computer Graphics Reference Model (CGRM) standard is in progress within the ANSI X3H3.2 committee. It is an international standard that explains the relationships between existing graphics standards and defines relationships between standards in computer graphics and those in other areas. It will form the basis for the next generation of computer graphics standards. Broadly speaking, CGRM provides a framework within which relationships between standards can be described.

There are five types of standards in the current family:

— *Application Program Interface (API) Standards:* These define a programming interface for application programmers. GKS, GKS-3D, PHIGS, and Xlib are examples of standards in this area.

— *Metafile and Archive Standards:* These standards define representations of graphics for storage and transfer between systems. These are basically file format and file transfer encoding standards. CGM (Computer Graphics Metafile) and PHIGS Archive files are of this type.

— *Device Independent Interface Standards:* These standards define the interface between device-independent graphics systems software and one or more device-dependent graphics device drivers. CGI (Computer Graphics Interface) is the standard in this area.

— *Language Binding Standards:* API and device interface standards are functional specifications defined independently from particular programming languages. Each standard has attached language binding standards that state how the functionality should be accessed from a variety of programming languages.

— *Framework Standards:* These include the standardization of a reference model for computer graphics, conformance criteria, and the registration of graphical items.

The CGRM describes the current family of graphics standards in terms of the following four levels of abstraction:

— *Application Level:* This is the level at which applications-related information is composed into abstract graphics related to the application.

— *Virtual Level:* At this level, the graphical output to be displayed is described in terms of output primitives

— *Logical Level:* At this level, the information necessary to render a primitive on a particular device is assembled.

— *Physical Level:* This level is associated with a particular output device and a collection of input devices. The physical level need not correspond to real devices such as a pen plotter. There could be further layers of the system between the physical level and the hardware, such as the window system.

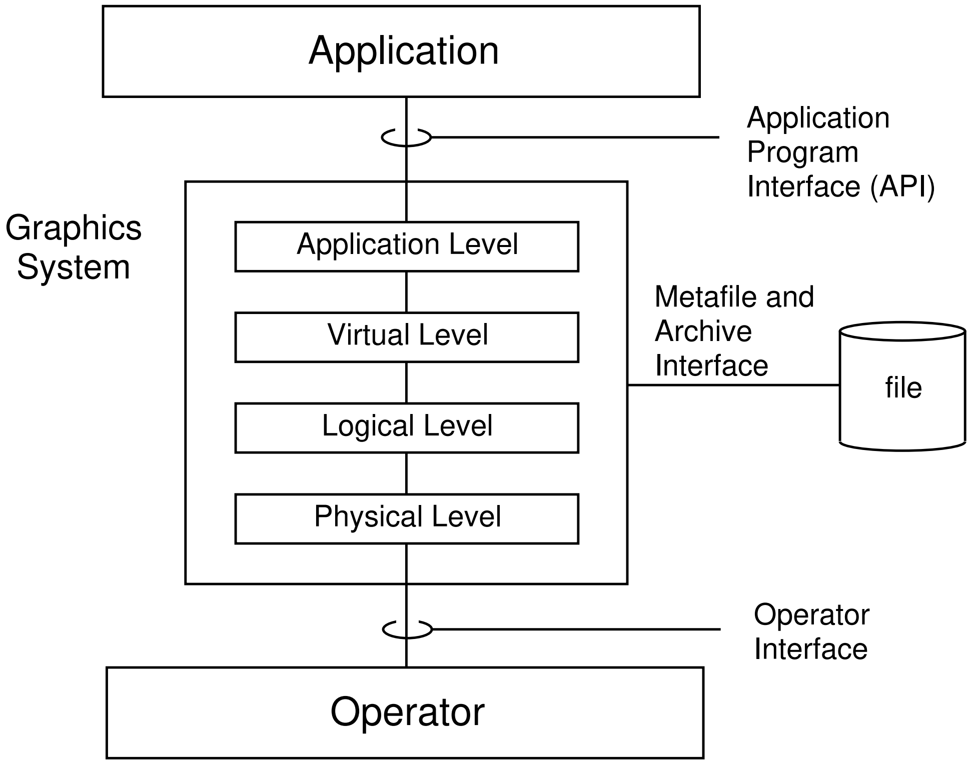

Figure 4-15 – Computer Graphics Reference Model Level Structure

The Application Program Interface (API) is the interface between the application and the graphics system. There are also interfaces to metafiles and archives and to the operator. Here the operator need not mean human operator, but the user of the graphics system; for example, the window system.

The Computer Graphics Reference Model can be incorporated into the POSIX OSE reference model as depicted in Figure 4-16. It provides the context for the discussion of graphics services and shows that the graphics services is a component of the overall POSIX OSE and is available to the the application through the POSIX OSE API.

The entities and interfaces specific to the graphics services are identified in this clause.

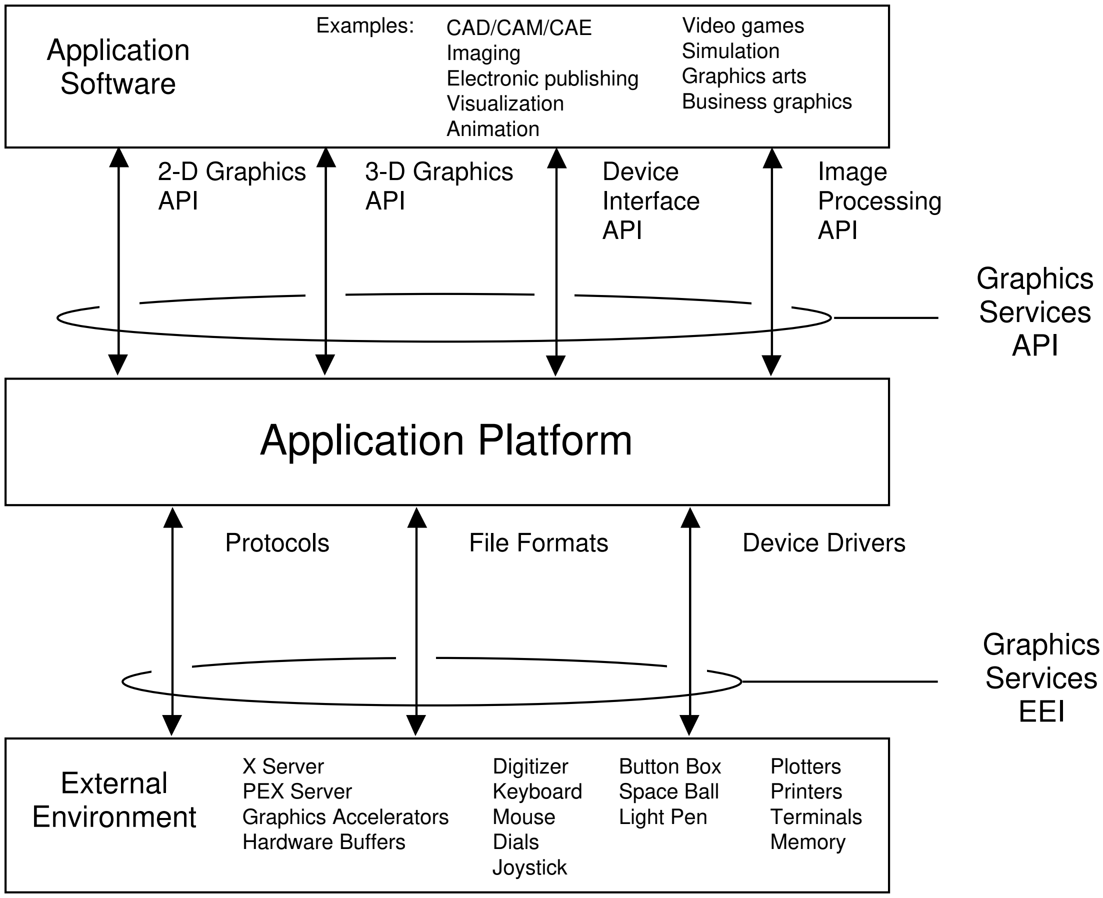

Figure 4-16 – POSIX OSE Graphics Service Reference Model

The entities are:

(1) **Application Software**, such as CAD/CAM/CAE applications, imaging applications, electronic publishing, etc.

(2) **Application Platform**, which consists of graphics libraries such as GKS, PHIGS and Xlib.

(3) **External Environment**, consisting of external entities with which the application platform exchanges information such as input devices, X/PEX servers, hardware buffers, etc.

The interfaces are:

(1) **Application Program Interface (API)**, which is the programming interface between the application and the application platform. It standardizes the conceptual model, calling sequence, functions, and syntax that a programmer uses to develop a graphics application. Each API standard has an attached language-binding standard that allows the

functionality to be accessed from a variety of programming languages. A standard API in conjunction with a standard language binding promotes application portability, by isolating the programmer from most hardware peculiarities and providing language features readily implemented on a broad range of processors. Examples of APIs in the graphics services area are GKS, PHIGS, PIK, PostScript, etc.

(2) **External Environment Interface (EEI)**, which is the interface between the application platform and the External Environment described earlier. In the graphics services area these can be device drivers that are used for communication between the device-independent and the device-dependent functions as well as protocols and file formats.

The standardization efforts in the graphics area focus on these two interfaces.

4.7.4 Service Requirements

4.7.4.1 Graphics Concepts

Computer Graphics Services can be discussed in terms of the following fundamental graphics concepts:

Output Primitives

The output primitives are the building blocks used to construct graphical objects for display or storage in an archive file. Common output primitives are:

— *Line*

— *Polyline* used to represent a series of straight lines from a set of points.

— *Marker* is a special symbol used to represent semantics of graphical objects.

— *Fill area* is an area with an edge and an interior which may be filled with a solid color or some form of pattern or hash.

— *Text* is an output primitive used to represent strings in two or three dimensional space.

— *Annotation text* is text that is always displayed facing the viewer.

— *Cell arrays* are areas with rectangular grids which can take on individual colors.

— *Triangle strip* is a set of triangles defined by a particular ordering of vertices.

— *Quadrilateral mesh* is a set of quadrilaterals defined by a grid of vertices.

— *Surfaces:* NURBS (Nonuniform Rational B-Spline)

— *Curves:* NURBS (Nonuniform Rational B-Spline)

— *Conics:* Circles, ellipses, parabolas, and hyperbolas

Primitive Attributes

Attributes of primitives determine the style of the display of the primitive. For example, lines and edges may have different line styles such as dotted or dashed, text may have different fonts, orientation, and character spacing. A polymarker may be an asterisk or a small triangle. They all may be red in color. General type attributes that apply to almost any output primitive are color, visibility, pickability, and highlight method.

Input Primitives

Input primitives or logical devices are virtual devices designed to insulate the application from the real input devices. Logical devices include picking devices, locator devices, choice devices, valuator devices, etc. In terms, of actual devices, a locator device might be associated with the first mouse button.

Input Model

The input model describes how input primitives and logical devices are related to physical input devices and the degree of control provided to the application over the devices. For example, one control choice might be how feedback is echoed to the operator when a logical locator device is attached to a depressed mouse button. The feedback might be a rectangular cursor or the highlighting of geometry as a cross-hair cursor moves over it. When the button is released the device coordinates are placed in the locator data record and an event is placed in an event queue for which the application can check asynchronously. The method the application uses to determine if a device has data for it is usually described in terms of modes. A common mode is event mode. The application waits a finite time for some event to appear in a queue. If no event comes in the finite time, the application does other processing and eventually comes back to check the queue with the wait for some event. If an event appears, the application determines what type it is and gets the data for that type of event. For a pick device, the data might be all possible graphical primitives that could intersect some aperture, possibly specified in the device coordinate system.

Coordinate Systems and Clipping

Part of the graphics services is a means to utilize various coordinate systems. Graphical output has to be described to the graphics system in terms of some coordinate system, relevant to the application and presented to the display device in terms of its own coordinate system, the device coordinate system. It is unlikely that these two coordinate systems will be the same. A graphics system may therefore involve a number of coordinate systems and hence the need to define transformations between them. Some standard types of transformations are scaling, rotating, translating, reflecting, and projection, such as parallel and perspective. They are used to manipulate objects in a coordinate system and to map from one coordinate system to another. The coordinate systems commonly used are modeling coordinates, world coordinates, view-reference coordinates, normalized projection coordinates, and device coordinates.

Clipping is the process of specifying a region in space and restricting graphical output to that region. Only those primitives that define objects in that region will have their output displayed.

Output Model

The output model is the concept of how graphics objects are created, displayed, and controlled on output devices. The output model defines how to position and organize objects on the screen, and the visual state of these objects such as visible or invisible, hidden lines removed or not removed, picture matches retained structure, picture not consistent with retained structure, etc.

More specifically, the output model concept is made up of the:

— Transformation pipeline

— Rendering pipeline

— Retained structures

— Nonretained structures

— Graphics state

— Window systems

Error Handling

TBD

Storage/Archiving

TBD

4.7.4.2 Graphics Requirements

The graphics service requirements of all users of this system can be generalized as:

— The ability to create, delete, and modify output primitives.

— The ability to specify and edit the primitive attributes globally and individually.

— The ability to transform (i.e., scale, translate, rotate, reflect, project, etc.) primitives for construction of more complex objects and for arrangement in the viewing space.

— The ability to create and manipulate a database of primitives, to define and edit attributes, to create and combine transformations, and to selectively control the display of graphics primitives.

— The ability to display graphical objects constructed in a retained database, or the ability to display primitives immediately, or to display from both a retained database and immediately.

— The ability to apply lighting and shading algorithms to collections of graphical objects with multiple light types and sources.

— The ability to prepare display data and control the timing of the actual display of the display data. On some systems this is referred to as frame buffer control.

— The ability to store and retrieve graphical objects from files.

— The ability to control input devices and retrieve data from input devices.

— The ability to direct output to a meta-file and retrieve graphics data from a meta-file.

— The ability to inquire about all aspects of the graphics environment; e.g., the state of the system at any given time, the actual capabilities of a given hardware platform, the attributes and primitives supported by a given implementation, etc.

— The ability to distribute graphics.

— The ability to control errors.

4.7.4.3 Application Program Interface Services

The major categories of graphics services available in the POSIX OSE API area include:

— 2-D graphics API services

— 3-D graphics API services

— Device interface API services

— Image processing API services

For most of these API standards there exist standard language bindings so that applications using different programming languages can access the same functionality.

The choice of which graphics standard API to use will depend on a number of factors: application profile, overall system architecture, equipment available, existing application database interaction, system performance considerations, user interface requirements, management policy, and other external factors. The aim of producing a compatible set of graphics standards in GKS, GKS-3D, PHIGS, PHIGS PLUS, etc. (described in the Standards subclause) is to allow that choice to be made in the most flexible way.

4.7.4.4 External Environment Interface Services

The major categories of graphics services in the POSIX OSE EEI area include:

— Protocols

— File Formats

— Device Drivers

The choice of which standard to use depends on a number of factors: application profile, system architecture, equipment available, system performance considerations, and other factors

4.7.4.5 Interapplication Software Entity Services

TBD

4.7.4.6 Graphics Resource Management Services

TBD

4.7.5 Standards, Specifications, and Gaps

There are several major standards existing in the computer graphics industry today, that have been approved by National/International organizations such as ANSI, ISO, and IEEE. There are also standards efforts going on in related areas such as application data exchange. These official graphics standards are complemented by de facto standards that have been accepted by the graphics industry at large. This document provides a general explanation of these standards, their specifications, and interrelationships.

4.7.5.1 Current Standards in the POSIX OSE

National/International

PHIGS — ISO 9592 Parts 1–3
Fortran Language Binding — ISO 9593-1
Ada Language Binding — ISO 9593-3
C Language Binding — DIS 9593-4

The Programmer's Hierarchical Interactive Graphics Standard (PHIGS) is a functional specification of the interface between an application program and its graphics support system. It is an ANSI/ISO standard and provides the following graphics functionality:

— A high degree of interactivity

— Multilevel, hierarchical structuring of graphics data

— Easy modification of graphics data and the relationships among the data

— 3-D, as well as 2-D, graphical input and output

— Offline storage (and retrieval) of graphics data

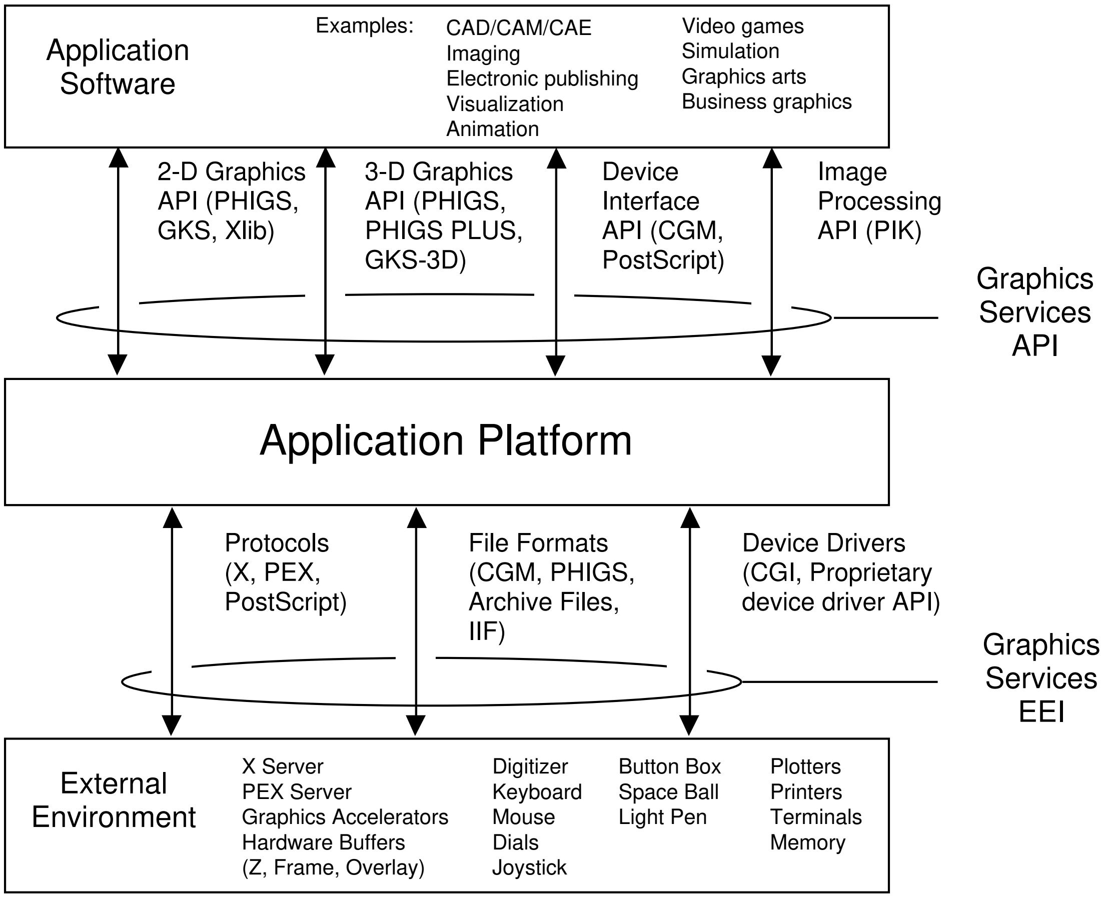

Figure 4-17 – POSIX OSE Graphics Service Reference Model Standards

PHIGS controls the definition, modification, and display of hierarchical graphics data and specifies functional descriptions of systems capabilities, including the definition of internal data structures, editing capabilities, display operations, and device control functions. PHIGS manages the organization and display of data in a centralized database, allowing programmers to define and organize graphical data in a manner most convenient to the application. Such a hierarchical approach is a big benefit and is not available in GKS, another international standard.

Objects are defined in the PHIGS graphical database by a sequence of elements, including output primitives, attributes, transformations, and invocations of other object and object part definitions. These elements are grouped into entities called structures. Structures may be related in a number of ways, including geometrically, hierarchically, or according to inherent properties or characteristics, as defined by an application.

PHIGS provides tools to use hierarchical data structures with minimal effort by the application programmer. Pictures constructed from geometric models often have a clearly evident structure. This structure can sometimes be easily seen in the repeated use of symbols, in the connections and geometric relationships between objects, or in the overall organization of a complex image. Even if the object's structure is not evident, its underlying data organization may be quite rigorous, well defined, and well understood by the application. PHIGS supports both these cases by separating the definition of graphics data from the actions required to display them.

The structured definition of graphics data inherently reduces repetition and connectivity problems. The repeated use of component objects and the relationships between them can automatically be made a part of an object's definition.

The structured definition of data allows images to share component objects, making it faster and easier for application programs to define and modify picture descriptions. Sharing component objects will also reduce storage requirements for graphics data.

PHIGS permits rapid dynamic access to a centralized graphics database. This allows PHIGS to support interactive end user application programs and, depending on the capability of the hardware, realtime definition, and modification of graphics data. PHIGS is capable of performing three-dimensional modeling transformations, workstation transformations, and viewing. It also handles two dimensions through a shorthand functionality of three dimensions. In workstation transformations, PHIGS provides another level of display control after the viewing operation that can isolate a section of an image for pan and zoom operations.

The National Institute of Standards and Technology (NIST) has developed a test system to help determine whether implementations of PHIGS conform to the specifications of the ANSI standard X3.144. The PHIGS Validation Test (PVT) suite consists of highly portable Fortran programs which examine test conditions and report the results.

PHIGS PLUS — DIS 9592-4

PHIGS Plus Lumiere Und Surfaces (PLUS) specifies a set of extensions to PHIGS that addresses some of the deficiencies in the graphics functionality provided by PHIGS. PHIGS does not include ''higher level'' primitives such as curves and surfaces, and techniques for lighting and shading. Recognizing this, an ad hoc working group was formed to propose a set of extensions to PHIGS to enable these capabilities to be addressed in a standard manner, compatible with the overall philosophy of PHIGS. This set of proposed extensions was submitted to ISO and has since been developed into PHIGS PLUS. PHIGS PLUS enhances PHIGS by providing:

- Primitives for defining curves and surfaces
- Lighting models
- Shading of surfaces
- Depth cueing
- Color mapping and direct color specification

PHIGS PLUS is not an international standard yet and is currently at the stage of committee draft.

GKS — ISO 7942; FIPS 120
Fortran Language Bindings — ISO 8651-1
Pascal Language Bindings — ISO 8651-2
Ada Language Bindings — DIS 8651-3
C Language Bindings — DIS 8651-4

GKS Information Bulletin

The Graphical Kernel System (GKS) is a 2-D graphics system and provides no support for 3-D. It is a 2-D graphics API that shields the programmer from differences among various computers and graphic devices. It allows for portability of graphics applications by standardizing the basic graphic functions and the method and syntax for accessing these functions.

GKS is an ANSI, ISO standard and is widely used today. It has standard language bindings for Fortran and Pascal. Language bindings for C, Ada, and LISP are currently being worked on.

GKS supports the grouping of logically related primitives such as lines, polygons, strings, and their attributes into collections called segments, which cannot be nested.

GKS supports many graphical input and output devices such as black/white and color displays, printers, plotters, mice, data tablets, joysticks, and digitizers.

GKS-3D — ISO 8805
Fortran Language Bindings — DIS 8806-1
Pascal Language Bindings — CD 8806-2
Ada Language Bindings — DIS 8806-3
C Language Bindings — DIS 8806-4

Graphical Kernel System for Three Dimensions (GKS-3D) is an ISO standard and specifies extensions to GKS for defining and viewing three-dimensional wire-frame objects. In addition, the GKS input model has been extended to provide three-dimensional locator and stroke input. GKS-3D allows the operator to obtain information from three-dimensional input devices and to perform hidden line/hidden surface removal (HLHSR) at the workstation. It does not, however, provide specific functions for controlling rendering techniques such as light source, shading, texturing, and shadow computations that must be done locally at the workstation. Conceptually, all workstations are three-dimensional in GKS-3D, which is made possible by shielding the hardware peculiarities as in GKS.

CGI — DIS 9636 Parts 1−6
Fortran Language Bindings — DIS 9638-1
C Language Bindings — CD 9638-4

The Computer Graphics Interface (CGI) specifies a standard functional and syntactical specification of the control and data exchange between device-independent graphics software and one or more device-dependent graphics device drivers. Unlike the graphics standards discussed earlier, CGI specifies an interface at the device-driver level, rather than at the application level.

Unlike CGM, which only handles graphical output, CGI handles both input and output, which makes all devices appear as identical, virtual graphics devices. Therefore, this protocol is also known as the Virtual Device Interface (VDI). It provides a standard graphics escape mechanism to access nonstandard graphics device capabilities. CGI allows programmers to write portable device-driver software that is independent of the physical graphics device characteristics. This makes the software portable and compatible with a wide variety of devices.

CGM — ISO 8632 Parts 1−4

The Computer Graphics Metafile for storage and transfer of picture description information (CGM) is a mechanism for retaining and/or transporting graphics data and control information. This information contains a device-independent description of a picture at the level of the Computer Graphics Virtual Device Interface described above. It provides a standard graphics escape mechanism to access nonstandard graphics device capabilities via the metafile.

Pictures are described in CGM as a collection of elements of different kinds, representing, for example, primitives, attributes, and control information. It is multipart ANSI, ISO standard. Part 1 contains the semantics of all the elements. Parts 2, 3, and 4 contain the syntax of three different bindings of the standard, namely: character-coded, binary, and clear-text encodings.

PHIGS Archive files — ISO 9592 Parts 2−3

Parts 2 and 3 of the PHIGS standard define an archive file format for storage and transfer of PHIGS structures and structure network definitions from the CSS (Central Structure Store). Part 2 describes the file format and Part 3 a clear text encoding. This encoding is constructed using the same techniques as used by CGM.

4.7.5.2 Emerging Standards in the POSIX OSE

IPI — JTC 1# 1002

Image Processing and Interchange is a functional specification and several language bindings for an Application Programmer Interface to Imaging. The standard defines the data objects, primitive operations, and a reference model. The API supplies the basic building blocks upon which applications requiring imaging functionality can be built within conventional, distributed, and image oriented computing environments.

The International Standard for Image Processing and Interchange includes three parts:

Part 1 Common Imaging Architecture

Part 2 Programmer's Imaging Kernel (PIK)

Part 3 Image Interchange Format

PEX — PHIGS Extensions to X

PEX is a network protocol extension to the X Window System. As many applications require 3-D graphics and other forms of input devices such as dials and button boxes, all of which are not supported by X, it became necessary to extend the X Protocol to include 3-D graphics. PHIGS was selected as the application program interface because of its acceptance as a 3-D standard, its high degree of input ability, and its powerful database editing capabilities. In 1988, the MIT X Consortium contracted to add 3-D and extended input extensions to the X protocol and the first release of PEX as a sample implementation (PEX-SI) was made in January 1991 but is not yet available commercially. Using PEX, PHIGS workstations would be defined as X Windows. For the programmer, X, PHIGS, and PEX standards provide portability.

Conformance Testing of Implementations of Graphics Standards — DIS 10641

The existence of any standard brings up the question of how one can be sure whether a product claiming to conform to the standard does in fact conform. If this question is not addressed then the process of standardization becomes pointless.

The general approach to software validation is through testing. The method is to subject the software to a collection of test cases and observe the results. If the results are different from what is expected, the software does not conform to the specification. The ANSI X3H3.7 committee is working on a standard that specifies the characteristics of standardized test sets for use in determining the conformance of implementations of graphics standards. It will also provide guidance to functional standards developers concerning the content of their standards and the conformance rules within standards.

4.7.5.3 Gaps in Available Standards

(1) Applications have different behaviors for similar functions which hinders user portability. By adopting a uniform approach (Graphics_Style_Guide) users can switch between applications without a lot of training.

(2) Current existing standards allow a wide interpretation for implementors of the standards thus denying the applications useful controls. In order to achieve true portability in a distributed environment, applications will need control and deterministic functionality.

(3) How window standard fits into CGRM

(4) Current existing standards do not address solids.

(5) The ability in a standard defined way to perform cut and paste between applications.

(6) Current standards do not allow nonretained graphic methods to do lighting and shading.

4.7.6 OSE Cross-Category Services

Not applicable.

4.7.7 Related Standards

IGES , NBSIR 86-3359

See 4.5.

X Window System Data Stream Definition Parts 1-4

(Being worked on in ANSI X3H3.6)

Part 1: Functional specification
Part 2: Data Stream Encoding
Part 3: KEYSYM Encoding
Part 4: Mapping onto Open Systems Interconnection (OSI) Services

The X Window System is a network based windowing and 2-D graphics system. It uses the client-server model. The client and server can reside on the same or different platforms. The client is an application program executing anywhere on the network and displaying on the screen. It does this by making calls to a library called Xlib to generate protocols. The X server is the software that accepts protocols sent by the client and processes them for display. It also accepts input from a mouse or keyboard for return to the application program. The X protocol specifies the data stream encoding between the server and the clients. The X Protocol originally developed by the X Consortium at MIT, is being standardized by the ANSI X3H3.6 committee. The