DKUUG Dansk UNIX-system Bruger Gruppe
Danish UNIX systems User Group

Årets X-event

Odense 29-30 sept. 1993

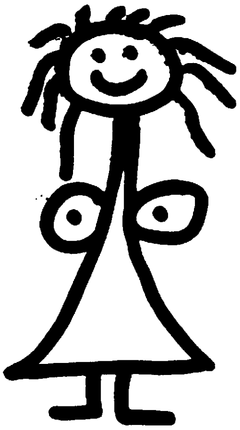
Bd. 1

Sekretariat:
Københavns Forskerby Symbion
Fruebjergvej 3, DK-2100 Kbh. Ø
Giro: 1 37 86 00
Telefon: +45 3917 9944
Telefax: +45 3120 8948
Email: sek@dkuug.dk

Formand (Chairman): Keld Simonsen
Telefon: +45 - 33 13 00 23
Email: keld@dkuug.dk
Netpasser (Backbone manager):
Telefon: +45 3917 9000
Telefax: +45 3917 9897
Email: netpasser@dkuug.dk

Tilknyttet *(Affiliated):*
The European Forum for Open Systems
EurOpen
UniForum
The International Association of UNIX Systems Users

X11R



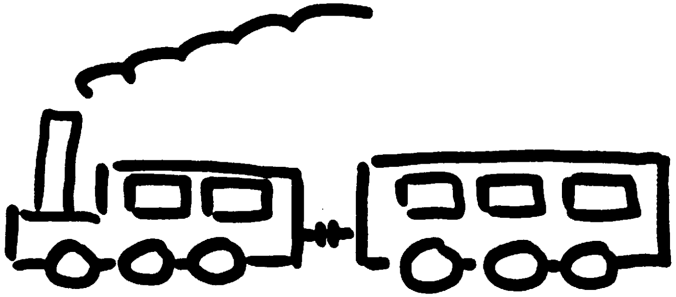
```
From rws@x.org Tue Sep 28 18:40 MET 1993
Received: from wilson by supercooper.superusers.dk (5.61/DKnet2.0.6) with SMTP
        id AA10191; Tue, 28 Sep 93 17:40:53 +0100
X-Organization: SuperUsers a/s, The UNIX SoftWare House
Received: by wilson.superusers.dk (5.65/DKnet2.0.6) with UUCP
        id AA04380; Tue, 28 Sep 93 18:40:51 +0200
Received: from x.org by dkuug.dk with SMTP id AA03975
  (5.65c8/IDA-1.4.4j for <brian@superusers.dk>); Tue, 28 Sep 1993 18:22:21 +0100
Received: from explain.x.org by x.org; Tue, 28 Sep 93 13:23:13 -0400
Received: from localhost by explain.x.org id AA03194; Tue, 28 Sep 93 13:23:11 -04
Message-Id: <9309281723.AA03194@explain.x.org>
X-Address-Reminder: The X Consortium is now in domain "X.ORG"
To: brian@superusers.dk (Brian Eberhardt)
Subject: Re: Important, please
In-Reply-To: Your message of "Tue, 28 Sep 1993 17:04:55 BST."             <930928
Date: Tue, 28 Sep 1993 13:23:07 EDT
From: Bob Scheifler <rws@x.org>
X-Charset: US-DK
X-Char-Esc: 29
Status: ORp
```

I am working to get a release out at the end of the week; you are lucky I'm even reading such email. Attached is plain text of slides we used at Xhibition. I have no time to verify how accurate the information still is. Sorry.

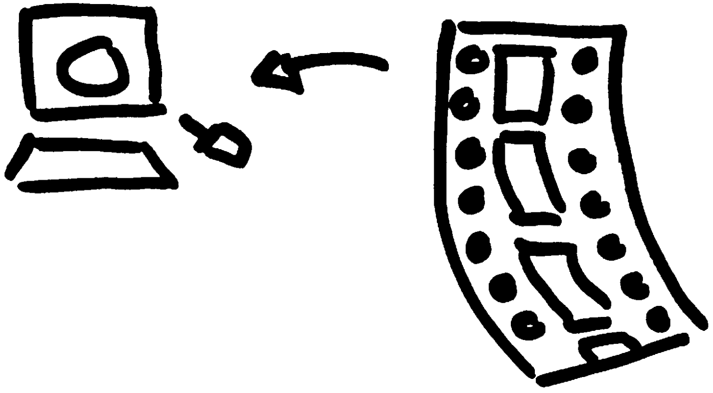
```
X: The Next Generation
Bob Scheifler, Director, MIT X Consortium
Luther Abel, President, X Consortium, Inc.

Release 6
 o target date: April 15, 1994
 o lots of new functionality
 o potentials, not commitments
 o new server extensions
 o security
 o multithreading support
 o internationalization
 o new toolkit
 o desktop management

Low-bandwidth X (LBX)
 o "make X usable at 9600 baud"
 clients <-> proxy <-> server <-> clients
         high      low        high
      bandwidth bandwidth  bandwidth

LBX
 o run over any underlying transport
    PPP, CSLIP, DDCMP, TCP
 o coexist with other protocols
    telnet, NFS, font service
 o compression
    X-specific and general
 o cache data in proxy
    window properties
    keyboard mapping tables
    font metrics and properties

X Image Extension (XIE)
 o compressed image transport
    G3/G4 FAX, JPEG, TIFF
 o not general image processing
 o rendition processing
    color space conversion
    e.g., YCbCr to RGB
    contrast enhancement
    convolution
    dithering
    geometric transforms
```

3IX

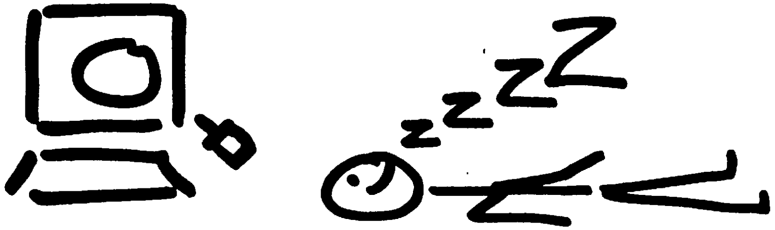
LBX



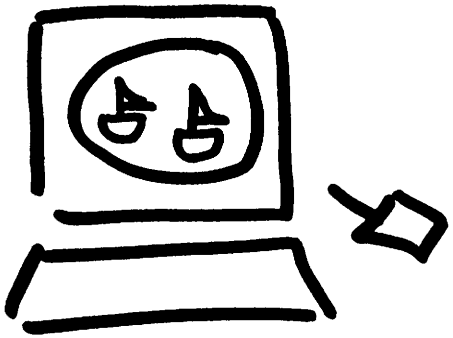
Sync Extension



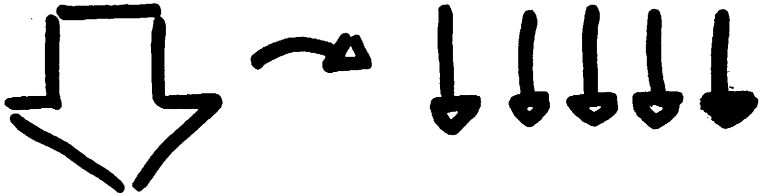
Screen Savers



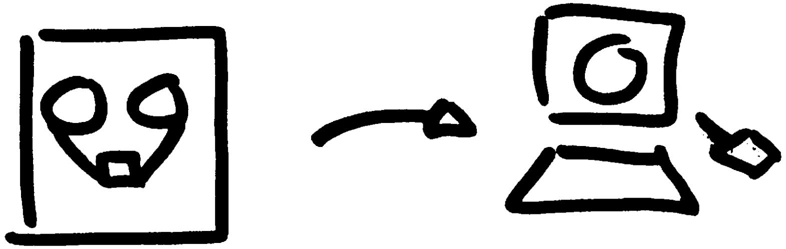
- Deep Frame Buffers



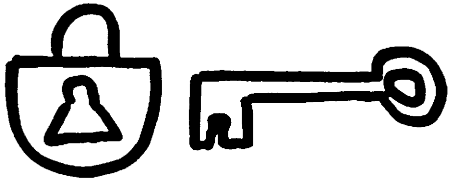
- Multithreaded



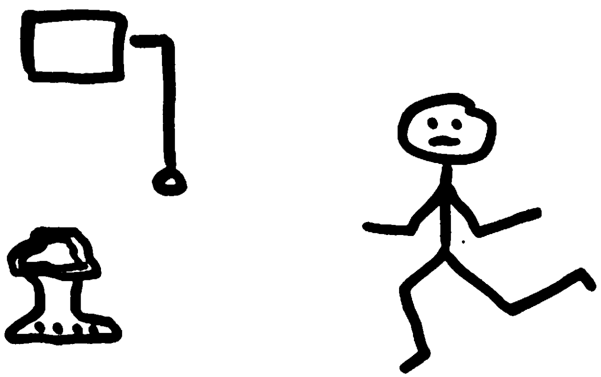
- X Test Extension



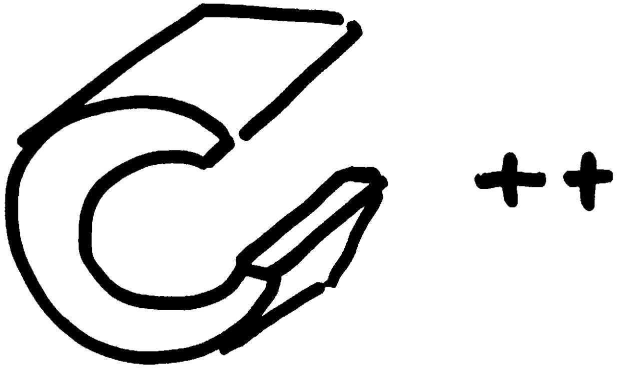
- Security



Drag & Drop



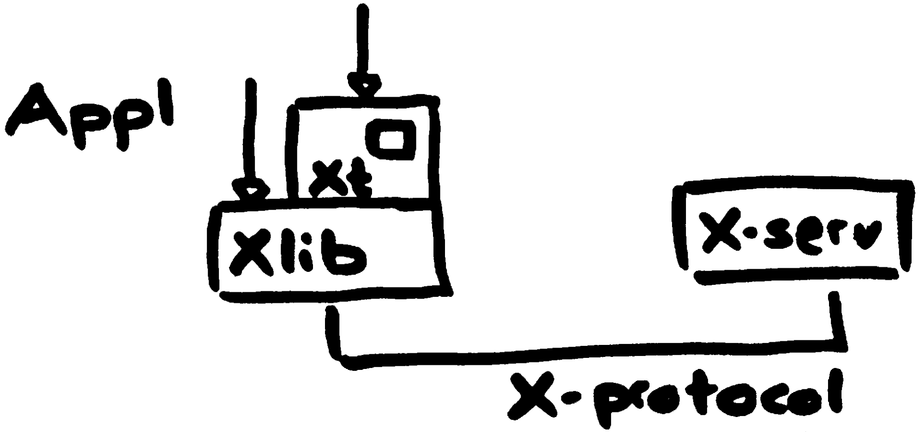
- Fresco



Det
Nye
X- Consortium

- Xt

X-Client X-Server



```
From rws@x.org Tue Sep 28 18:40 MET 1993
Received: from wilson by supercooper.superusers.dk (5.61/DKnet2.0.6) with SMTP
        id AA10191; Tue, 28 Sep 93 17:40:53 +0100
X-Organization: SuperUsers a/s, The UNIX SoftWare House
Received: by wilson.superusers.dk (5.65/DKnet2.0.6) with UUCP
        id AA04380; Tue, 28 Sep 93 18:40:51 +0200
Received: from x.org by dkuug.dk with SMTP id AA03975
  (5.65c8/IDA-1.4.4j for <brian@superusers.dk>); Tue, 28 Sep 1993 18:22:21 +0100
Received: from explain.x.org by x.org; Tue, 28 Sep 93 13:23:13 -0400
Received: from localhost by explain.x.org id AA03194; Tue, 28 Sep 93 13:23:11 -04
Message-Id: <9309281723.AA03194@explain.x.org>
X-Address-Reminder: The X Consortium is now in domain "X.ORG"
To: brian@superusers.dk (Brian Eberhardt)
Subject: Re: Important, please
In-Reply-To: Your message of "Tue, 28 Sep 1993 17:04:55 BST."           <930928
Date: Tue, 28 Sep 1993 13:23:07 EDT
From: Bob Scheifler <rws@x.org>
X-Charset: US-DK
X-Char-Esc: 29
Status: ORp
```

I am working to get a release out at the end of the week; you are lucky I'm even reading such email. Attached is plain text of slides we used at Xhibition. I have no time to verify how accurate the information still is. Sorry.

```
X: The Next Generation
Bob Scheifler, Director, MIT X Consortium
Luther Abel, President, X Consortium, Inc.

Release 6
 o target date: April 15, 1994
 o lots of new functionality
 o potentials, not commitments
 o new server extensions
 o security
 o multithreading support
 o internationalization
 o new toolkit
 o desktop management

Low-bandwidth X (LBX)
 o "make X usable at 9600 baud"
 clients <-> proxy <-> server <-> clients
         high      low       high
       bandwidth bandwidth  bandwidth

LBX
 o run over any underlying transport
    PPP, CSLIP, DDCMP, TCP
 o coexist with other protocols
    telnet, NFS, font service
 o compression
    X-specific and general
 o cache data in proxy
    window properties
    keyboard mapping tables
    font metrics and properties

X Image Extension (XIE)
 o compressed image transport
    G3/G4 FAX, JPEG, TIFF
 o not general image processing
 o rendition processing
    color space conversion
    e.g., YCbCr to RGB
    contrast enhancement
    convolution
    dithering
    geometric transforms
```

DIS
- o Document Imaging Subset of XIE
- o focus: bitonal image display
  e.g., faxes
- o scale, rotate, mirror, recolor
- o support "scale to gray"
  convert bitonal to grayscale
- o small enough to be ubiquitous even in low-end X terminals

Sync Extension
- o synchronization between clients
  independent graphics streams
  graphics and video
- o synchronization with real-time
  animation sequencing
- o synchronize within X server
  avoid round-trip requests
  utilize system buffering
  (client, network, server)

Sync Extension
- o counters
  system defined
  e.g., server time, v. retrace
  user defined
- o wait for counters to reach values
- o alarms
  asynchronous notification
- o priority scheduling

Screen-Saver Extension
- o stay competitive with Macintosh
- o conform to industry standards for swimming fish, flying toasters
- o have some fun

X Test Extension
- o first version already released
- o used in automated X Test Suite
- o provides basic "playback" support
  synthesize device events
- o now discussing "record" support
  capture device events
  capture synchronization data

PEX 5.1
- o PEX 5.1 already released
- o fixes the most pressing problems
  support for Escapes
  find supported drawables
  renderer-based picking, echo
  hierarchical Z-buffer control
- o PEXlib already released
- o low-level interface to PEX
- o complete access to PEX
  immediate mode graphics

PEX 6.0
- o major redesign
- o reconcile with final PHIGS PLUS
  e.g., add data mapping
- o advanced rendering
  anti-aliasing, alpha blending, transparency, texturing, motion blur,
  depth of field, shadowing
- o better performance, extensibility
- o strive for backward compatibility

PEX 6.0
- o support large data primitives
- o change structure naming

introduce CSS resource

- drop workstation resource
  - implement on client side
- peacefully coexist with OpenGL
  - common resource semantics
- spec done about same time as R6
- probably no code in R6

Deep Frame Buffers

- finalize existing Multibuffering draft standard
- "deep" frame buffer configurations
  - describe, enable, disable
- overlays, underlays
- alpha buffers
- depth (Z) buffers
- stencil buffers
- accumulation buffers (?)

Security

- Kerberos Version 5
- User-to-User authentication
- mutual authentication between client and server
- authorize individual users to connect to your display

Multi-threaded X server

- existing server is single-threaded
  - executes one request at a time
- desire interactivity in the presence of long-executing requests
- e.g., send mail while rendering 3D
- time-slice among requests
  - improves interactivity
- execute requests concurrently
  - improves throughput on multiprocessor hardware

Multi-threaded Xlib

- threads soon to be commonplace in Unix (and about time!)
- need to support applications that use multiple threads to drive their user interface
- Xlib interface design already capable of supporting threads
- existing implementation is not
- fix the implementation

Thread-safe Xt

- single thread of execution is pretty intrinsic to the Xt design, but
- need to support multi-threaded applications that use Xt
- make Xt "safe" for threads
- guarantee that only one thread executes inside Xt at a time

Input Method Protocol

- R5 added internationalization (i18n) support to Xlib and Xt
- standardized application interface for keyboard input and text display
- common implementation for Asian input uses separate IM server
- two different protocols in R5
- produce a single unified protocol
- increase interoperability

Other I18N

- dynamic linking of Xlib locale code
- ISO 10646 support
  - Universal Character Set
- ISO 9995 support
  - keyboard layout and semantics
- related keyboard support
  - disability access
  - PC keyboards

Xt Intrinsics

- thread safety
- safe signal handling

- o dispatching of extension events e.g., from input devices
- o allow widgets to be C++ objects
- o gadget / widget cache support
- o hooks
  disability access, editres, D&D
- o session management
- o new selection features

Fresco
- o "next generation" toolkit
- o take advantage of C++ features
- o support "inside" of application too
- o very lightweight objects
- o structured graphics
- o resolution independence, printing
- o window system independent
- o edit-in-place object embedding
- o distributed object components
- o multi-threading

Drag & Drop
- o familiar desktop metaphor
- o multiple vendor implementations
- o define a single standard
- o requirement: interoperability
- o goal: even with "foreign" windows
- o work in secure systems env.
- o define the underlying mechanism, not the user interface
- o separate drag, drop protocols

Session Management
- o uniform mechanism for users to save and restore their sessions
- o preserve screen layout
- o preserve application state
- o support application checkpointing
- o support clients from multiple hosts
- o support heterogenous systems
- o assume separate process start/ restart mechanism(s)

Other ICCCs
- o embedded clients ("insets")
  geometry, focus, menu mgmt
- o dynamic data updates
  notification rather than poll
- o formalize selection attributes
  VALUE, LENGTH, FONT
- o parameterized selection targets
  LENGTH of <target>
  ATTRIBUTES of <target>

The New X Consortium

X Window System
- o Developed at MIT in mid-80s
- o Adopted by major workstation mfrs as graphics/UI standard in 1987
- o MIT-sponsored consortium
  formed in 1988 to fund continuing evolution

MIT X Consortium
- o Vendor-neutral leadership in user interface swr stds
- o Open membership
- o Financed by mbrship fees
- o Rely on mbr participation
- o "Product" is not just X std, but sample implementations

Change
- o Consortium spinning out of MIT
- o Separate, nonprofit org
- o New leadership

Goals for NXC

- o Don't break anything
- o Get even better; address shortcomings
- o Keep X in the lead

Transition

- o Orderly move out of MIT
- o Recruit new staff
- o Get R6 out

New Goals

- o Broaden membership
- o Greater attention to strategic direction of X
- o Work with new XIA

Keep X in the Lead

- o Define "What is X?"
- o Articulate 5 year vision
- o Define R7

Beyond R6

- o PostScript Extension
- o Fresco V2
- o Audio
- o Printers as "displays"
- o Compound Documents
- o Couple w/related stds
  - Remote procedure call
  - OMG CORBA
  - Internet MIME
- o Gen'l server enhancemts

The X Journal

Serving the X Window System Community

July–August 1993 A SIGS Publication Vol 2 No 6/$6.50 US, $7 Canada

MOTIF BASKS IN THE SUN

COSE AGREEMENT ENDS GUI WARS

Automated GUI testing

OVERLAY WINDOWS BRING NEW DIMENSIONS TO X

NT's effect on X

Runtime widget subclassing

A PEEK AT X11R6

Modular techniques for layout design

CenterLine's ObjectCenter
AGE's XoftWare/32 for Windows
FrameMaker 3.0

X11R6—an overview

Dan Heller

BEFORE THE X Consortium is spun off from MIT at the end of the year to become its own entity, it will release the sixth version of the X Window System using the X11 protocol. Otherwise known as X11R6, this will be the last release under the direction of Bob Scheifler, the current director and founder of the X Consortium, which was established in 1988. The target dates for X11R6 are uncertain, but the tenative schedule is:

Alpha	October 1
Beta	February 4
Final	April 15

The new version is packed with so many new features, extensions, and performance improvements, that it could be considered the most substantial release since it moved from the older X10 protocol to X11 in 1988. On the list of new items with the new extensions are security, multi-threading support, further internationalization support, a new toolkit, and desktop management specifications (with support for features such as drag-and-drop).

One particularly interesting technological feature to come is a new method for transmitting the X11 protocol from the client to the server called LBX (Low Bandwidth X). This could possibly have a pronounced effect on the commercial viability of X, especially as it branches out from the UNIX arena and into the PC area or onto non-desktop devices.

To give you an idea of why this technology is interesting, first recall how X works. Any X-based application communicates with the X server through some sort of communication channel. Normally, this is a TCP/IP layer, but any other protocol is possible—X doesn't care. However, problems come up that make this model "difficult" to use in some environments. There are cases where the communication mechanism is "slow," such as through a modem connection where the X server is local to the user (on the desktop) and the application (the X client) is remote—on other end of the phone connection. Passing X protocol back and forth requires so much bandwidth that performance is not sufficient for any reasonable level of user productivity to take place. LBX will help alleviate this problem.

There are other examples, such as those running PPP (Point to Point Protocol) or CSLIP (Compressed Serial Line Internet Protocol). Because LBX is handled entirely by X, it can coexist with other protocols occurring on the network, such as telnet, NFS, or a font server. LBX is partially a compression scheme, but it is X-specific so it can examine X protocol packets going over the wire and optimize the amount of information being sent. Furthermore, it maintains its own caches of various information such as window properties, keyboard mapping tables, font metrics, and properties. With LBX, the commercial use of X will be able to expand far beyond its conventional boundaries.

For imaging, the XIE (X Image Extension) will be added to the X11R6 standard distribution. This will allow compressed image display for G3/G4 fax, JPEG and TIFF formats, although it is not a general image processing extension. As part of XIE, there will be DIS, the Document Imaging Subset. The purpose of DIS is bi-tonal image display for fax images; the ability to rotate, mirror, and recolor images; and support for "scale to gray" (bitonal grayscale). XIE and DIS are small enough to be ubiquitous, even in low-end X terminals.

The Sync Extension will allow synchronization between clients. This will allow for independent graphics streams to be sent by multiple clients to the X server and be displayed synchronously. This will help animation sequencing and avoid round-trip requests to the X server by using system buffering (whether the buffering takes place in the client, network, or server).

A Postscript Extension is finally being added to the core distribution. Not much more needs to be said on this one—the Display Postscript Extension has been available for X servers for a long time, but only from certain vendors.

The X Test Extension will be used in the automated X Test Suite. It will provide basic "playback" support by synthesizing device events, but it currently does not support "recording" events.

The current version of Phigs Extension to X (PEX), version 5.1, which is already available, will be a part of X11R6. What is being developed independently and not with X11R6, is PEX version 6.0. This is said to be a major redesign reconciling with final PHIGS PLUS (data mapping), advanced rendering such as anti-aliasing, alpha blending, trans-

Dan Heller is author of MOTIF PROGRAMMING and president of Z-Code Software. He can be reached by email at uunet!zander!z-code.com!argv.

parency, texturing, motion blur, depth of field, and shadowing. PEX 6.0 will coexist with OpenGL, and will have better performance and extensibility, although it's not quite clear how backwards-compatible it will be with PEX 5.1.

The Buffering Extension will extend the existing Multibuffering draft standard, support advanced rendering, permit PEX and OpenGL to share resource mechanisms, provide mono and stereo image buffers, alpha and Z (depth) buffers, and stencil and ˋccumulation buffers.

Finally, there will be a Screen Saver Extension, so applications can be written consistently to allow for flying toasters, swimming fish, and so on.

X11R6 *will be the last release under the direction of* Bob Scheifler *and could be considered the most substantial release since* X *moved from* X10 *to* X11 *in* 1988.

A significant enhancement to X11R6 will be support for Security using Kerberos Version 5. This will support user to user authentication and mutual authentication between client and server. It will also authorize individual users to connect to displays. This will help make X11R6 easier to get into certain government procurements and pass certain security-conscious environments.

A Multi-threaded X server (and Xlib layer) will also be available to allow for interactivity in the presence of long-executing Xlib requests (PEX, etc). Having time-slicing and concurrent requests will improve perceived performance substantially on multiprocessor hardware. Since Xlib will be able to support threads, the X Toolkit Intrinsics (Xt) will have to be modified to make .hreads "safe." (Currently, single threading is pretty intrinsic to Xt.)

In the internationalization area, X11R5 provided some i18n support to Xlib and Xt, but X11R6 will provide an Input Method Protocol to standardize the application interface to keyboard input and text display. X11R5 had two different protocols, but the consolidating to a common implementation will help simplify Asian input implementations. This is an area that is not yet resolved—the hot issue is whether to make the IM protocol part of the X protocol. If the protocol is separate, it could be implemented in window systems and other applications independently of X. (And why not? The protocol is independent of X.) However, compatibility with existing Xlib interfaces makes this difficult.

Support for ISO 10646 (Universal Character Set) and ISO 9995 (keyboard layout and semantics) are also slated for X11R6.

For Xt, in addition to making it safe of multi-threaded applications, there will be support for safe signal handling. (See TXJ, March/April 1992, "Handling signals in X".) While signals are specific to UNIX, the API will still port over to non-UNIX platforms.

Xt will also support dispatching of extension events, make widgets be C++ objects, and have the editres protocol support the GetValues method. There will also be standard support for the "virtual keysym" paradigm. For example:

```
<Key>Activate: activate()
```

Because Activate is not a recognized key on the keyboard, it is considered a virtual key that can be redefined. The system might indicate that the Return key is the same as the Activate key, or maybe:

```
Ctrl<Key>A
```

A new entry into X11R6 will be the introduction of Fresco, a "next generation" toolkit designed to take advantage of C++ features with very lightweight objects, support for structured graphics, resolution-independence, window-system independence (wow!), edit-in-place object embedding, distributed object components and multi-threading.

One of the most familiar features of graphical desktop interfaces has been the concept of drag-and-drop. While it has never been necessary to introduce a protocol specifically for drag-and-drop, it is a common desktop metaphor that has been integrated into a variety of window systems and user interface design specifications such as Motif and OPEN LOOK. There are currently multiple vendor implementations that are incompatible with one another, so R6 will define a single standard and define the underlying mechanism (not the user interface). Included will be both a drag protocol and a drop protocol, but the question arises again on whether the protocol should be independent of X. It would be nice if other window systems could comply with a single protocol for drag-and-drop. After all, X can co-exist with the Mac desktop and Microsoft Windows—wouldn't it be nice if users could drag and drop objects between them?

Along with drag-and-drop, X11R6 will also address the issue of *session management.* A new, uniform mechanism for users to save and restore their desktop sessions, preserve screen layout and application state, and support application checkpointing. This will all work with clients from multiple hosts on heterogenous systems. Once more does the question of having this protocol specification be separate from the X protocol itself arise. (Reader alert: this is called *foreshadowing.)*

Other Inter-Client Communications Conventions (ICCC) that will be addressed is the concept of "embedded clients" (where there can be nested window management). In my column entitled "Multimedia Applications under the X Window System" (TXJ, September 1992), I discussed the issue in depth. Under the new ICCC specifications, the user-interface portion of the issue concerning compound documents in a single window will be more clearly defined. Currently, the Object Management Group (Sun and HP) have a lower-level architecture and protocol designed to allow different applications on a network (not just the same machine or desktop) to talk to one another and exchange information. That design has no provision for user interaction or presentation on the display—nor should it. The newly proposed ICCC modifications addresses that issue. If this sounds familiar to you, it is analogous to Microsoft's Object Linking and Embedding (OLE), where multiple applications on the same desktop can share information and talk to one another. The difference, of course, is that the ICCC specs and the OMG specs are window system-independent, operating system-independent, and vendor independent. (Go Open Systems!)

Of course, with all these new features and extensions in X11R6, it is expected to have an incredible influx of new applications, better looking user interfaces, faster performance, and further desktop independence so X technology can expand as far as possible.

But what's next after X11R6? The X consortium is splitting off from MIT into a separate, non-profit organization with a new President and a larger staff. At the Advisory Committee meeting of 24 September 1991, Bob Scheifler announced his intention to step down as director of the X Consortium at the end of 1993 and return to doing research. At the same time, the MIT Laboratory for Computer Science expressed its belief that Bob's departure would provide a good opportunity for the X Consortium to move out from underneath MIT's wing.

In response, the Future Committee was established to examine the various options before the Consortium and recommend a course of action. On 14 March 1992, it published a proposal to the membership that the X Consortium become an independent, not-for-profit corporation.

The Future Committee announced its unanimous recommendation that Dr. Luther Abel of Procon Systems and formerly of Data General, Via Systems, Applicon, and Digital Equipment Corporation be selected to form and lead the new X Consortium. It will work closely with the X Industry Association, a newly formed business-oriented group, to help pay closer attention to the business needs of vendors that use and promote X technology.

So, what about the technology? The recurring theme expressed here (and that the X Consortium has been aware of during this process) is that new protocols are developing that are independent of one another. This is a valuable piece of information, since if these protocols can develop independently, the X Consortium can expand its charter to affect more than just the X Window System. It could act as a standards body for defining various open systems protocols and conventions associated with windowing systems (and beyond?) that are vendor-independent. **X**

Developments in Motif

Paul Davey Motif Project Manager
IXI Limited

pd@x.co.uk +44 (0)223 236555

Contents

Intro

COSE

So What

Motif

Who is IXI

Vendor of Open Systems GUI Solutions

Provide Easy to use environment for Distributed Computer Users

X/Motif based technology

Products:

- X.desktop, Deskworks
- Motif User and Development Packs for Sun

Platforms : Sun, IBM, HP, Intel, ...

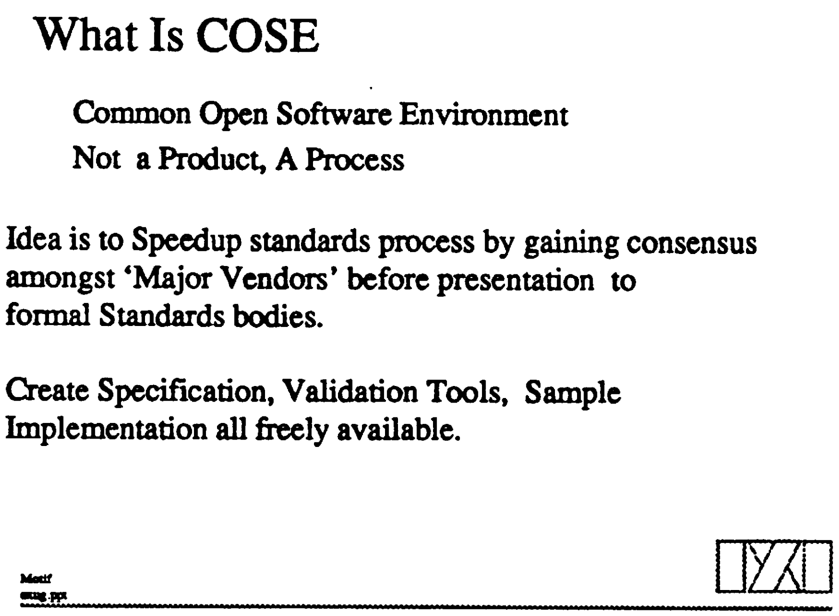
What is Motif

User Interface Specification and Implementation

Comprised of

- Style Guide
- Toolkit
- Window Manager
- User Interface language Compiler

- Formal Application Environment Specification
- Validation/ Test Suite



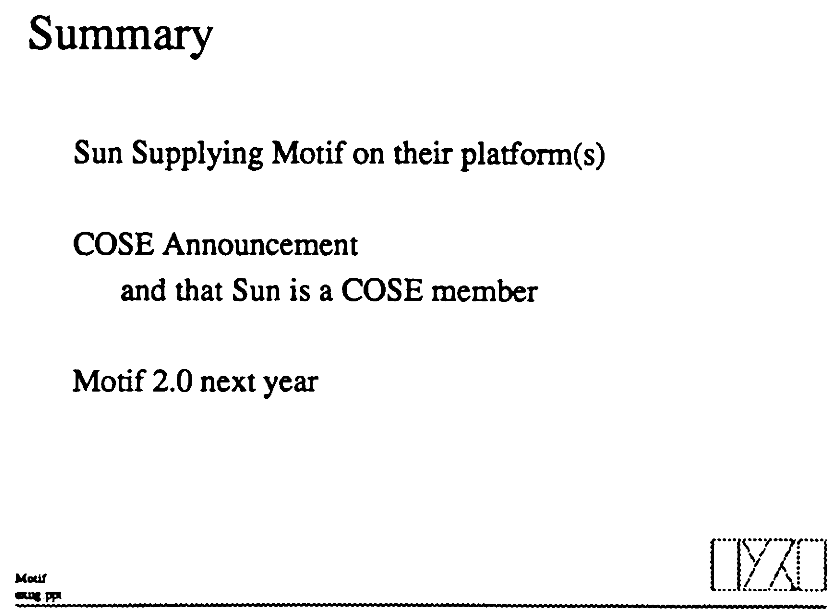
What Is COSE

Common Open Software Environment

Not a Product, A Process

Idea is to Speedup standards process by gaining consensus amongst 'Major Vendors' before presentation to formal Standards bodies.

Create Specification, Validation Tools, Sample Implementation all freely available.



Summary

Sun Supplying Motif on their platform(s)

COSE Announcement

and that Sun is a COSE member

Motif 2.0 next year

Why COSE

Open Systems Growth ‘slowed by excessive Competition’

Users and ISV’s want more similar systems

- More consistent, Interoperable

Provide Foundation for future development

Unstated

Fear of Competition

(From another ‘M’ word)

Standardise Industry on own technology

COSE Who

Core: HP, IBM, USL, SUN, Univel, SCO

First 4 are the technology providers

COSE Components

Common Desktop Environment (CDE)
    Motif Look and feel

Networking
    ONC+/DCE/Netware
Graphics
    X/XLIB, PEX/PEXlib, XIE/XIElib
Multimedia
    DMS/DIME
Distributed Object technology
    CORBA
System Management

CDE

User Accessable/User Interface Component of COSE

Desktop/FileManager/EndUser Tools

Combination of elements from
    VUE, CUA Workplace Shell
    OPENLOOK + Deskset
    USL 4.2 Desktop Manager

OSF Motif 1.2 API, Look and Feel

Motif Component

Motif 1.2 (plus)

Motif 1.2 base

Bug Fixes fm OSF + COSE members

Optimized for Performance and Size

Multi Threading

Some M2.0 Features

- Colour Selection
- Some Support for OW Drag and Drop
- Widgets - Container + IconGadget, Notebook

TimeTable

Currently

Pre June	Prototype
June	Specification
	High level Overview Only
Oct	Developers Conference - Sample Code
Dec	Complete Specification - X/Open
94/Q2	Product (?)

Risks to COSE

Implementations are poor quality and late (Motif?)

Differing implementations cross platform.

Break backward compatibility - Freeze in industry.

User Resistance - Solutions are not to User problems

Internal Member Politicing/ Positioning

Immediate Importance

Sun to provide Motif for its platforms
(IXI Motif)

Motif X/Open Standard
(Hence the above)

Vendor Motif stabilise at M1.2

Comments On COSE

Current public 'lack of substance'

Lack of 'End User' Input (or requests for same)
Supposed to input through X/Open process.

Acceptance of Motif 'easiest' part.

Currently Users and ISV's sceptical
current proposals seen as only to vendors benefit

Sun Position

Attempting Migrate OS base to Solaris
Supplying Motif 1.2
Migrate UI and tools to Motif

Initially provide Motif Product

Later bundle Motif with OW.

Motif History

3 Major Releases

1.0 Initial release, relatively Unstable

1.1 Mature and Stable
In Common Use

1.2 1.1 Base plus some New Features

Proposed New Version coming
M 2.0

Motif 1.2 Major Features

3 Major enhancements from M1.1

Internationalisation and locale Support
Migration to X11R5 base

Drag and Drop
Proprietary Protocol -
Use MIT protocol when available
Standardised API (Incredibly baroque)

Tear Off Menus
Comparable to OL pushpin

Motif 1.2 Minor Features

'Tidy Up' M1.1 Behaviour

- Additional Insensitive Visuals
- Visual Changes
  - Access to Color API (was internal)
- Titled Frames
- Better Traversal Support
- Resource management for Enumerated Types

Many many more...

Motif 2.0

TimeTable

Snapshot Program starts in September

Two Snapshot drops

Release to OEM's Q1 94

X11R5 Based..

Motif 2.0 Features

New Widgets
Enhance Widget Extensibility
Drag and Drop - Performance Improvement
- Possible New API

Enhance XmString
- Tab support
- Rendition (Colour, underline, strikethrough)

Right To Left Layout Support
Visuals:
- Check mark on Toggle
- Joint Arrows (XmScale)

Motif 2.0 New Clients

Enhance existing clients also

Session Manager

WorkSpace manager

Mods to Window Manager to support above two

Motif 2.0 New Widgets

Container and IconGadget
Notebook
Combobox
SpinButton
Scale

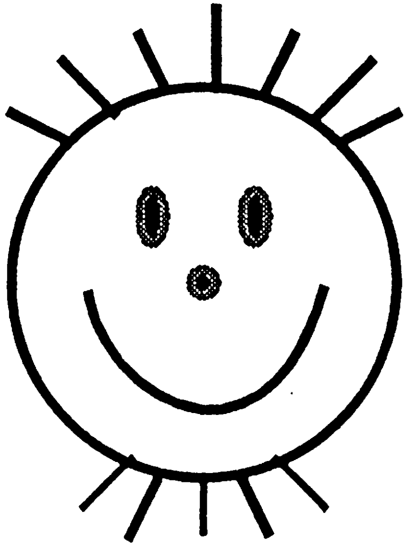
'Text' Widget supporting XmString

Motif 2.0 Widget Extensibility

'Traits' Mechanism - specify traits want widget to have
Subclass off XmPrimitive and XmManager (only)

Widget Writers Tools and Docs
- Extensible libXm
- Documentation - Reference and Instructional
- Example Code

MOTIF 1.2.2 DEVELOPER'S TOOLKIT IS SHIPPING!!!



AGENDA

1. Windows Toolkits Roadmap

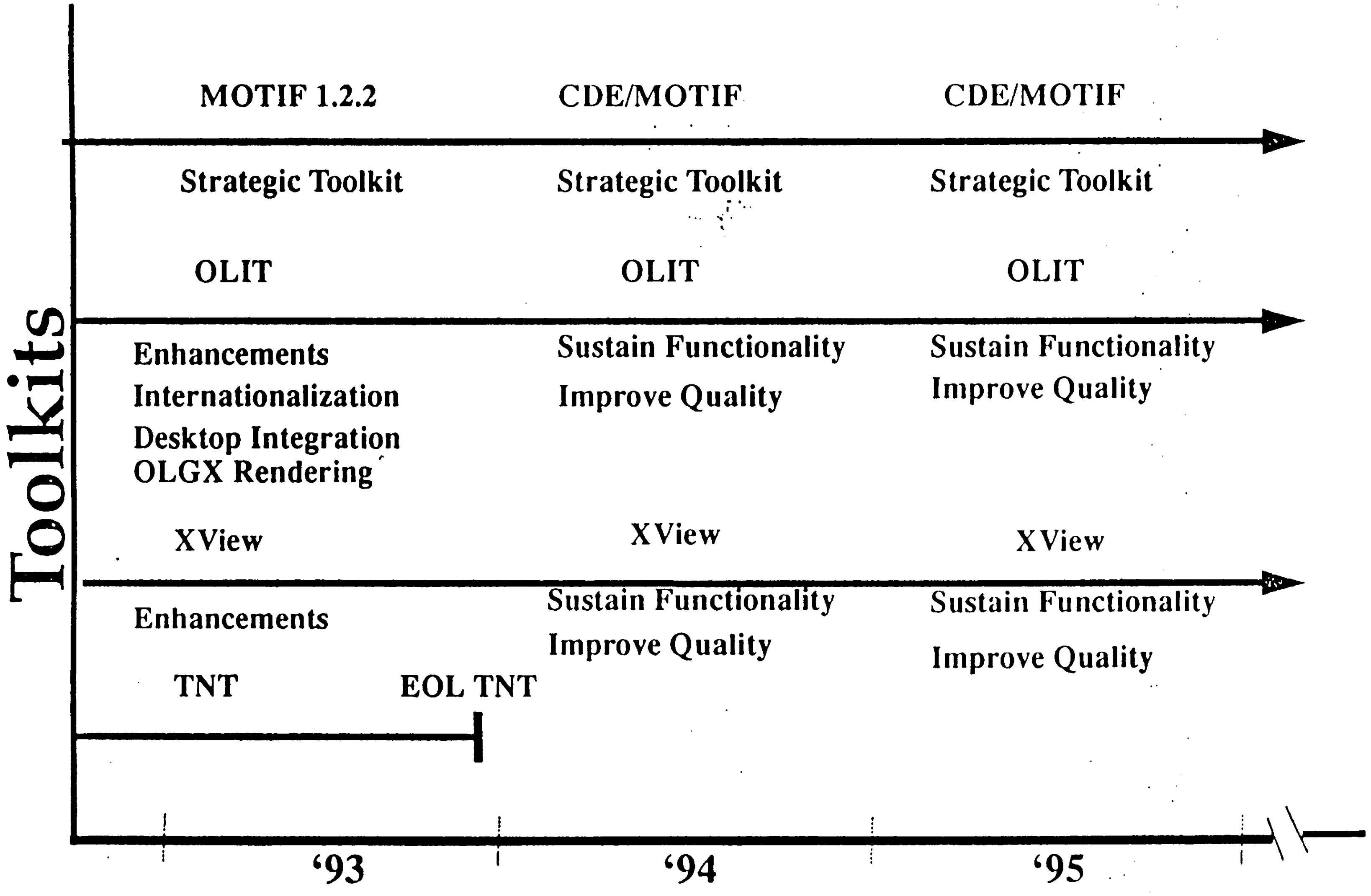
2. MOTIF/SDK Timeline Interaction

3. Developer Toolkits Messages

4. MOTIF 1.2.2 Toolkit for Solaris 2.2 (SPARC)

5. OpenLook to MOTIF Transition Tools

SunSoft's Toolkits Roadmap



Toolkits

MOTIF 1.2.2

Strategic Toolkit

CDE/MOTIF

Strategic Toolkit

CDE/MOTIF

Strategic Toolkit

OLIT

Enhancements
Internationalization
Desktop Integration
OLGX Rendering

OLIT

Sustain Functionality
Improve Quality

OLIT

Sustain Functionality
Improve Quality

XView

Enhancements

XView

Sustain Functionality
Improve Quality

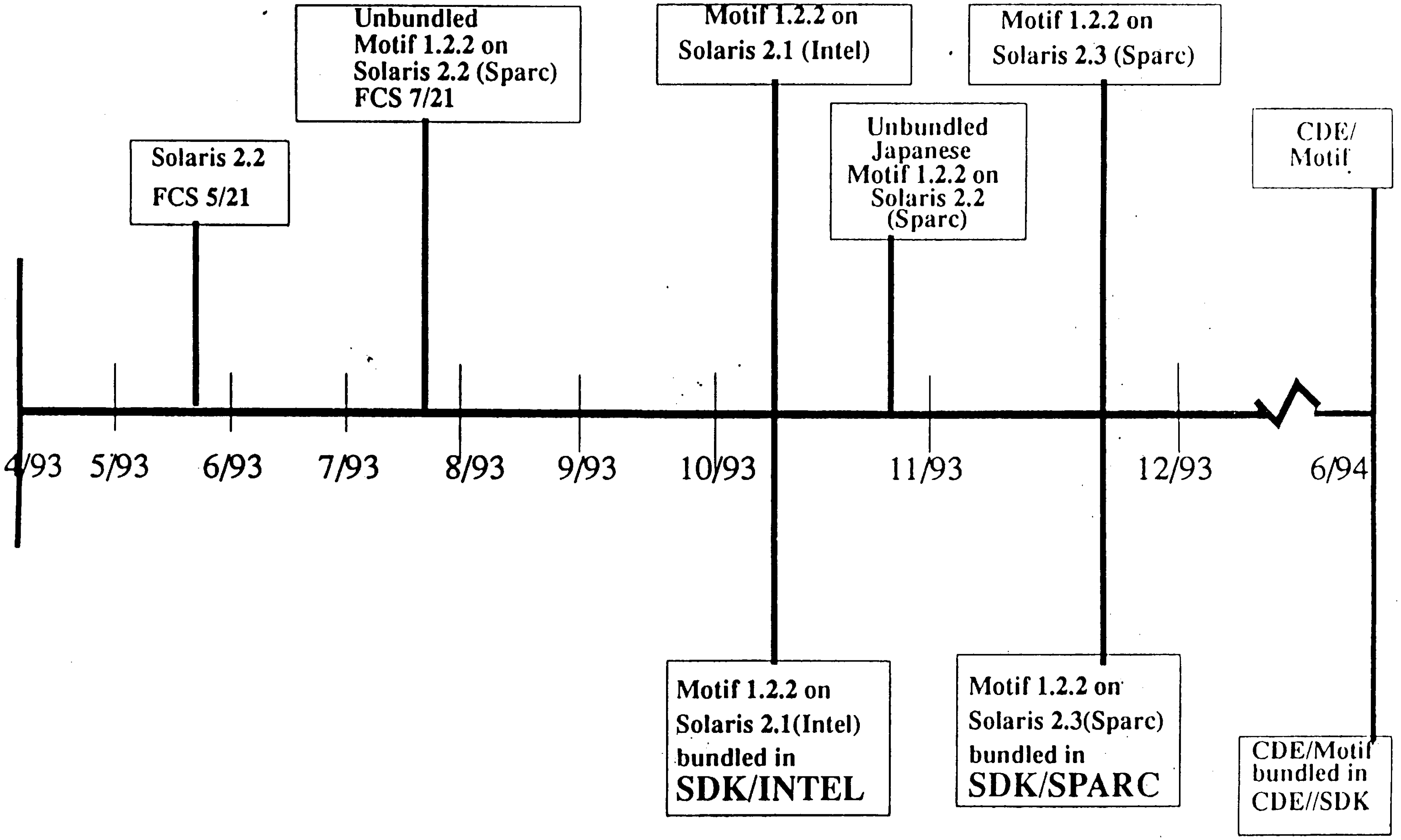
XView

Sustain Functionality
Improve Quality

TNT

EOL TNT

TIME LINE FOR MOTIF TOOLKIT & SDK INTERACTION



Solaris 2.2
FCS 5/21

Unbundled
Motif 1.2.2 on
Solaris 2.2 (Sparc)
FCS 7/21

Motif 1.2.2 on
Solaris 2.1 (Intel)

Unbundled
Japanese
Motif 1.2.2 on
Solaris 2.2
(Sparc)

Motif 1.2.2 on
Solaris 2.3 (Sparc)

CDE/
Motif

Motif 1.2.2 on
Solaris 2.1(Intel)
bundled in
SDK/INTEL

Motif 1.2.2 on
Solaris 2.3(Sparc)
bundled in
SDK/SPARC

CDE/Motif
bundled in
CDE//SDK

DEVELOPER TOOLKITS KEY MESSAGES

- Motif 1.2.2 on Solaris 2.2 is an Interim Product
  Use Motif Toolkit for NEW applications

- XView and OLIT will be sustained and quality improved
  Don't panic! For EXISTING applications use XView, OLIT

- Devguide will provide Motif code generator (gmf) to produce
  Motif C code, and UIL code generator (guil), from GIL
  Available Fall '93 bundled in SDK/Sparc

- OpenLook to MOTIF Transition Tools
  Third Party solutions exist; Re. May '93 SunExpert
  SunSoft Transition Tools White Paper in SDK/Sparc

MOTIF 1.2.2 FOR SOLARIS 2.2 (SPARC)

- Solaris 2.2 port of Motif 1.2.2 product from IXI, Cambridge, UK

- Shared and Archived libraries (libXm, libMrm, libUil)
- Motif Window Manager (mwm)
- User Interface Language Compiler (UIL)

- Three Major Enhancements from Motif 1.1

  Internationalization and Locale support (Migration to X11R5 base)

  Supports Drag and Drop

  Tear Off Menus (comparable to OL pushpin)

- Very competitively priced at $295/RTU license with volume discounts

  No Royalty Fees due to SunSoft or to OSF for Motif libraries (statically or dynamically linked)

  OSF Motif license is not required; all developers and end-users covered under SunSoft Unlimited Distribution Rights license

MIGRATION FROM OPENLOOK TO MOTIF

◆ Third Party Migration Solutions exist:

- ✱ Integrated Computer Solutions (ICS)
- ✱ Qualix Group Inc
- ✱ National Information Systems Inc (NIS)

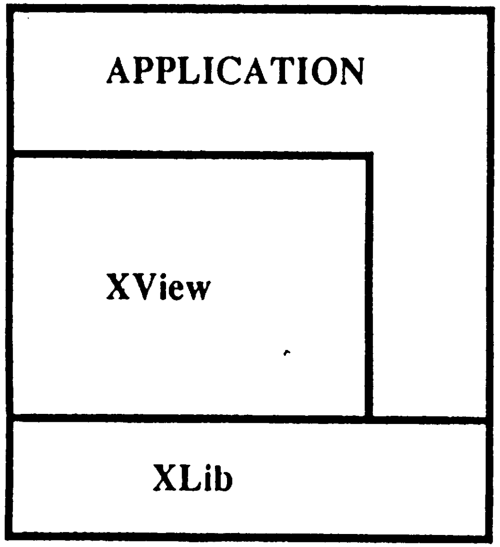
◆ Good Reference: GUI Development Tools, May '93 SunExpert, pg 54 - 63

SIMILARITIES BETWEEN XView & MOTIF

- Object Oriented in design
- Use a simple API (although a different one to create objects/widgets)
- Use an event-driven programming model
- Use callbacks and have an event loop
- Allow the creation of interfaces with menus and dialogue boxes

DIFFERENCES BETWEEN XView & MOTIF

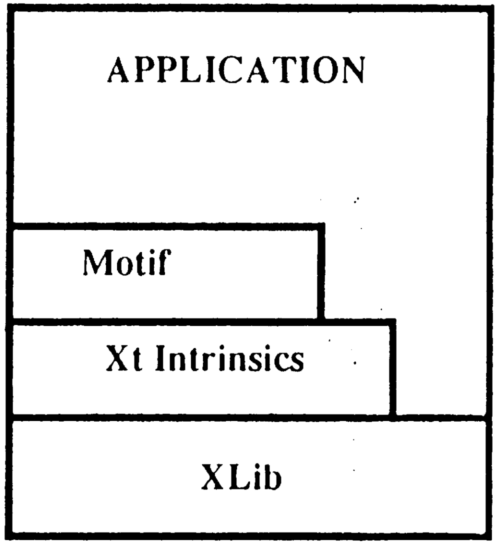
- There is a difference in the layers of X that XView and Motif use:



APPLICATION

XView

XLib



APPLICATION

Motif

Xt Intrinsics

XLib

- XView is a set of objects and an API which is not Xt-Intrinsics based
- Motif is an Xt-Intrinsics based widget set

DIFFERENCES BETWEEN XView & MOTIF

- Motif does not have a "Notifier" as in XView, although most of that functionality is available via Xt-Intrinsics
- In XView, there is no Map, Manage and Realize steps that are necessary for each object as in Motif
- All XView programs are variable argument calls by default
- An "owner" in XView is known as a "parent" in Motif
- There are visual differences in GUIs created using XView and Motif

OPENLOOK TO MOTIF FROM ICS

- ◆ With SUN's adoption of Motif Toolkit, many OpenLook developers will be looking to move existing applications to the Motif environment

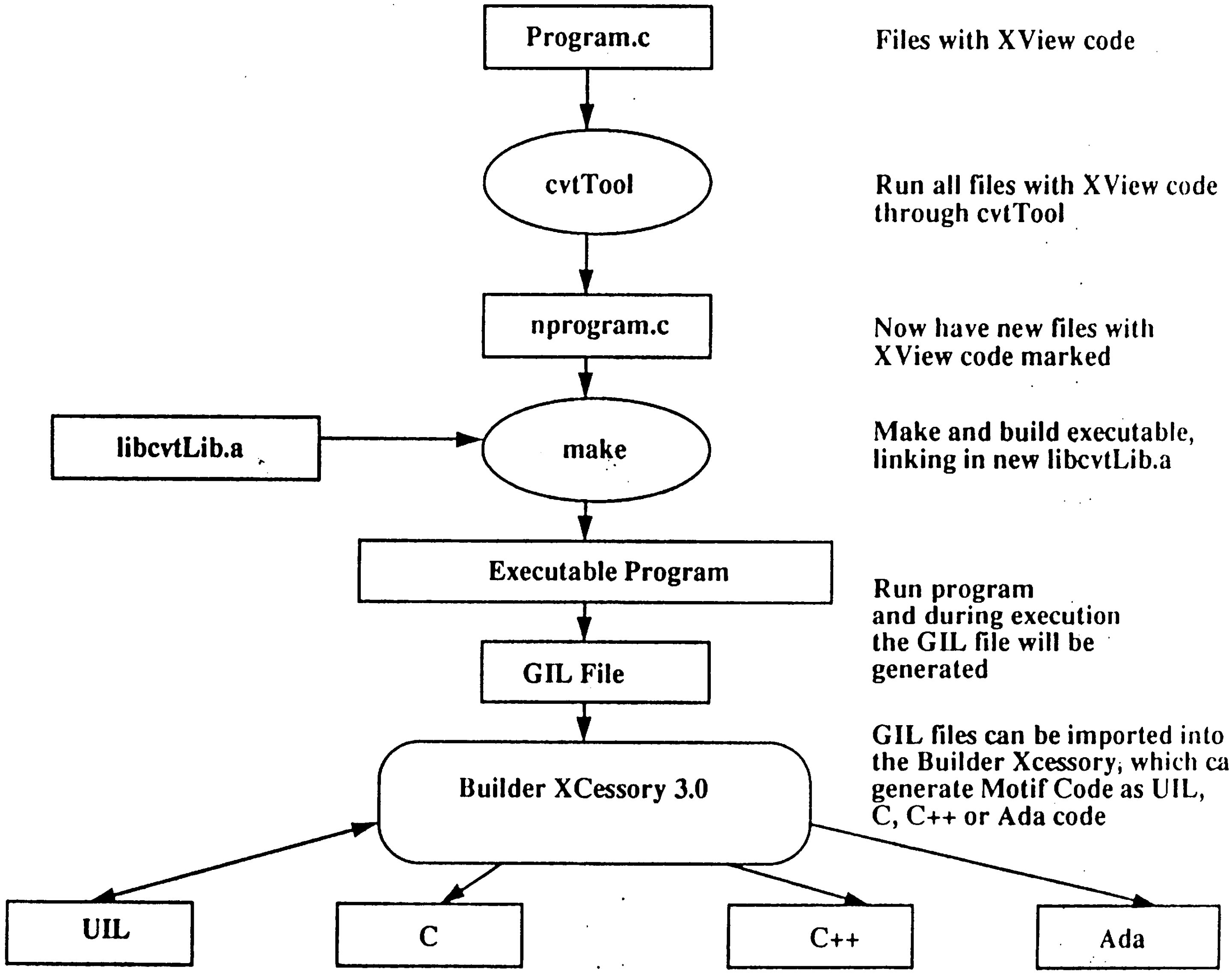
- ◆ Integrated Computer Solutions (ICS) will provide:

  ✻ An Interface Converter (GILConvert) that translates XView application files into Graphics Intermediate Language (GIL) files

  ✻ An import mechanism in Builder Xcessory 3.0 that reads GIL files and generates User Interface Language (UIL), C, C++ or Ada code that represents the GUI in Motif

- ◆ These products will allow XView code that was developed by hand or GIL files produced by Devguide to be moved to a native Motif development environment

OPENLOOK TO MOTIF: ICS PROCESS



Files with XView code

Run all files with XView code
through cvtTool

Now have new files with
XView code marked

Make and build executable,
linking in new libcvtLib.a

Run program
and during execution
the GIL file will be
generated

GIL files can be imported into
the Builder Xcessory, which can
generate Motif Code as UIL,
C, C++ or Ada code

WHAT IS GILconvert 1.1?

- ◆ Developed by Expert Object Corporation, distributed exclusively by Integrated Computer Solutions (ICS)

  GILconvert is a library that links with XView files and extracts XView Objects, then outputs GIL files

- ◆ Not Planned for GILconvert 1.1

  Translation of code within notify and event procs

  Translation of Notice Objects (not available in GIL)

  Translation of dynamically created Interface Objects (GILconvert can be run multiple times to capture as much of a dynamic application interface as possible)

- ◆ Product Structure

  ICS is planning to bundle GILconvert with Builder Xcessory 3.0

  GILconvert will also be sold separately, pricing is under review, but is expected to be around $2,500 per RTU license

WHAT IS BUILDER Xcessory 3.0?

- Interactive Design Tool/GUI Builder from Integrated Computer Solutions

- Imports GIL Files

- New in Builder Xcessory 3.0
  - Motif 1.2 support
  - Outputs C++ classes (in addition to C code)
  - Configurable, for developers with differing skill sets
  - Support for large project team development
  - Internationalization

SCHEDULES

- Builder Xcessory 3.0

  Beta: September 13 '93

  FCS: Early Fourth Quarter '93

  Initial Platform: Solaris 1.x

  CYQ493 Platforms: Solaris/Intel, Solaris 2.x

  Pricing: $3,200 per developer license (with volume discount)

- GILconvert 1.1

  Beta: September 13 '93

  FCS: Early Fourth Quarter '93

  Initial Platform: Solaris 1.x

  Pricing: Bundled with BX 3.0 (available separately for $2,500 per RTU)

- Substantial rollout is planned for UNIX Expo (Sept '93) focusing on these tools

- First demonstrated at Xhibition, June'93; reviewed by Devguide team

ICS CONTACTS

- ICS is looking for European Beta sites for testing Migration Tools
- Product and Availability Information

  Judy Lazaro
  ICS Market Development Mgr.
  lazaro@ics.com
  (617) 621-0060

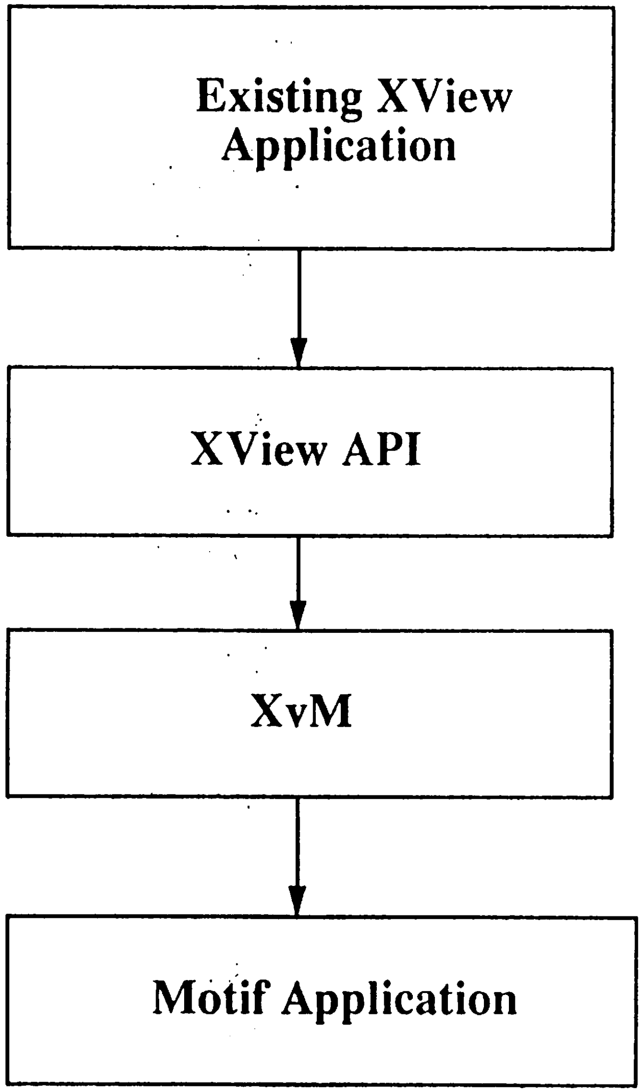
- Technical Information

  Ed Lee
  ICS Product Marketing Mgr.
  elee@ics.com
  (503) 752-7901

OPENLOOK TO MOTIF FROM QUALIX

- Fast, Easy process to convert XView applications to Motif
- Based on XView 3.2 API and Uses Motif 1.2 Libraries
- Recompile and Relink existing XView application with XvM library to provide Motif look-and-feel
- Minimizes transition time and cost- simply recompile and run application
- Preserves investment in current XView applications and programming expertise
- Minimizes maintenance costs for current applications- no new bugs introduced by hand recoding translations
- Minimum risk- Source code is available so there is no risk of vendor obsolescence
- XvM provides C++ interface, which results in future flexibility for current XView applications

OPENLOOK TO MOTIF: QUALIX PROCESS



Existing XView Application

XView API

XvM

Motif Application

XVM BETA TEST PROGRAM

- Beta Test program starts mid-September '93
- Requirements: Solaris 1.x, XView 3.2, Motif 1.2.2, C compiler, Linker
- Beta release: Binary libraries with full functionality for limited time period at no cost for participation
- Packages supported in initial Beta: Server, Frame, Panel, PanelItem, Server-Image, Icon, Cursor, Menu, TextSW
- Packages supported in second Beta: Font, Color, Canvas, ScrollBar, Full Screen
- For Beta site inclusion: e-mail xvm.beta.interest@qualix.com
- Qualix Contact: Doug Shaker : (415)570-2985; dshaker@qualix.com
  Shirley Kumamoto: (415)572-0200; shirleyk@qualix.com

XVM FCS INFORMATION

- FCS availability: November '93
- Platforms: Initially: Solaris 1.x, Motif 1.2.2

  Late Q493: Solaris/X86 and Solaris 2.x
- Pricing: Single-developer Binary: $495 with volume discounts available

  Source access per building: $5,000
- European distribution: Currently handled through Qualix's USA HQ

  Qualix is actively seeking European distribution partners
- For more information: e-mail xvm.interest@qualix.com
- Qualix Contact: XvM Product Manager:

  Shirley Kumamoto: (415)572-0200; shirleyk@qualix.com

OPENLOOK TO MOTIF FROM NIS

- ACCENT STP (Sun Transition Pack) converts OpenLook applications to Motif applications
- Requirements: Solaris 1.x or Solaris 2.x, X11R5 Window System, Motif 1.2.,
- Converts XView C or C++ source code to Motif C or C++ source code
- Converts OLIT C or C++ source code to Motif C or C++ source code
- Generates UIL files or Motif C code from Devguide GIL files
- Closely approximates original Window Geometry Management
- Leaves all Header files, Make files and Data structures intact
- Currently available ACCENT toolkit provides Motif or OpenLook GUI at runtime from saved source code
- Saves hundreds of hours of hand coding and testing

OPENLOOK TO MOTIF:ACCENT STP PRICING

◆ XView Conversion Option	$4,995
◆ OLIT Conversion Option	$4,995
◆ Devguide Conversion Option	$4,995
◆ WindowMaker GUI Editor	$1,495
◆ ACCENT Toolkit (Motif to OpenLook)	$2,495
◆ HotLine, FAX, E-mail support	20%

MOTIF CUSTOMER TRAINING SERVICES

IN EUROPE

- In Europe Motif Training by Instruction Set is offered under the auspices of:

  Computer College Ltd.,
  216 Cambridge Science Park
  Cambridge CB4 4WA
  UK
  Telephone: (44)-(223)-420002
  FAX: (44)-(223)-426868

- Contact: Cliff Booth (e-mail: cliff@unipalm.co.uk)
  Richard Smith (e-mail: richards@unipalm.co.uk)

OPENWINDOWS USER'S GROUP ALIAS

- owd-ig@cbzoo.att.com

  is the Worldwide alias for OpenWindows Developer's Interest Group (OWDIG) maintained by Dr. Neil Ostrove, AT&T Bell Labs, Columbus,Ohio.

  We encourage you STRONGLY to use it to communicate directly with SunSoft Engineering, Marketing & Mgmt and discuss OW issues within the group

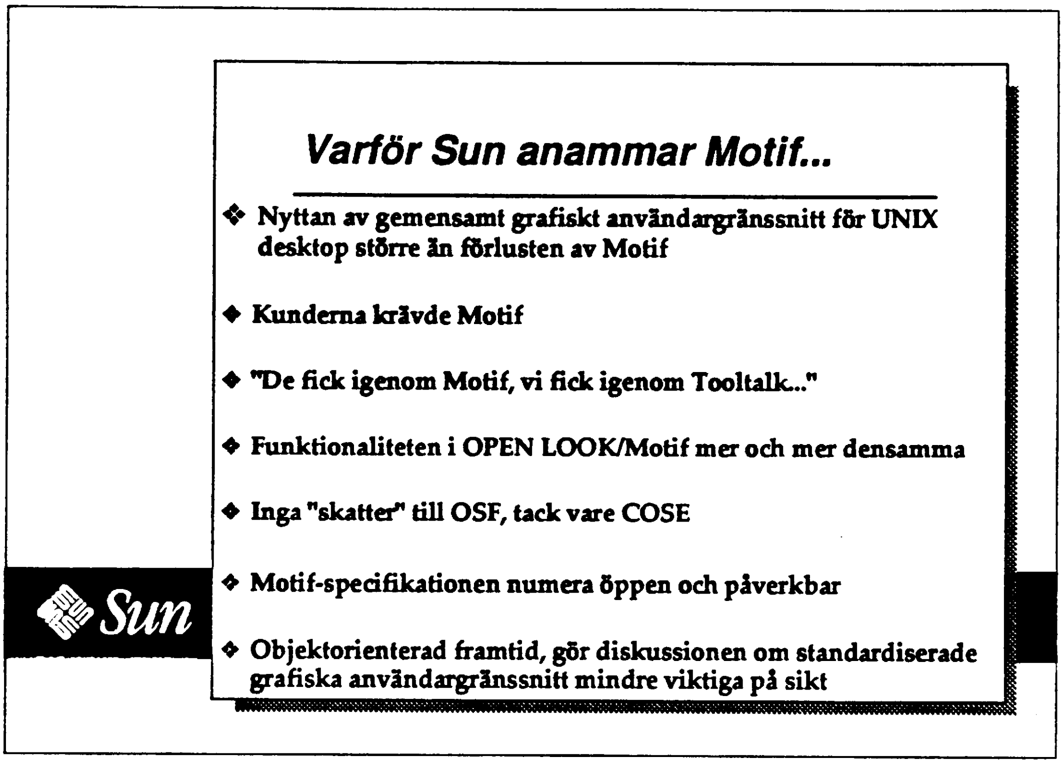
- To be added to this alias, please send e-mail to:

  owd-ig-request@cbzoo.att.com

  OR as a last resort, e-mail Neil Ostrove at:

  ost@cbnews.att.com

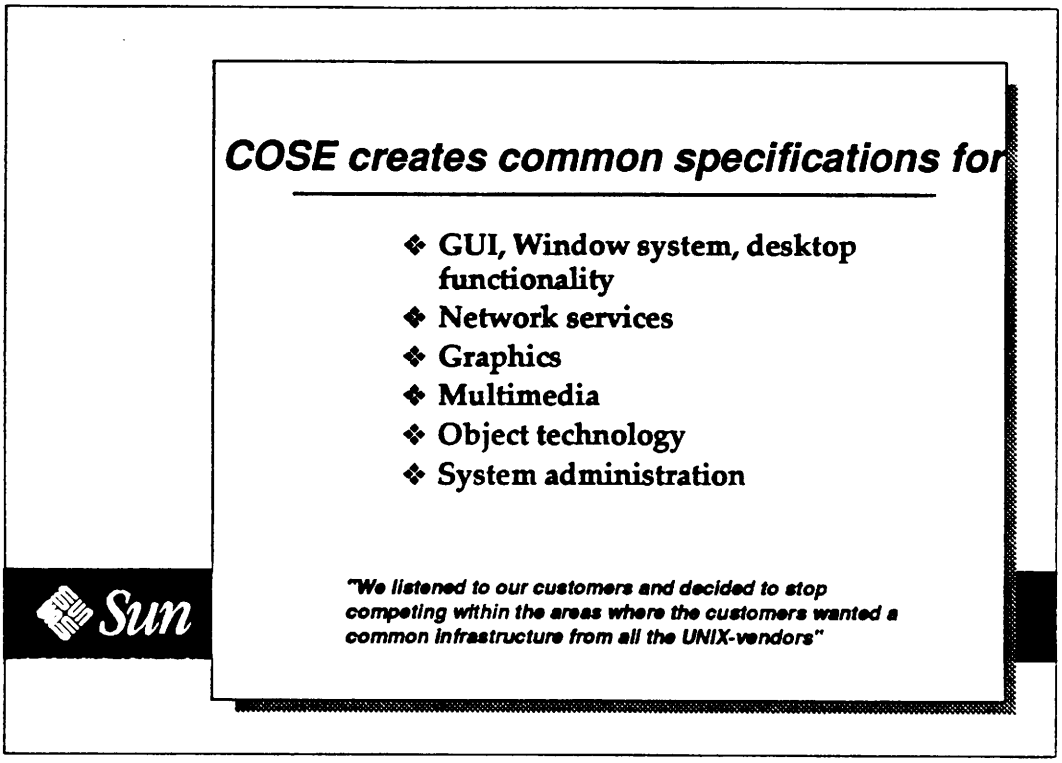
Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



Varför Sun anammar Motif...

- Nyttan av gemensamt grafiskt användargränssnitt för UNIX desktop större än förlusten av Motif
- Kunderna krävde Motif
- "De fick igenom Motif, vi fick igenom Tooltalk..."
- Funktionaliteten i OPEN LOOK/Motif mer och mer densamma
- Inga "skatter" till OSF, tack vare COSE
- Motif-specifikationen numera öppen och påverkbar
- Objektorienterad framtid, gör diskussionen om standardiserade grafiska användargränssnitt mindre viktiga på sikt

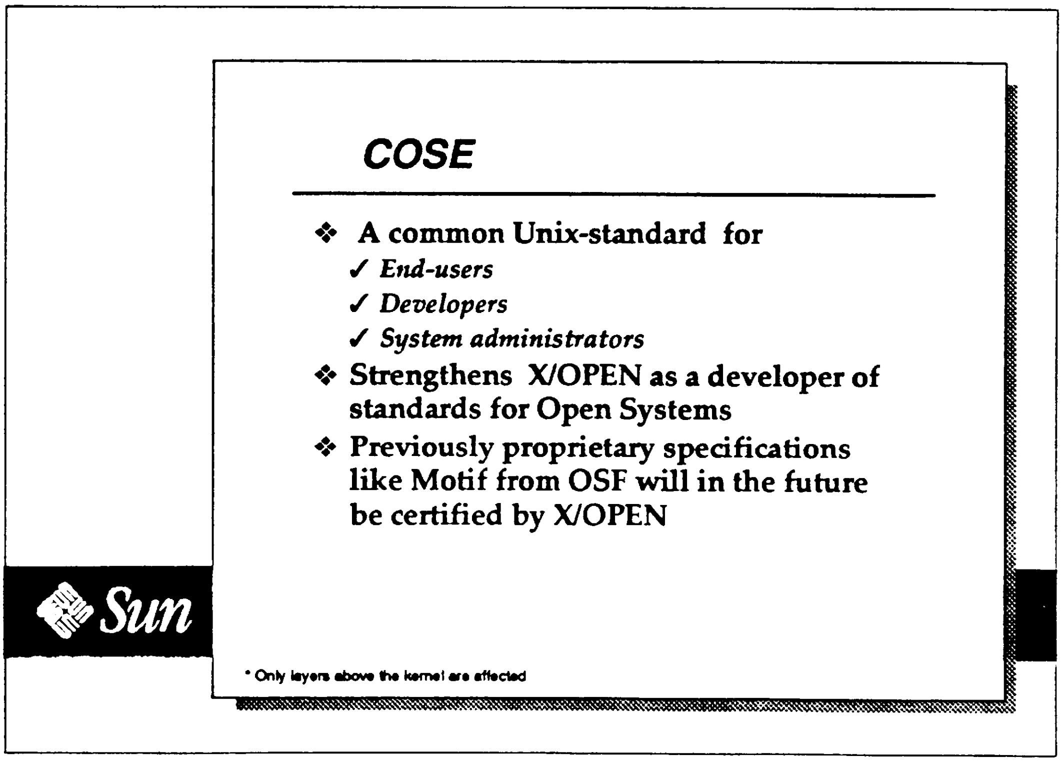
Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



COSE creates common specifications for

- ❖ GUI, Window system, desktop functionality
- ❖ Network services
- ❖ Graphics
- ❖ Multimedia
- ❖ Object technology
- ❖ System administration

"We listened to our customers and decided to stop competing within the areas where the customers wanted a common infrastructure from all the UNIX-vendors"

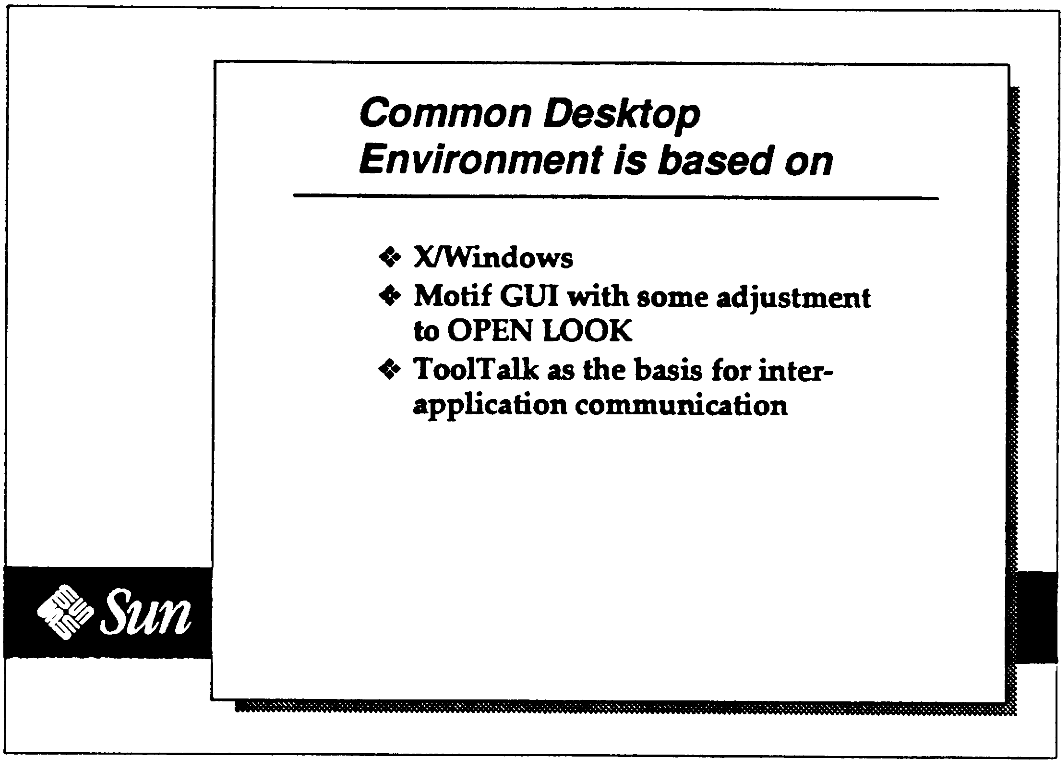


COSE

- ❖ A common Unix-standard for
  - ✓ *End-users*
  - ✓ *Developers*
  - ✓ *System administrators*
- ❖ Strengthens X/OPEN as a developer of standards for Open Systems
- ❖ Previously proprietary specifications like Motif from OSF will in the future be certified by X/OPEN

* Only layers above the kernel are affected

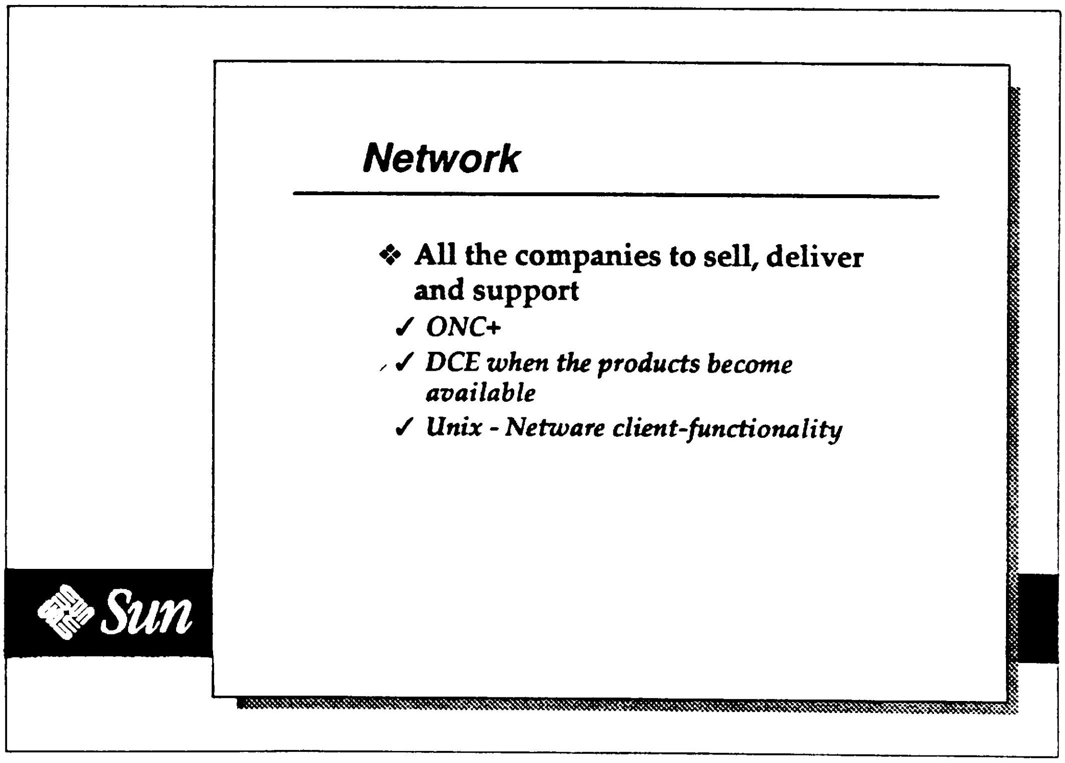
Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



Common Desktop Environment is based on

- ❖ X/Windows
- ❖ Motif GUI with some adjustment to OPEN LOOK
- ❖ ToolTalk as the basis for inter-application communication

Sun

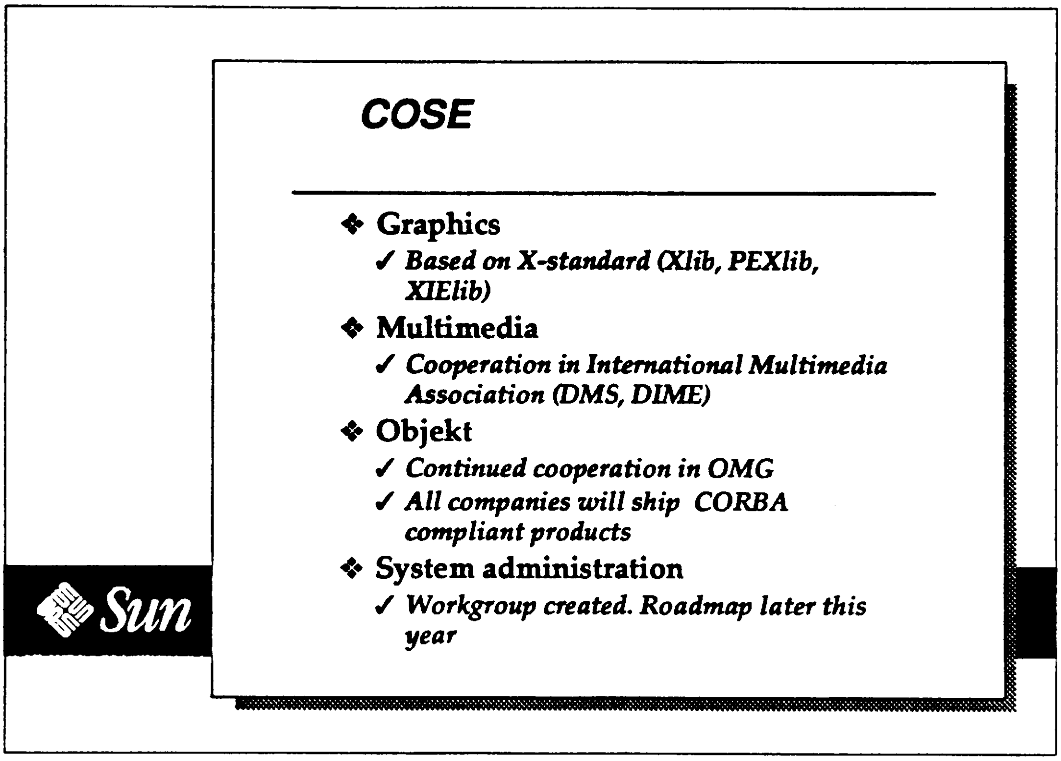


Network

- ❖ All the companies to sell, deliver and support
  - ✓ *ONC+*
  - ✓ *DCE when the products become available*
  - ✓ *Unix - Netware client-functionality*

Sun

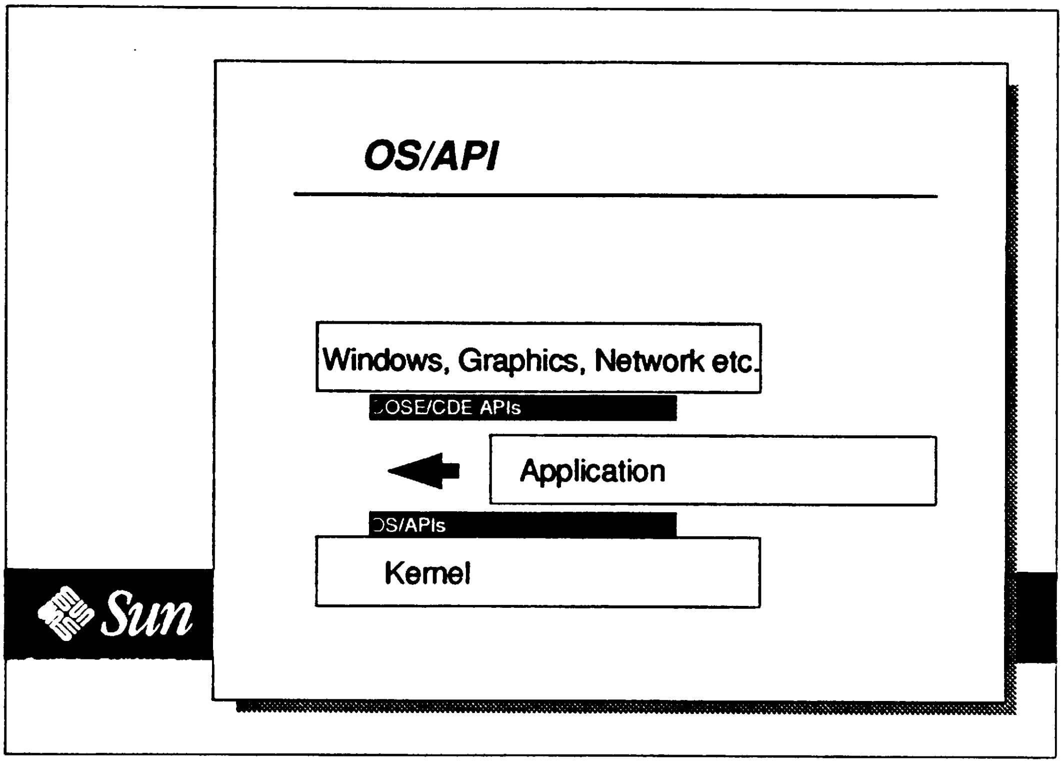
Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



COSE

- Graphics
  - ✓ *Based on X-standard (Xlib, PEXlib, XIElib)*
- Multimedia
  - ✓ *Cooperation in International Multimedia Association (DMS, DIME)*
- Objekt
  - ✓ *Continued cooperation in OMG*
  - ✓ *All companies will ship CORBA compliant products*
- System administration
  - ✓ *Workgroup created. Roadmap later this year*

Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



OS/API

Windows, Graphics, Network etc.

COSE/CDE APIs

Application

OS/APIs

Kernel

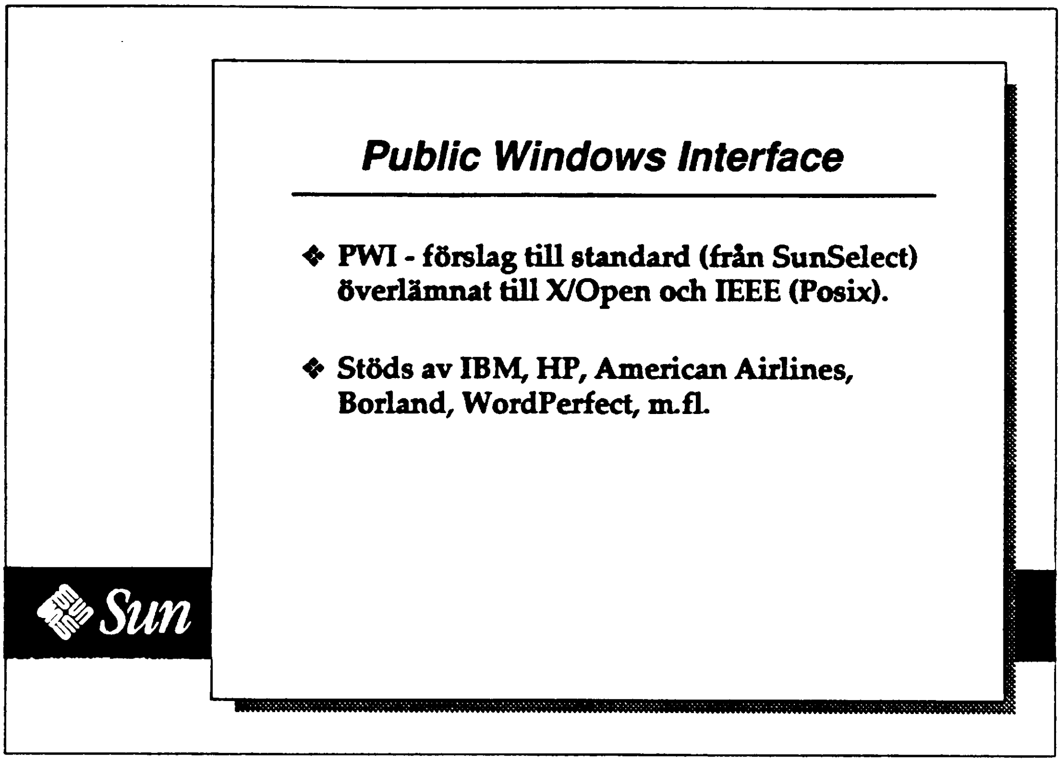
Sun

Endorsing ISVs

Acer, Amdahl Corporation, Apple Computer, Banyan Systems, Bentley Systems, Inc., Bull, Cadre Technologies, Inc., Computer Associates International, Inc., Convex Computer Corp., Cray Research, Inc., Chorus Systems, Computer, Digital Equipment Corp., Data General Corp., EDS Unigraphics, Encore Computer Corp., Fuji Xerox, 88open Consortium, Ltd., Fujitsu OSSI, HaL Computer Systems, Harris Corp., Hewlett-Packard Company, Hitachi, Ltd. Corp., Locus Computing Corporation, IBM Corp., ICL, Intel Corp., Motorola, NCR Corp., NEC, Novell-UNIX System Laboratories (USL), Oki Electric Industries Co.

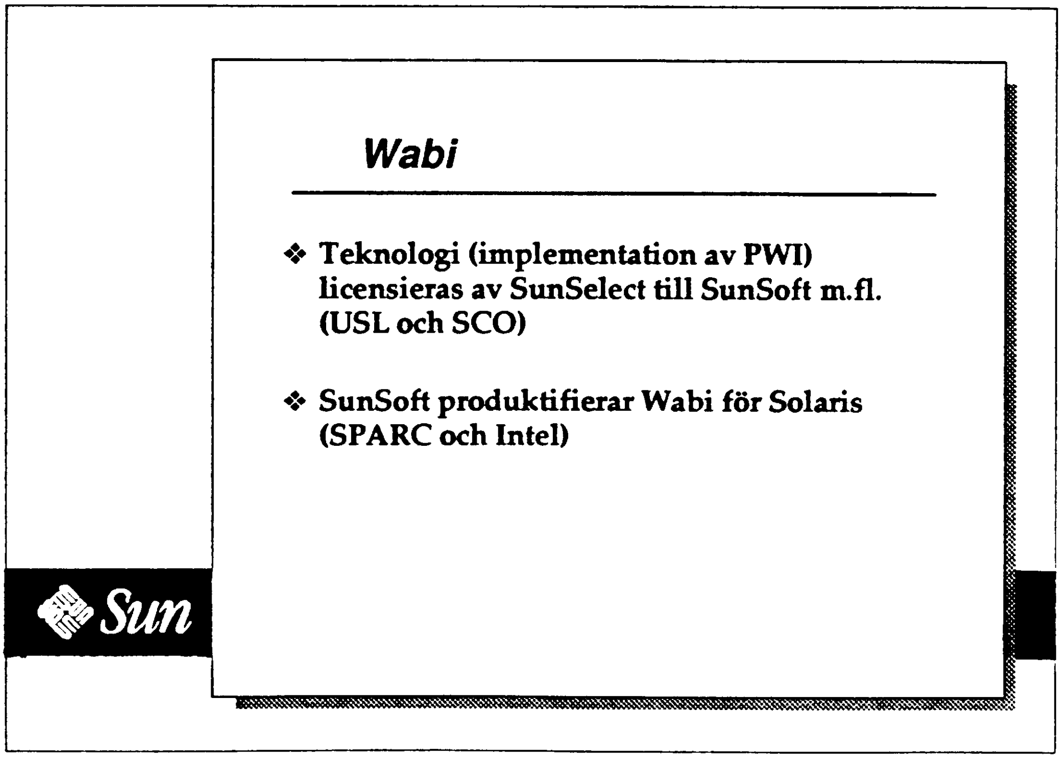
Ltd., Olivetti, the Open Software Foundation, the Precision RISC Organization, the PowerOpen Association, Inc., Pyramid, the Santa Cruz Operation, Sequent Computer Systems, Inc., Sequoia Systems, Inc., SHARP Corporation, Siemens-Nixdorf, Silicon Graphics, Inc., Sony Corp., SPARC International, Stratus Computer Inc., Sun Microsystem Computer Corp., SunSelect, SunSoft, Inc., Tadpole, Tandem Computer Corp., Texas Instruments, THOMPSON-CSF/CETIA, Toshiba, Unisys, UNIX International, Inc., VERITAS Software, Wang Laboratories, Inc. and WordPerfect Corporation

Diverse bilder till DKUUG/Bert Rubaszkin, Sun Microsystems AB/September 1993



Public Windows Interface

- ❖ PWI - förslag till standard (från SunSelect) överlämnat till X/Open och IEEE (Posix).
- ❖ Stöds av IBM, HP, American Airlines, Borland, WordPerfect, m.fl.



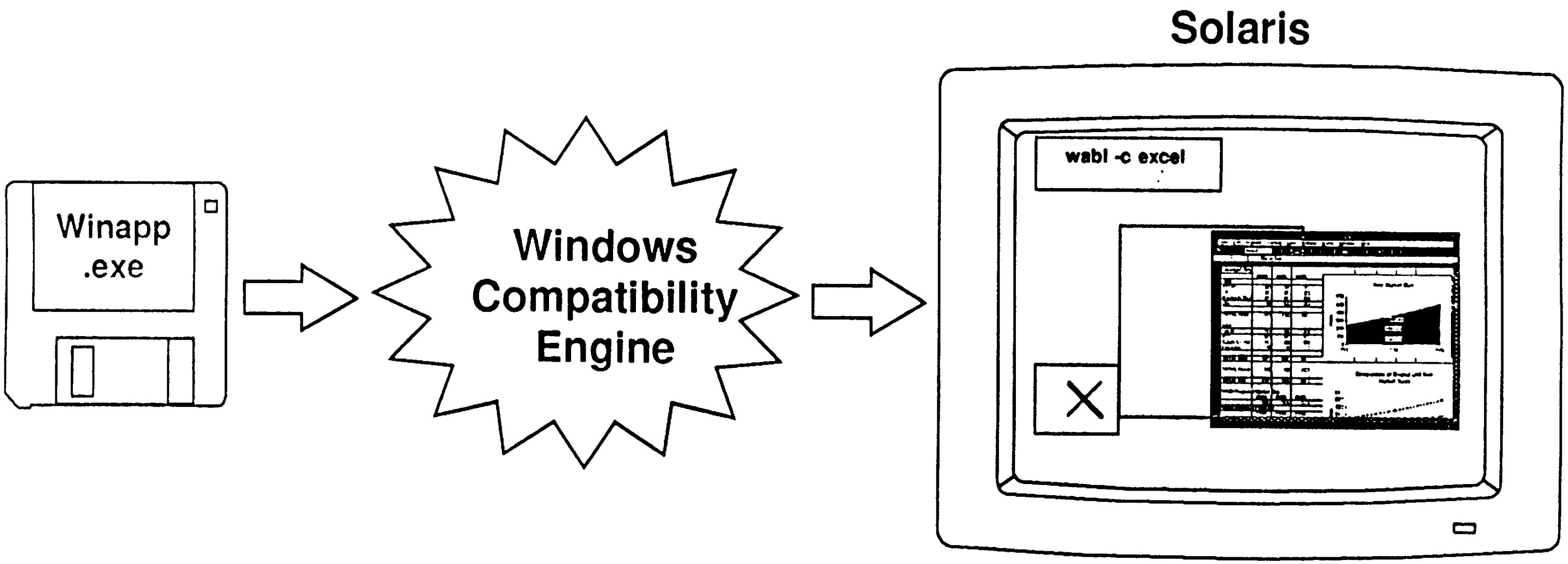
Wabi

- ❖ Teknologi (implementation av PWI) licensieras av SunSelect till SunSoft m.fl. (USL och SCO)
- ❖ SunSoft produktifierar Wabi för Solaris (SPARC och Intel)

What is Wabi?

- *Wabi* is a UNIX based technology that permits "shrink-wrapped" Windows 3.X applications to execute unmodified on X based desktops.

How does Wabi Work?

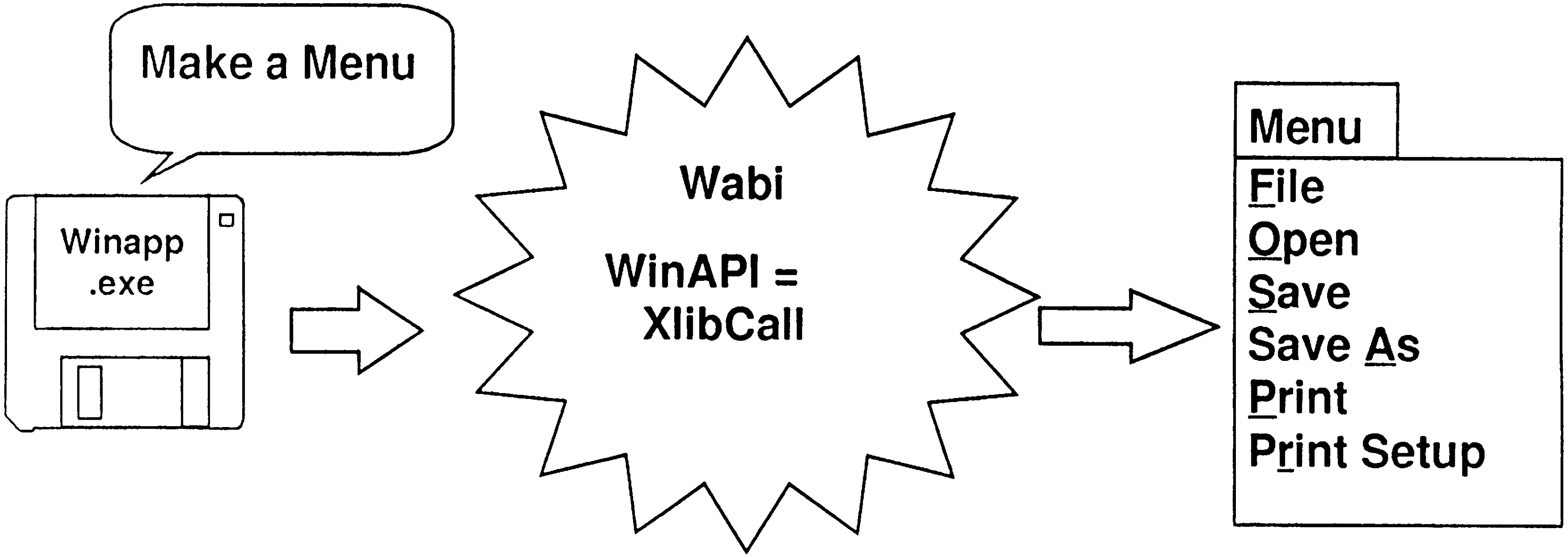


How does Wabi Work?

	UNIX		
PC	X86	RISC	
MS-Windows APPLICATION	MS-Windows APPLICATION	MS-Windows APPLICATION	
GRAPHICAL USER INTERFACE	Wabi	Wabi	
WINDOWS ENVIRONMENT	X-WINDOWS	X-WIN	CPU
DOS KERNEL	UNIX	UNIX	Simulator
DEVICE DRIVERS	DEVICE DRIVERS	DEVICE DRIVERS	
ROM BIOS	X86 CPU		
X86 CPU		RISC CPU	

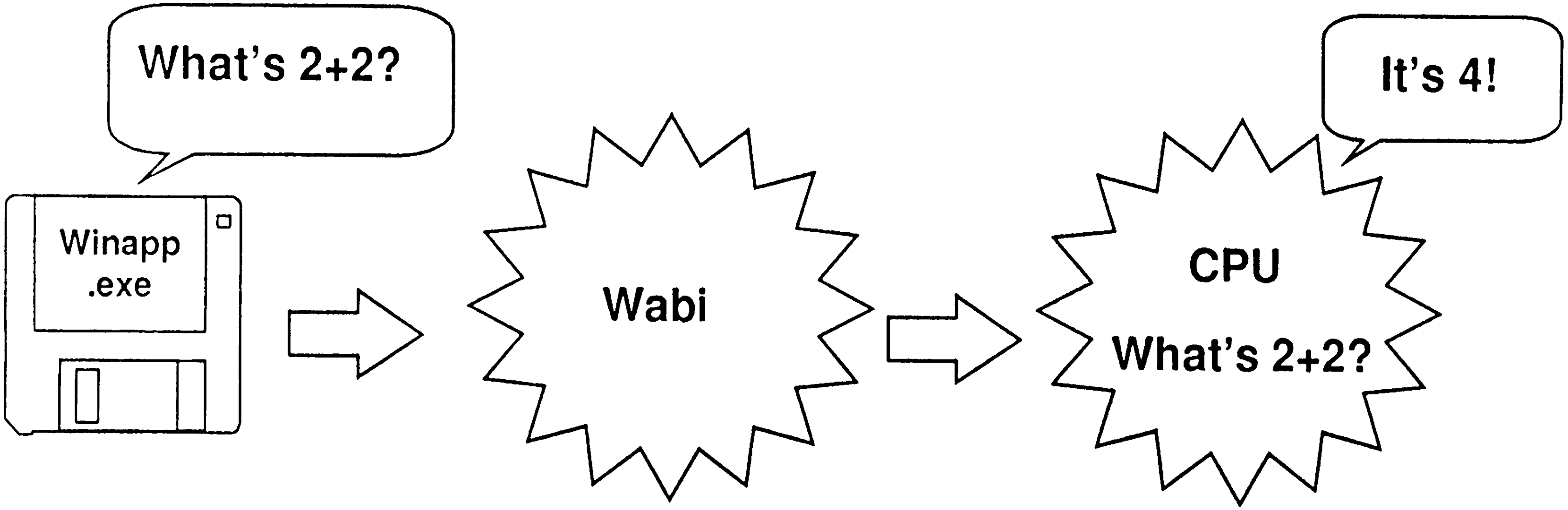
SunSelect
A Sun Microsystems, Inc Business

How does Wabi Work?



- Wabi translates Windows 3.1 API calls to Xlib calls
  - Window controls
  - Text
  - Graphics

How does Wabi Work?



What's 2+2?

Winapp .exe

Wabi

CPU

What's 2+2?

It's 4!

- Wabi passes Intel specific instructions to the Intel processor or the CPU simulator (RISC)

Wabi Features and Benefits

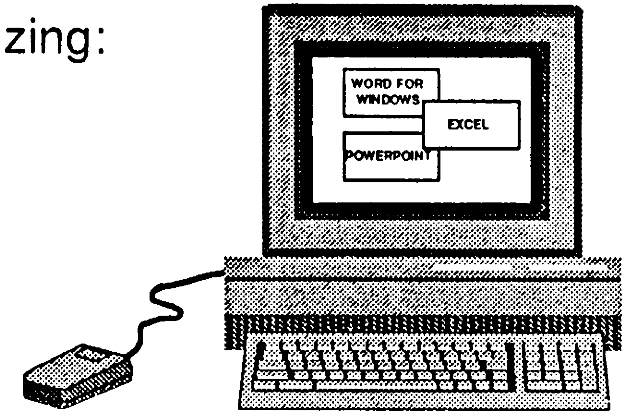
Feature	Benefit
•Windows Applications are translated to Native Desktop	•Support for multiple, simultaneous Windows applications •Native video performance •Native mouse and keyboard performance •Support for multiple users (i.e. X terminals)
•Windows Application's "richness" is preserved	•OLE/DDE works between Windows applications •Cut, Copy, and Paste supported •Drag and Drop Supported

Wabi Features and Benefits

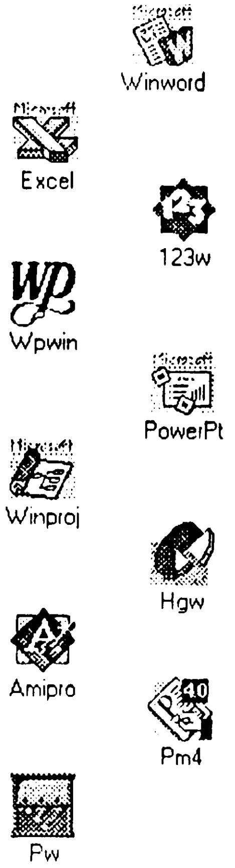
Feature	**Benefit**
*Tight integration with Native Desktop	*Support of Cut and Paste, Drag and Drop *Utilize Network Printers *Use native serial ports
* Virtual Disk Drives for Windows Applications	*Windows applications can access any UNIX file locally or on the network

Wabi 1.0

- Wabi 1.0 will support Windows 3.1 API compliant applications
- Wabi will run on Solaris 2.X SPARC and X86 systems
- SunSelect will “certify” top selling Windows applications
  - Expect many more applications will execute correctly through aggressive Beta program
- Application Manger (Program Manager replacement) that enables application launching and installation
- Windows based configuration tools
- Ability to launch DOS applications by utilizing:
  - SunPC
  - Locus' Merge
  - VP/ix



Wabi 1.0 Application List



- Word for Windows
- Excel for Windows
- Lotus 123 for Windows
- WordPerfect for Windows
- PowerPoint
- Project for Windows
- Harvard Graphics for Windows
- AmiPro for Windows
- Aldus Pagemaker
- ProComm Plus for Windows
- CorelDraw
- Paradox for Windows
- Windows 3.1 Applets

SunSelect
A Sun Microsystems, Inc. Business

Rev5a

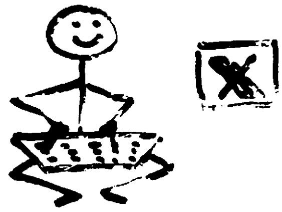
Wabi Futures

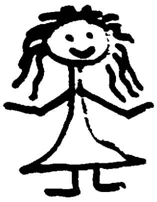
- Future Possibilities
  - Better UNIX/Windows Integration
  - More Certified Applications
  - Tight Integration with SunPC Product
  - Windows Socket API Support
  - Win-32s API Support
  - Multimedia Support
    - » Sound
    - » CDROM

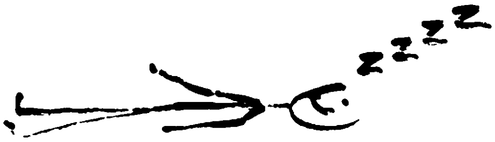
Brian Eberhardt
SuperUsers a/s

Døgnet har 24 timer !

6 timer arbejde

6 timer hobby 

6 timers fritid 

6 timers søvn 

Will Cosy COSE end the UNIX wars ?

Hvad ønsker vi ?

Operativsystem

Grafik

GUI

Utilities { bruger
udvikler
systemadm

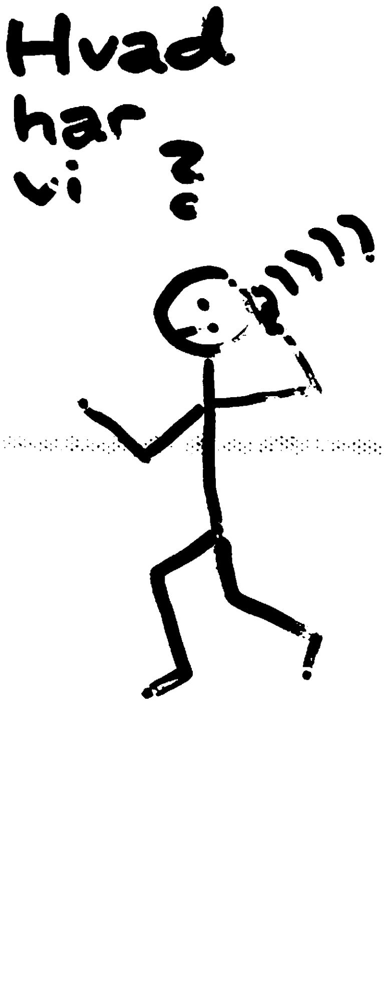
UNIX V.4 tilsat OSF (POSIX X/OPEN)

X (X Consortium X/Open)

⋮

MOTIF / OW

Utilities i OW



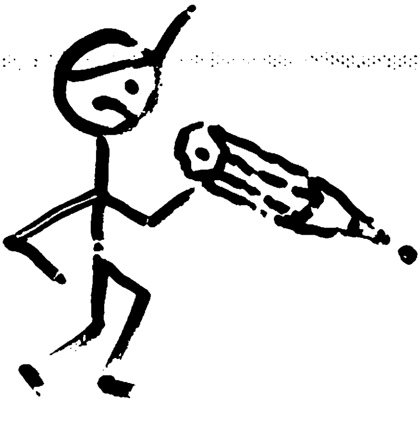
KONKLUSION:

Enige om

- basalt OS
- hvordan man tegner en prik

Uenige om

- GUI, firkantede/runde knapper
- Utilities, filemanager
  (pwd, cd, mv, cp ...)
  editor (vi)
  kalender (cal)
  mailsystem (mail)
  ⋮



Hvem ka' ?

UNIX 1969 - 1993 : 24 år ?

Over 100 UNIX leverandører ?

Nej !

Bill ka' !

NT ⇨ COSE

Microsoft Vapor-OS ⇨ UNIX VaporStandard

Microsoft har bevist talent (1M MSW per md.)

Kloge spåmænd siger: 1996 NT
6% server
20% client

Hvem skal frygte ?

- Novel (66%) NetOS
- SCO (20%) PC-UNIX
- Leverandører af servere
(Pentium * 12 ⇨ 100 vis MIPS)
- Alle (NT på INTEL, ALPHA,
MIPS, CLIPPER, SPARC...)

UNIX-argumenter:

UNIX is

- proven, stable & mature
- open system
- from any vendor
- more scalable

Gates svar:

Hvilken version af UNIX ?

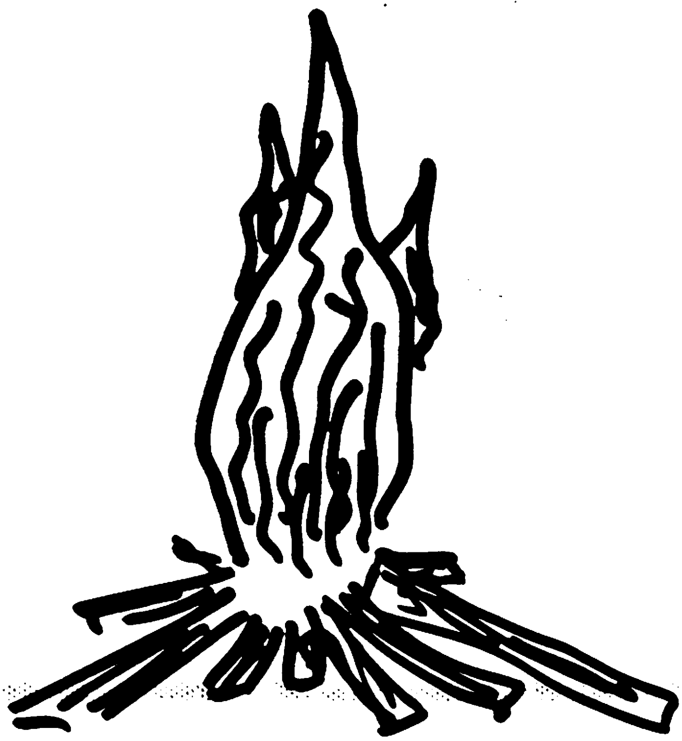
Gates har ret,
prøv at tænde for en:

- SUN Runde Knapper, OW desktop
- HP Prop. desktop
- IBM Anden Prop. desktop
- SGI Tredje Prop. desktop
- SCO .....

Prøv at flytte et program
på tværs at leverandør

Prøv at flytte et program
internt på een leverandør

- SUN SunOS → Solaris
- HP HP9000 700 → 800



COSE annonceres

- Common Open System Environment

- Een del heraf: CDE Common Desktop Environment

- HP, IBM, SCO, SUN, UNIVEL & USL (DEC på vej ind)

CDE

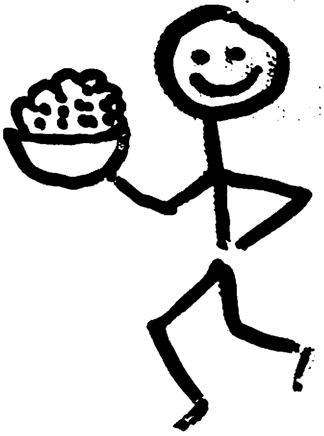
UNIX V.4 tilsat OSF

✓ X

og nu

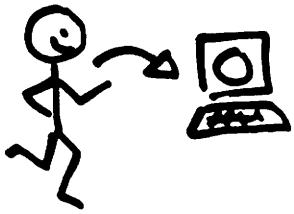
✓ GUI (MOTIF)

✓ Utilities

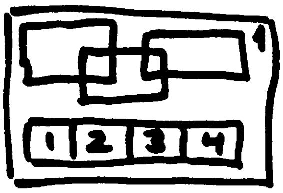


CDE er:

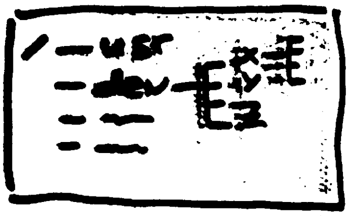
Login



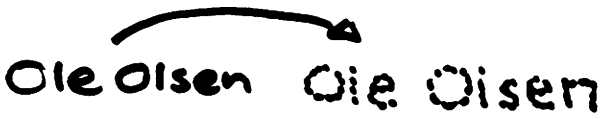
FrontPanel



FileManager

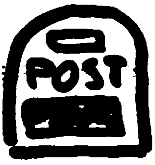


Cut/Paste

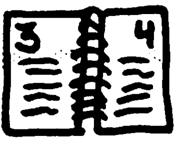


Drag/Drop

Mail



Kalender



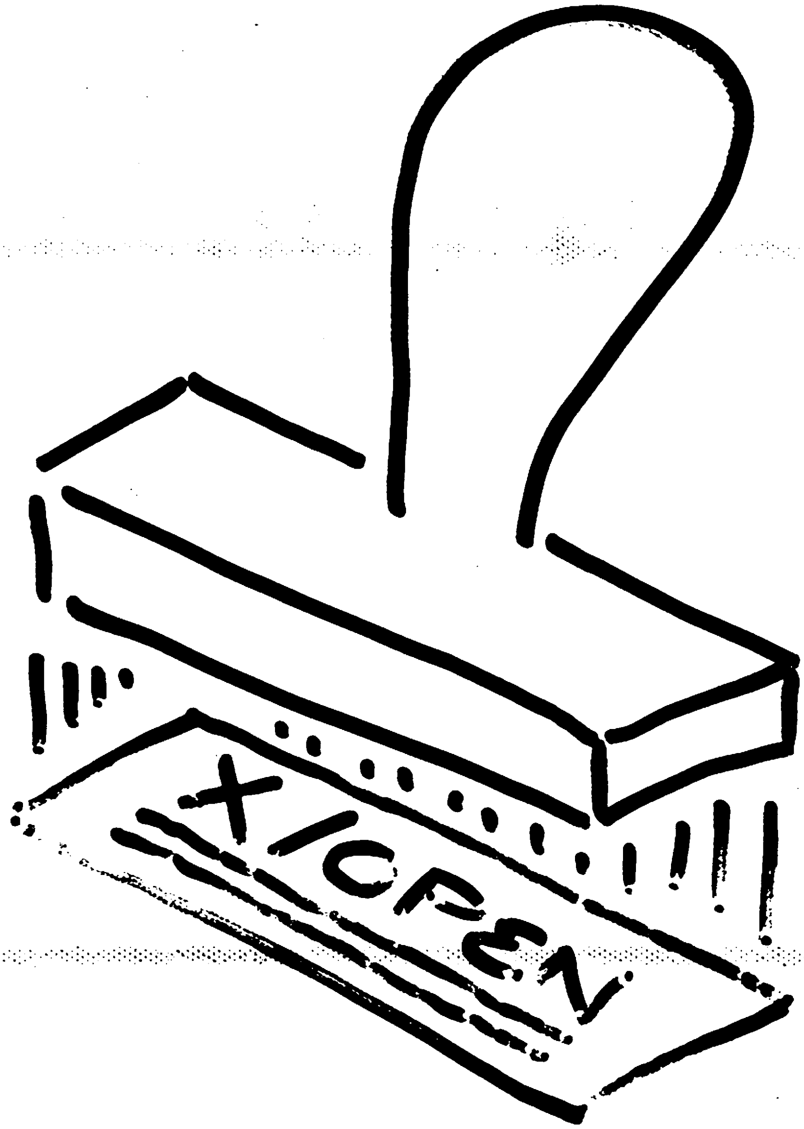
Text

Icon

Help System
med
Hypertext

Dialog
Script
m. GUI

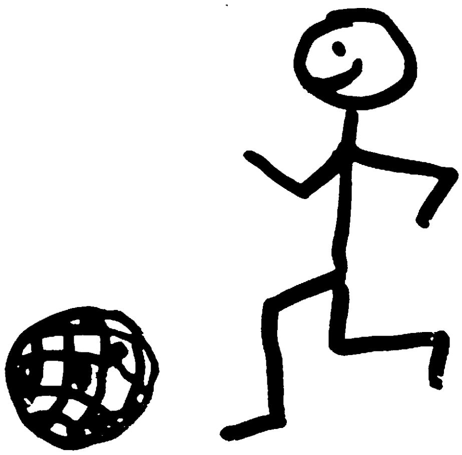
Developers tools
GUI builders
Test-tools
Konfig af appls


X/OPEN

UNiX

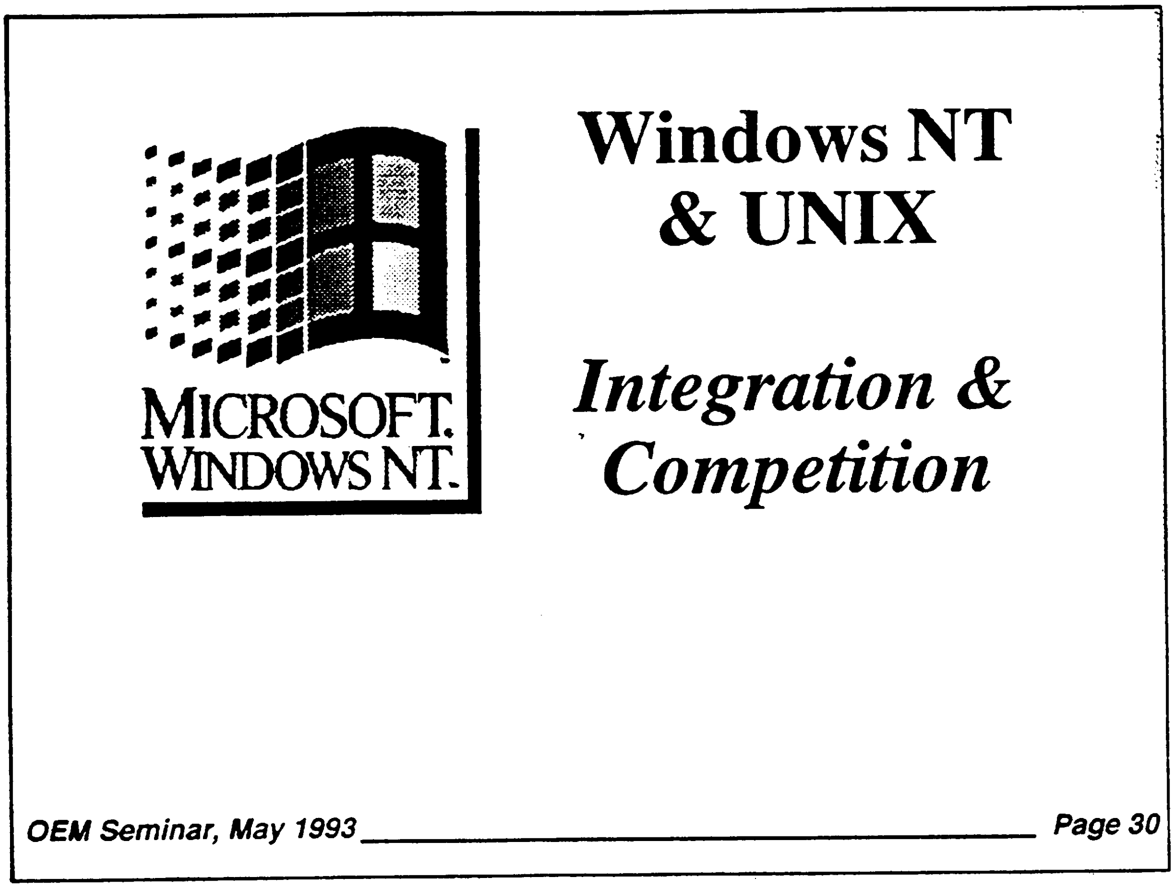
versus

NT



Hvorfor SuperUsers
og
NT ?

- Vi tør ikke lade være !
- Ny "åben" OS-teknologi
- "Let" at overføre viden
- Debut : NT i Parken !!

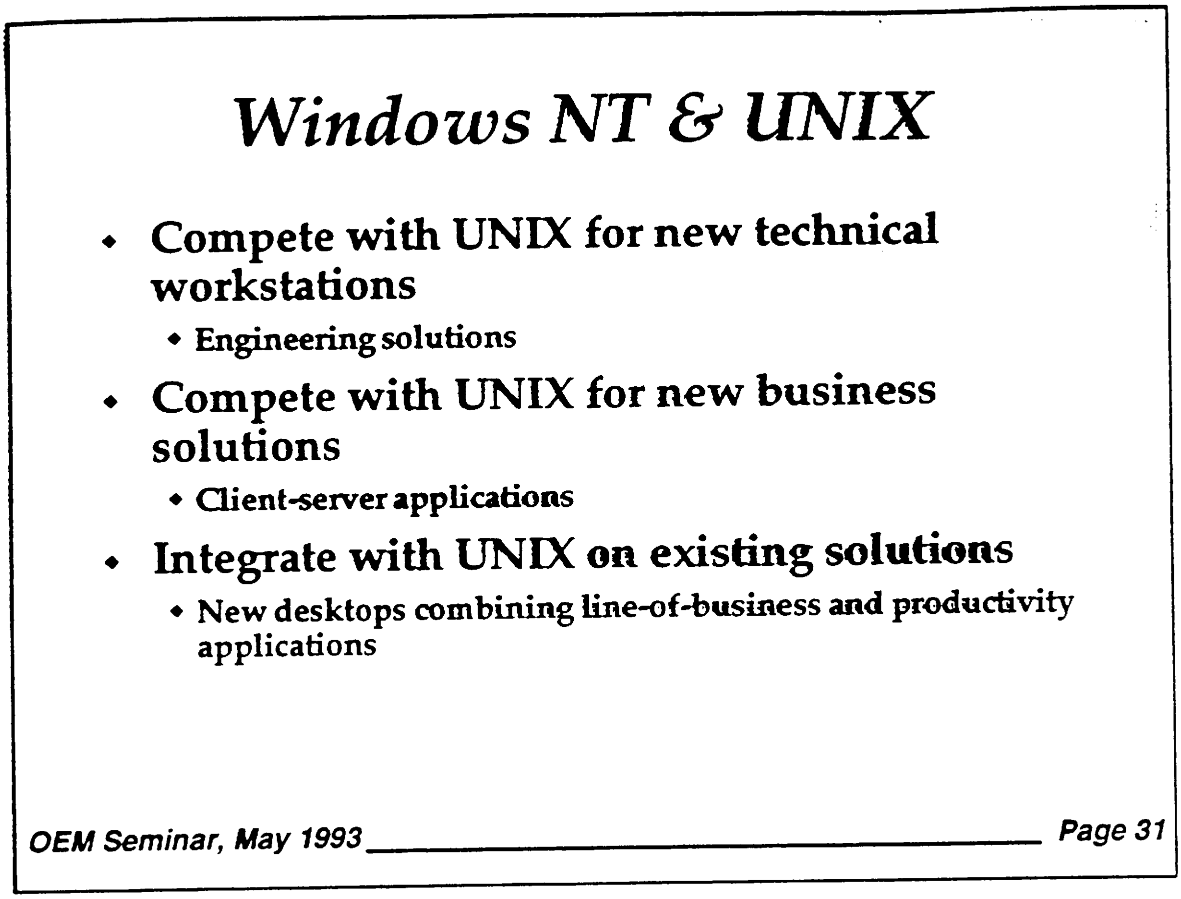


MICROSOFT.
WINDOWS NT.

Windows NT & UNIX

Integration & Competition

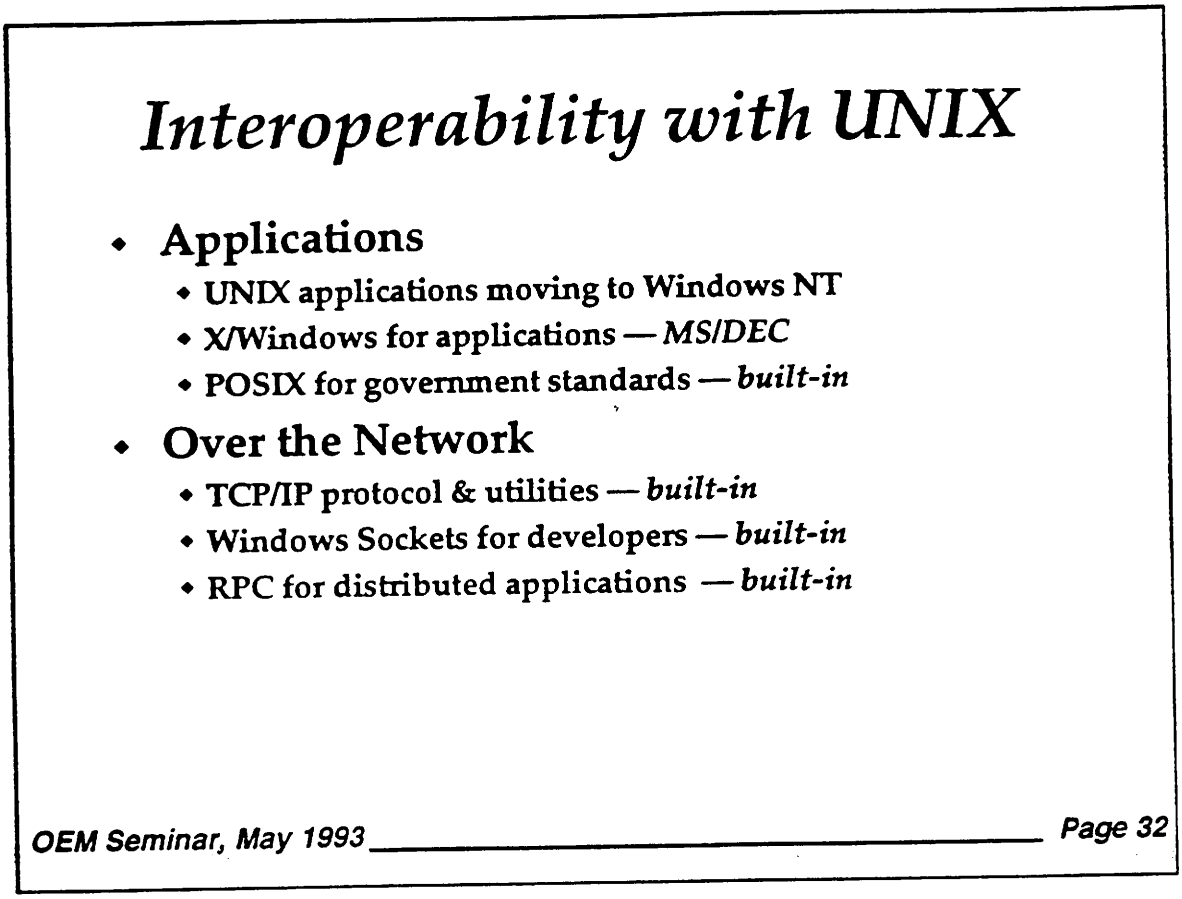
OEM Seminar, May 1993 Page 30



Windows NT & UNIX

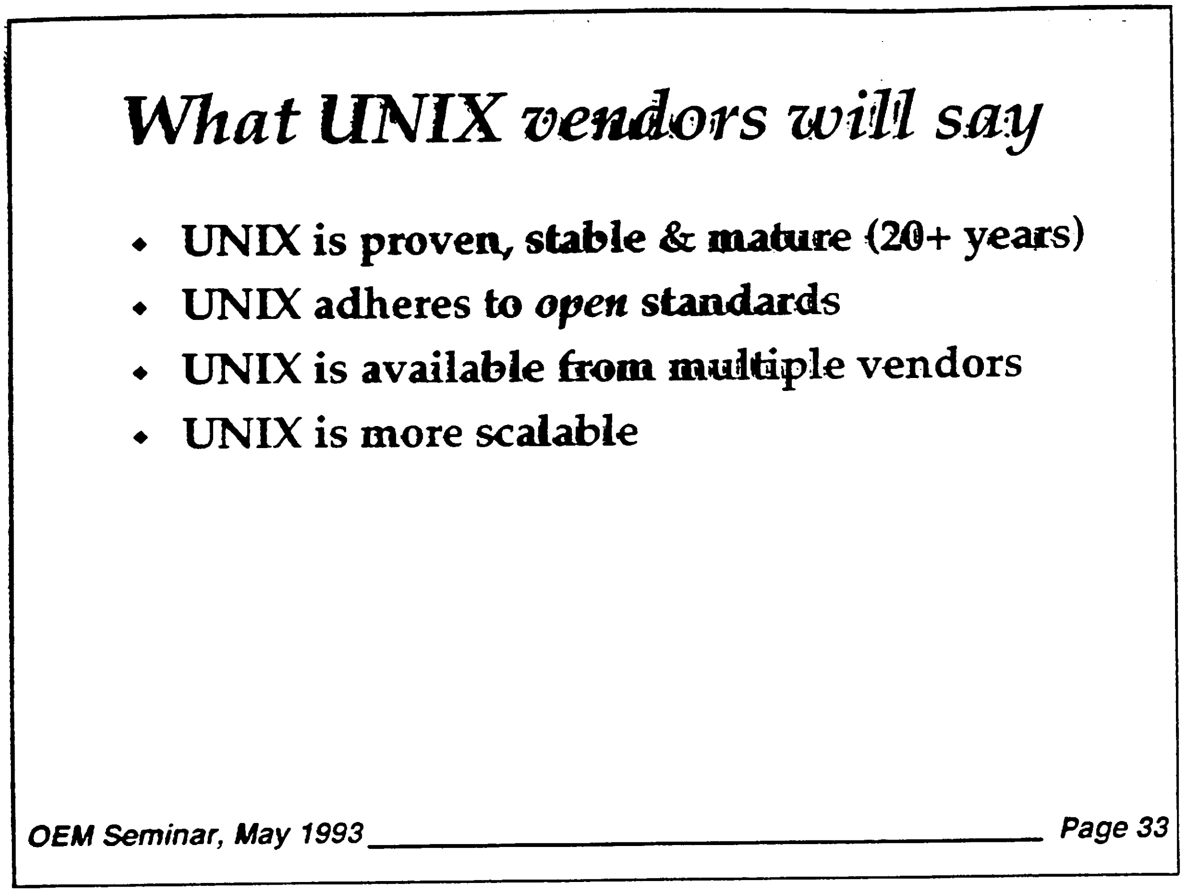
- Compete with UNIX for new technical workstations
  - Engineering solutions
- Compete with UNIX for new business solutions
  - Client-server applications
- Integrate with UNIX on existing solutions
  - New desktops combining line-of-business and productivity applications

OEM Seminar, May 1993 Page 31



Interoperability with UNIX

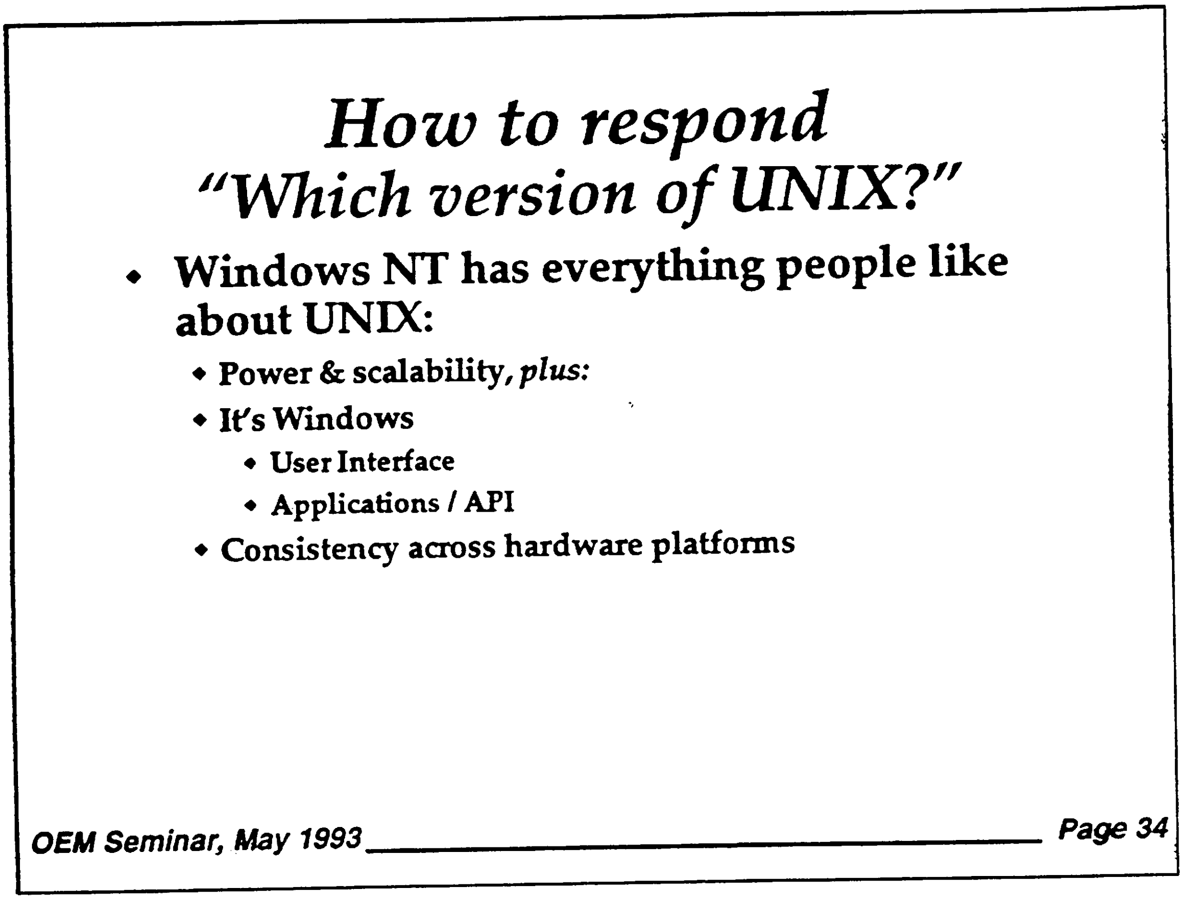
- Applications
  - UNIX applications moving to Windows NT
  - X/Windows for applications — *MS/DEC*
  - POSIX for government standards — ***built-in***
- Over the Network
  - TCP/IP protocol & utilities — ***built-in***
  - Windows Sockets for developers — ***built-in***
  - RPC for distributed applications — ***built-in***



What UNIX vendors will say

- UNIX is proven, stable & mature (20+ years)
- UNIX adheres to *open* standards
- UNIX is available from multiple vendors
- UNIX is more scalable

OEM Seminar, May 1993 *Page 33*

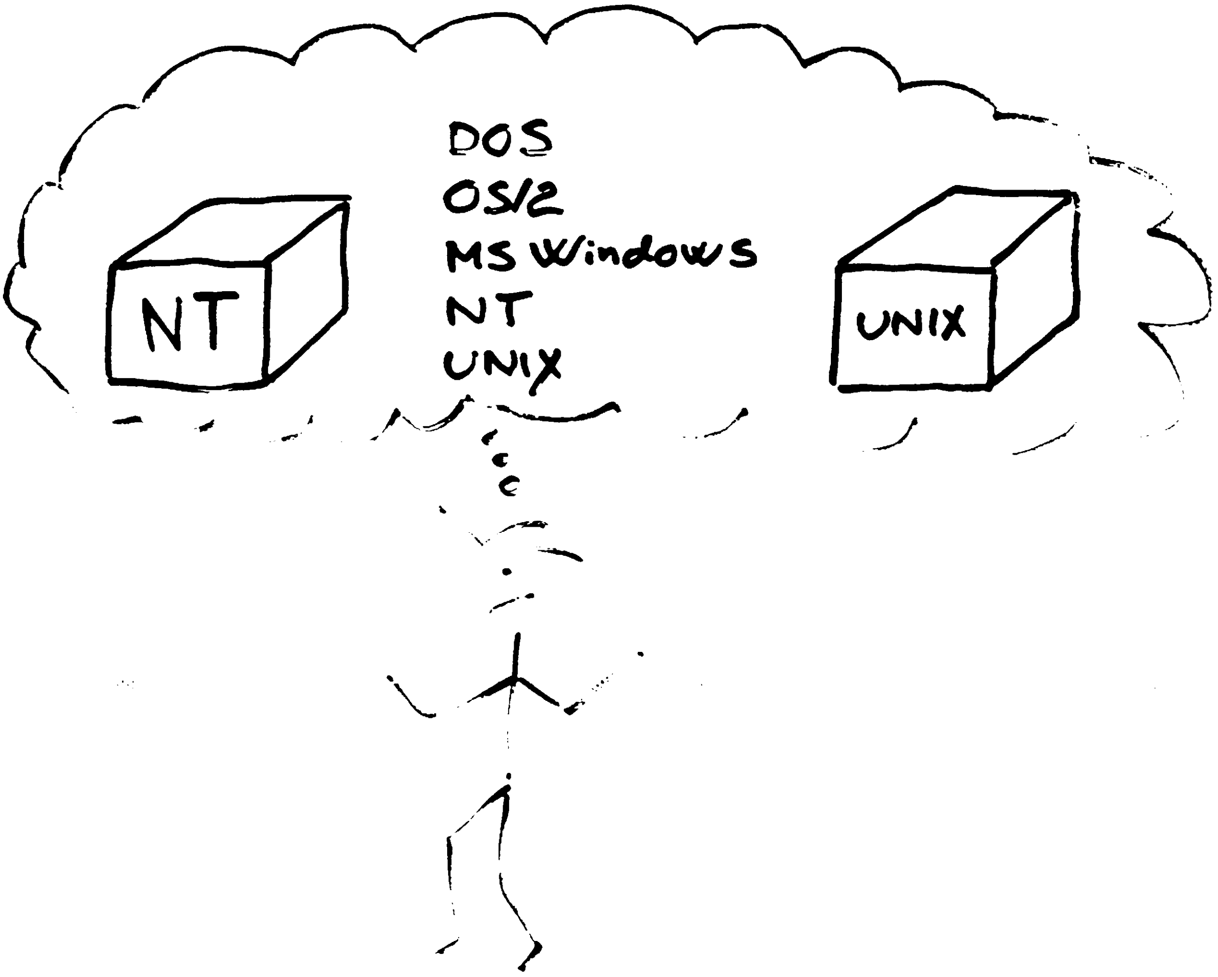


How to respond "Which version of UNIX?"

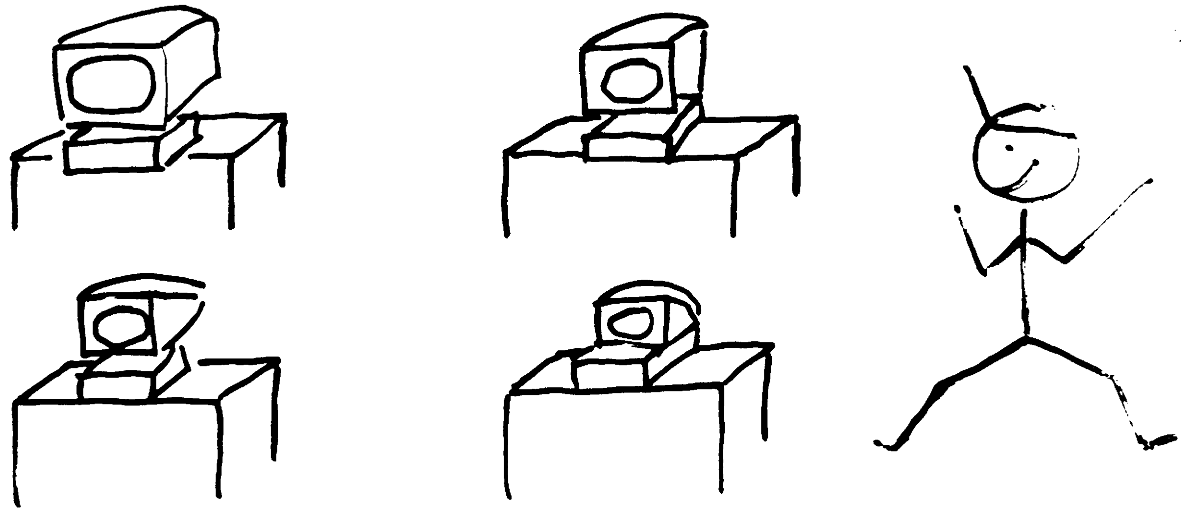
- Windows NT has everything people like about UNIX:
  - Power & scalability, *plus:*
  - It's Windows
    - User Interface
    - Applications / API
  - Consistency across hardware platforms

OEM Seminar, May 1993 ________ *Page 34*

Hvad kan eksekveres hvor ?



Multibruger



Krav til hw !

Husk: Alt er GUI-baseret !

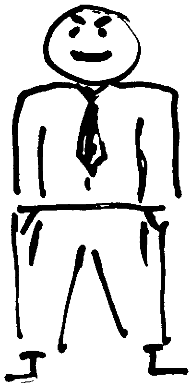
Performance

Ressourcer

Bruger

Editor
Mail
Calendar
Desktop

SystemAdm



Backup
Brugeradm
Diskadm
Terminaladm
Printeradm
Netværk
SystemConfig

Udvikler

Utilities
Portering

System Administration

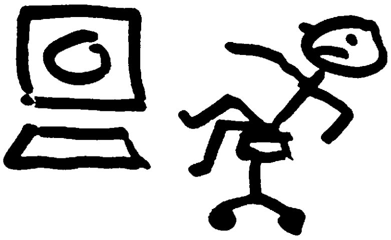
NT har et sæt homogene utilities

- User / Group
- Disk / Printer
- Net
- Backup
- sikkerhed
- Overvågning
- Performance

Men ....

NT har et problem

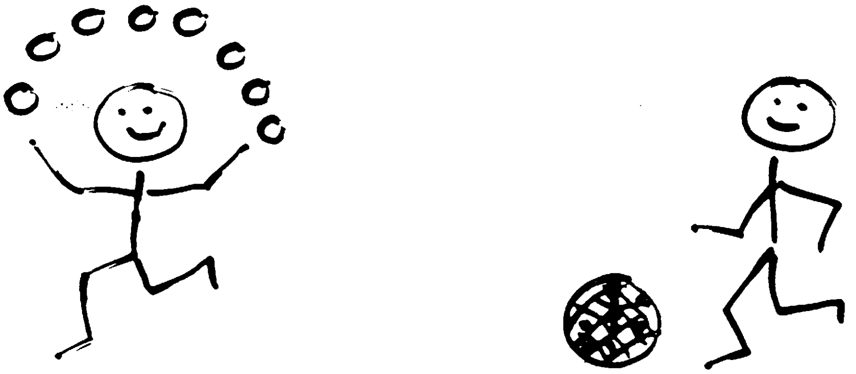
- Ingen "rigtig shell"
- Ingen "utilities"



GUI:

X/Motif ↔ NT

Netværksbaseret LowLevel Protocol ↔ Direkte i Videoram

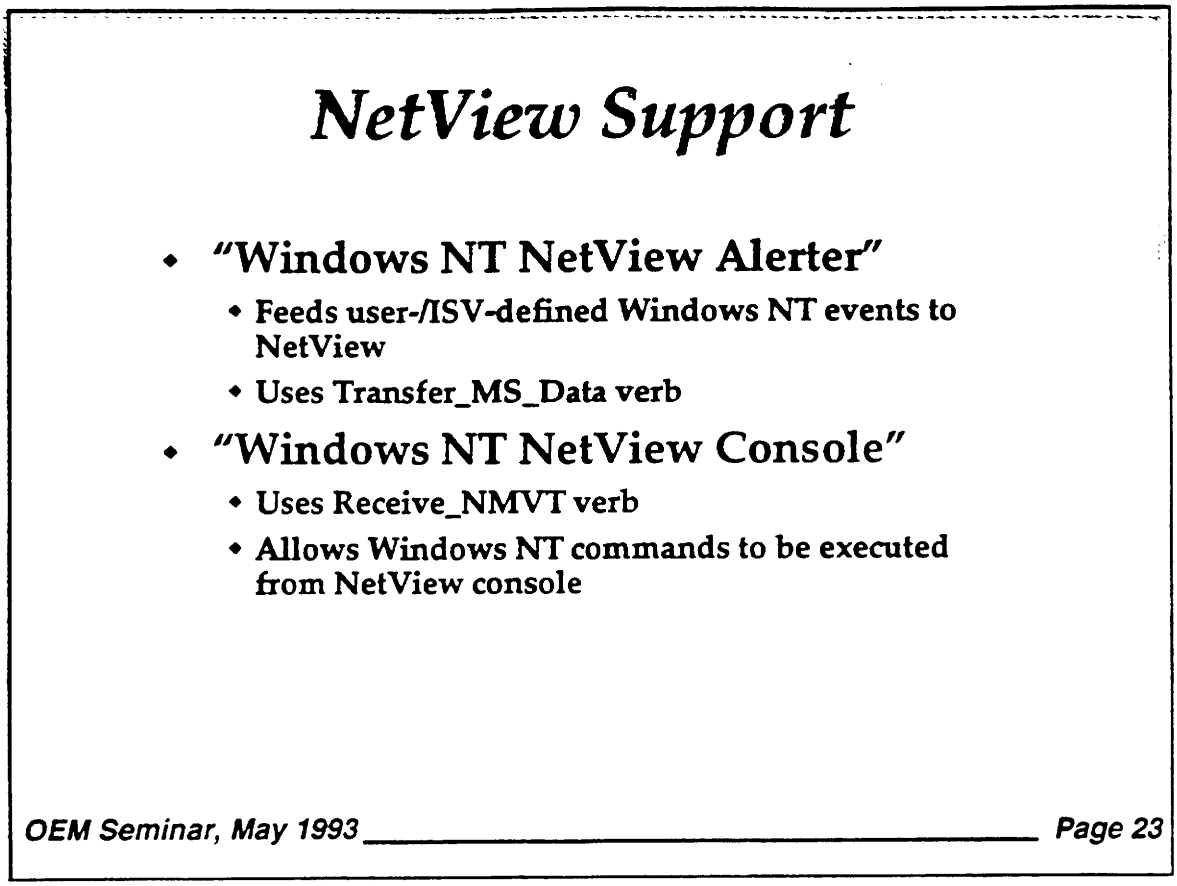


NETVÆRK

NFS eller ....

RPC (DCE)

TCP/IP



NetView Support

- "Windows NT NetView Alerter"
  - Feeds user-/ISV-defined Windows NT events to NetView
  - Uses Transfer_MS_Data verb
- "Windows NT NetView Console"
  - Uses Receive_NMVT verb
  - Allows Windows NT commands to be executed from NetView console

INTEGRATION

PC/NFS

LAN Manager

Novel Netware

TCP/IP

NT X-server

WABI

routine, a message is sent to the server that implements the API routine via the NT executive's *local procedure call* (LPC) facility, a locally optimized message-passing mechanism. The server replies by sending a message back to the caller.

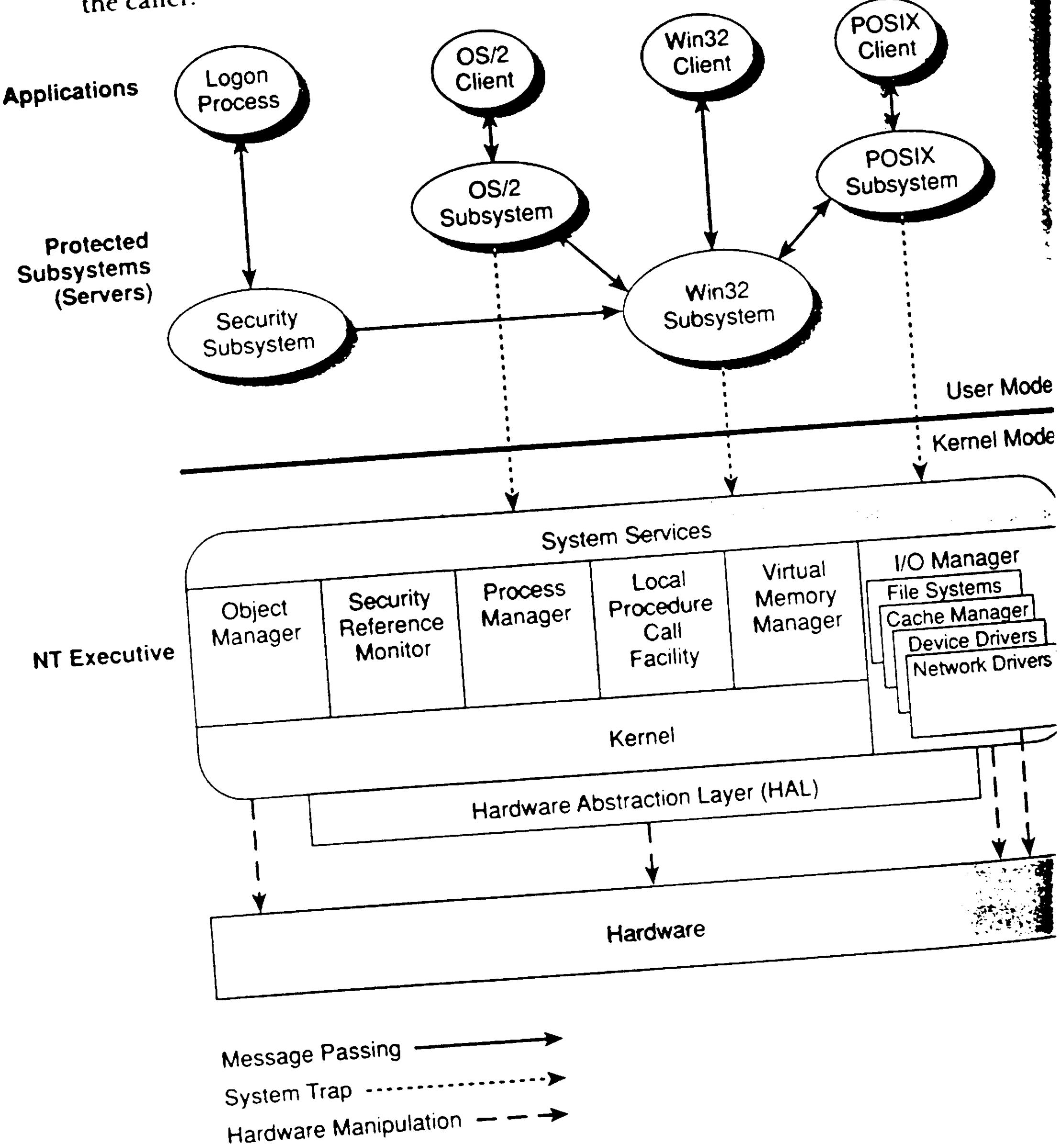


Figure 2-6. Windows NT Block Diagram

Win32 API'en

Preemptive Multitasking

- Threads / Processer

NT Objects og Handles

- User Objects
- GDI Objects
- Kernel Objects

Process Synkronisering

- Events
- Semaforer
- Mutex

IPC

- Sendmessage
- Named Pipes
- Mapped Files
- Mailslots

Filhåndtering

Portering

X/Open.

- posix 1 + 2
- Div Prog. Lang.
- X, COSE
- NFS

Novell Netware.

UUCP, Simple Mail Protocol

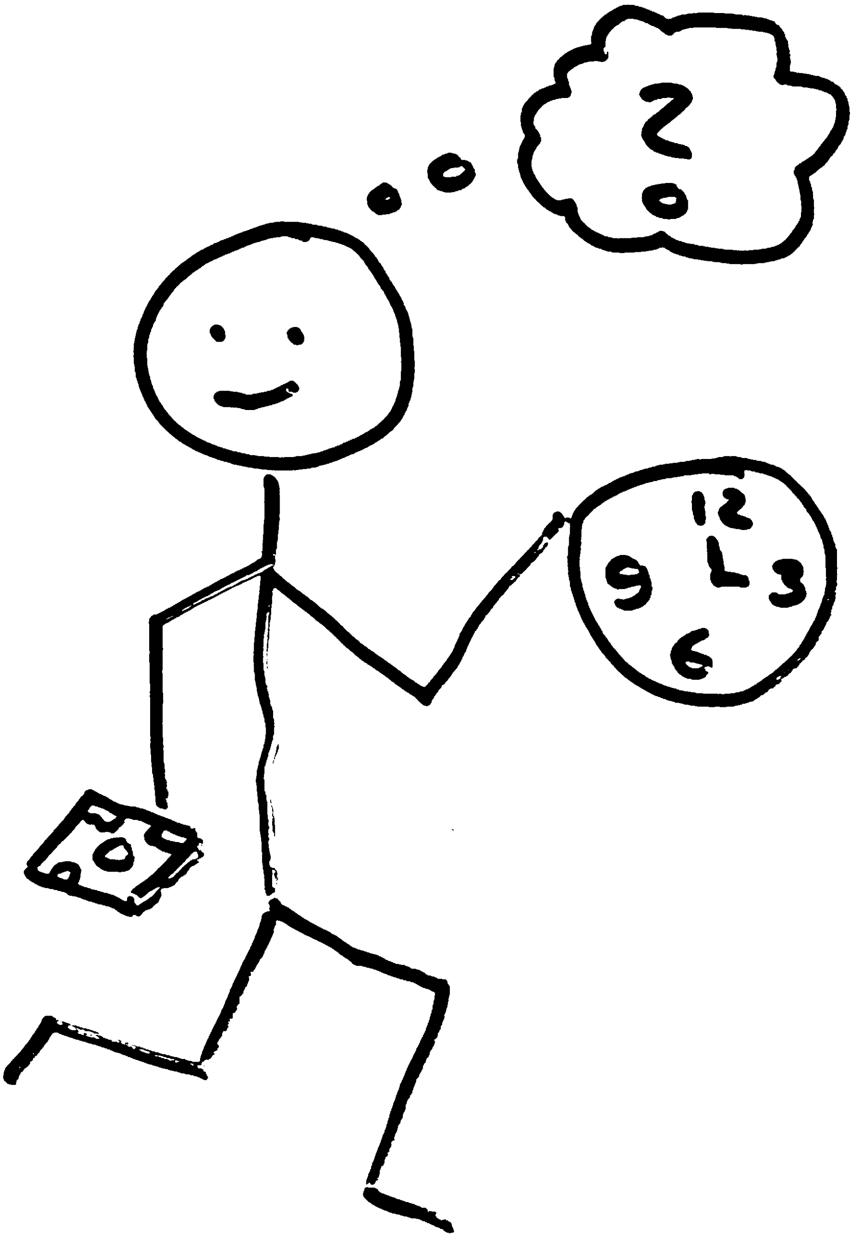
Device drivers

System Adm.

IPC

C++ !!

Portering til NT



```
main ()
{
  long time ;
  if ( ! Maskine running )
    time += 3 ;
  if ( ! Maskine running NT )
    time += 2 ;
  if ( ! NMAKE )
    time += 3 ;
  if ( ! C )
    Exit(1) ;
  if ( filhåndtering )
     small ;
  if ( preces håndtering )
     medium ;
  if ( GUI )
     Xlarge ;
  if ( networking )
     Xlarge ;

   .
   .
   .
```

Hvor Mange ?

150 M	DOS
25 M	MS Windows
1.5 M	OS/2
5 M	UNIX

NT

Konkurrenter ?

NT ↔ OS/2
Novell Netware
UNIX

som
client / server

INDHOLD

1. **Hvorfor portere til NT**

   1.1 **Udbredelse**
   1.2 **Nye markeder**
   1.3 **Signaler om åbenhed**

2. **Hvad er portering**

   2.1 **Fra/Til**
   2.2 **ABI'er på OS'er**
   2.3 **API'er på OS'er**
   2.4 **Ønskeliste**

3. **Hvad opfylder/mangler NT**

   3.1 **Binær portering (ABI)**
   3.2 **Source portering (API)**

4. **Værktøjer**

   4.1 **Standarder**
   4.2 **Virtuelle toolkits**
   4.3 **NT Tools**
   4.4 **Tredje Part Tools**

5. **Hvad vil det sige at være "Porteret til NT"**

1. Hvorfor portere til NT

1.1 Udbredelse

	DOS	MS-WIN	OS/2	NT	MAC	Next Step	Unix
DK	500K	100K			50K	200	25K
Total	125M	17M	2M	70M	10M		5M
Total per md		1M					

Source: Brian International 1993

1. Hvorfor portere til NT

1.2 Nye Markeder

✓ Fra DOS/Windows til "et ægte operativsystem"

✓ Fra UNIX til "Gates-tilstande"

✓ Fra mini'er & mainframes til downsized miljø

1. Hvorfor portere til NT

1.3 Signaler om åbenhed

✓ Første non-UNIX PC-operativsystem med indbygget:

POSIX, TCP/IP, RPC, ...

✓ Med EF-indkøbsdirektiver etc. er dette også kommercielt interessant.

2.1 Fra/Til

- ✓ Fra non-NT til NT

- ✓ Fra NT til NT

2.2 ABI'er på OS'er

- ✓ DOS applikationer
  - Tilgår hardware
  - Tilgår ej hardware
- ✓ MS Windows applikationer
  - Well-behaved 16-bit applikationer
  - Tilgår "internal code" i 16-bit GDI
  - Tilgår "internal code" i 16-bit window manager
  - Brug af DLL
- ✓ OS/2 applikationer
  - 1.x tegnorienterede applikationer
  - 2.x grafiske applikationer (PM)
- ✓ POSIX-applikationer
  - For NT: NEJ

2.3 API'er på OS'er

DOS Windows

OS/2 UNIX

NT

2.3 API'er på OS'er

NT's Win32 API sammenlignet med Windows

✓	Widened:	1.070 stk
✓	Changed:	26 stk
✓	Dropped:	137 stk
✓	New:	633 stk

2. Hvad er Portering

2.4 Ønskeliste

✓ X/Open XPG4:

POSIX.1, POSIX.2
C, Cobol, Pascal, Fortran, ADA, SQL
X Window, COSE
NFS, RPC

✓ Netværk/kommunikation:

LAN Manager
Novell Netware
TCP/IP
ISO/OSI
Sockets
TLI
UUCP, SendMailProtocol

✓ Ømme punkter:

Threads
C++
Device Drivers
System Administration

3. Hvad opfylder/mangler NT

3.1 Binær Portering

✓ Goto 2.1 og 2.2

✓ Opsummering, binær kompatibel når:

NT:	På samme processor-type
DOS:	Non hardware access
Windows:	Well-behaved
OS/2:	1.x tegnorienteret
POSIX:	Nej

3. Hvad opfylder/mangler NT

3.2 Source Portering, hvad opfyldes

✓ Win32

✓ Win32s

✓ POSIX.1

✓ C, (C++)

✓ LAN Manager

✓ RPC

✓ Sockets

✓ TCP/IP

3. Hvad opfylder/mangler NT

3.2 Source Portering, hvad mangler

✓ X/Open, d.v.s.

POSIX.2

Diverse programmeringssprog

X, COSE (bl.a. OSF/Motif)

NFS

✓ Novel Netware

UUCP, Sendmail Protocol

✓ Ømme punkter:

C++ standard
Device Driver standard
System Administration standard

✓ Man kan kun anvende eet subsystem per applikation

4. Værktøjer

4.1 Standarder

- ✓ Bedste værktøj til portering

4. Værktøjer

4.2 Virtuelle Toolkits

- ✓ Der findes 3. parts produkter, som kan danne GUI til stort set enhver platform.

4.3 NT Tools, til portering

✓ Ved portering fra Windows til NT:

PORTTOOL

Finder steder i Windows-kode som skal rettes

4.3 NT Tools, Debugging

✓ Process/Task Viewer

✓ Spy

5. Porteret til NT

5. Hvornår er et program "porteret til NT" ?

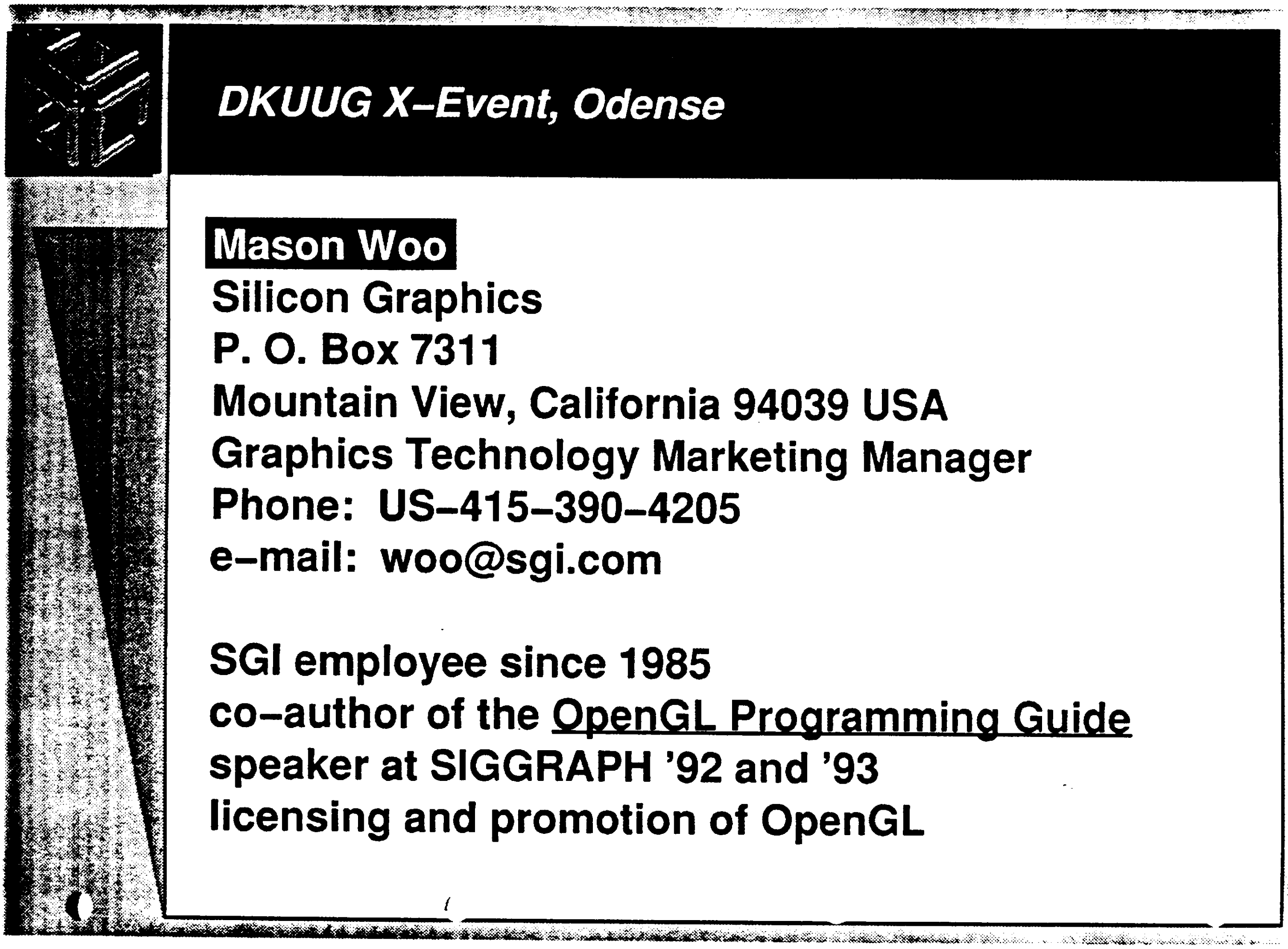
✓ Når et DOS, Windows, OS/2 1.x program **emuleres** under NT ?

✓ Når et Windows 16-bit program er **gen-compileret** på (læs tilpasset til) NT til 32 bit ?

✓ Når en vilkårlig portering **skrives om** til at:

Anvende "**Asynkron I/O**" under NT ?

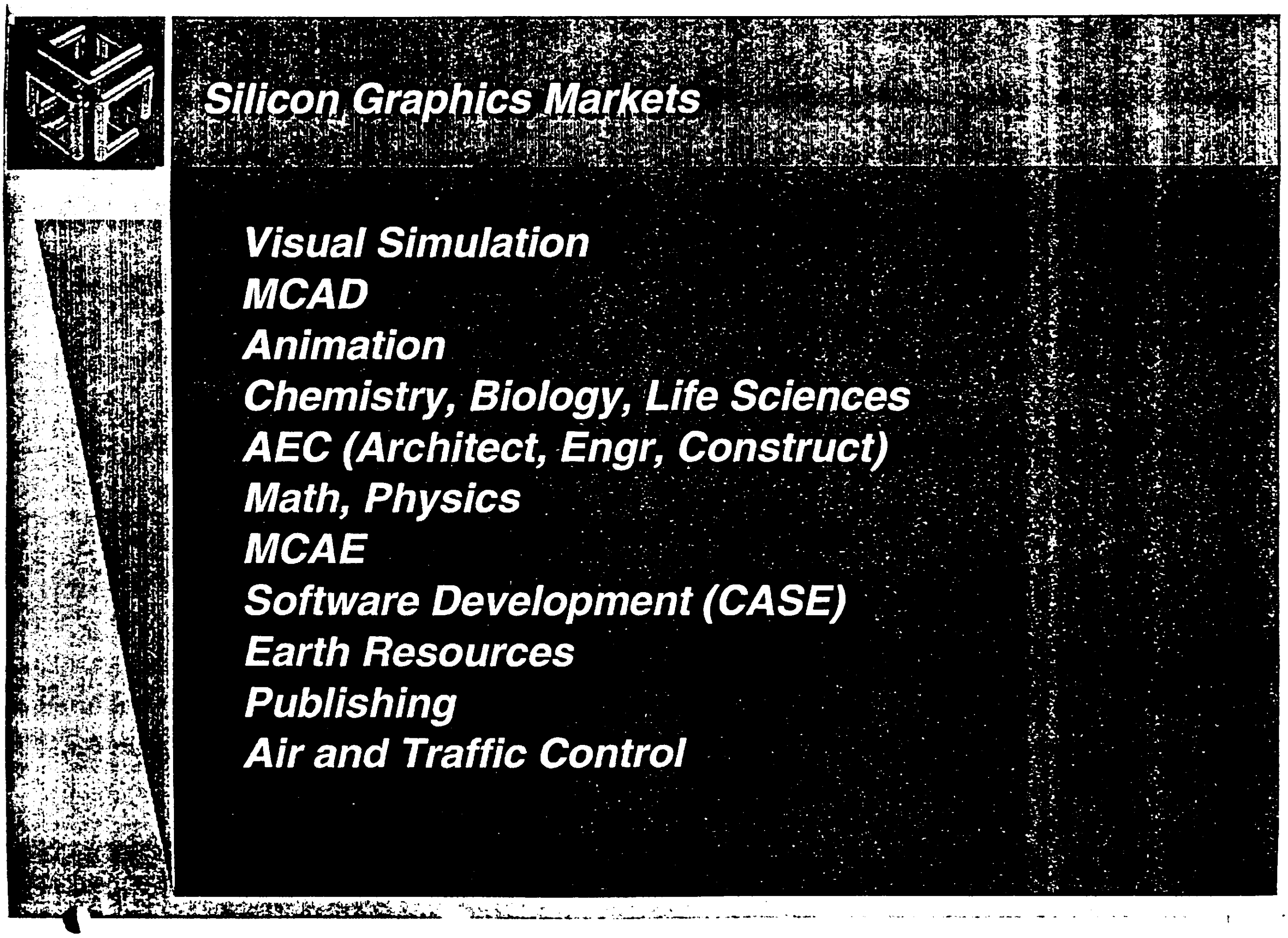
Anvende **threads** (og dermed udnytte multi-CPU) ?



DKUUG X-Event, Odense

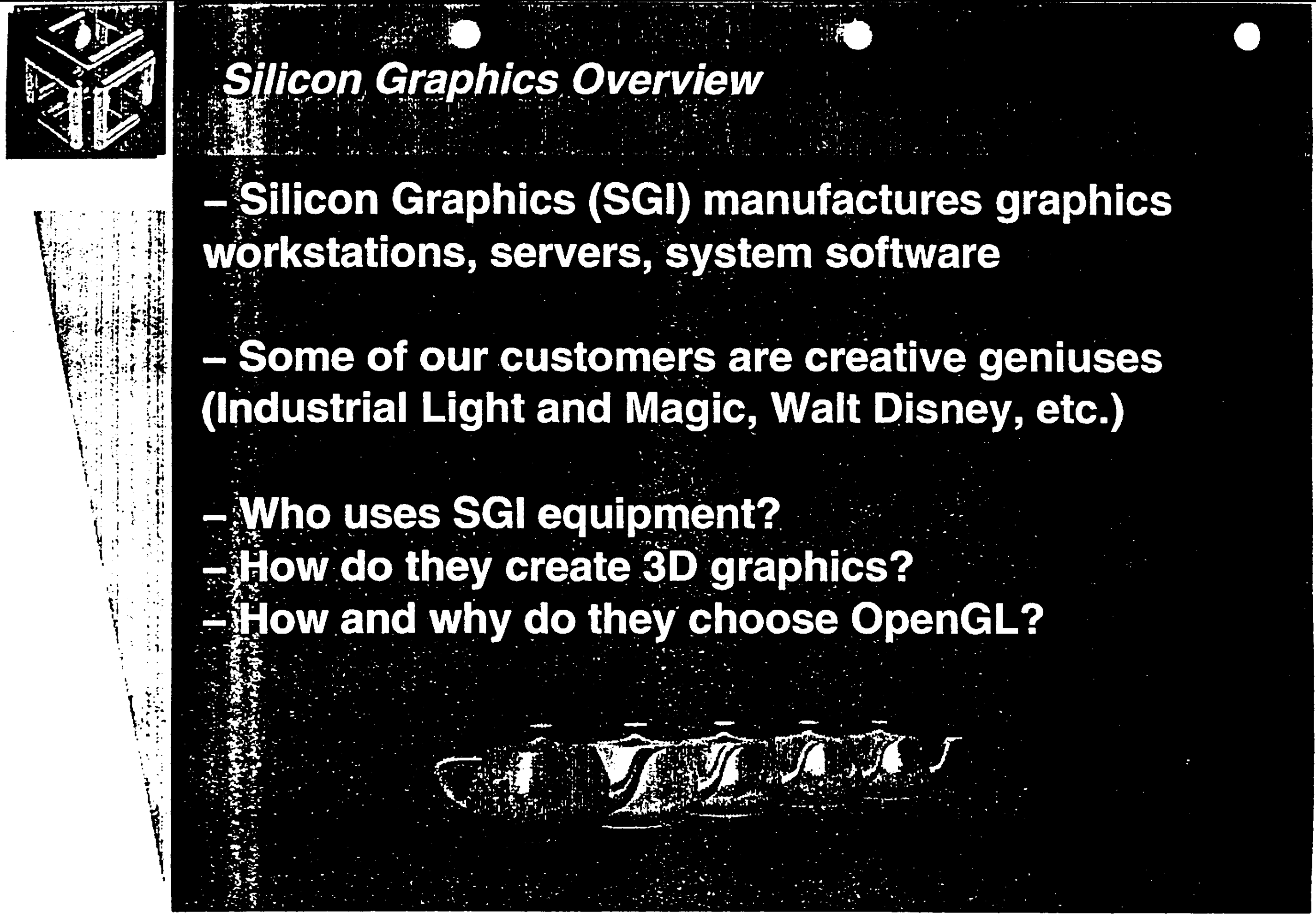
Mason Woo
Silicon Graphics
P. O. Box 7311
Mountain View, California 94039 USA
Graphics Technology Marketing Manager
Phone: US-415-390-4205
e-mail: woo@sgi.com

SGI employee since 1985
co-author of the OpenGL Programming Guide
speaker at SIGGRAPH '92 and '93
licensing and promotion of OpenGL



Silicon Graphics Markets

Visual Simulation
MCAD
Animation
Chemistry, Biology, Life Sciences
AEC (Architect, Engr, Construct)
Math, Physics
MCAE
Software Development (CASE)
Earth Resources
Publishing
Air and Traffic Control

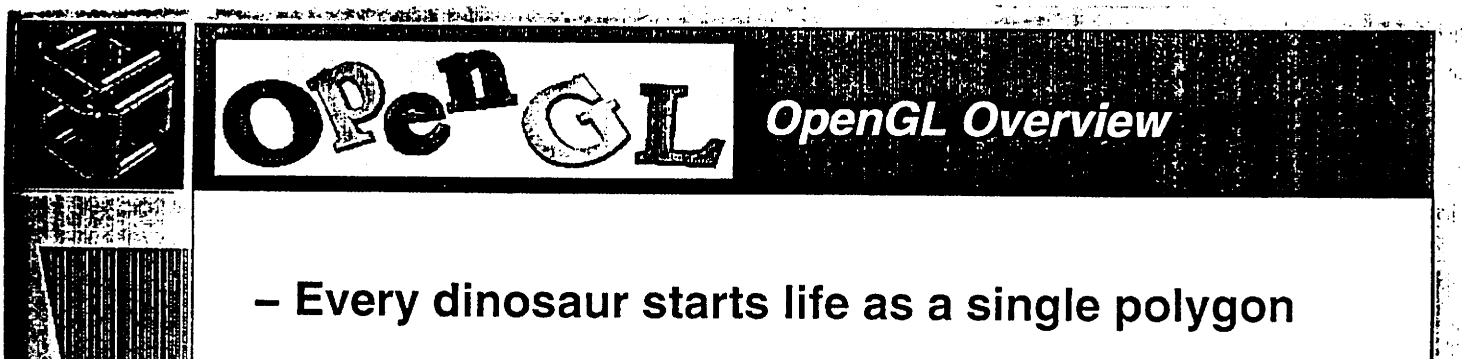


Silicon Graphics Overview

– Silicon Graphics (SGI) manufactures graphics workstations, servers, system software

– Some of our customers are creative geniuses (Industrial Light and Magic, Walt Disney, etc.)

– Who uses SGI equipment?
– How do they create 3D graphics?
– How and why do they choose OpenGL?



OpenGL Overview

– Every dinosaur starts life as a single polygon

– OpenGL is an Application Programming Interface (API) for 2D and 3D graphics

– OpenGL works with the X Window System/Motif/UNIX or Microsoft Windows NT

– OpenGL is supported by > 20 companies

– How do you learn more about OpenGL?

Choosing a Standard

How Do I Make a Decision Among Graphics Standards?

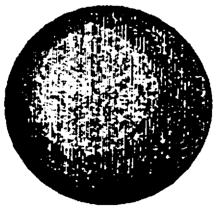
- "Is it widely available?
Does application code port easily?"

- "Is it powerful enough?
Will new features be added?"

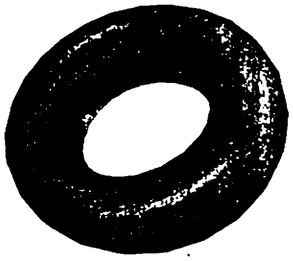
- "Will hardware evolve faster than the software?"

- "Will I ever be able to understand it?"

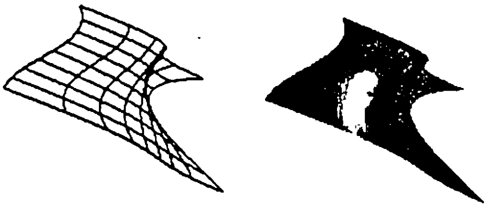
OpenGL Features



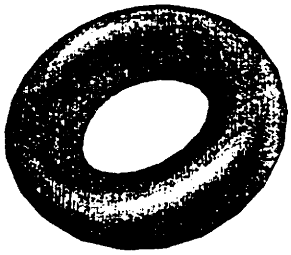
Gouraud Shaded,
Lighted Sphere



Gouraud Shaded,
Lighted Torus



Wireframe and
Lighted NURBS surface



Texture Mapped,
Lighted Torus

OpenGL Features

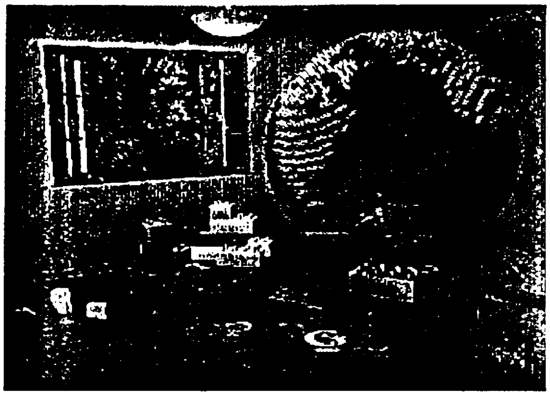
What Makes OpenGL Stand Out?

- OpenGL is the Evolution of the IRIS GL

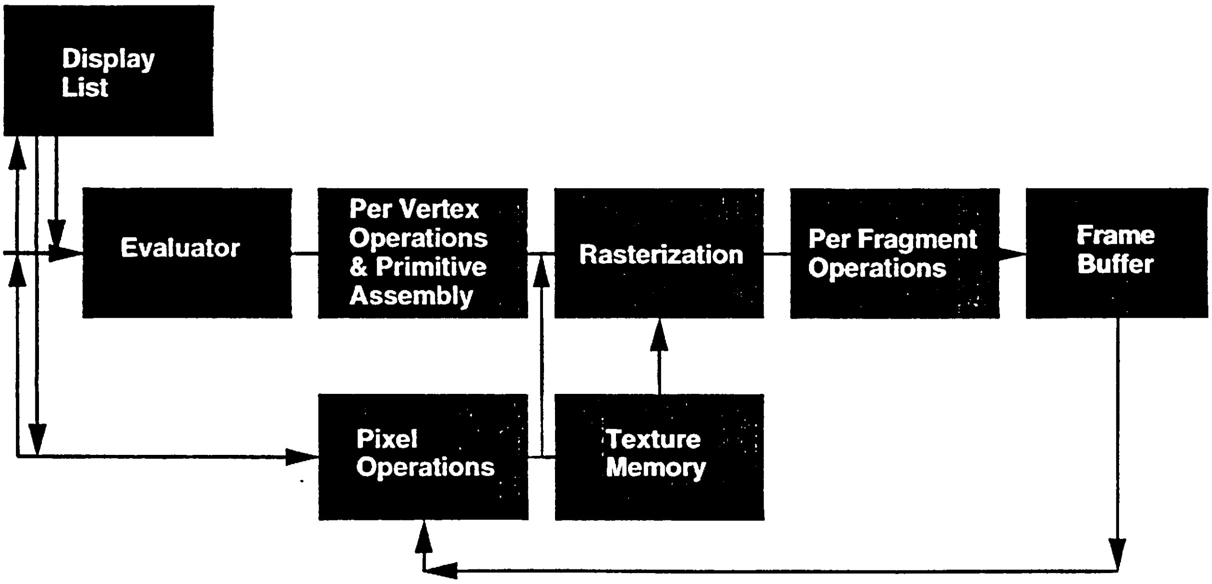
IRIS GL has 10 years of use; 1500 IRIS GL applications

- Governed by small Review Board; flexible, but not radical changes
- Enhanced portability/interoperability

clear conformance requirements; most functionality is mandatory



- 3D graphics features
  - geometric primitives
  - matrix transformations
  - display lists
  - lighting
  - alpha
  - texture mapping
  - anti–aliasing
  - fog
  - stencil planes
  - accumulation buffer



Display List

Evaluator

Per Vertex Operations & Primitive Assembly

Rasterization

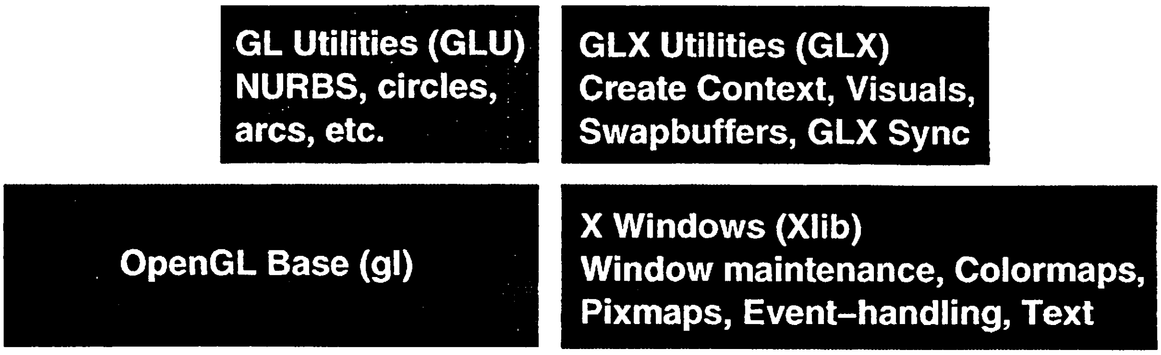
Per Fragment Operations

Frame Buffer

Pixel Operations

Texture Memory

OpenGL API Hierarchy



GL Utilities (GLU)
NURBS, circles, arcs, etc.

GLX Utilities (GLX)
Create Context, Visuals, Swapbuffers, GLX Sync

OpenGL Base (gl)

X Windows (Xlib)
Window maintenance, Colormaps, Pixmaps, Event–handling, Text

- OpenGL depends upon the implementations's window/input/event system
- for X, addition of GLX extension for distributed computing
- for other environments, other extensions (Windows NT)

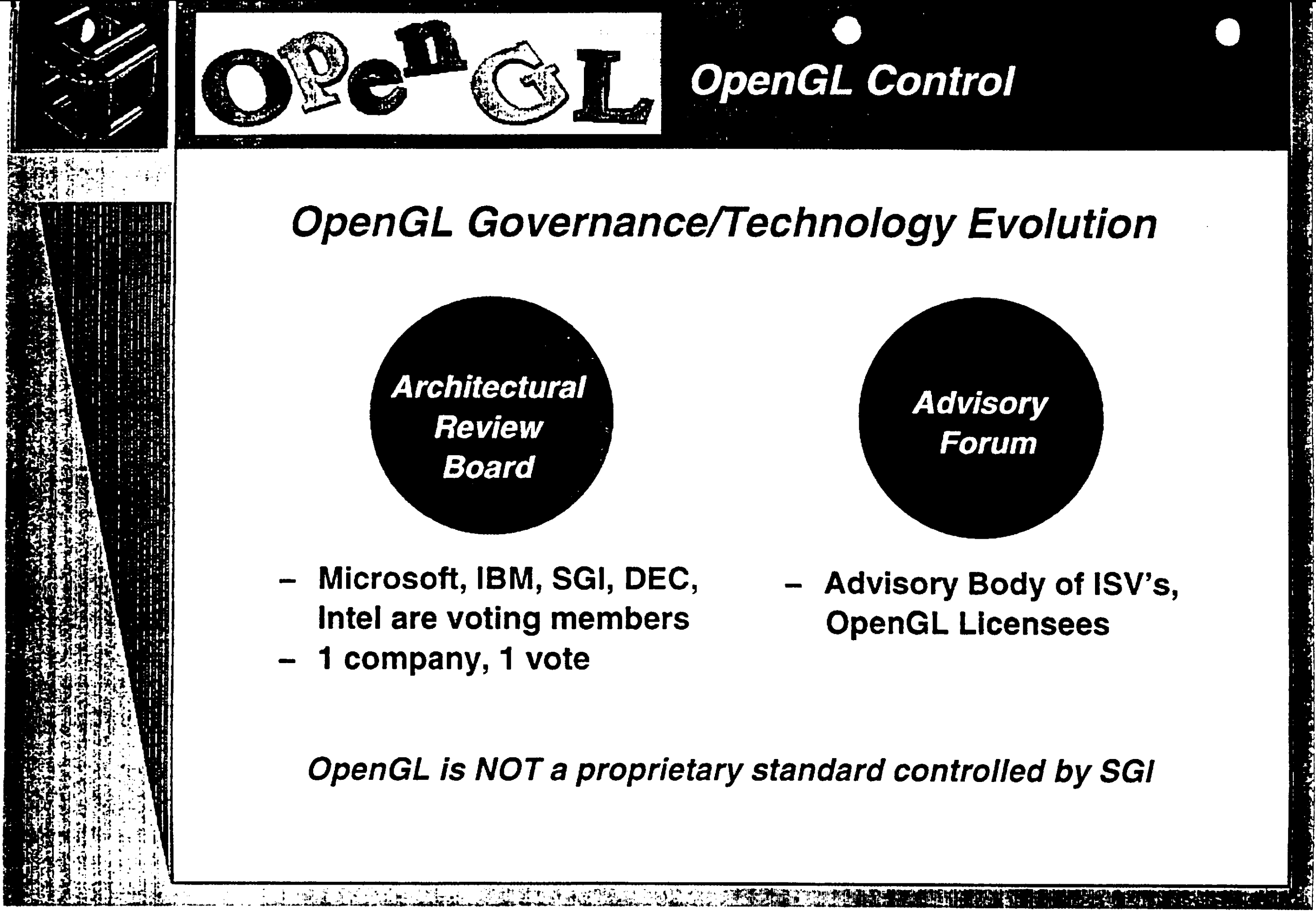
OpenGL Licensees

Workstations

DEC Daikin
Du Pont Pixel (Sun)
E&S Harris
Hitachi IBM
Intergraph Kendall Square
Kubota NEC
Portable Graphics (Sun, HP)
Samsung SGI
SONY Univel/USL

Personal Computers

Microsoft
Intel
Pellucid (MediaVision)
Austek



OpenGL

OpenGL Control

OpenGL Governance/Technology Evolution

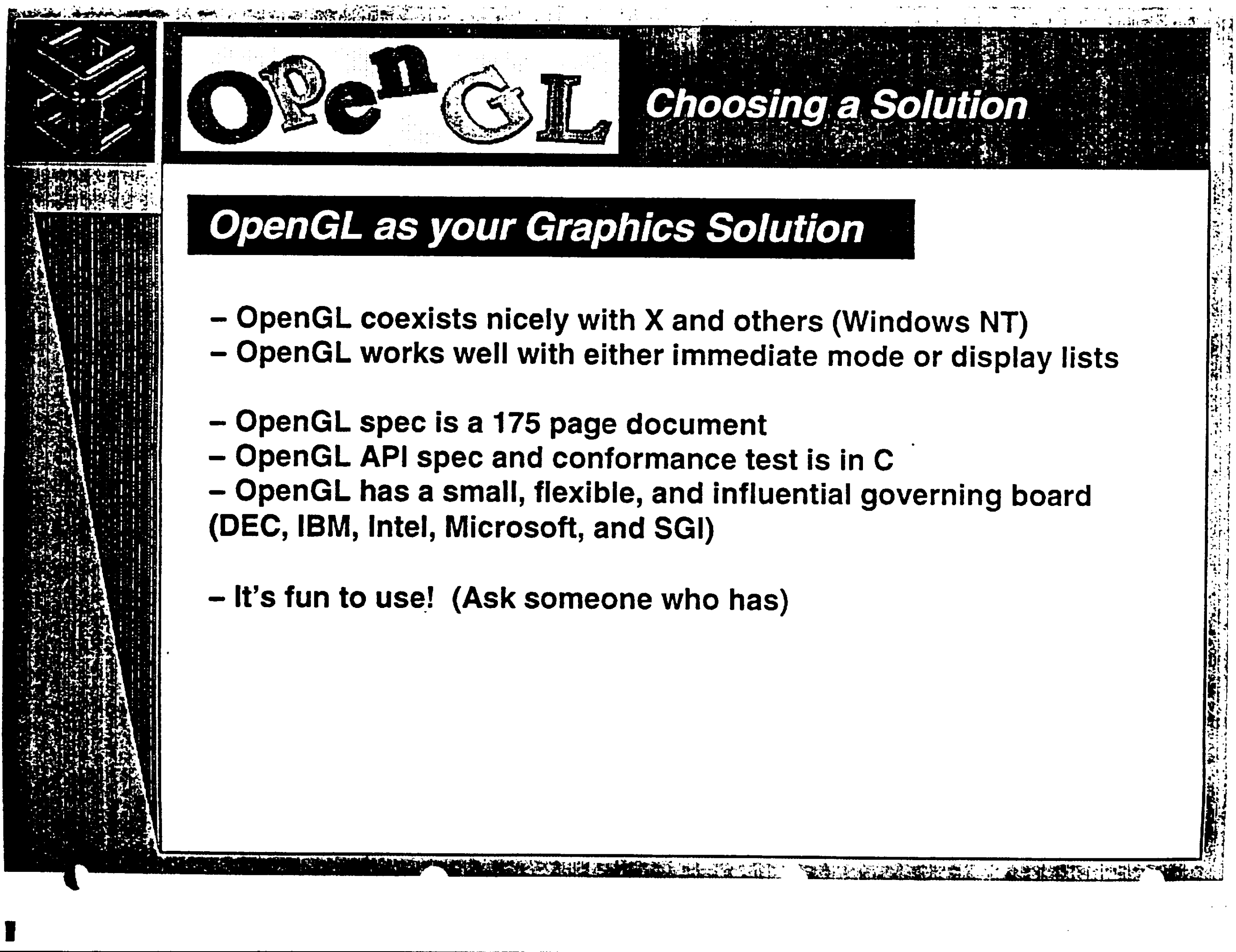
Architectural Review Board

- Microsoft, IBM, SGI, DEC, Intel are voting members
- 1 company, 1 vote

Advisory Forum

- Advisory Body of ISV's, OpenGL Licensees

OpenGL is NOT a proprietary standard controlled by SGI



OpenGL

Choosing a Solution

OpenGL as your Graphics Solution

- OpenGL coexists nicely with X and others (Windows NT)
- OpenGL works well with either immediate mode or display lists

- OpenGL spec is a 175 page document
- OpenGL API spec and conformance test is in C
- OpenGL has a small, flexible, and influential governing board (DEC, IBM, Intel, Microsoft, and SGI)

- It's fun to use! (Ask someone who has)

ILM: A Case Study

Alias Research -- modeling
SoftImage -- animation
Renderman (not GL) -- rendering
IRIS Inventor -- internal development

IRIS Inventor is an object-oriented 3D toolkit, built atop GL and Motif

Information Sources

- Usenet Group comp.graphics.opengl

- FTP OpenGL 1.0 Man Pages from sgi.com
  in ~ftp/OpenGL/doc directory -> {gl,glu,glx}.shar.Z files

- Technical White Paper 'An Analysis of PEX 5.1 vs. OpenGL 1.0'

- Addison-Wesley Publishing Company
  OpenGL Programming Guide
  OpenGL Reference Manual

- OpenGL 1.0 Specification

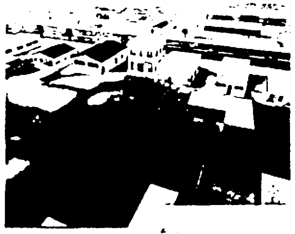
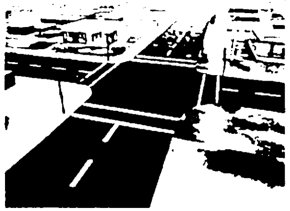
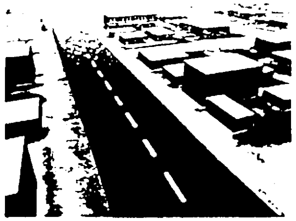
```
     intro.c -- setup needed to create
     OpenGL visual for an X Window
/

nclude <stdio.h>
nclude <GL/glx.h>
nclude <GL/gl.h>

atic int attributeList[] = { GLX_RGBA, None };

atic Bool WaitForNotify \
  (Display *d, XEvent *e, char *arg) \
  { return (e->type == MapNotify) && \
  (e->xmap.window ==  (Window)arg); }

t main(int argc, char **argv) {
  Display *dpy;
  XVisualInfo *vi;
  Colormap cmap;
  XSetWindowAttributes swa;
  Window win;
  GLXContext cx;
  XEvent event;

  /* get a connection */
  dpy = XOpenDisplay(0);

  /* get an appropriate visual */
  vi = glXChooseVisual(dpy,
      DefaultScreen(dpy), attributeList);

  /* create a GLX context */
  cx = glXCreateContext(dpy, vi, 0, GL_TRUE);

  /* create a colormap */
  cmap = XCreateColormap(dpy,
      RootWindow(dpy, vi->screen),
      vi->visual, AllocNone);

  /* create a window */
  swa.colormap = cmap;
  swa.border_pixel = 0;
  swa.event_mask = StructureNotifyMask;
    n = XCreateWindow(dpy,
      RootWindow(dpy, vi->screen),
      0, 0, 100, 100,
      0, vi->depth, InputOutput, vi->visual,
      CWBorderPixel|CWColormap|CWEventMask, &swa);
  XMapWindow(dpy, win);
  XIfEvent(dpy, &event, WaitForNotify, (char*)win);

  /* connect the context to the window */
  glXMakeCurrent(dpy, win, cx);

  /* now you can call OpenGL */
```

```
/*
 * smooth.c
 * This program demonstrates smooth shading.
 * A smooth shaded polygon is drawn in a 2-D projection.
 */
#include <GL/gl.h>
#include <GL/glu.h>
#include "aux.h"

/*  GL_SMOOTH is actually the default shading model.  */
void myinit (void)
{
    glShadeModel (GL_SMOOTH);
}

void triangle(void)
{
    glBegin (GL_TRIANGLES);
    glColor3f (1.0, 0.0, 0.0);
    glVertex2f (5.0, 5.0);
    glColor3f (0.0, 1.0, 0.0);
    glVertex2f (25.0, 5.0);
    glColor3f (0.0, 0.0, 1.0);
    glVertex2f (5.0, 25.0);
    glEnd ();
}

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    triangle ();
    glFlush ();
}

void myReshape(GLsizei w, GLsizei h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        gluOrtho2D (0.0, 30.0,
        0.0, 30.0 * (GLfloat) h/(GLfloat) w);
    else
        gluOrtho2D (0.0,
        30.0 * (GLfloat) w/(GLfloat) h, 0.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
}

/*  Main Loop
 *  Open window with initial window size, title bar,
 *  RGBA display mode, and handle input events.
 */
int main(int argc, char** argv)
{
    auxInitDisplayMode (AUX_SINGLE | AUX_RGB);
    auxInitPosition (0, 0, 500, 500);
    auxInitWindow (argv[0]);
    myinit();
    auxReshapeFunc (myReshape);
    auxMainLoop(display);
}
```

```
*
*  scene.c
*  This program demonstrates the use of the
*   , lighting model.  Objects are drawn using
*  a grey material characteristic.
*  A single light source illuminates the objects.
*/
include <GL/gl.h>
include <GL/glu.h>
include "aux.h"

*  Initialize material property and light source.
*/
oid myinit (void)

   GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
   GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
*      light_position is NOT default value      */
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
   ~lLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    .Lightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
   glLightfv (GL_LIGHT0, GL_POSITION, light_position);

   glEnable (GL_LIGHTING);
   glEnable (GL_LIGHT0);
   glDepthFunc(GL_LESS);
   glEnable(GL_DEPTH_TEST);


oid display (void)

   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glPushMatrix ();
   glRotatef (20.0, 1.0, 0.0, 0.0);

   glPushMatrix ();
   glTranslatef (-0.75, 0.5, 0.0);
   glRotatef (90.0, 1.0, 0.0, 0.0);
   ~uxSolidTorus (0.275, 0.85);
    .PopMatrix ();

   glPushMatrix ();
   glTranslatef (-0.75, -0.5, 0.0);
   glRotatef (270.0, 1.0, 0.0, 0.0);
   auxSolidCone (1.0, 2.0);
   glPopMatrix ();

   glPushMatrix ();
   glTranslatef (0.75, 0.0, -1.0);
   auxSolidSphere (1.0);
   glPopMatrix ();

   glPopMatrix ();
   glFlush ();


oid myReshape (GLsizei w, GLsizei h)

   ~lViewport (0, 0, w, h);
   MatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   if (w <= h)
```

```
        glOrtho (-2.5, 2.5, -2.5*(GLfloat)h/(GLfloat)w,
            2.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
    else
        glOrtho (-2.5*(GLfloat)w/(GLfloat)h,
            2.5*(GLfloat)w/(GLfloat)h,
            -2.5, 2.5, -10.0, 10.0);
    glMatrixMode (GL_MODELVIEW);
}

/*  Main Loop
 *  Open window with initial window size, title bar,
 *  RGBA display mode, and handle input events.
 */
int main(int argc, char** argv)
{
    auxInitDisplayMode (AUX_SINGLE | AUX_RGB | AUX_DEPTH);
    auxInitPosition (0, 0, 500, 500);
    auxInitWindow (argv[0]);
    myinit ();
    auxReshapeFunc (myReshape);
    auxMainLoop (display);
}
```

CHALLENGING
BUSINESS AS USUAL
IN SOUTH CENTRAL L.A.

BY CYNTHIA M. MARSHALL

Like any big city, Los Angeles is operated by an intricate machinery of politicians, civil servants and various representatives of both public and private causes. They may not agree on everything, but at the moment, there's one thing on which many seem to concur: the old ways just aren't working.

In the aftermath of this spring's riots, there's no question that things in south central Los Angeles haven't been working, and a new set of graphics boards is playing an important role in shaping the area's economic and social recovery.

South central Los Angeles isn't what you'd call an easy place to live. The two square-mile area packs in somewhere between two-thirds and three-quarters of a million people, including many immigrants who don't know English and many more — as many as 50 percent in some areas — who live below the federally established poverty line.

"It's a low income, high poverty, high unemployment area with a large number of people who are completely outside of the economy," explains Paul Ong, associate professor of architecture and urban planning at the University of California at Los Angeles (UCLA).

Community organizers had staged a rally for May 1 to call attention to the area's dire situation. But for the three days beforehand, the neighborhood erupted in its own violent convocation, leaving in its wake $476 million in structural damage and untold losses to inventories and destroyed businesses, according to estimates from UCLA.

To say that the organizations that are leading the rebuilding effort face a crisis in community confidence is more than a little bit of an understatement.

"Before the riots, the community was already organizing to demonstrate its dissatisfaction with losing money for parks and redevelopment while wealthier parts of the city had lots of money in their coffers. Now, there's a definite sense of urgency that something different has to happen," according to Eduardo P. Reyes, chief deputy for Los Angeles City Council member Mike Hernandez, who represents Pico Union, a neighboring area that was also hard hit by the disturbances.

REVAMPING THE PROCESS

In particular, Reyes explains, for any rebuilding effort to work, community members have to be considered in the decision-making process, with the end result better addressing their needs.

"We want to allocate resources in such a way that there's an improvement, not just a rebuilding of what was there. If you put up what was there, they'll burn it down again," Reyes stresses.

Enter Bill Jepson, the director of computing at UCLA's graduate school of architecture and urban planning. Under Jepson's technological direction, the school had already participated in a number of planning projects in other low-income areas of Los Angeles and its successes made it a natural place for city leaders to turn for assistance in planning the rebuild.

Using processes that had been perfected on the other projects, RealityEngine graphics boards and other hardware on loan from Silicon Graphics, and a suite of software applications from Silicon Graphics, Software Systems and Digital Vision, Jepson, Ong and two students scanned in aerial photos of the south central neighborhood and created a detailed digitized model of the area.

Using the model as a starting point, the group visually "rebuilt" the neighborhood's homes, businesses, streets and public spaces. Unlike standard architectural plans, the visual plan is easy to understand and can be modified quickly and easily, both necessities in the interactive, consensus-driven environment that the city council hopes to create when it allows the community at large to see and evaluate it.

"By their very nature, architectural plans are very schematic and difficult to

3D IN DAILY PRACTICE

BEFORE
AFTER

The computer visualization of south central Los Angeles is a dramatic example of the way that Silicon Graphics high-end workstations are used in architecture and urban planning. Visualization and rendering is available, to varying degrees, on all of Silicon Graphics workstations and even the entry-level Indigo has been used to help shape public policy.

Take HDA Consulting for example. This Sacramento, California based site visualization and rendering company operates successfully with a single IRIS Indigo workstation and an IBM-based personal computer. "I got into business for around $12,000,

understand. With this technology, we can put people in the neighborhood and ask them if this is what they envision and get buy-in on their part," Jepson says.

"What we have here is something that lets you say, 'Here's our proposal,'" explains Leo Estrada, associate professor of architecture and urban planning at UCLA and a member of the board of directors of RLA, an organization whose mission is to lead the rebuilding efforts in the south central Los Angeles area.

"You could actually locate a person and show them how the block that they live on will look and allow them to say, 'I see a problem here,' and take that into account," Estrada explains.

"Arguments about how many liquor stores the neighborhood needs can be resolved and locations can be shown quickly," he further describes. In fact, according to Estrada, one community member's inquiries about street lighting at a recent showing of the plan revealed a number of dark pockets which were subsequently revised to improve the neighborhood's safety.

The visual medium, by transcending language barriers, also offers the prospect of encouraging non-English speakers to participate more actively in the community. In south central Los Angeles," where Reyes estimates that more than half of the residents speak Spanish, finding new ways to communicate is essential.

"How do you communicate with a diverse culture that has so many languages and outlooks?" Reyes ponders.

WHAT THE FUTURE HOLDS

So far, the UCLA visualization appears to have received a warm reception from the representatives of City Council and RLA who have seen it. "I think it's technology at its best," says Bernard Kinsey, co-chair of RLA.

One of the next big tests will be how the community at large responds to it. Although no dates have been set, plans are underway to unveil the visualization in a forum such as a neighborhood meeting.

"We don't want to create false expectations, so we need to be convinced that Silicon Graphics can articulate the technology in the community," Reyes explains.

"In these areas, the likelihood of having TVs, let alone computers, isn't great," he stresses. As such, the way that the technology is presented is as important as the technology itself.

In the meantime, Jepson and his staff are working to permanently secure the hardware they need to continue the south central project and to begin a digitized model of the Pico Union area.

"This tool will be useful in all areas of Los Angeles county. It will facilitate the notion of urban planning in a way that people can understand it," according to Kinsey. "We see it as a constructive tool in the community, and government and industry, too. Everything will be centered on what we can do at one desk with this program," he says.

Professor Estrada's long-term wish is to digitize the entire city map of Los Angeles. "I can't imagine that in ten years every city in America will not have digitized city maps like these," he says.

If Estrada is correct, then it's safe to say that 3D graphics is affecting the fields of architecture and urban planning on a scale that will forever alter the way that its practitioners look at their work. Hardly business as usual.

RealityEngine graphics is a set of three graphics boards that facilitate interactive realism on the IRIS Crimson and IRIS POWER Series deskside and rack systems.

Cynthia Marshall is a freelance writer in Mountain View, California

including software, peripherals and upgrades," says the firm's principal, Hayward Anderson.

The firm is regularly engaged by city governments as well as engineering firms faced with difficult to envision or controversial projects. For instance, an engineering firm in Sacramento brought Anderson in to produce visual plans for a shopping center that the local planning commission had refused four different times over the course of three years.

"The engineering firm had brought in 2D drawings and washes, but the commission just couldn't see how the shopping center would fit into the community," Anderson explains. "They needed to visualize what it would look like and how it would harm or not harm the environment around it," he continues.

With just a month and a half of HDA's assistance, the plan was accepted on the fifth try, with just one slight modification.

HDA has also worked extensively with the City of Big Bear Lake, a ski resort town in Southern California. Recognizing the importance of projecting a polished, up-to-date image to visitors, the city applied for scenic corridor status for its main street, Big Bear Boulevard.

CalTrans, the state agency that issues scenic corridor status, rejected the application on the basis of the number of utility poles lining the road. Big Bear Lake's City Council had earlier decided not to relocate the utility poles, but was anxious to have the main drag obtain scenic status. To help them sort out their options, they contacted HDA.

Anderson produced before and after renderings of the boulevard. "The difference was like night and day," Anderson says. "Before, the street looked like it had a line of Christmas trees going down it, there were so many poles," he explains. His work helped the Council make the somewhat expensive, but more aesthetic decision to relocate the poles. Now, CalTrans is reconsidering their application.

More recently, the firm is helping the city decide the best bridge to build over Big Bear Lake Dam. Anderson has produced four different alternatives, which he says the city will be presenting to residents, so they can have a say in the decision making.

"With visualization, clients can see how everything fits together, new and old," Anderson says.

Path: kicksave.asd.sgi.com!woo
From: woo@kicksave.asd.sgi.com (Mason Woo)
Newsgroups: comp.graphics.opengl,comp.answers,news.answers
Subject: comp.graphics.opengl Frequently Asked Questions (FAQ) [1/1]
Distribution: world
Followup-To: poster
Summary: This contains a list of Frequently Asked Questions about OpenGL (TM)
Expires: 30 September 1993 00:00:00 GMT
Keywords: FAQ
Reply-To: woo@kicksave.asd.sgi.com
Approved: news-answers-request@MIT.Edu

Archive-name: graphics/opengl-faq
Last-modified: 1993/09/09
Version: 1.7

* marks recently modified answers or new questions

Q1: What is OpenGL?
Q2: What is the relationship between IRIS GL and OpenGL?
Q3: What does the .gl or .GL file format have to do with OpenGL?
Q4: Which vendors are supporting OpenGL?
Q5: When will OpenGL implementations be available?
* Q6: What documentation is available for OpenGL? Where is the source code which accompanies that documentation?
Q7: Why doesn't SGI provide a free implementation of OpenGL?
Q8: What are the conformance tests?
Q9: How do I contribute OpenGL code examples to a publicly accessible archive?
Q10: Will OpenGL code be source code or binary code compatible with IRIS GL code?
Q11: Why should I port my IRIS GL application to OpenGL?
Q12: How much work is it to convert an IRIS GL program to OpenGL? What are the major differences between them?
Q13: How does a university or research institution acquire access to OpenGL source code?
Q14: How is a commercial license acquired?
Q15: How is the OpenGL governed? Who decides what changes can be made?
Q16: Who are the current ARB members?
Q17: What is the philosophy behind the structure of the ARB?
Q18: How does the OpenGL ARB operate logistically? When does the ARB have meetings?
Q19: How do additional members join the OpenGL ARB?
Q20: So if I'm not a member of the ARB, am I shut out of the decision making process?
Q21: What is the OpenGL Advisory Forum?
Q22: Are ARB meetings open to observers?

Q1: What is OpenGL?

A: OpenGL is the software interface for graphics hardware that allows graphics programmers to produce high-quality color images of 3D objects. OpenGL is a rendering only, vendor neutral API providing 2D and 3D graphics functions, including modelling, transformations, color, lighting, smooth shading, as well as advanced features like texture mapping, NURBS, fog, alpha blending and motion blur. OpenGL works in both immediate and retained (display list) graphics modes. OpenGL is window system and operating system independent. OpenGL has been integrated with Windows NT and with the X Window System under UNIX. Also, OpenGL is network transparent. A defined common extension to the X Window System allows an OpenGL client on one vendor's platform to run across a network to another vendor's OpenGL server.

Q2: What is the relationship between IRIS GL and OpenGL?

A: IRIS GL is the predecessor to OpenGL. After other implementors and experience trying to port the IRIS GL to their own machines, it was learned that the IRIS GL was too tied to a specific window system or hardware. Based upon consultations with several implementors, OpenGL is much more platform independent.

IRIS GL is being maintained and bugs will be fixed, but SGI will no longer add enhancements. OpenGL is now the strategic interface for 3-D computer graphics.

Q3: What does the .gl or .GL file format have to do with OpenGL?

.gl files have nothing to do with OpenGL. It's a file format for images, which has no relationship to IRIS GL or OpenGL.

Q4: Which vendors are supporting OpenGL?

A: OpenGL is supported by SGI and many other hardware vendors. As of August, 1993, OpenGL has been licensed to DEC, IBM, Intel, Microsoft, Portable Graphics (formerly Nth Portable Graphics; supporting Sun and HP), Du Pont Pixel Systems (supporting Sun and a Du Pont Pixel graphics accelerator board), Evans & Sutherland, Kubota Pacific, Sony, NEC, Hitachi, Daikin, Intergraph, miro, pellucid, RasterOps, Samsung, SPEA, USL, Harris Computer, Kendall Square Research, and SHOGraphics.

The machines supported by OpenGL licensees constitute over 95% of the graphics workstation marketplace, as well as the majority of the PC market.

Q5: When will OpenGL implementations be available?

A: On SGI machines, OpenGL is available on both Onyx and Indy. Availability on at Indigo, Indigo 2, Crimson, and other SGI workstations will be part of the next release of IRIX to upgrade all products to 5.x.

Digital Equipment has shipped its first OpenGL product as part of the Open3D 2.0 layered product for Alpha AXP OSF/1. The server support in this release is only for the PXG graphics adaptor, but the client libraries are of course universal if thats all you need. For information on ordering the Open3D 2.0 product from Digital contact your Digital sales representative.

SGI does not speak for any other company. However, this space is available for any company who wishes to state status reports or release dates for their OpenGL implementation. Please send e-mail to woo@sgi.com to add to this section.

Q6: What documentation is available for OpenGL?

A: A 2 volume set, The OpenGL Technical Library, is being published by Addison-Wesley. The OpenGL Reference Manual is ISBN 0-201-63276-4. The OpenGL Programming Guide is ISBN 0-201-63274-8.

You can purchase the books in extremely large volume by calling Robert Shepard of Addison-Wesley (617) 944-3700 ext 2435.

The man pages are also available via anonymous, public ftp. On the machine sgi.com, the directory with man pages for OpenGL, its Utility Library (glu), and the X server extension API (glx) is ~ftp/OpenGL/doc.

If you have access to ftp, you can get the source code examples which are found in the OpenGL Programming Guide via anonymous, public ftp on another machine: sgigate.sgi.com in the file ~ftp/pub/opengl/opengl.tar.Z

Q10: Will OpenGL code be source code or binary code compatible with IRIS GL code?

A: OpenGL code is neither binary nor source code compatible with IRIS GL code. It was decided to bite the bullet at this time to make OpenGL incompatible with IRIS GL and fix EVERYTHING that made IRIS GL difficult to port or use. For example, the gl prefix has been added to every command: glVertex(), glColor(), etc.

Q11: Why should I port my IRIS GL application to OpenGL?

SGI will be maintaining the old IRIS GL, but not enhancing it. OpenGL is the API of choice on all new SGI machines.

OpenGL has no subsets. You can use the same functionality on all machines from SGI or from other vendors.

OpenGL is better integrated with the X Window System than the old IRIS GL. For example, you can mix OpenGL and X or Display PostScript drawing operations in the same window.

The OpenGL naming scheme, argument list conventions, and rendering semantics are cleaner than those of IRIS GL. This should make OpenGL code easier to understand and maintain.

Q12: How much work is it to convert an IRIS GL program to OpenGL? What are the major differences between them?

There is a fair amount of work, most of which is in substituting for window management or input handling routines, for which the equivalents are not OpenGL, but the local window system, such as the X Windows System or Windows NT. And all routine names have changed, at least, minimally; for example: ortho() is now glOrtho().

To help ease the way, port to "mixed model" right away, mixing the X Window System calls to open and manage windows, cursors, and color maps and read events of the window system, mouse and keyboard. You can do that now with IRIS GL, if you are running IRIX 4.0.

In the X Window System, display mode choices (such as single or double buffering, color index or RGBA mode) must be declared before the window is initially opened. You may also substitute for other IRIS GL routines, such as using a OSF/Motif menu system, in place of the IRIS GL pop-up menus. You should use glXUseXFont(), whenever you were using the font manager with IRIS GL.

Tables for states such as lighting or line and polygon stipples will be gone. Instead of using a def/set or def/bind sequence to load a table, you turn on the state with glEnable() and also declare the current values for that state.

Colors are best stored as floating point values, scaled from 0.0 to 1.0 (0% to 100%). Alpha is fully integrated in the RGBA mode and at least source alpha will be available on all OpenGL implementations. OpenGL will not arbitrarily limit the number of bits per color to 8. Clearing the contents of buffers no longer uses the current color, but a special "clearing" color for each buffer (color, depth, stencil, and accumulation).

The transformation matrix has changed. In OpenGL, there is no single matrix mode. Matrices are now column-major and are post-multiplied, although that does NOT change the calling order of these routines from IRIS GL to OpenGL. OpenGL's glRotate*() now allows for a rotation around an arbitrary axis, not just the x, y, and z axes. lookat()

of IRIS GL is now gluLookAt(), which takes an up vector value, not merely a twist. There is no polarview() in OpenGL, but a series glRotate*()s and glTranslate*()s can do the same thing.

There are no separate depth cueing routines in OpenGL. Use linear fog.

Feedback and selection (picking) return values, which are different from those found on any IRIS GL implementation. For selection and picking, depth values will be returned for each hit. In OpenGL, feedback and selection will now be standardized on all hardware platforms.

Q13: How does a university or research institution acquire access to OpenGL source code?

A: There is a university/research institution licensing program. A university license entitles the institution to generate binaries and copy them anywhere, so long as nothing leaves the institution. The OpenGL source and derived binaries can only be used for non-commercial purposes on-campus. A university license costs $500 and can be obtained by contacting woo@sgi.com.

Q14: How is a commercial license acquired?

.. Call Mason Woo at (415) 390-4205 or e-mail him at woo@sgi.com. There are licenses available restricted to site (local) usage, or permitting redistribution of binary code. The limited source license provides a sample implementation of OpenGL for $50,000. The license for commerical redistribution of OpenGL binaries has two most commonly chosen levels. Level 1 costs $25,000. Level 2 costs $100,000, and includes the sample implementation of OpenGL. Both levels require a $5 royalty for every copy of the OpenGL binary, which is redistributed.

Since many implementations will be a shared library on a hardware platform, the royalty sometimes will be charged for each hardware platform, not each application which uses OpenGL.

Q15: How is the OpenGL governed? Who decides what changes can be made?

A: OpenGL is controlled by an independent board, the Architectural Review Board (ARB). Each member of the ARB has one vote. The founding mbers of the ARB are DEC, IBM, Intel, Microsoft, and SGI. .ditional members will be added over time. The ARB governs the future of OpenGL, proposing and approving changes to the specification, new releases, and conformance testing.

Q16: Who are the current ARB members?

A: In alphabetical order: Digital Equipment, IBM, Intel, Microsoft, and Silicon Graphics.

Q17: What is the philosophy behind the structure of the ARB?

A: The ARB is intended to be able to respond quickly and flexibly to evolutionary changes in computer graphics technology. The ARB is currently "lean and mean" to encourage speedy communication and decision-making. Its members are highly motivated in ensuring the success of OpenGL.

Q18: How does the OpenGL ARB operate logistically? When does the ARB have meetings?

ARB meetings are held about once a quarter. The meetings rotate among sites hosted by the ARB members.

The next ARB meeting will be held the week of December 6th in Austin, Texas. This meeting will be hosted by IBM. There will also be a day or two of interoperability testing as part of the next ARB meeting, and licensees are invited to test the interoperability of their OpenGL implementations.

Meetings are run by a set of by-laws, which are currently being approved. When they are approved, the by-laws will be publicly available for inspection.

Q19: How do additional members join the OpenGL ARB?

A: The intention is that additional members may be added on a permanent basis or for a one-year term. The one-year term members would be voting members, added on a rotating basis, so that different viewpoints (such as ISV's) could be incorporated into new releases. Under the currently proposed (but not yet ratified) by-laws, SGI formally nominates new members.

Q20: So if I'm not a member of the ARB, am I shut out of the decision making process?

A: There are many methods by which you can influence the evolution of OpenGL.

1) Contribute to the comp.graphics.opengl news group. Most members of the ARB read the news group religiously.
2) Contact any member of the ARB and convince that member that your proposal is worth their advocacy. Any ARB member may present a proposal, and all ARB members have equal say.
3) Come to OpenGL Advisory Forum and speak directly to ARB in person.

Q21: What is the OpenGL Advisory Forum?

A: Preceding every ARB meeting will be a meeting of the OpenGL Advisory Forum. Members of the ARB will attend this meeting to listen to proposals and discussions. If you want to attend, you want to notify the Secretary of the ARB. Until a more generic mail alias is set up for the Secretary, you can just send mail to woo@sgi.com

The Advisory Forum is intended to start as an informal, self-governing group of people, highly interested in OpenGL. The ARB will invite a representative of the Advisory Forum to attend the ARB meetings, in a non-voting capacity. Advisory Forum members can be candidates for both short-term and permanent membership in the ARB. The Advisory Forum is encouraged to formulate its own structure and rules and move into any direction it wants.

Corporations, universities, or individuals can be members of the Advisory Forum.

The next OpenGL Advisory Forum meeting is Monday, December 6th hosted by IBM, somewhere in Austin, Texas (same place as the ARB meeting).

Q22: Are ARB meetings open to observers?

A: The ARB meeting will be open to observers, but we want to keep the meeting small. For the next meeting in Austin, Texas, the ARB will allow the Advisory Forum to nominate five representatives of the Advisory Forum to attend the ARB meeting, in a non-voting capacity. After the meeting in Austin, the ARB will once again re-evaluate the number of observers, based upon the success of this experiment.

The OpenGL Graphics Interface

Mark Segal
Kurt Akeley
Silicon Graphics Computer Systems
2011 N. Shoreline Blvd., Mountain View, CA 94039
USA

Abstract

Graphics standards are receiving increased attention in the computer graphics community as more people write programs that use 3D graphics and as those already possessing 3D graphical programs want those programs to run on a variety of computers.

OpenGL is an emerging graphics standard that provides advanced rendering features while maintaining a simple programming model. Its procedural interface allows a graphics programmer to describe rendering tasks, whether simple or complex, easily and efficiently. Because OpenGL is rendering-only, it can be incorporated into any window system (and has been, into the X Window System and the soon-to-be-released Windows NT) or can be used without a window system. Finally, OpenGL is designed so that it can be implemented to take advantage of a wide range of graphics hardware capabilities, from a basic framebuffer to the most sophisticated graphics subsystems.

1 Introduction

Computer graphics (especially 3D graphics, and interactive 3D graphics in particular) is finding its way into an increasing number of applications, from simple graphing programs for personal computers to sophisticated modeling and visualization software on workstations and supercomputers. As the interest in computer graphics has grown, so has the desire to be able to write an application so that it runs on a variety platforms with a range of graphical capabilites. A graphics standard eases this task by eliminating the need to write a distinct graphics driver for each platform on which the application is to run.

Several standards have succeeded in integrating specific domains of 2D graphics. The PostScript page description language[4] has become widely accepted, making it relatively easy to electronically exchange, and, to a limited degree, manipulate static documents containing both text and 2D graphics. The X window system[7] has become standard for UNIX workstations. A programmer uses X to obtain a window on a graphics display into which either text or 2D graphics may be drawn; X also provides a standard means for obtaining user input from such devices as keyboards and mice. The adoption of X by most workstation manufacturers means that a single program can produce 2D graphics or obtain user input on a variety of workstations by simply recompiling the program. This integration even works across a network: the program may run on one workstation but display on and obtain user input from another, even if the workstations on either end of the network are made by different companies.

For 3D graphics, several standards have been proposed, but none has (yet) gained wide acceptance. One relatively well-known system is PHIGS (Programmer's Hierarchical Interactive Graphics System). Based on GKS[5] (Graphics Kernel System), PHIGS is an ANSI (American National Standards Institute) standard. PHIGS (and its descendant, PHIGS+[9]) provides a means to manipulate and draw 3D objects by encapsulating object descriptions and attributes into a *display list* that is then referenced when the object is displayed or manipulated. One advantage of the display list is that a complex object need be described only once even if it is to be displayed many times. This is especially important if the object to be displayed must be transmitted across a low-bandwidth channel (such as a network). One disadvantage of a display list is that it can require considerable effort to re-specify the object if it is being continually modified as a result of user interaction. Another difficulty with PHIGS and PHIGS+ (and with GKS) is that they lack support for advanced rendering features such as texture mapping.

PEX[8], which is often said to be an acronym for PHIGS Extension to X, extends X to include the ability to manipulate and draw 3D objects. (PEXlib[6] is the programmer's interface to the PEX protocol.), Among other extensions, PEX adds *immediate mode* rendering to PHIGS, meaning that objects can be displayed as they are described rather than having to first complete a display list. One difficulty with PEX has been that different suppliers of the PEX interface have chosen to support different features, making program portability problematic. PEX also lacks advanced rendering feaures, and is available only to users of X.

2 OpenGL

OpenGL ("GL" for "Graphics Library") provides advanced rendering features in either immediate mode or display list mode. While OpenGL is a relatively new standard, it is very similar in both its functionality and its interface to Silicon Graphics' IRIS GL, and there are many succesful 3D applications that currently use IRIS GL for their 3D rendering.

Like the graphics systems already discussed, OpenGL is a software interface to graphics hardware. The interface consists of a set of several hundred procedures and functions that allow a programmer to specify the objects and operations involved in producing high-quality graphical images, specifically color images of three-dimensional objects. Like PEX, OpenGL integrates 3D drawing into X, but can also be integrated into other window systems (e.g. Windows/NT) or can be used without a window system.

OpenGL draws *primitives* into a framebuffer subject to a number of selectable modes. Each primitive is a point, line segment, polygon, pixel rectangle, or bitmap. Each mode may be changed independently; the setting of one does not affect the settings of others (although many modes may interact to determine what eventually ends up in the framebuffer). Modes are set, primitives specified, and other OpenGL operations described by sending *commands* in the form of function or procedure calls.

Geometric primitives (points, line segments, and polygons) are defined by a group of one or more *vertices*. A vertex defines a point, an endpoint of an edge, or a corner of a polygon where two edges meet. Data (consisting of positional coordinates, colors, normals, and texture coordinates) are associated with a vertex and each vertex is processed independently, in order, and in the same way. The only exception to this rule is if the group of vertices must be *clipped* so that the indicated primitive fits within a specified region; in this case vertex data may be modified and new vertices created. The type of clipping depends on which primitive the group of vertices represents.

OpenGL provides direct control over the fundamental operations of 3D and 2D graphics. This includes specification of such parameters as transformation matrices, lighting equation coefficients, antialiasing methods, and pixel update operators. It does not provide a means for describing or modeling complex geometric objects. Another way to describe this situation is to say that OpenGL provides mechanisms to describe how complex geometric objects are to be rendered rather than mechanisms to describe the complex objects themselves.

The model for interpretation of OpenGL commands is client-server.

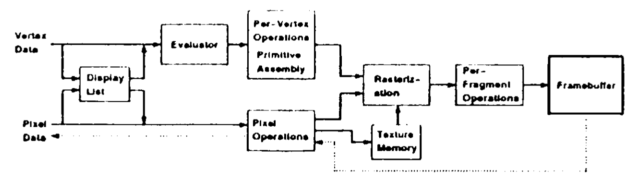


Figure 1. Block diagram of OpenGL.

That is, a program (the client) issues commands, and these commands are interpreted and processed by OpenGL (the server). The server may or may not operate on the same computer as the client.

The effects of OpenGL commands on the framebuffer are ultimately controlled by the window system that allocates framebuffer resources. It is the window system that determines which portions of the framebuffer that OpenGL may access at any given time and that communicates to OpenGL how those portions are structured. Similarly, display of framebuffer contents on a CRT monitor (including the transformation of individual framebuffer values by such techniques as gamma correction) is not addressed by OpenGL. Framebuffer configuration occurs outside of OpenGL in conjunction with the window system; the initialization of an OpenGL context occurs when the window system allocates a window for OpenGL rendering. Additionally, OpenGL has no facilities for obtaining user input, since it is expected that any window system under which OpenGL runs must already provide such facilities. These considerations make OpenGL independent of any particular window system.

3 Basic OpenGL Operation

Figure 1 shows a schematic diagram of OpenGL. Commands enter OpenGL on the left. Most commands may be accumulated in a *display list* for processing at a later time. Otherwise, commands are effectively sent through a processing pipeline.

The first stage provides an efficient means for approximating curve and surface geometry by evaluating polynomial functions of input values. The next stage operates on geometric primitives described by vertices: points, line segments, and polygons. In this stage vertices are transformed and lit, and primitives are clipped to a viewing volume in preparation for the next stage, rasterization. The rasterizer produces a series of framebuffer addresses and values using a two-dimensional description of a point, line segment, or polygon. Each *fragment* so produced is fed to the next stage that performs operations on individual fragments before they finally alter the framebuffer. These operations include conditional updates into the framebuffer based on incoming and previously stored depth values (to effect depth buffering), blending of incoming fragment colors with stored colors, as well as masking and other logical operations on fragment values.

Finally, pixel rectangles and bitmaps bypass the vertex processing portion of the pipeline to send a block of fragments directly through rasterization to the individual fragment operations, eventually causing a block of pixels to be written to the framebuffer. Values may also be read back from the framebuffer or copied from one portion of the framebuffer to another. These transfers may include some type of decoding or encoding.

3.1 The OpenGL Utility Library

A guiding principle in the design of OpenGL has been to provide program portability without mandating how higher-level graphical objects must be described. As a result, the basic OpenGL interface does not support some geometric objects that are traditionally associated with graphics standards. For instance, an OpenGL implementation need not render concave polygons. One reason for this omission is that concave polygon rendering algorithms are of necessity more complex than those for rendering convex polygons, and different concave polygon algorithms may be appropriate in different domains. In particular, if a concave polygon is to be drawn more than once, it is more efficient to first decompose it into convex polygons (or triangles) once and then draw the convex polygons.

A general concave polygon decomposer is provided as part of the OpenGL Utility Library, which is provided with every OpenGL implementation. The Utility Library also provides an interface, built on OpenGL's polynomial evaluators, to describe and display NURBS curves and surfaces (with domain space trimming), as well as a means for rendering spheres, cones, and cylinders. The Utility Library serves both as a means to render useful geometric objects and as a model for building other libraries that use OpenGL for rendering.

Object	Interpretation of Vertices
point	each vertex describes the location of a point
line strip	series of connected line segments, each vertex after first describes the endpoint of next segment
line loop	same as line strip but final segment added from final vertex to first vertex
separate line	each pair of vertex describes a line segment
polygon	line loop formed by vertices describes the boundary of a convex polygon
triangle strip	each vertex after the first two describes a triangle given by that vertex and the previous two
triangle fan	each vertex after the first two describes a triangle given by that vertex, the previous vertex, and the first vertex
separate triangle	each consecutive triad of vertices describes a triangle
quadrilateral strip	each pair of vertices after the first two describes a quadrilateral given by that pair and the previous pair
independent quad	each consecutive group of four vertices describes a quadrilateral

Table 1: **glBegin/glEnd** objects.

4 The OpenGL Pipeline

4.1 Vertices and Primitives

In OpenGL, most geometric objects are drawn by enclosing a series of coordinate sets that specify vertices and optionally normals, texture coordinates, and colors between **glBegin/glEnd** command pairs. For example, to specify a triangle with vertices at $(0,0,0)$, $(0,1,0)$, and $(1,0,1)$, one could write:

```
glBegin(GL_POLYGON);
  glVertex3i(0,0,0);
  glVertex3i(0,1,0);
  glVertex3i(1,0,1);
glEnd();
```

The ten geometric objects that are drawn this way are summarized in Table 4.1. This particular group of objects was selected because each object's geometry is specified by a simple list of vertices, because each admits an efficient rendering algorithm, and because it was determined that taken together these objects satisfy the needs of nearly all graphics applications.

Each vertex may be specified with two, three, or four coordinates (four coordinates indicate a homogeneous three-dimensional location). In addition, a *current normal*, *current texture coordinates*, and *current color* may be used in processing each vertex. OpenGL uses normals in lighting calculations; the current normal is a three-dimensional vector that may be set by

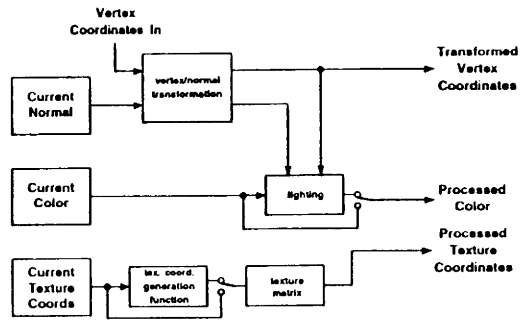


Figure 2. Association of current values with a vertex.

sending three coordinates that specify it. Color may consist of either red, green, blue, and alpha values (when OpenGL has been initialized to RGBA mode) or a single color index value (when initialization specified color index mode). One, two, three, or four texture coordinates determine how a texture image maps onto a primitive.

Each of the commands that specify vertex coordinates, normals, colors, or texture coordinates comes in several flavors to accomodate differing application's data formats and numbers of coordinates. Data may also be passed to these commands either as an argument list or as a pointer to a block of storage containing the data. The variants are distinguished (in the C language) by mnemonic suffixes.

Most OpenGL commands that do not specify vertices and associated information may not appear between **glBegin** and **glEnd**. This restriction allows implementations to run in an optimized mode while processing primitive specifications so that primitives may be processed as efficiently as possible.

When a vertex is specified, the current color, normal, and texture coordinates are used to obtain values that are then associated with the vertex (Figure 2). The vertex itself is transformed by the *model-view matrix*, a 4 × 4 matrix which can represent both linear and translational transformations. The color is obtained from either computing a color from lighting or, if lighting is disabled, from the current color. Texture coordinates are similarly passed through a *texture coordinate generation function* (which may be the identity). The resulting texture coordinates are transformed by the *texture matrix* (this matrix may be used to effectively scale or rotate a texture that is applied to a primitive). Figure 4 shows some results of using texture coordinate generation functions.

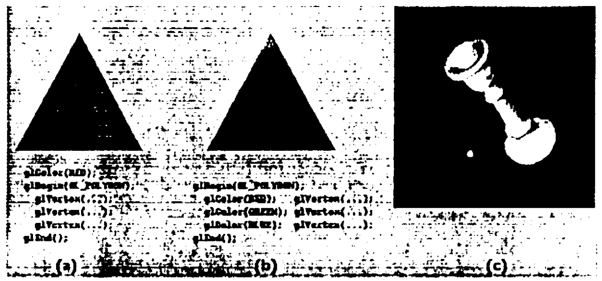
A number of commands control the values of parameters used in processing a vertex. One group of commands manipulates transformation matrices; these commands are designed to form an efficient means for generating and manipulating the transformations that occur in hierarchical 3D graphics scenes. A matrix may be loaded or multiplied by a scaling, rotation, translation, or general matrix. Another command controls which matrix is affected by a manipulation: the model-view matrix, the texture matrix, or the *projection* matrix (to be described presently). Each of these three matrix types also has an associated stack onto which matrices may be pushed or popped.

Lighting parameters are grouped into three categories: material parameters, that describe the reflectance characteristics of the surface being lit, light source parameters, that describe the emission properties of each light source, and lighting model parameters, that describe global properties of the lighting model. Lighting is performed on a per-vertex basis; lighting results are eventually interpolated across a line segment or polygon. The general form of the lighting equation includes terms for constant, diffuse, and specular illumination, each of which may be attenuated by the distance of the vertex from the light source. A programmer may sacrifice realism in favor of faster lighting calculations by indicating that the viewer, the light sources, or both should be assumed to be infinitely far from the scene. Figure 3 shows some results with lighting disabled and enabled.

4.2 Clipping and Projection

Once a primitive has been assembled from a group of vertices, it is subjected to clipping by *clip planes*. The positions of these planes (every OpenGL implementation must provide at least six) is specifiable using the **glClipPlane** command. Each plane may be enabled or disabled individually.

In the case of a point, the clip planes either have no effect on the point or annihilate it depending as the point lies inside or outside the intersection of



```
glColor(RED);
glBegin(GL_POLYGON);
    glVertex(...);
    glVertex(...);
    glVertex(...);
glEnd();
```

```
glBegin(GL_POLYGON);
    glColor(RED);   glVertex(...);
    glColor(GREEN); glVertex(...);
    glColor(BLUE);  glVertex(...);
glEnd();
```

Figure 3. (a) A uniformly colored triangle. (b) A Gouraud shaded triangle. (c) A scene consisting of many lit, shaded polygons.

the half-spaces determined by the clip planes. In the case of a line segment or polygon, the clip planes may have no effect on, annihilate, or alter the original primitive. In the later case, new vertices may be created between edges described by original vertices; color and texture coordinate values for these new vertices are found by appropriately interpolating the values assigned to the original vertices.

After the clip planes (if any) have been applied, the vertex coordinates of the resulting primitive are transformed by the projection matrix. Then *view frustum* clipping occurs. View frustum clipping is like clip plane application, but with fixed planes: if coordinates after transformation are given by (x, y, z, w), then the six half spaces defined by these planes are $-w \leq x$, $x \leq w$, $-w \leq y$, $y \leq w$, $-w \leq z$, $z \leq w$.

With view frustum clipping completed, each group of vertex coordinates is projected by computing x/w, y/w, and z/w. The resulting values (which must each lie in [-1,1]) are multiplied and offset by parameters that control the size of the viewport into which primitives are to be drawn. The **glViewport** (for x/w and y/w) and **glDepthRange** (for z/w) commands control these parameters.

4.3 Rasterization

Rasterization converts a projected, viewport-scaled primitive into a series of *fragments*. Each fragment comprises a location of a pixel in the framebuffer along with color, texture coordinates, and depth (z). When a line segment or polygon is rasterized, these associated data are interpolated across the primitive to obtain a value for each fragment.

The rasterization of each kind of primitive is controlled by a corresponding group of parameters. One width affects point rasterization and another affects line segment rasterization. Additionally, a stipple sequence may be specified for line segments, and a stipple pattern may be speicifed for polygons.

Antialiasing may be enabled or disabled individually for each primitive type. When enabled, a coverage value is computed for each fragment describing the portion of that fragment that is covered by the projected primitive. This coverage value is used after texturing has been completed to modify the fragment's alpha value (in RGBA mode) or color index value (in color index mode).

4.3.1 Pixel Rectangles and Bitmaps

Pixel rectangles and *bitmaps* are the two primitives that are unaffected by the geometric operations that occur in the pipeline prior to rasterization. A pixel rectangle is a group of values destined for the framebuffer (typically the values represent colors, although provision is made for other types of data, such as depth values). The values, stored as a block of data in host memory, are sent using **glDrawPixels**. Arguments to **glDrawPixels** indicate the memory address of the data, the type of data, and the width and height of the rectangle that the data values form. In addition, two groups of parameters are maintained that control the decoding of the stored values. The first group describes how the values are packed in memory and provides a means for selecting a subrectangle from a larger containing rectangle. The second group controls conversions that may be applied to the values after they obtained: values may be scaled, offset and mapped by means of look-up tables. These various parameters form a flexible means for specifying rectangular images stored in a variety of formats.

Once obtained, the resulting values produce a rectangle of fragments. The location of this rectangle is controlled by the *current raster position*, which is treated very much like a point (including associating a color and texture coordinates with it), except that it is set with a separate command (**glRasterPos**) that does not occur between **glBegin** and **glEnd**. The rectangle's size is determined by its specified width and height as well as the setting of pixel rectangle zoom parameters (set with **glPixelZoom**).

A bitmap is similar to a pixel rectangle, except that it specifies a rectangle of zeros and ones, and is designed for describing characters that can be placed at a projected 3D location (through the current raster position). Each one in the bitmap produces a fragment whose associated values are those of the current raster position, while each zero produces no fragment. The **glBitmap** command also specifies offsets that control how the bitmap is placed with respect to the current raster position and how the current raster position is advanced after the bitmap is drawn (thus determining the relative positions of sequential bitmaps).

4.4 Texturing and Fog

OpenGL provides a general means for generating texture-mapped primitives (Figure 4). When texturing is enabled, each fragment's texture coordinates index a texture image, generating a *texel*. This texel may have between one and four components, so that a texture image may represent, for example, intensity only (one component), RGB color (three components), or RGBA color (four components). Once the texel is obtained, it modifies the fragment's color according to a specifiable texture *environment*.

A texture image is specified using **glTexImage**, which takes arguments similar to those of **glDrawpixels**, so that the same image format may be used whether that image is destined for the framebuffer or texture memory. In addition, **glTexImage** may be used to specify mipmaps[3] so that a texture may be filtered as it is applied to a primitive. The filter function (and whether or not it implies mipmaps) is controlled by a number of specifiable parameters using **glTexParameter**. The texture environment is selected with **glTexEnv**.

Finally, after texturing, a fog function (if enabled) is applied to each fragment. The fog function blends the incoming color with a constant (specifiable) fog color according to a computed weighting factor. This factor is a function of the distance (or an approximation to the distance) from the viewer to the 3D point that corresponds to the fragment. Exponential fog simulates atmospheric fog and haze, while linear fog may be used to produce depth-cueing.

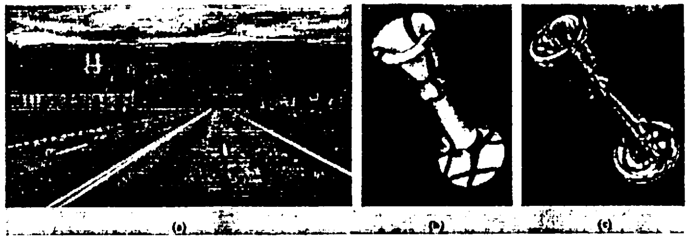


Figure 4. (a) A scene with a number of textures mapped onto primitives. (b) Contouring achieved with texture mapping and a texture coordinate generation function. (c) Reflectance mapping with a texture coordinate generation function.

4.5 The Framebuffer

The destination of rasterized fragments is the framebuffer, where the results of OpenGL rendering may be displayed. In OpenGL, the framebuffer consists of a rectangular array of pixels corresponding to the window allocated for OpenGL rendering. Each pixel is simply a set of some number of bits. Corresponding bits from each pixel in the framebuffer are grouped together into a *bitplane*; each bitplane contains a single bit from each pixel.

The bitplanes are grouped into several logical buffers: the *color*, *depth*, *stencil*, and *accumulation* buffers. The color buffer is where fragment color information is placed. The depth buffer is where fragment depth information is placed, and is typically used to effect hidden surface removal through z-buffering. The stencil buffer contains values each of which may be updated whenever a corresponding fragment reaches the framebuffer. Stencil values are useful in multi-pass algorithms, in which a scene is rendered several times, to achieve such effects as CSG (union, intersection, and difference) operations on a number of objects and capping of objects sliced by clip planes.

The accumulation buffer is also useful in multipass algorithms; it can be manipulated so that it averages values stored in the color buffer. This can effect such effects as full-screen anti-aliasing (by jittering the viewpoint for each pass), depth-of-field (by jittering the angle of view), and motion blur (by stepping the scene in time)[2]. Multi-pass algorithms are simple to implement in OpenGL, because only a small number of parameters must be manipulated before each pass, and changing the values of these parameters is both efficient and without side effects on the values of other parameters that must remain constant.

OpenGL supports both double-buffering and stereo, so the color buffer is further subdivided into four buffers: the front left & right buffers and the back left & right buffers. The front buffers are those that are typically displayed while the back buffers (in a double-buffered application) are being used to compose the next frame. A monoscopic application would use only the left buffers. In addition, there may be some number of auxiliary buffers (these are never displayed) into which fragments may be rendered. Any of the buffers may be individually enabled or disabled for fragment writing.

A particular copy of OpenGL may not provide depth, stencil, accumulation, or auxiliary buffers. Further, only some subset of the left & right front and left & right back buffers may be present. Different buffers may be available (each with varying numbers of bits) depending on the platform and window system on which OpenGL is running. Every window system must, however, provide at least one window type with a front (left) color buffer, and depth, stencil, and accumulation buffers. This guarantees a minimum configuration that a programmer may assume is present no matter where an OpenGL program is run.

4.6 Per-Fragment Operations

Before being placed into its corresponding frame buffer location, a fragment is subjected to a series of tests and modifications, each of which may be individually enabled, disabled, and controlled. The tests and modifications include the stencil test, the depth buffer test (typically used to achieve hidden surface removal), and blending. We briefly describe only a subset of the tests; for specifics, the reader should consult [10].

The stencil test, when enabled, compares the value in the stencil buffer corresponding to the fragment with a reference value. If the comparison succeeds, then the stored stencil value may be modified by a function such as increment, decrement, or clear, and the fragment proceeds to the next test. If the test fails, the stored value may be updated using a different function, and the fragment is discarded. Similarly, the depth buffer test compares the fragment's depth value with the corresponding value stored in the depth buffer. If the comparison succeeds, the fragment is passed to the next stage, and the fragment's depth value replaces the value stored in the depth buffer (if the depth buffer has been enabled for writing). If the comparison fails, the fragment is discarded, and no depth buffer modification occurs.

Blending mixes a fragment's color with the corresponding color already stored in the framebuffer (blending occurs once for each color buffer enabled for writing). The exact blending function may be specified with **glBlendFunction**.

Blending is the operation that actually achieves antialiasing for RGBA colors. Recall that the coverage computation only modifies a fragment's alpha value; this alpha value must be used to blend the fragment color with the already stored background color to obtain the antialiasing effect. Blending is also used to achieve transparency.

In addition to modifying individual framebuffer values with a series of fragments, a whole buffer or buffers may be cleared to some specifiable constant value. Clear values are maintained for the color buffers (all color buffers share a single value), the stencil buffer, the depth buffer, and the accumulation buffer.

4.7 Miscellaneous Functions

4.7.1 Evaluators

Evaluators allow the specification of polynomial functions of one or two variables whose values determine primitives' vertex coordinates, normal coordinates, color, or texture coordinates. A polynomial map, specified in terms of the Bezier basis[1] may be given for any of these groups of values. Once defined and enabled, the maps are invoked in one of two ways. The first way is to cause a single evaluation of each enabled map by specifying a point in the maps' domain using **glEvalCoord**. This command is meant to be placed between **glBegin** and **glEnd** so that individual primitives may be built each of which approximates a portion of a curve or surface. The second method is to specify a grid in domain space using **glEvalMesh**. Each vertex of the evaluated grid is a function of the defined polynomials. **glEvalMesh** generates its own primitives, and thus cannot be placed between **glBegin** and **glEnd**.

The evaluator interface provides a basis for building a more general curve and surface package on top of OpenGL. One advantage of providing the evaluators in OpenGL instead of a more complex NURBS interface is that applications that represent curves and surfaces as other than NURBS or that make use of special surface properties still have access to efficient polynomial evaluators (that may be implemented in graphics hardware) without incurring the costs of converting to a NURBS representation.

4.7.2 Display Lists

A display list encapsulates a group of OpenGL commands so that they may be later issued (in the order originally specified) by simply naming the display list. This is accomplished by surrounding the commands to be encapsulated with **glBeginList** and **glEndList**. **glBeginList** takes an integer argument that is the numeric name of the display list.

Display lists may be redefined, but not edited. The lack of editing simplifies display list memory management in the OpenGL server, eliminating the performance penalty such management would incur. Display lists may, however, be nested (one display list may invoke another). An effect similar to display list editing may thus be obtained by: (1) building a list containing a number of subordinate lists; (2) redefining the subordinate lists.

A single display list is invoked with **glCallList**. **glCallLists** calls a series of display lists in succession. Arguments to **glCallLists** specify an array of integers that are added to a *list base* to form the series of display list numbers. **glCallLists** is useful to display a string of characters when the commands that generate each character have been encapsulated in their own display list. Section 6 gives an example using **glCallLists**.

4.7.3 Feedback and Selection

As described so far, OpenGL renders primitives into the framebuffer. OpenGL has two additional modes. *Feedback* mode returns information about primitives (vertex coordinates, color, and texture coordinates) after they have been processed but before they are rasterized. This mode is useful, for instance, if OpenGL output is to be fed to a pen plotter instead of a framebuffer.

In *selection* mode, OpenGL returns a *hit* whenever a (clipped) primitive lies within the view frustum. This mode is used, for instance, to determine which portions of a scene lie within a region of the window centered around the mouse position (this is often termed *picking*). The Utility Library provides routines to manipulate the transformations so that when the scene is redrawn, only those portions that lie within a specified region about a specified position will return hits. Each hit returns the contents of the *selection stack*, which may be manipulated as the scene is drawn. By appropriately manipulating the stack, the application can identify the scene features that intersected the selection region.

4.7.4 OpenGL State

Finally, the value of nearly any OpenGL parameter may be obtained by an appropariate *get* command. There is also a stack of parameter values that may be pushed and popped. For stacking purposes, all parameters are divided into 21 functional groups; any combination of these groups may be pushed onto the attribute stack in one operation (a pop operation automatically restores only those values that were last pushed). The get commands and parameter stacks make it possible to implement various libraries, each without interfering with another's OpenGL usage.

5 Integration in a Window System

OpenGL draws 3D and 2D scenes into a framebuffer, but to be useful in a heterogeneous environment, OpenGL must be made subordinate to a window system that allocates and controls framebuffer resources. We describe how OpenGL is integrated into the X Window System, but integration into other window systems (Windows NT, for instance) is similar.

X provides both a procedural interface and a network protocol for creating and manipulating framebuffer windows and drawing certain 2D objects into those windows. OpenGL is integrated into X by making it a formal X extension called *GLX*. GLX consists of about a dozen calls (with corresponding network encodings) that provide a compact, general embedding of OpenGL in X. As with other X extensions (two examples are Display PostScript and PEX), there is a specific network protocol for OpenGL rendering commands encapsulated in the X byte stream.

OpenGL requires a region of a framebuffer into which primitives may be rendered. In X, such a region is called a *drawable*. A *window*, one type of drawable, has associated with it a *visual* that describes the window's framebuffer configuration. In GLX, the visual is extended to include information about OpenGL buffers that are not present in unadorned X (depth, stencil, accumulation, front, back, etc.).

X also provides a second type of drawable, the *pixmap*, which is an off-screen framebuffer. GLX provides a *GLX pixmap* that corresponds to an X pixmap, but with additional buffers as indicated by some visual. The GLX pixmap provides a means for OpenGL applications to render off-screen into a software buffer.

To make use of an OpenGL-capable drawable, the programmer creates an OpenGL context targeted to that drawable. When the context is created, a copy of an OpenGL renderer is initialized with the visual information about the drawable. This OpenGL renderer is conceptually (if not actually) part of the X server, so that, once created, an X client may *connect* to the OpenGL context and issue OpenGL commands (Figure 5). Multiple OpenGL contexts may be created that are targeted to distinct or shared drawables. Any OpenGL-capable drawable may also be used for standard X drawing (those buffers of the drawable that are unused by X are ignored by it). Calls are provided to synchronize drawing between OpenGL and X; it is the client's responsibility to carry out this synchronization if required.

A GLX client that is running on a computer of which the graphics subsystem is a part may avoid passing OpenGL tokens through the X server. Such direct rendering may result in increased graphics performance since the overhead of token encoding, decoding, and dispatching is eliminated. Direct rendering is supported but not required by GLX (a client may determine whether or not a server provides direct rendering). Direct rendering is feasible because sequentiality need not be maintained between X commands and OpenGL commands except where commands are explicitly synchronized.

6 Example: Three Kinds of Text

To illustrate the flexibility of OpenGL in performing different types of rendering tasks, we outline three methods for the particular task of displaying text. The three methods are: using bitmaps, using line segments to generate outlined text, and using a texture to generate antialiased text.

The first method defines a font as a series of display lists, each of which contains a single bitmap:

```
for i = start + 'a'  to  start + 'z' {
  glBeginList(i);
    glBitmap( ... );
  glEndList();
}
```

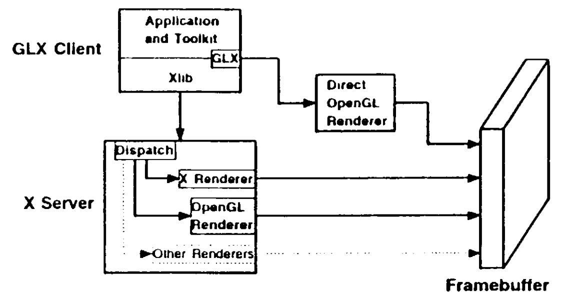


Figure 5. GLX client, X server, and OpenGL renderers.

Recall that glBitmap specifies both a pointer to an encoding of the bitmap and offsets that indicate how the bitmap is positioned relative to previous and subsequent bitmaps. In GLX, the effect of defining a number of display lists in this way may also be achieved by calling glXUseXFont. glXUseXFont generates a number of display lists, each of which contains the bitmap (and associated offsets) of a single character from the specified X font. In either case, the string "Bitmapped Text" whose origin is the projection of a location in 3D is produced by

```
glRasterPos3i(x, y, z);
glListBase(start);
glCallLists("Bitmapped Text", 14, GL_BYTE);
```

See Figure 6a. glListBase sets the display list base so that the subsequent glCallLists references the characters just defined. The second argument to glCallLists indicates the length of the string; the third argument indicates that the string is an array of 8-bit bytes (16- and 32-bit integers may be used to access fonts with more than 256 characters).

The second method is similar to the first, but uses line segments to outline each character. Each display list contains a series of line segments:

```
glTranslate(ox, oy, 0);
```

```
glBegin(GL_LINES);
  glVertex(...);
    ...
glEnd();
glTranslate(dx-ox, dy-oy, 0);
```

The initial glTranslate updates the transformation matrix to position the character with respect to a character origin. The final glTranslate updates that character origin in preparation for the following character. A string is displayed with this method just as in the previous example, but since line segments have 3D position, the text may be oriented as well as positioned in 3D (Figure 6b). More generally, the display lists could contain both polygons and line segments, and these could be antialiased.

Finally, a different approach may be taken by creating a texture image containing an array of characters. A certain range of texture coordinates thus corresponds to each character in the texture image. Each character may be drawn in any size and in any 3D orientation by drawing a rectangle with the appropriate texture coordinates at its vertices:

```
glTranslate(ox, oy, 0);
glBegin(GL_QUADS)
  glTexCoord( ... );
  glVertex( ... );
    ...
glEnd();
glTranslate(dx-ox, dy-oy, 0);
```

If each group of commands for each character is enclosed in a display list, and the commands for describing the texture image itself are enclosed in another display list called TEX, then the string "Texture mapped text!!" may be displayed by:

```
glCallList(TEX);
glCallLists("Texture mapped text!!", 22, GL_BYTE);
```

One advantage of this method is that, by simply using appropriate texture filtering, the resulting characters are antialiased (Figure 6c).

7 Conclusion

OpenGL is a flexible procedural interface that allows a programmer to describe a variety of 3D rendering tasks. It does not enforce a particular method of describing 3D objects, but instead provides the basic means by which those objects, no matter how described, may be rendered. This mechanistic view of rendering provides for efficient use of graphics hardware, whether that hardware is a simple framebuffer or a graphics subsystem capable of directly manipulating 3D data. OpenGL is rendering-only, so it is independent of the methods by which user input and other window system functions are achieved, making the rendering portions of a graphical program that uses OpenGL platform-independent.

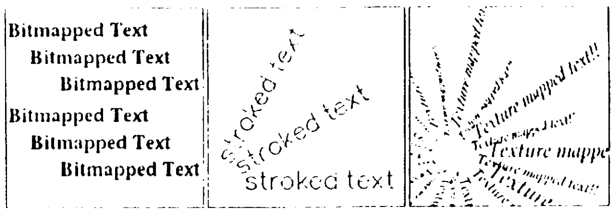


Figure 6 (a) Bitmap example (b) Stroke font example. (c) Texture mapped font example.

Because OpenGL imposes minimum structure on 3D rendering, it is an excellent base on which to build libraries for handling structured geometric objects, no matter what the particular structures may be. Examples of such libraries include object-oriented graphics toolkits that provide methods to display and manipulate complex objects endowed with a variety of attributes[11][12]. A library that uses OpenGL for its rendering inherits OpenGL's platform independence, making such a library available to a wide programming audience.

References

[1] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Boston, Ma., second edition, 1990.

[2] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. In *Proceedings of SIGGRAPH '90,*

pages 309-318, 1990.

[3] Paul S. Heckbert. A survey of texture mapping. *IEEE CG & A*, pages 56-67, November 1986.

[4] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley, Reading, Mass., 1986.

[5] International Standards Organization. International standard information processing systems — computer graphics — graphical kernel system for three dimensions (GKS-3D) functional description. Technical Report ISO Document Number 9905:1988(E), American National Standards Institute, New York, 1988.

[6] Jeff Stevenson. PEXlib specification and C language binding, version 5.1P. *The X Resource*, Special Issue B, September 1992.

[7] Adrian Nye. *X Window System User's Guide*, volume 3 of *The Definitive Guides to the X Window System.* O'Reilly and Associates, Sebastapol, Ca., 1987.

[8] Paula Womack, ed. PEX protocol specification and encoding, version 5.1P. *The X Resource*, Special Issue A, May 1992.

[9] PHIGS+ Committee, Andries van Dam, chair. PHIGS+ functional description, revision 3.0. *Computer Graphics*, 22(3):125-218, July 1988.

[10] Mark Segal and Kurt Akeley. The OpenGL graphics system: A specification. Technical report, Silicon Graphics Computer Systems, Mountain View, Ca., 1992.

[11] Paul S. Strauss and Rikk Carey. An object-oriented 3D graphics toolkit. In *Proceedings of SIGGRAPH '92*, pages 341-349, 1992.

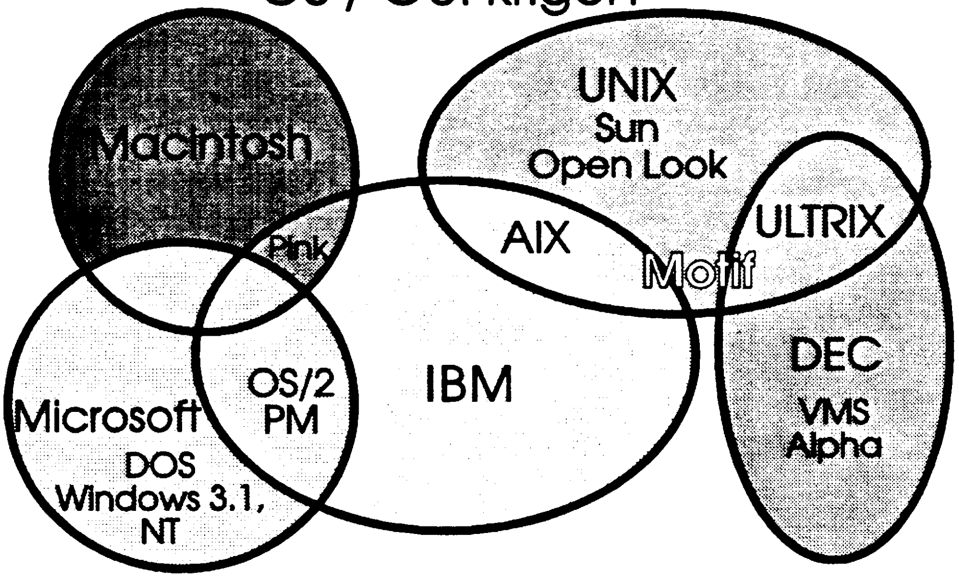
[12] Garry Wiegand and Bob Covey. *HOOPS Reference Manual, Version 3.0.* Ithaca Software, 1991.

Toolkit til udvikling af brugergrænseflader uafhængig af platforme

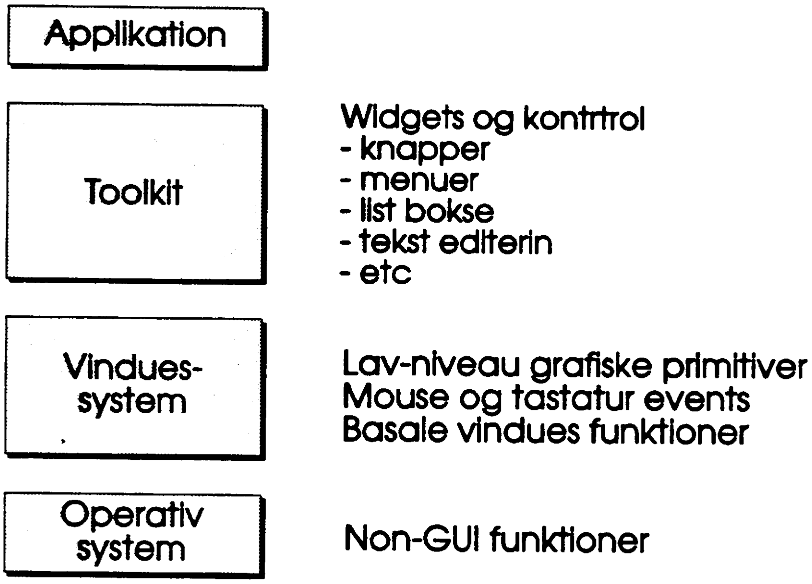
Kjeld Gammelgaard

Scanex • Lærkevænget 17 • 2970 Hørsholm • tlf / fax 45 76 86 20

Udviklernes udfordring
OS / GUI krigen

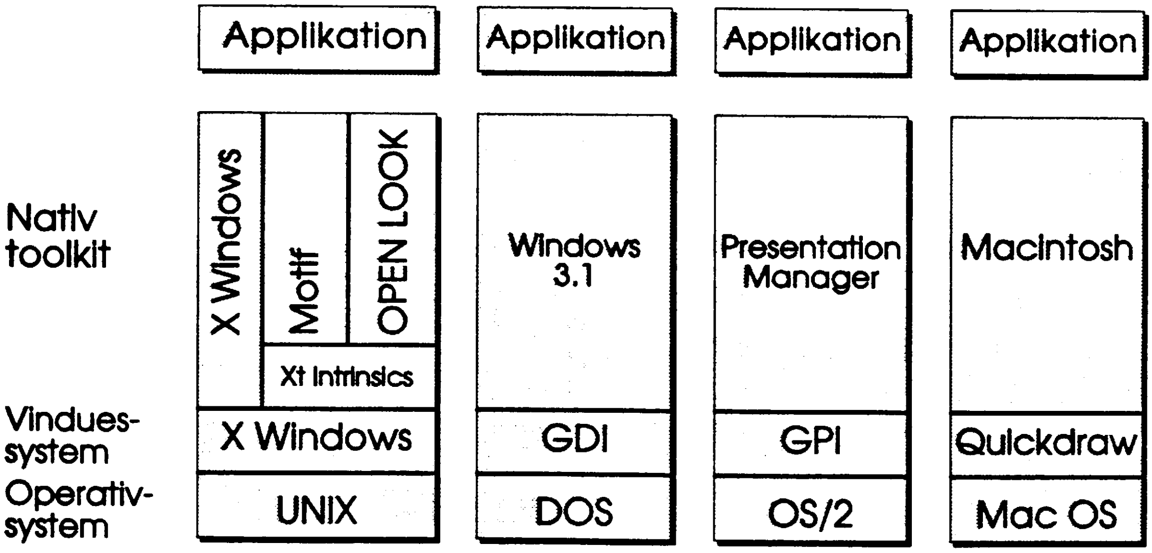


Standard GUI arkitektur



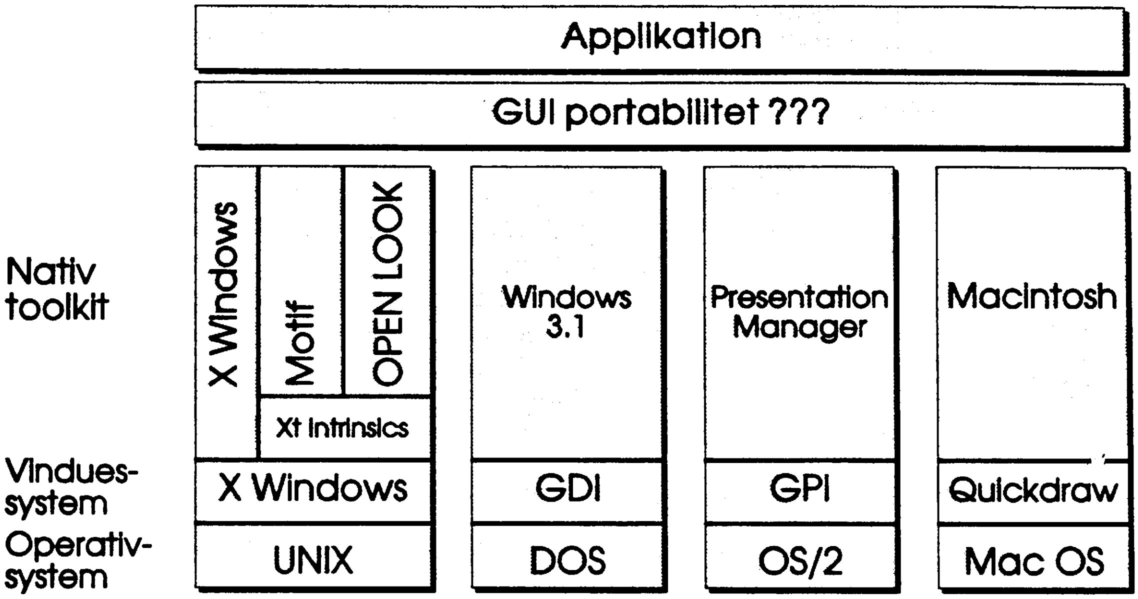
Applikation	
Toolkit	Widgets og kontrtrol - knapper - menuer - list bokse - tekst editerin - etc
Vindues-system	Lav-niveau grafiske primitiver Mouse og tastatur events Basale vindues funktioner
Operativ system	Non-GUI funktioner

GUI teknologier



	Applikation	Applikation	Applikation	Applikation
Nativ toolkit	X Windows / Motif / OPEN LOOK Xt Intrinsics	Windows 3.1	Presentation Manager	Macintosh
Vindues-system	X Windows	GDI	GPI	Quickdraw
Operativ-system	UNIX	DOS	OS/2	Mac OS

Fælles GUI API toolkit



Applikation

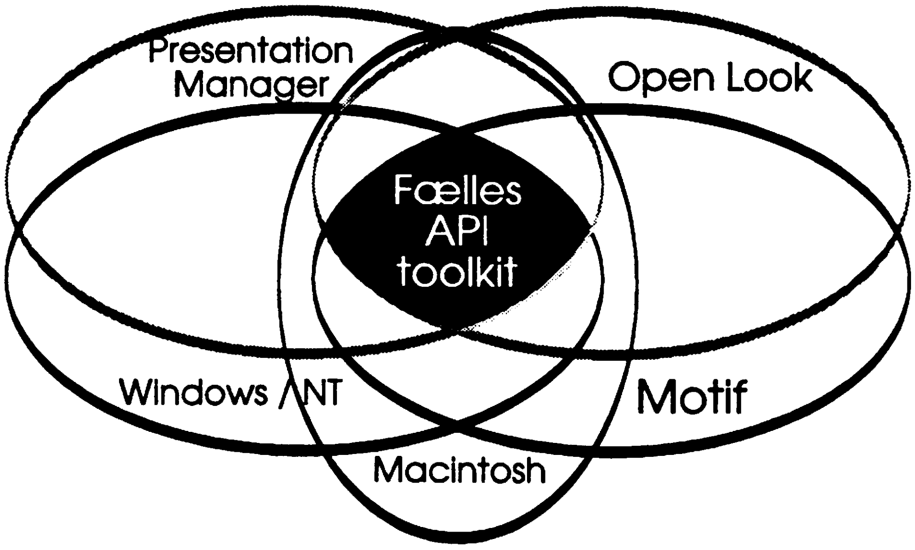
GUI portabilitet ???

Nativ toolkit: X Windows, Motif, OPEN LOOK, Xt Intrinsics | Windows 3.1 | Presentation Manager | Macintosh

Vindues-system: X Windows | GDI | GPI | Quickdraw

Operativ-system: UNIX | DOS | OS/2 | Mac OS

Mindste fællesnævner



Ønskeseddel til det idéelle toolkit

- Staben behøver kun lære ét tool, som kan anvendes på alle platforme
  Ikke behov for ekspertise på hver platform og vinduesmiljø (også karakter mode)
- Kort indlæringskurve
- Komplet widget bibliotek
  Bibliotek af widgets fra alle vinduesmiljøer tilsammen plus tredjeparts widgets og eget udviklede widgets
- Anvendelig til både små og store projekter
- Udviklingsmiljøet kan være forskellig fra afviklingsmiljøet
- C-baseret
- Objekt-orienteret paradigma
- Muligt at udvide

SCANEX

Minimumskrav til toolkit

1 Grafisk kapacitet på pixel niveau
2 Åben arkitektur
3 Portabelt
4 Flexiblet
5 Komplet widgets sæt
6 Udvidelsesmulighed
7 Performance ved afvikling

SCANEX

Minimumskrav
1. Grafisk kapacitet på pixel niveau

- Tegning af linier og andre primitive objekter
- Ikoner og grafisk display
- Farver
- Fonte
- Mulighed for at anvende ethvert image på skærmen

SCANEX

Minimumskrav
2. Åben arkitektur

- Samme applikation kan køre på mange platforme
- Toolkit uafhængig af platforme

SCANEX

Minimumskrav
3. Portabilitet

- Platform til udvikling kan være en anden end til afvikling
- Minimal kode skal ændres, når platforme ændres

Minimumskrav
4. Flexibilitet

- Fra små til store applikationer
- Simpel kode
- Genbrug af kode
- Native toolkit kan også anvendes

Minimumskrav
5. Komplet widgets sæt

- Der kan anvendes et super-set af widgets
  - sum af widgets fra alle de understøttede platforme
- Høj-niveau toolkit, der håndtere interface på den aktuelle platform
  - udviklere aflastes for at beskæftige sig med lav-niveau detaljer

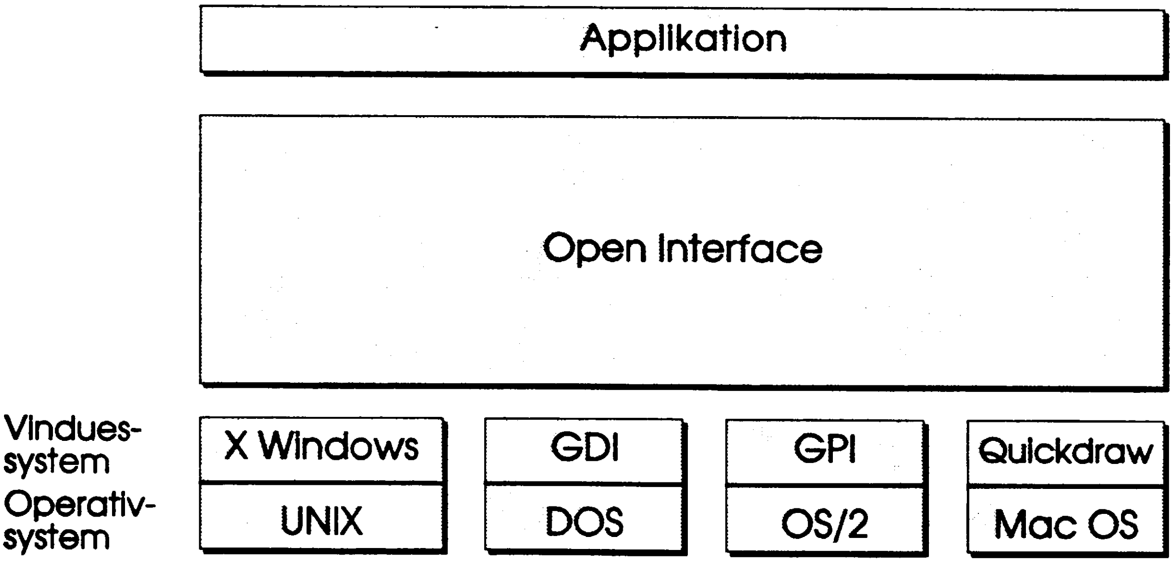
Minimumskrav
6. Udvidelsesmulighed

- Muligt at udvide toolkittet
- Muligt at ændre basis-widgets til konkrete applikationer
- Muligt at tilføje nye attributter til basis-widgets
- Muligt at tilføje hel nye komplette widgets

Minimumskrav
7. Performance ved afvikling

- Display
- Eksekvering

100% GUI portabilitet

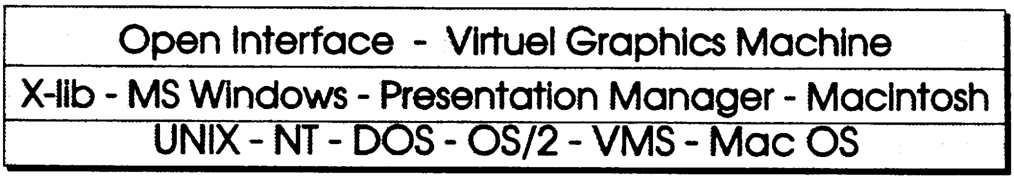


Applikation

Open Interface

Vindues-system	X Windows	GDI	GPI	Quickdraw
Operativ-system	UNIX	DOS	OS/2	Mac OS

Virtuel Graphics Machine



Open Interface - Virtuel Graphics Machine

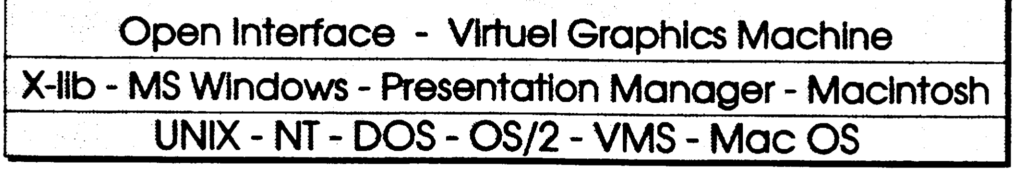
X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

SCANEX

Kærne bibliotek

FUNKTION	MODUL
Basale konstanter, datatyper, makroer:	Base, Err
Maskinafhængige moduler:	Mch
Memory mangagement:	Ptr
Streng håndtering:	Str, VStr
Arrays, træstrukturer m.v.	Array, Cell



Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

SCANEX

Ressource bibliotek

FUNKTION	MODUL
Kontrol af biblioteker:	RLib
Kontrol af ressourcer:	Res
Kontrol af faste data:	PFld

Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

VGM bibliotek

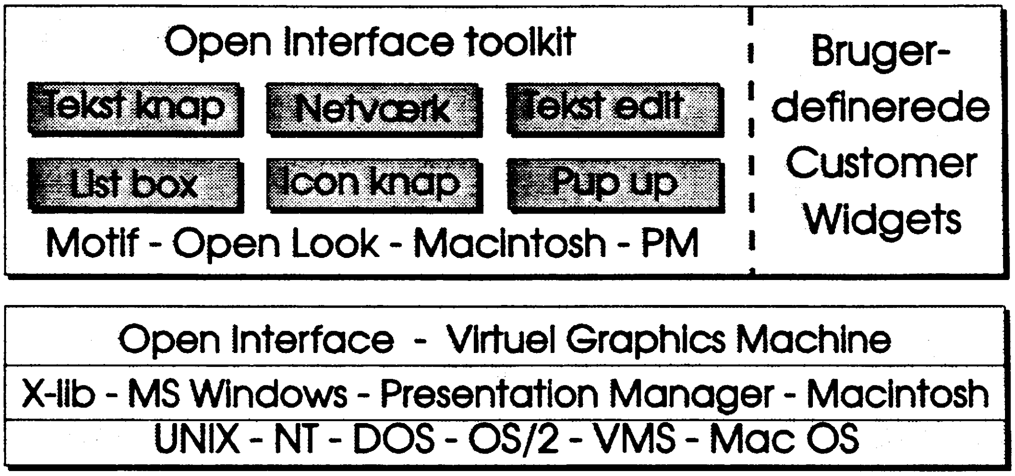
FUNKTION	MODUL
Tegning:	Rect, Rgn, Draw
Event manager:	Event
Grafiske ressourcer:	Color, Font, Icon, Curs, Pen, Patt
Tastatur ressourcer:	KyElt, KyLst
Widgets, paneler og vinduer:	Wgt, Panel, Win, Menu, MBar
Display og vinduessystener:	Dsply
Platformsspecifik. ressourcer:	Mac, MSW, NT, PM, X
Print:	Print

Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

Open Interface arkitektur



Open Interface toolkit

Tekst knap | Netværk | Tekst edit

List box | Icon knap | Pup up

Motif - Open Look - Macintosh - PM

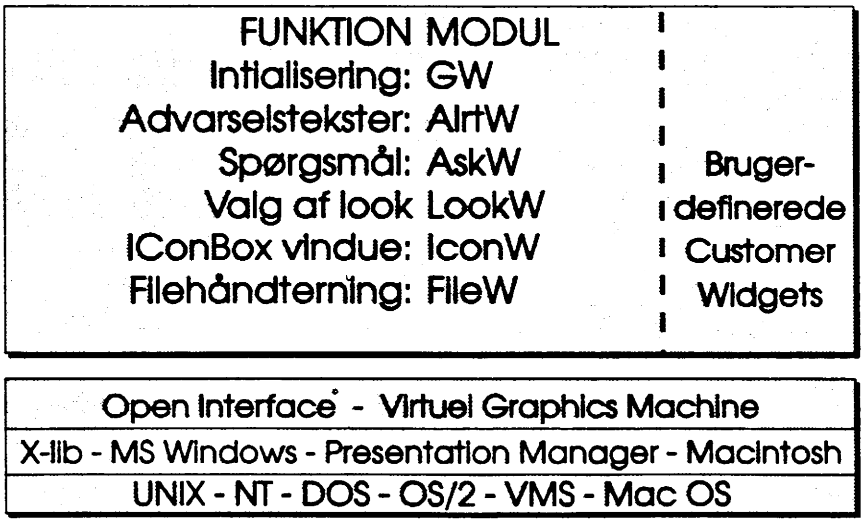
Bruger-
definerede
Customer
Widgets

Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

Open Interface Standard vinduesbibliotek



FUNKTION	MODUL
Intialisering:	GW
Advarselstekster:	AlrtW
Spørgsmål:	AskW
Valg af look	LookW
IConBox vindue:	IconW
Filehåndterning:	FileW

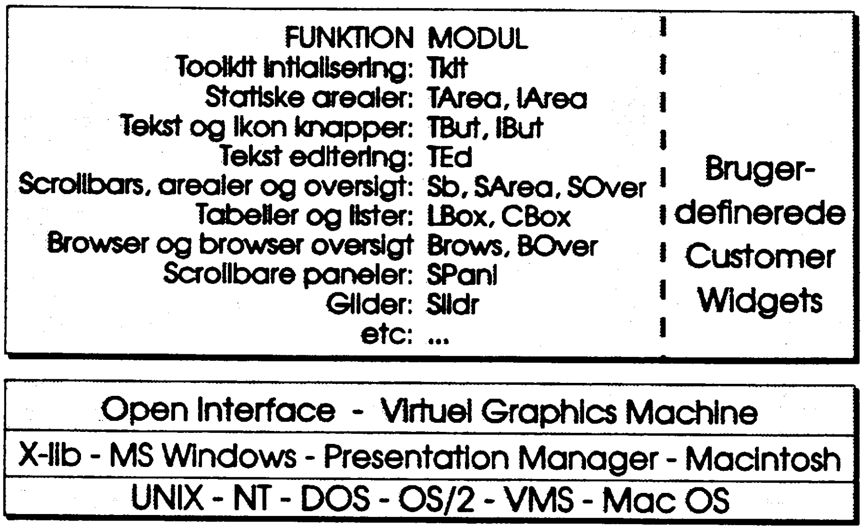
Bruger-
definerede
Customer
Widgets

Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

Open Interface toolkit bibliotek



FUNKTION	MODUL
Toolkit Intialisering:	Tkit
Statiske arealer:	TArea, IArea
Tekst og ikon knapper:	TBut, IBut
Tekst editering:	TEd
Scrollbars, arealer og oversigt:	Sb, SArea, SOver
Tabeller og lister:	LBox, CBox
Browser og browser oversigt	Brows, BOver
Scrollbare paneler:	SPanl
Glider:	Slidr
etc:	...

Bruger-
definerede
Customer
Widgets

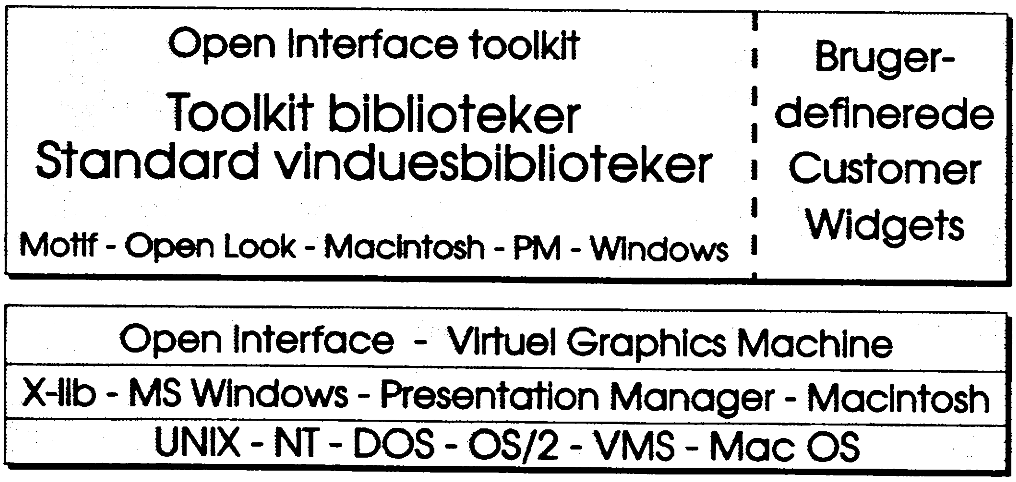
Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

SCANEX

Open Interface arkitektur



Open Interface toolkit

Toolkit biblioteker
Standard vinduesbiblioteker

Motif - Open Look - Macintosh - PM - Windows

Bruger-
definerede
Customer
Widgets

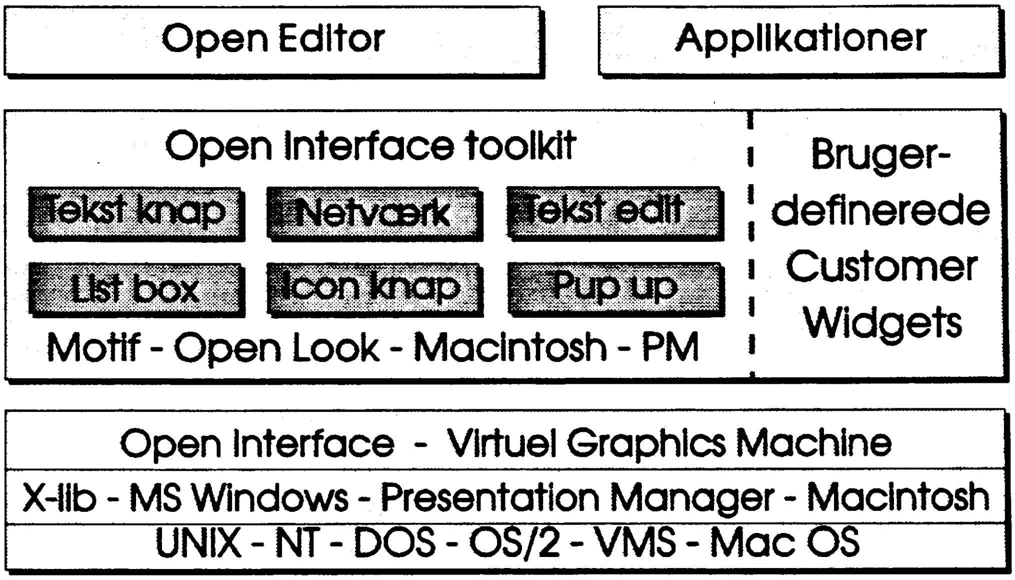
Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

SCANEX

Open Interface arkitektur



Open Editor

Applikationer

Open Interface toolkit

Tekst knap

Netværk

Tekst edit

List box

Icon knap

Pup up

Motif - Open Look - Macintosh - PM

Bruger-
definerede
Customer
Widgets

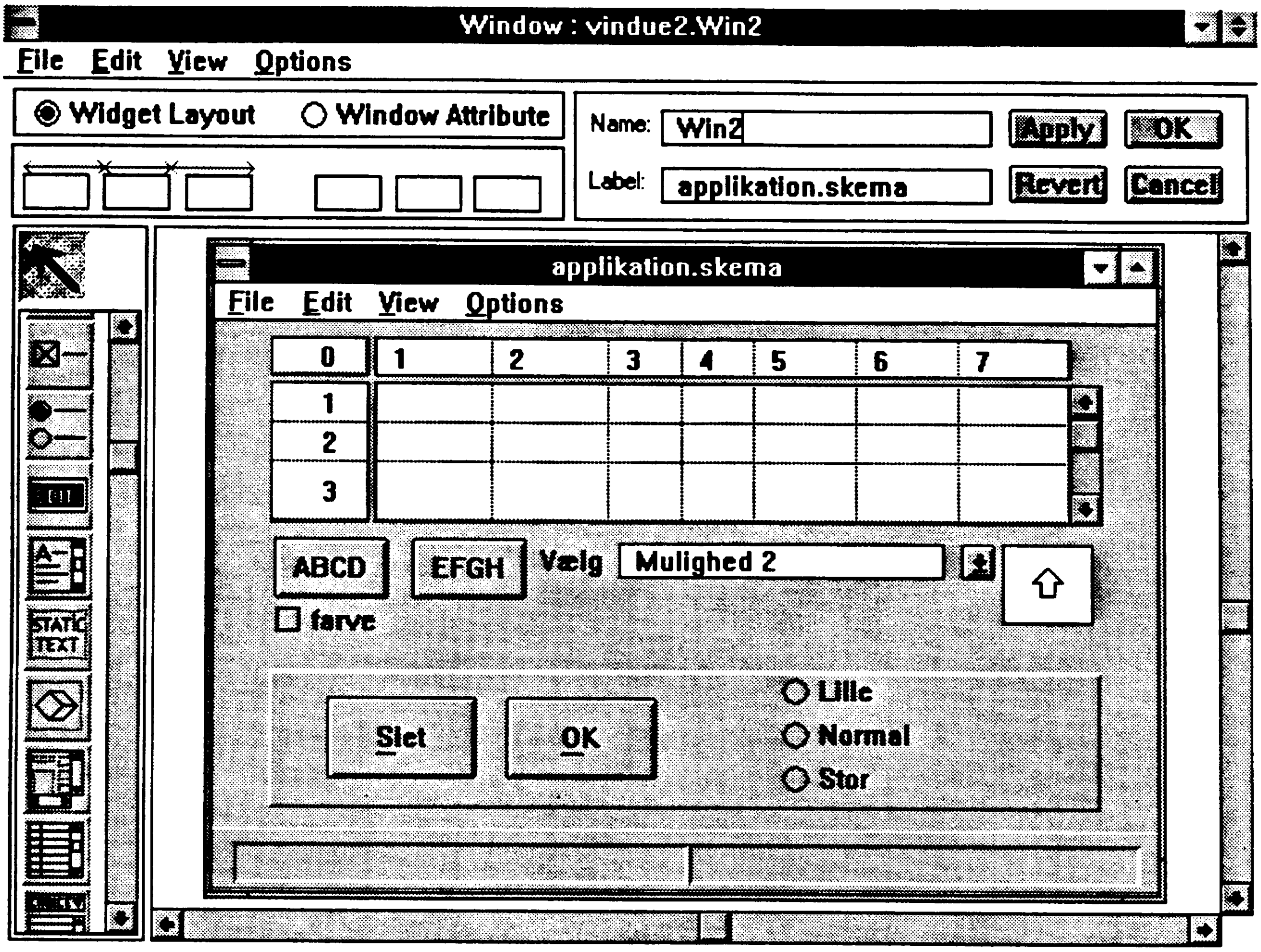
Open Interface - Virtuel Graphics Machine

X-lib - MS Windows - Presentation Manager - Macintosh

UNIX - NT - DOS - OS/2 - VMS - Mac OS

Open Interface

- Template kode generering
- Regenerering af kode
- Håndtering af lav-niveau detaljer
  - ikke behov for at være guru på hver platform
- Stærk og konsistens intern struktur
- Objekt-orienteret paradigme
- Kan bruge widgets på en platform selv om de ikke indgår i det originale vinduesmiljø
- C portabilitet med makro support
- Muligt at ændre GUI uden at ændre kode



Name: Win2

Label: applikation.skema

0	1	2	3	4	5	6	7
1							
2							
3							

Vælg Mulighed 2

Open Interface Elements indhold

Widgets

- Push button
- Radio button
- Check button
- List button
- Tabel
- Netværks browser
- Menu bar
- Popup menu
- Editérbar felt med format kontrol
- Multi font / multi linie tekst edit
- Valg liste
- Skrollbar panel
- Push pin
- Slider og Gauge
- Business grafik
  - **Lagkage diagram**
  - **Linie diagrammer**
  - **Kasse diagrammer**
- Farvepaletter
- Farve ikoner

Dialoger

- Advarsels vinduer
- Hjælpe vinduer
- Printer setup
- grafisk brower oversigt
- Farve vælger
- Font vælger
- File vælger

Script Element

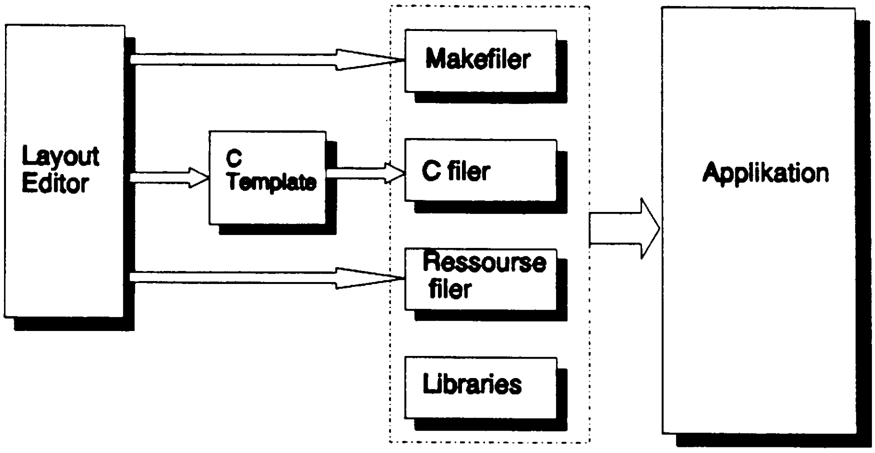
- 3 GL performance
- message baseret
- Event og verber
- Udvidelsesmuligheder og sprog uafhængighed

Access forbindelse

- Access til tværgående databaser
  - **relations databaser**
  - **hierakisek databaser**
  - **flade databse**
  - **data object mappped**
  - **standard SQL acces**
  - **understøter sammenkobling af flere databaser**

Understøttede platforme

- UNIX
  - Data General Aviion
  - DEC RISC Ultrix
  - DEC Alpha AXP OSF/1
  - HP 900 serie 300/400/700/800
  - IBM RS/6000 AIX
  - ISC UNIX
  - MIPS ABI
  - SCO ODT
  - Silicon Graphics
  - Solaris for SPARC og Intel
  - SONY NEWS (RISC og CISC)
  - Unixware
- Microsoft Windows NT
- Microsoft Pen Windows
- Digital Alpha NT
- DOS.Microsoft Windows
- DOS-karakter mode
- OS/2 Presentatione Manger
- Macintosh
- Digital Open VMS Alpha AXP og VAX/VMS



Layout Editor

C Template

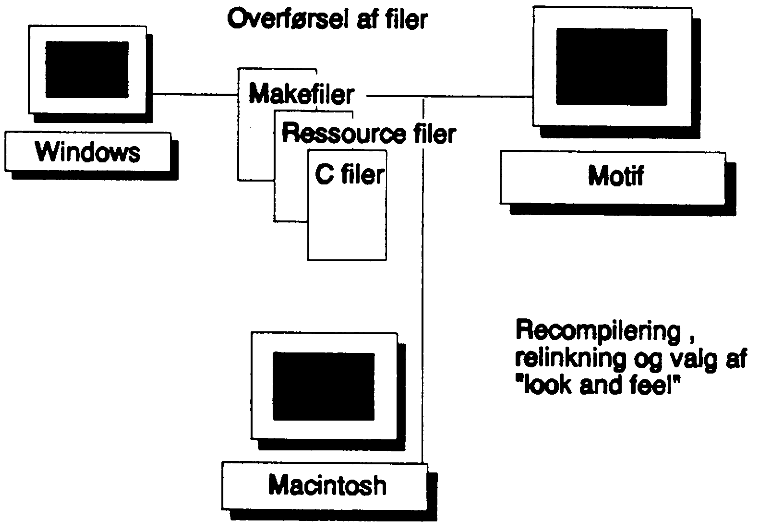
Makefiler

C filer

Ressourse filer

Libraries

Applikation



Overførsel af filer

Windows

Makefiler

Ressource filer

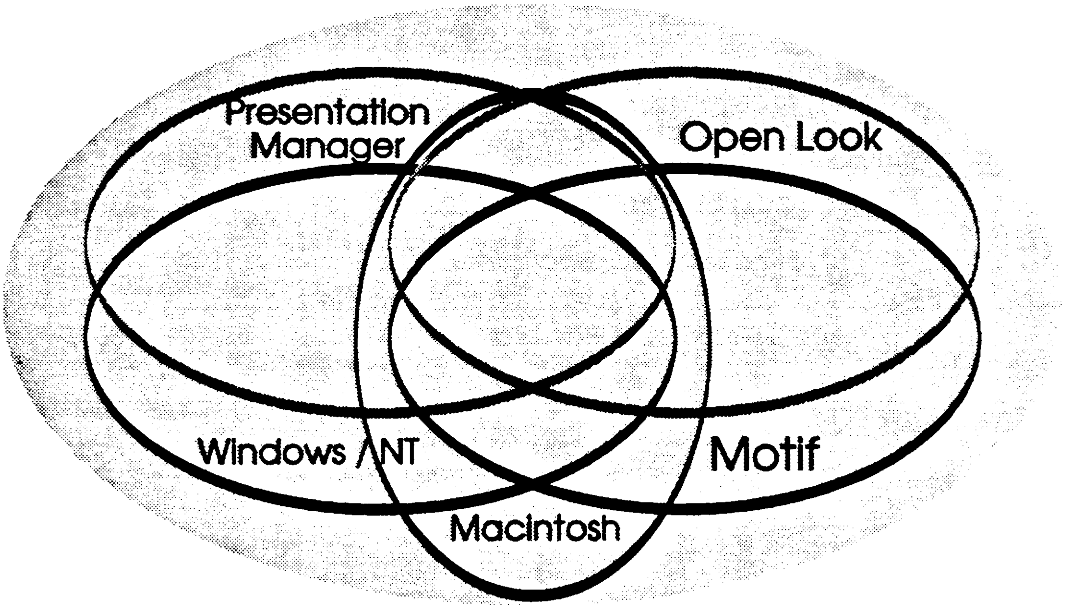
C filer

Motif

Macintosh

Recompilering , relinkning og valg af "look and feel"

Open Interface - superset arkitektur



Open Interface
100% GUI portabilitet

UNIX
Sun
Open Look

Macintosh

Pink

AIX

ULTRIX

Motif

DEC
VMS
Alpha

OS/2
PM

IBM

Microsoft
DOS
Windows 3.1,
NT

Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

Special Reprint

Publisher
PATRICIA B. SEYBOLD

This article is a reprint from the July, 1992 issue of *Patricia Seybold's Open Information Systems*. It has been reproduced here exactly as originally published.

The Patricia Seybold Group (PSG) is a computer industry publishing, research, and consulting firm in Boston, Massachusetts. The PSG publishes three monthly research newsletters—*The Office Computing Report, Distributed Computing Monitor, and Open Information Systems* —and a monthly audionewsletter titled *Paradigm Shift*.

Patricia Seybold Group
148 State Street
7th Floor
Boston, MA 02109

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 7, No. 7 • ISSN: 1058-4161 • July 1992

Tool Focus: Neuron Data

Will Open Interface 2.0 Nullify the GUI Wars?

By Andrew D. Wolfe, Jr.

Development of the Virtual Toolkit Approach

The promise of graphical user interfaces (GUIs) for Unix was to give novice users all the power of Unix but to hide the complexities. Unfortunately, when developers encountered the voluminous libraries of the X Window System and its minimal—at best—interface templates, the graphical revolution went into slow motion. Developing X-based programs was a grueling process. When standardized look-and-feel specifications and common widget libraries, such as OSF/Motif and OpenLook, arrived on the scene, the situation improved somewhat. But twiddling C source code to design and edit graphical objects has been a handicap for X Window programming compared to the ease of writing to the more abstract interfaces of the Macintosh and Microsoft Windows. Developers began to address this problem, and GUI-builder programs appeared in the X Window community, some of which rolled over as publicly available products. In these environments, programmers could paint the display layout of a GUI on the screen, have the C code generated for them, and then plug in regular software subroutines underneath. X Window development productivity improved markedly with the availability of these tools. (See "Integrating Applications in the Real World," *Open Information Systems*, Vol. 7, No. 7.)

Sammenligning af GUI toolkits	Portabel													Ikke portabel
	Super-set	Pseudo superset					Virtuelle tool kits							
	Fuld porta. C API	C API		O.O.			4GL			O.O.	C API			
	Open Interface	WNDX	Galaxy	Obj. Interf. Lib.	Object Works	Quest	Dialog Manager	JAM	Easel	CommomView	XVT	Aspekt	Open UI	
C bibliotek	+	+	+		-	-	+	+			(+)	+	+	
Grafisk layout toolkit	+	dårlig	+		-	+	+	(+)			(+)	+	(+)	
Interaktiv test	+	-	?		+	+	script	+			-	-	+	
Indeholder et script sprog	+	-	-		-	-	+	+			-	-	-	
Kodegenerering	+	-	?		-	+	-	-			+	-	header	
C++ support	+	+	?		-	-	?	?			+	-	+	
Portabel printer support	+	-	-		-	-	-	-			-	-	-	
Portabel font support	+	?	?		?	?	-	-			-	-	-	
Portable bruger def. widgets	+	-	?		+	-	-	-			-	-	(+)	
Dynamiske widgets	+	?	?		?	?	?	?			-	?	?	
Understøtter shared library	+	?	?		?	?	?	?			-	?	?	
Nat. sprog konvertering	+	+	?		-	+	+	+			-	-	-	
Ingen runtime fortolkning	+	+	?		smalltalk	+	-	-			+	+	+	
Performance	+	+	+		-		-	-			(+)	-	?	
	+													
Motif	+	+	+		+	+	+	+			+	+	+	
OpenLook	+	-	+		+	-	?	?			+	+	+	
Windows	+	+	+		+	-	+	+			+	+	+	
OS/2	+	-	?		+	-	+	-			+	-	-	
Mac	+	+	?		+	-	-	-			+	-	-	
Karakter mode	+	-	?		-	-	+	+			+	+	+	

Tool Focus: Neuron Data

Will Open Interface 2.0 Nullify the GUI Wars?

However, the market for GUI applications has remained fragmented, not only between the X Window look-and-feel camps, OSF/Motif and OpenLook, but also between them and the much larger Macintosh and Windows markets. Neuron Data (Palo Alto, California) was one of the few vendors that recognized that GUI portability among all these platforms could be both possible and profitable, for itself as well as for developers. This was the impetus behind Open Interface, a Neuron Data product that allows developers to design graphical interfaces which run unmodified on OSF/Motif, Microsoft Windows 3.x, Apple Macintosh, OpenLook, and OS/2 Presentation Manager platforms. (See "GUI Portability at Last," *Unix in the Office* Vol. 6, No. 3, March 1991, which covered the first release of Open Interface.) This product helped get developers past the toil of GUI programming and the turmoil of choosing a GUI platform and onto the business of delivering GUI applications to users.

Amplifying the "Kitchen-Sink" Approach

SUPPORTING *ALL* FUNCTIONS ON *ALL* PLATFORMS. When you look at the various GUI platforms, it quickly becomes apparent not only that they differ in appearance but also that each has many unique functions. OpenLook has its pushpin, Apple has its menu bar, Microsoft Windows has its "Task List" dialogue, and so forth. Even functions that look similar are often implemented in very different fashions, such as rubber-band vs. sticky drop-down menus. To build a virtual GUI toolkit supporting all of the GUI specs is a pretty complicated affair. Other cross-platform GUI builders have implemented a subset of the universe of widgets comprising those found across all supported GUIs. But, instead of this approach, Open Interface has implemented a toolkit that is a functional *superset* of all the widget collections. That is, *every* function of *every* GUI look-and-feel is supported on *all* Open Interface platforms. The resulting set of GUI widgets is very large, potentially overwhelming developers, but Neuron Data has opted for richness in order to have the same virtual toolkit on all supported environments without sacrificing functionality.

The Open Interface widget set is not simply a concatenation of all the others. It is a consolidation in which corresponding widgets—such as different pull-down menus—are merged into one representation. Open Interface implements each of the application-level widgets with distinct presentations for each of the target look-and-feel specs. All of these implementations are available on all target platforms and can be selected and changed on all platforms even while the application is running. Neuron Data often demonstrates this by running an application under Windows with Macintosh look-and-feel and on a Macintosh with a Windows look-and-feel. Many skeptics might dismiss such capabilities as frivolous, but they would be surprised, as were we, to hear of two large corporate users of Open Interface that were running the same look-and-feel—in one case Motif, in the other Presentation Manager—on all their platforms, including Macintosh.

OBJECT-ORIENTED STRUCTURE HELPS INTEGRATION. Good GUIs have a simplicity that masks their tremendous complexity at the code level. The intricacy and interrelationships of pull-down menus, text regions, and scrollbars can be an organizational nightmare. The X Window System and most GUI builders address the difficulty with a form of object-oriented programming, such as sharing type definitions among widget sets and packaging widgets within other widgets. Neuron Data goes further than most in supporting this development paradigm. The first step is encapsulating the data and functions into objects. Open Interface generates fully structured C code and uses C "typedef" definitions, header files, and macro definitions to enforce widget encapsulation. This prevents hand-written implemen-tation software from corrupting the widgets' internal data. Open Interface also supports

inheritance, but it doesn't allow a widget to inherit capabilities from more than one parent widget type—that is, it doesn't have multiple inheritance, but it does have basic polymorphism. By using the inheritance mechanism, the developer can build custom widgets that share common characteristics and can thus be integrated more easily into an application framework.

USING EXISTING TEXT-BASED SOURCE MANAGEMENT TOOLS. All GUI components, such as message files and resource files, that are developed under Open Interface are stored as text, not just C source code. Neuron Data uses ANSI C as its target language, since it is most directly portable and widely implemented. Unfor-tunately, the C language is not completely amenable to object orientation, which gives rise to the limitations in Neuron Data's object approach. When C++ standards are finalized, we expect Neuron Data to move quickly in that direction. The message files and other GUI resources that are stored in text form can be managed under a version control facility like SCCS or RCS along with the C source code. For use at run-time, Open Interface includes a resource compiler that builds resource files into a more efficiently-used binary format.

INTERNATIONALIZED TEXT CAPABILITY. In most software, the text labels, questions, and messages seen on the screen are fixed. The developer has programmed the messages directly into the source code. Consequently, it is difficult to make the software usable for those who speak other languages. Neuron Data addresses this problem by supporting the definition of message libraries. All text that an application presents to the user can be compiled into a file. Using a special filter program, the developer or integrator can generate different versions of the file with messages in different languages. While it might be preferable to have an "internationalization editor," allowing GUI entry of foreign-language messages right next to their developer-language versions, Open Interface's approach is adequate. The internationalization mechanism supports non-roman character sets like Cyrillic and includes multi-byte sets like Japanese Kanji. Unfortunately, text can only be written left-to-right today, which excludes the right-to-left Semitic languages like Hebrew and Arabic. Apart from this, however, Open Interface's internationalization is comprehensive and serves multilingual application projects admirably.

API ALLOWS EXTENSIONS, PORTABLE WIDGETS ... While support for standard widgets is necessary for GUI programming, it is not sufficient. Specific applications—such as cash flows, network management, or customer information—should have distinct widgets as a part of the GUI. Ordinarily, the application developer would have to implement these capabilities using the drawing primitives of the native windowing system on each target platform—largely bypassing the user interface toolkit. This represents a severe compromise to application portability. To address the need for widget extension, Neuron Data has developed a portability layer—implemented as an API—which is identical on all targets. This layer, called the Virtual Graphics Machine (VGM), consists primarily of drawing primitives, plus some windowing functions.

Understandably, the VGM is not a superset of all platform-specific drawing tools. However, it does allow the developer to build existing GUI widgets into new types of widgets. To build the examples mentioned above, a bar-graph widget could illustrate cash flows, a line-drawing widget with some sort of routing indicators could show network topology, and customer information could show maintenance schedules along a time line. The payoff of this approach is near-complete GUI portability. Not only will Open Interface support the developer in creating new widgets, but also the new widgets are inherently portable across all supported platforms.

...BUT REQUIRES A SPECIAL RUN-TIME. Open Interface applications run on a special run-time library. Neuron Data's software is considerably more optimized than standard Motif and OpenLook widget libraries. Almost all Open Interface widgets are actually "gadgets"—to use the X terminology—because they are not built using the window primitives of X.

Will Open Interface 2.0 Nullify the GUI Wars?

Neuron Data took this route to cut out the high overhead of X Window primitives, and it has added programming workarounds to flesh out standard widget functionality. The result is performance that matches and often surpasses that of the standard widget libraries. Applications running under X, moreover, are smaller than native applications, and library sizes are smaller.

The downside of Open Interface's powerful run-time is that it adds more layers to GUI software and supersedes standard GUI implementations. An application can behave differently from standard look-and-feel specifications, which is particularly bothersome when it runs side-by-side with native applications. Worse, it must be licensed for each machine on which it is run, adding to the cost. And while X hostings of Open Interface applications tend to be smaller and faster than ordinary Motif and OpenLook applications, the wholesale replacement of look-and-feel code makes applications running on the Macintosh larger than native applications.

Neuron Data packages its run-time library in shared-library form whenever the platform supports shared libraries. This includes most Unix hostings, VMS, Windows 3.x, and OS/2. This approach reduces application size on these particular platforms. While shared libraries are often a hassle to administer from the standpoint of shared memory parameters and file configuration, they can significantly improve memory and disk space management by reducing application size.

Enhancements in Open Interface 2.0

NEW WIDGETS AMPLIFY GUI COMPONENTS. Neuron Data has taken some of the initiative in developing GUI functionality beyond what is provided in the standard GUI toolkits. These basic widget sets include primitive interface components, such as buttons and text fields, as well as higher-level components, such as pull-down menus and scrolling lists. Open Interface 2.0 gives developers additional functionality in grouping and organizing existing widgets. Version 2.0 includes an enhancement of its "List Box" widgets, which implement various forms of row-column organization of data and can be used to build spreadsheet programs. The enhancements allow the display attributes of a List Box cell or range of cells to be changed from the attributes of the rest of the List Box. The List Box also allows more sophisticated input handling; about all it doesn't do now is clone the 1-2-3 macro language for you. A new Scrollable Panel extends a panel widget beyond the dimensions of its enclosing window. The extended area is accessed by using scroll bars. This is one of those "why didn't someone do that before?" capabilities. A "Choice Box" widget set allows selection of an item from a list—but not just a list of text strings. The Choice Box can allow selection of icons for things like drawing tools or text attributes. It's a natural for implementing tool palettes. Other widgets from Open Interface 1.0 have been enhanced in a similar fashion.

UPDATES TRACK SUPPORTED PLATFORMS. Since the first release of Open Interface, some of its supported platforms have had software upgrades. Open Interface 2.0 includes enhancements to adapt to the release of Microsoft Windows 3.1 and IBM OS/2 Version 2.0. Windows support now includes Win32 and TrueType font-rendering. Open Interface can also build software for 32-bit-mode OS/2 operation. Unfortunately, it has been scooped by the recent release of OSF/Motif 1.2; customers needing Motif 1.2 compliance will need to wait for the next release or a software patch. Tear-off menus and drag-and-drop will have to wait until the next release of Open Interface.

POSTSCRIPT TOOLS GIVE WYSIWYG PRINTING TO UNIX APPS. High-resolution printing of text and graphics was the enabling technology for desktop publishing, but the Unix community is largely left out. The PostScript imaging mechanism pioneered on Macintosh and embraced by DOS and Windows has not been standardized under Unix, which has resulted in few PostScript-aware Unix applications. Part of the difficulty is the fundamental difference between X Window imaging, which is bit-mapped, and PostScript, which is a higher-level abstraction. Under PostScript, if you draw a one-inch by one-inch square, it comes out that way on any printer. But under X, if you draw a 100-pixel by 100-pixel box, it can be displayed in practically any size and shape. Open Interface 2.0 supports a PostScript printing capability that is transparent to the application. By changing a switch, VGM rendering commands that would ordinarily drive a screen display are changed to generate a PostScript file. This relieves developers of having to generate their own PostScript printer files.

OPEN EDITOR NOW EDITS CUSTOM WIDGETS. Under Open Interface 1.0, developers could create new widgets but could not edit or adapt them with the tool's Open Editor. Since sizing and customization are part of the way GUI builders earn their keep, this inability to modify custom widgets was a serious deficiency in the product. Open Interface 2.0 has augmented the Open Editor so that custom widgets can be placed, sized, reshaped, colored, and otherwise modified just like those supplied by Neuron Data.

STRENGTHENED CODE MANAGEMENT. No matter how powerful the GUI builder, it won't write your entire application for you. Open Interface, like its competitors, still requires coding the application logic in C. All of these tools generate stub code into which the developer writes the implementation of the various functions. Once the stub has been fleshed out into a full source file, other GUI builders can't handle it any more. And if the developer has to change the GUI, the stub code must be regenerated and source text from the prior implementation must be pasted in and adapted. Open Interface 2.0 keeps these program texts in the development loop. GUI alterations are propagated to implementation code without damaging the handwritten portions of software. This eases maintenance of GUI definitions and implementation code. Added to its textual representation of GUI resources, these features give Open Interface superior project management capabilities.

Cheaper on the Desktop, Too. One nontechnical change is a significant reduction in run-time license fees. Run-times have been lowered from $250 to $95 for the PC and Mac, and from $500 to $190 on other systems. We feel that this makes Open Interface a much more realistic choice, especially for developers. The previous pricing structure simply couldn't be handled by ISVs trying to sell shrinkwrapped applications, even with bulk or bundling licenses. This reduction in run-time fees makes it feasible to use Open Interface for commercially sold applications.

Who's Committing to Open Interface?

NUMEROUS LICENSES—MOSTLY LARGE END-USER ORGANIZATIONS. Neuron Data has piled up the impressive figure of 100,000 run-times since its first release in June 1991. Most of these licenses have gone to large corporations for in-house application development. Neuron Data's customer list reads like the Fortune 1000, which is, in fact, a primary target market for Open Interface. For these companies, multiple desktop platforms are a fact of life, and Open Interface promises to bring some unity and order without compromising functionality. Unfortunately, a lot of the business in that market boils down to client/server database applications running on PCs and using some sort of mainframe data storage. However, several solid graphical database application builders have already staked out this market with GUI builder products. We suspect Open Interface will find limits to penetration except where there is already a significant number of Unix workstations. Indeed, customers cited by Neuron Data are largely in the engineering or financial industries, which do have significant Unix workstation penetration.

WHERE ARE THE APPLICATIONS? While Open Interface promises a brave new world, it will be off-the-shelf applications that deliver it. Such applications, however, remain rare, and Neuron Data can't give out the names of unannounced products or their developers. A product like Open Interface should really be paying off in several applications already. But where are they? In our view, many of the ISVs have largely gone ahead with their own GUI efforts, which may not be compatible with Open Interface. Of course, the aforementioned qualms about portability and the perennially cash-starved nature of the Unix software market might also explain vendors shying away. But what we suspect is that a not-inventedhere attitude prevails. Developers may simply not want to be dependent on another software company.

SOME ISVS AT WORK. Unfortunately, participation of independent software vendors in the Open Interface revolution is rather sparse. Neuron Data's biggest success so far is ARC-VIEW, from Environmental Systems Research Institute (ESRI) in Redlands, California. ESRI is one of the leading providers of Geographical Information Systems, and it believes Open Interface helped ARC-VIEW get to market much faster than it would have otherwise. American Management Systems is also developing products with Open Interface. Two aren't exactly a flood, but these are good, solid companies. Open Interface is moving well in Japan, however, a key payoff of its multi-language, multi-alphabet internationalization.

INTEGRATORS: KEY TO BROADER MARKET PENETRATION? The capabilities of Open Interface were really seized by system integrators. Customers like Anderson Consulting and EDS constitute a testimonial to Open Interface. Consultants and integrators don't take the time for setting corporate policies or for building from scratch. To have custom applications come out identically on multiple platforms with a short development time may catch the interest of large purchasers and developers more than a laundry list of features. The upscale integrators are, in our view, the acid test of Open Interface.

Conclusions Fills Some Important Gaps in X

Like Consistency. One of the serious defects in the common X Window programming model is that it is an amalgam of components operating at different levels. Even using a GUI builder, the developer may have to code to the look-and-feel toolkit, the basic X Window toolkit, the Xlib interface—even to socket-level functions—in the same application. In contrast, the Open Interface toolkit is completely self-contained. There are few real reasons, if any, to go outside the toolkit. Open Interface provides the comprehensive and consistent interface that none of the Unix/X Window vendors provide.

And Profitability. Another minor benefit Open Interface brings to X Window is the possibility that ISVs will make decent money. If a vendor can deliver on Windows and Macintosh, it could earn enough money to support on-going development that would pay off on X-based systems as well. The Windows/Macintosh market is as much as two orders of magnitude larger than the Unix desktop market. Open Interface allows Unix vendors to bring their products into this considerably more lucrative arena. We hope they have enough breadth of vision to make use of it. Or is it just pragmatism they require?

POWERFUL DEVELOPMENT AND ORGANIZATIONAL FRAMEWORK. The principal lessons that Open Interface and other GUI builders embody are that programs are built from more than source code and that each component should be built with an appropriate tool. The dreary chore, inevitable in early X Window develop-ment, of constructing pixel coordinates and bit-map data in source code, is not only time-consuming but inappropriate to the data being devised. Bit maps aren't numbers; they're images. Pull-down menus aren't null-terminated arrays of strings; they're pull-down menus. Text messages aren't character arrays; they're mes-sages. By supporting the partitioning of applications into distinct types of components, Open Interface is not only speeding development, it is also improving soft-ware organization.

PORTABILITY AND COMPLIANCE A QUESTION MARK. Notwithstanding its strengths and its flexibility, a product like Open Interface sometimes goes against the grain in the Unix market. Portability is such an innate concern that programmers often shy away from third-party software libraries like those in Open Interface—after all, it doesn't run on AT&T 3B2s. (What? You don't even remember the 3B2?) Perhaps more to the point, since its implementation omits standard run-times for look-and-feel specs, it could fall out of compliance with those specs. However, we feel such concerns are really immaterial. End users are little affected by minor deviations in look-and-feel and often want the best of all worlds anyway. And scrupulous developers should think about the piles of Windows PCs and Macintoshes they can support by "compromising" their portability.

HOW FAST CAN DEVELOPERS ABSORB AND MASTER IT? Neuron Data has clearly driven toward a portable GUI model that embodies heavy emphasis on high-level abstraction. It also embodies broad and varied functionality lifted from its supported platforms. Both characteristics will create a steep learning curve for the rank-and-file developer in creating applications. Simply navigating a widget set can be a daunting experience, even in single look-and-feel GUI builders. Under-standing in entirety all of Open Interface's libraries could be a very long process. The abstraction of Open Interface's add-on widgets, however, poses a different problem. The power in these widgets is veiled under an API that is deceptively simple but extremely, though subtly, rich. Neuron Data found that its Open Interface 1.0 customers didn't perceive all of the functionality they were getting, so Release 2.0 has substantial amounts of example software bundled in. Nonetheless, we feel it will take some time for developers to really leverage the capabilities of Open Interface.

IMPORTANT PIECE OF FUTURE PROGRAMMING. Open Interface goes a step beyond current programming tools. Not only does it build GUIs for multiple platforms—and very disparate ones, at that—it also structures applications to grow into new computing environments, like the workgroup-centered initiatives at Apple and Microsoft, and the distributed Unix environments like ONC and DCE. It brings GUI programming out of the text editor and lifts it above the GUI wars to where it should be: delivering functionality and usability to the end user. Open Interface 2.0 brings the promise of its first release closer to completion. GUI extensions are more thoroughly supported, the add-on widget set has been enlarged and enhanced, and niceties like PostScript printing are fleshing out the end-user capabilities available under the product. In our view, Open Interface is not only a valuable product, but it also could become the key to preserving the Unix beachhead on the desktop market.

August's *Unix in the Office* will address
Windows NT 3.1.

For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

Patricia Seybold's Computer Industry Reports

ORDER FORM

Please start my subscription to:		U.S.A.	Canada	Foreign
□ *Patricia Seybold's Office Computing Report*	12 issues per year	$385	$397	$409
□ *Patricia Seybold's Open Information Systems*	12 issues per year	$495	$507	$519
□ *Patricia Seybold's Distributed Computing Monitor*	12 issues per year	$495	$507	$519
□ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*	6 issues & tapes per year	$395	$407	$419
□ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*	6 issues per year	$295	$307	$319

Please send me a sample of: □ *Distributed Computing Monitor* □ *Open Information Systems* □ *Office Computing Report* □ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*

Please send me information on: □ Consulting □ Special Reports □ Conferences

□ **My check for $________ is enclosed.** □ **Please bill me.** □ **Please charge my subscription to: Mastercard/Visa/American Express** (circle one)

Name: ________________ Title: ________________

Company Name: ________________ Dept.: ________________

Address: ________________________________

City, State, Zip code, Country: ________________

Fax No.: ________________ Bus. Tel. No.: ________________

Card #: ________________

Exp. Date: ________________

Signature: ________________

Checks from Canada and elsewhere outside the United States should be made payable in U.S. dollars. You may transfer funds directly to our bank: Shawmut Bank of Boston, State Street Branch, Boston, MA 02109, into the account of Patricia Seybold Group, account number 20-093-118-6. Please be sure to identify the name of the subscriber and nature of the order if funds are transferred bank-to-bank.

Send to: Patricia Seybold Group: 148 State Street, Boston MA 02109; FAX: 1-617-742-1028; MCI Mail: PSOCG
To order by phone: call (617) 742-5200

IUR-892

Topics covered in Patricia Seybold's Computer Industry Reports in 1991 & 1992:

Back Issues are available, call (617) 742-5200 for more information.

Office Computing Report

1991—Volume 14

#	Date	Title
11	Nov.	Keyfile—Bringing Imaging and Workflow to the Desktop
12	Dec.	IBM/Lotus Relationship—Building a Platform for Communicating Applications

1992—Volume 15

#	Date	Title
1	Jan.	The Groupware Phenomenon—Does It Focus on the Proper Issues?
2	Feb.	Digital's TeamLinks—A Renewed Focus on the Client Desktop
3	Mar.	Requirements for Workflow—What Should We Expect from the Vendors?
4	Apr.	Desktop Multimedia—Moving beyond the Chicken and the Egg
5	May	Borland International—A Database-Centric, Object-Oriented Approach to Desktop Integration
6	June	Apple's Macintosh—Can It Become "the Cadillac of Collaboration"?
7	July	Business Intelligence—A Framework for Data Analysis Applications
8	Aug.	The Quest for Common Mail APIs—Clearing up the Confusion

UNIX in the Office

1991—Volume 6

#	Date	Title
11	Nov.	The SQL Standard—Can It Take Us Where We Want to Go?
12	Dec.	Positioning Desktop Options—How Does Unix Fit in the Client Environment?

1992—Volume 7

#	Date	Title
1	Jan.	Downsizing with Open Systems—Can Unix Symmetric Multiprocessing Systems Meet MIS Requirements?
2	Feb.	System V.4 and OSF/1—Matching up in the Marketplace
3	Mar.	Europe's Harness Project—Integrated Technology for an Open, Object-Oriented, Distributed Applications Platform
4	Apr.	The X Window System—Where is Its Future?
5	May	HP's Master Plan—Winning Is Everything in Palo Alto
6	June	Digital's DECworld Gems—Alpha and Accessworks Shine

Open Information Systems

7	July	Integrating Applications in the Real World—Evolution, Not Revolution
8	Aug.	Windows NT 3.1—Microsoft's Bid for Desktop Dominance

Network Monitor

1991—Volume 6

#	Date	Titl
11	Nov.	ANSA—A Model for Distributed Computing
12	Dec.	PowerBuilder—Graphical, Client/Server Database Applications Tool

1992—Volume 7

#	Date	Titl
1	Jan.	Securing the Distributed Environment—A Question of Trust
2	Feb.	HyperDesk DOMS—A Dynamic Distributed Object Management and Applications Development System
3	Mar.	Smart Hubs—Establishing a Manageable Internet Foundation for Distributed Computing
4	Apr.	Message Express—A Message Platform for Cooperative Processing
5	May	Novell NetWare 4.0—Building toward an Enterprise Distributed Object Computing Environment
6	June	Distributed Printing—Major New Approaches Begin to Relieve One of Distributed Computing's Most Frustrating Problems

Distributed Computing Monitor

7	July	The New E-Mail APIs—Finally, Real Progress toward Mail-Enabled Applications
8	Aug.	Distributed Object Computing—The Merger of Distributed Computing and Object-Orientation into a New Architecture

Printed on recycled paper.

OPEN INTERFACE ELEMENTS™
Data Sheet

Power, Portability, Extensibility for GUI Applications

Aug.6, 93

Introduction

OPEN INTERFACE ELEMENTS is the most powerful and flexible graphical user interface (GUI) application development tool available today. With OPEN INTERFACE, developers can design any type of graphical interface for their application, for virtually any environment, while they reduce development effort by as much as 50 percent. The most advanced version of OPEN INTERFACE yet, OPEN INTERFACE 3.0 gives developers even more flexibility, power, and extensibility with new widgets, a new script language, and new C++ support. Open Interface is your key to developer productivity.

Develop Once, Deploy Anywhere

Macintosh. Windows and Windows NT. OS/2 Presentation Manager. OSF Motif and OPEN LOOK. For end-users, these windowing systems spell a whole new world of possibilities, providing the ability to work within easy-to-use graphical interfaces to access, manipulate and analyze information. But for developers, the proliferation of windowing environments across different hardware platforms has created a quagmire of conflicting standards and methodologies. To date, cross-platform application development has required separate teams to write code for each target environment.

No more. With OPEN INTERFACE, a single development team can create any interface for their application, then port that application intact to any native windowing environment across more than 40 platforms. Unlike other tools on the market, OPEN INTERFACE delivers full portability without compromise-any interface, anywhere.

A Better Toolkit

For most developers, graphical user interface creation takes up as much as 80 percent of the total development effort. The reason: Native toolkit libraries are notoriously limited in their functionality. OPEN INTERFACE bridges the gap between what these widget sets provide and what you really need, with a powerful and full-featured toolset that gives you tables, tree browsers, business graphics, hypertext, multi-font text edits, color icons and color images, and much more. In fact, OPEN INTERFACE delivers all the widgets available within the various native widget sets, across all platforms.

Rich Application Development Environment

You may be accustomed to a traditional trade-off with your application development tools: they're either powerful or easy to use. OPEN INTERFACE is different. It provides all the power and functionality you need to design complex and sophisticated GUI applications, with two easy and extensible tools: Open Editor and a flexible script language. Open Editor, an interactive design tool, lets you manage the appearance of your interface with easy point-and-click, while the script lets you specify the behavior of those elements through a simple and powerful advanced language. Both are fully extensible, to give you even more control over your application.

Widest Selection of Power Widgets

OPEN INTERFACE provides a full range of interface objects, called widgets, from the simplest buttons and windows to the most complex tables, business graphics and browsers. Unlike other GUI application development tools on the market, OPEN INTERFACE delivers a superset of all widgets available on all supported platforms. With the OPEN INTERFACE toolkit, the power is in your hands.

But the functionality in our toolkit is just the beginning. You can create your own custom widgets as "sub-classes" of the widgets provided with OPEN INTERFACE. Your new widgets "inherit" all the power Neuron Data has built into its toolkit, and they're completely portable across all of your target environments.

Extensible Script

An optimized script language for linking elements to application functions, the script lets you prototype and run application functionality on the fly, directly from the development environment using a visual editor. Without compiling. What's more, you can extend the script language by integrating your own verbs and events. So you can have any widget call any function, anytime. You can even incorporate third-party libraries or legacy applications into our development environment for the widest range of flexibility and power.

C AND C++ Support

OPEN INTERFACE's C and C++ interface lets developers working in C and C++ take full advantage of that language's features and syntax. With the C++ interface, the widget data structures and routines in OPEN INTERFACE libraries are organized into C++ classes with complete member functions. So you can use familiar C++ programming techniques, such as multiple inheritance and function and operator overloading.

Character Mode Support

Character-based terminals still offer many end-user environments significant advantages, especially for applications that require dial-up access to network services. Developers who want to target both character-based and graphical end-user environments now have a simple and elegant solution with OPEN INTERFACE, which fully supports a wide variety of character-based terminals and DOS character mode. So you can preserve your investment in existing character terminals while you leverage the full functionality of OPEN INTERFACE for graphical environments.

Transparent Database Access

Access any database, from any client platform. That's the goal of Neuron Data Access Element option. With it, you can design decision-support and on-line transaction processing (OLTP) applications that access flat-file, hierarchical, relational and object-oriented databases, and other data sources through a single open module. So your graphical applications can mix and match data from Oracle or Sybase with DB2 and flat-file data. And you can build that capability seamlessly and easily, by pointing and clicking on the database you choose.

Portable Application System Services

Beyond OPEN INTERFACE's powerful GUI libraries and APIs lie a number of powerful system services that make OPEN INTERFACE a complete environment for GUI application development. Services like portable data types and memory and print management. Like error-handling, file I/O, file naming, timers, and single- and double-byte string manipulation.

Customers

Hundreds of companies have utilized Neuron Data OPEN INTERFACE for their mission-critical GUI application development. Commercial software houses, corporate developers and systems integrators in industries such as finance, aerospace, publishing, manufacturing, medicine, defense and telecommunications rely on Open Interface every day.

A Part of the Elements Architecture

The Neuron Data OPEN INTERFACE ELEMENTS product incorporates key components of the company's ELEMENTS Architecture, a comprehensive and integrated set of application development tools and integration layers that provide a flexible environment for accommodating multiple development styles, hardware platforms, windowing systems, data sources and third-party tools and libraries.

Neuron Data's ELEMENTS Architecture consists of two primary types of components:
- ELEMENTS, which are tool modules optimized for specific development needs; and,
- integration layers, including visual editors, script, C and C++ interfaces. Underlying all of these services are Neuron Data's powerful application services to enable easier, faster porting and management of ELEMENTS-built applications.

Neuron Data's ELEMENTS Architecture was built in C, with a design employing strict object-oriented programming methodologies and constructs. So, Neuron Data ELEMENTS retain the full portability and openness of the C language while still providing a robust object-oriented architecture and environment for maximum flexibility and performance in demanding real-world applications.

It's this design philosophy that enables the ELEMENTS Architecture to incorporate and support hundreds of diverse APIs, for maximum compatibility with third-party APIs, tools and libraries. And it's this level of openness, coupled with industrial-strength performance, that makes Neuron Data ELEMENTS Architecture product the leading GUI design and knowledge-base development tools among commercial software suppliers and systems integrators worldwide.

OPEN INTERFACE ELEMENTS Features

•Widgets:

- Push button
- Radio button
- Check button
- List box
- Table
- Tree Browser
- Menu Bar
- Pop-up Menu
- Edit Field with Format Checking
- Multi-Font/Multi-Line Text Edit
- Choice Box
- Scrollable Panel
- Pushpin
- Slider & Gauge
- Business Graphics **-New!**
  - Pie Charts
  - Line Charts
  - Bar Graphs
- Palette **-New!**
- Color Icon **-New!**
- Color Image **-New!**
- Supports GIF, TIFF, BMP, PICT
- Hypertext **-New!**

• Common Dialogs:

- Alert Windows
- Help Windows **-New!**
- Printer Set-up
- Tree Browser Overview
- Scrollable Area Overview
- Color Picker **-New!**
- Font Picker **-New!**
- File Picker **-New!**

*•Script Element — **All New!***

- 3GL performance
- Message-based
- Events and verbs
- Extensible and language-independent

*•Data Source Connections (optional) - **All New!***

- Access to cross-platform databases
  - Relational
  - Hierarchical
  - Flat-file

- Data object mapping
- Common SQL access
- Supports database joins

• Platform Support:

- UNIX
  - Data General Aviion
  - Digital RISC Ultrix
  - Digital Alpha AXP OSF/1
  - HP 9000 Series 300/400/700/800
  - IBM RS/6000 AIX
  - ISC UNIX
  - MIPS ABI
  - SCO ODT
  - Silicon Graphics
  - Solaris for SPARC and Intel
  - SONY NEWS (RISC and CISC)
  - Unixware
- Microsoft Windows NT
- Microsoft Pen Windows
- Digital Alpha NT
- DOS-Microsoft Windows
- DOS-Character Mode
- OS/2-Presentation Manager
- Macintosh
- Digital Open VMS Alpha AXP and VAX/VMS

Develop it Right with OPEN INTERFACE

There's a reason why OPEN INTERFACE is the industry's most popular graphical user interface application development tool: It's the best. With the most powerful capabilities and widest range of platform and windowing environment support, OPEN INTERFACE lets you realize the full potential of graphical user interfaces. So if you're developing graphical, client/server applications, you owe it to yourself to evaluate the power and flexibility of OPEN INTERFACE. Call us today at 1-800-876-4900, and develop it right.

About Neuron Data

Headquartered in Palo Alto, Calif., Neuron Data designs, manufactures and markets application development tools optimized for mission-critical corporate environments and commercial software.

###

OPEN INTERFACE

CASE STUDY

AT ESRI, NEURON DATA OPEN INTERFACE™ ENABLES QUICK SOFTWARE DELIVERY FOR EXPANSION INTO NEW MARKETS

"We look for partners to join forces with us and help us bring complete software solutions to the market. For ArcView, Open Interface turned out to be the ideal partner."

In the fast-paced software industry, time-to-market is one of the most critical ingredients for success. Leadership companies usually have two strong capabilities. First is the ability to identify new market opportunities very early. Then, the company must respond by quickly delivering quality products tailored to customer demand.

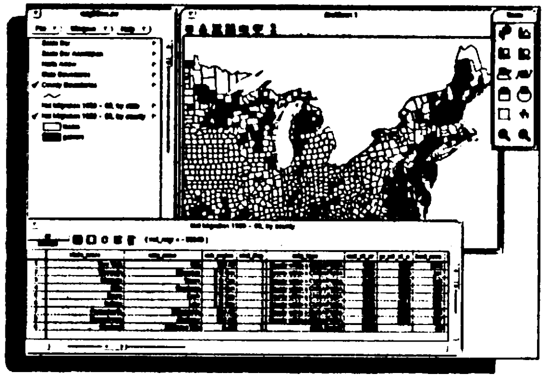
As the worldwide market leader in geographic information systems (GIS) software and a $100 million company, Environmental Systems Research Institute, Inc. (ESRI®) is a living example. After identifying an important new market opportunity, ESRI was able to deliver its new GIS software product, ArcView™, in record time — thanks to Neuron Data Open Interface.

"In the exploding commercial GIS market, competitive advantage is critical. We selected Open Interface to build and then port ArcView to more than a dozen Macintosh, PC, and UNIX platforms. As a result, we brought ArcView to market much earlier than we would have without Open Interface," said ESRI President Jack Dangermond.

RISING TO THE CHALLENGE

As a tool for geographical analysis of data through maps, GIS has been most widely used by local and federal government agencies and the military. With more than 600 employees in the U.S. and distributors in over 60 countries worldwide, ESRI is recognized as the industry leader in these traditional markets with ARC/INFO®, a high-end GIS system that runs on UNIX and PC platforms.

"In recent years, the commercial market has discovered that GIS has something to offer. For instance, insurance companies can perform risk analysis by plotting properties geographically by neighborhood. Or transportation and distribution companies can plan routes," said Clem Henriksen, manager of technology relationships at ESRI. "We saw a huge opportunity for commercial applications of GIS, and we wanted to be one of the first to deliver a desktop GIS product to those markets."



In addressing the commercial marketplace, ESRI encountered several challenges. By its very nature, the product would have to be graphically rich and easy to use. It would have to support all of the major desktop platforms, including the Macintosh and Windows on the PC. In addition, ESRI needed to be able to roll the product out as quickly as possible in order to acquire early market share.

"We've had a great deal of experience building GUIs for different UNIX environments. So we knew it would be virtually impossible to build screens for Mac and Windows from scratch and still meet time-to-market goals," said Henriksen. "So we looked for a tool that would make the job easier."

ESRI conducted an extensive evaluation of GUI development tools, examining native toolkits as well as other portable, cross-platform tools. "We chose Open Interface because it most fully met our requirements for portability, ease of use, and high-level graphics capabilities," said Henriksen. "Among the cross-platform tools, Open Interface is by far the most complete solution — the only one that has a consistent application programming interface (API) across all platforms. And the widget set is on par with those offered by native toolkits of individual windowing systems."

ANNOUNCING: ARCVIEW

The new software product, ArcView, runs on all standard UNIX windowing systems including Open Look and Motif, as well as Windows PCs. ESRI is using Open Interface to port ArcView to the Macintosh in its next release.

ArcView provides powerful tools that let users explore their data together with existing geographic databases and visualize it in many different ways. Users can perform sophisticated queries, explore extensive areas of geography, merge PC and workstation geographic databases, and produce high-quality, full-color maps at the touch of a button.

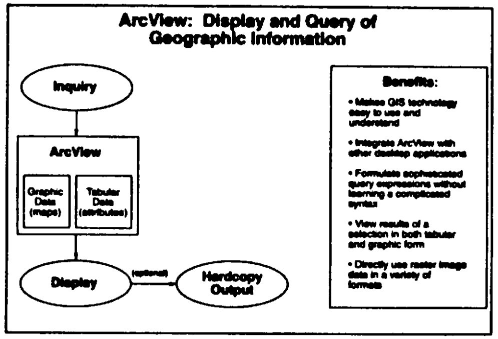
ArcView gives new meaning to the term "GUI," with an interface that is completely visual. Users can create graphical displays of geographic data and query geographic information through windows, pull-down menus, and other graphical features. ArcView makes full use of Open Interface's widget set, including slider bars, buttons, scrolling text, and text windows; in addition, ESRI used Open Interface to create custom geographical widgets for the software which are also completely portable.

ArcView has been highly successful, with over 5,000 copies sold in just over a year. "Our users have responded very well to ArcView's easy access to geographic information, facilitated by an intuitive GUI," said Henriksen. ArcView's markets include education, real estate and retail market research, and users in federal, state, and local government.

THE IDEAL PARTNERSHIP

"Using Open Interface, we were able to deliver ArcView very quickly on all platforms," said Henriksen. "It would probably have taken at least three years, had we ported it from scratch to each platform. And we would have had to hire someone with in-depth expertise for each individual windowing system that we wanted to support. But Open Interface's APIs allowed us to leverage our existing skill set instead of having to develop our own or hire that knowledge.

"Each graphical environment is extremely complex and requires a high level of expertise in terms of people and tools. But we're a GIS company — we're not experts in operating systems, networking protocols, windowing systems," he added. "Instead, we look for partners to join forces with us and help us bring complete software solutions to the market. For ArcView, Open Interface turned out to be the ideal partner."



ArcView: Display and Query of Geographic Information
Inquiry
ArcView
Graphic Data (maps)
Tabular Data (attributes)
Display
(optional)
Hardcopy Output
Benefits:
• Makes GIS technology easy to use and understand
• Integrate ArcView with other desktop applications
• Formulate sophisticated query expressions without learning a complicated syntax
• View results of a selection in both tabular and graphic form
• Directly use raster image data in a variety of formats

ArcView users produce high-quality, full-color maps at the touch of a button.

OPEN INTERFACE AT ESRI

Application:

Fast delivery of new software products to commercial markets, on a wide range of desktop platforms.

Environments:

Open Look, Motif, Windows, and (with next release) Macintosh.

NEURON DATA
156 University Avenue
Palo Alto, CA 94301
415.321.4488
FAX 415.321.3648

OPEN INTERFACE

CASE STUDY

ND + ALDUS

AT ALDUS CORPORATION, NEURON DATA OPEN INTERFACE™ ENABLES RAPID DELIVERY OF MISSION-CRITICAL APPLICATIONS

"OPEN INTERFACE IS A VERY IMPORTANT COMPONENT IN OUR LONG-TERM STRATEGY TO IMPLEMENT DISTRIBUTED, CLIENT/SERVER-BASED SOLUTIONS. WE BELIEVE THIS IS KEY TO STAYING COMPETITIVE THROUGH THE '90s."

What does it take to be a world-class company? A winning product, for one — and Aldus Corporation's PageMaker® is a household word for desktop publishing. But speedy and cost-effective deployment of internal, mission-critical applications is just as important to gaining competitive advantage. At Aldus, Neuron Data Open Interface is key to a powerful new software engineering architecture that allows quick development and deployment of strategic internal applications. Open Interface has been instrumental in the development of two such applications — a bug tracking system for product quality control, and a technical support database.

"Open Interface is very valuable to us — not just as a tool for building graphical user interfaces, but as the basis for an ongoing architecture for developing future applications consistently and quickly," said Parker Whittle, MIS systems engineer and chief architect of Aldus' client/server systems and architectures. "Open Interface is an important component in our long-term strategy to implement distributed, client/server-based computing solutions. We believe this strategy will be key to helping Aldus stay competitive through the '90s."

A PERFECT FIT

Based in Seattle, WA, Aldus Corporation employs a 25-person MIS department to develop and maintain computer systems for the company's worldwide operations. Over the past year, the department has begun to implement strategic client/server-based applications. "We needed centralized control of data management and integrity. But at the same time, we wanted our end-users to have maximum flexibility to tailor front-end applications to their needs, on the client platform of their choice," said Whittle.

Before Open Interface, Aldus' MIS engineers were developing client/server applications on the Macintosh, using hypermedia products such as SuperCard to build and integrate user interfaces. "SuperCard is not available for DOS, which meant we could not deploy new client/server applications to DOS-based PC users — about 30% of our user population," said Whittle. "We would have had to completely re-engineer the Mac user interfaces in Microsoft Windows, something we had neither time nor manpower to do."

A more efficient development environment became crucial as MIS began work on the new bug tracking system, Ant Farm. "Ant Farm was the first client/server application that specifically required us to develop simultaneously for both the Macintosh and PC Windows platforms," said Whittle. Aldus MIS searched for a tool that would facilitate multi-platform development without adding significantly to development time and costs. They explored a number of options, including object-oriented languages, fourth-generation languages (4GLs), and tools based on a portable C API, such as Open Interface.

"We ruled out 4GLs because of their slowness and limited architecture, and the libraries for C++ did not seem robust enough. Open Interface's widgets come equipped with a high degree of functionality, and it's written in a portable language. It was a perfect fit for our requirements," Whittle commented.

FORGING A CONSISTENT SOFTWARE ARCHITECTURE

Ant Farm, also known as the Symptom Tracking Database, is a client/server application based on ORACLE with Macintosh and DOS/Windows clients. Ant Farm tracks symptoms of possible "bugs" that are reported by Aldus quality control technicians as they test upcoming releases of software products, and routes the symptoms to engineers for further analysis. Once the engineers identify the problems, Ant Farm provides reports to product managers on the symptoms and their ongoing resolution.

Aldus relied on powerful Open Interface widgets such as tables and browsers to accelerate Ant Farm's development. Furthermore, the application framework provided by Open Interface simplified the integration of the user interface and the ORACLE database networking libraries. "Ant Farm was the first product we had developed using C, and Open Interface was critical to our timely delivery of the application," Whittle said.

Since all of the Open Interface code was completely portable, Whittle was able to simultaneously deploy Ant Farm to end-users on either PCs or Macs. "Developing for each platform from scratch would have increased our development time by at least 60%. But with Open Interface, we were able to deliver the application in essentially the same timeframe as if we were just deploying on one platform," he commented.

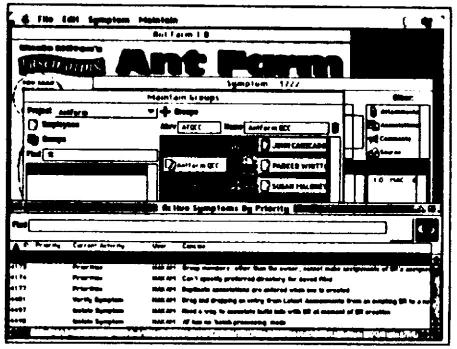
Currently, Aldus is at work on its next client/server application — a technical support database called Navigator. Navigator will provide a database of technical articles to support representatives to aid them in promptly resolving customer queries about Aldus products. Open Interface will access the SYBASE database using SYBASE's DBLib. The development team is also taking advantage of Open Interface's ability to develop portable custom widgets, which provides added flexibility without sacrificing portability.

Ant Farm and Navigator have brought Aldus successfully into the realm of client/server applications. By developing the two applications with Open Interface, Aldus MIS engineers have reached another important milestone — the adoption of a consistent software development environment that will allow future applications to be developed quickly and modified easily. Whittle anticipates that this software environment will eventually go far beyond the underlying architectures of individual applications to include widely-distributed modules of code that will form the basis for all of Aldus' corporate systems.

With Navigator, Whittle and his team are fine-tuning the software architecture and expect to easily meet the department's stated 90-day goal for turnaround time on software projects. "With a consistent architecture designed explicitly for our purposes, we'd be able to develop new projects for multiple platforms without significant increases in time and costs," he said. "We see great possibilities not only for new client/server applications, but for converting existing corporate systems to client/server, such as MRP, financials, and customer service and sales."

UNMATCHED GUI FUNCTIONALITY

Open Interface has gone far beyond Whittle's initial expectations. "Open Interface's libraries provide a high level of functionality that is unmatched by other GUI tools, and its widgets are robust and packed with features. But we also have the option of putting in our own behavior, at as low a level as we choose," he said.



Mac end-user screen for Ant Farm, Aldus' new client/server bug tracking system.

"Open Interface has enabled Aldus to develop client/server applications that can be easily ported to both Macintosh and MS Windows clients almost simultaneously, which is key to our development strategy," he said. "Open Interface has become an integral part of the Aldus information systems architecture for development of internal IS systems."

OPEN INTERFACE AT ALDUS

Application:

Rapid development and delivery of strategic internal applications.

Environments:

Development on Macintosh; simultaneous delivery on Macintoshes and DOS PCs.

ND
NEURON DATA
156 University Avenue
Palo Alto, CA 94301
TEL 415 321 4488
FAX 415 321 3648

OPEN INTERFACE

CASE STUDY

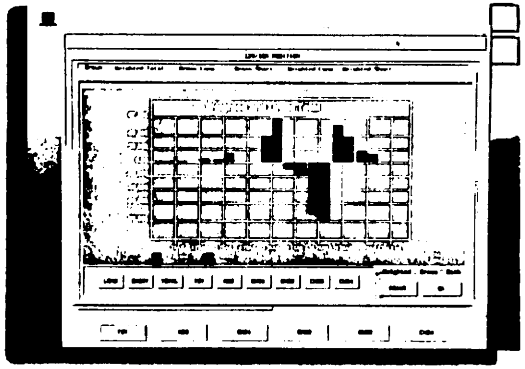
ND + SHEARSON LEHMAN BROTHERS

AT LEHMAN BROTHERS, NEURON DATA OPEN INTERFACE™ IS KEY TO FAST DEPLOYMENT OF TRADING SYSTEM

"THIS SYSTEM HAS MADE A HUGE DIFFERENCE IN THE EFFECTIVE MANAGEMENT OF OUR DIVISION. OPEN INTERFACE MADE IT POSSIBLE FOR US TO REALIZE ITS BENEFITS MUCH SOONER."

Advanced computing systems are a cornerstone of investment banking. The speed at which these systems can be developed and deployed to new platforms can be a make-or-break issue for an investment company like Lehman Brothers.

Headquartered in New York, NY, Lehman Brothers is the institutional division of Shearson Lehman Brothers, Inc., a wholly-owned subsidiary of American Express. In Lehman Brothers' Fixed Income Division, traders rely on a graphical fixed income trading system to help them track and manage more than $30 billion a day in transactions. Using conventional development tools, this system would have taken many months—perhaps years—to implement. Yet Lehman Brothers IS engineers were able to deploy the PC-based trading system in just over two months, and later port it to the UNIX environment in less than a month. How? with Neuron Data Open Interface.



"I've been involved in similar development efforts that took six to nine months for PC development and up to a year for UNIX applications," said Douglas Aronson, Lehman Brothers' system vice president. "Open Interface provided a very orderly and fast transition from one environment to the other, and saved us valuable time."

BEYOND SPREADSHEETS

Technology has played a major role in shaping the jobs of Lehman Brothers' mortgage traders, who were tracking their positions on index cards as recently as three years ago. Gradually, Lotus spreadsheets replaced the index cards; then traders switched to Microsoft Excel in the PC Windows environment.

"As the company and the markets grew, the traders' positions were getting too large to track in Excel," said Aronson. "And heads of desks (senior traders) had no way of tracking what the traders were doing at a given moment — they had to wait for reports at the end of the day or week. And, there was no way for management or the back office to reconcile what the traders were doing in real time, so they had to rely primarily on the traders' own abilities to manage their accounts. Occasional mistakes and deviations from our management strategy were keeping us from maximizing our profits."

In the meantime, the company planned to migrate all of its desktop computing applications to Sun Sparcstations under UNIX. "We wanted to build a state-of-the art trading system that would address all of the above issues," Aronson said. "Because seconds count in this business, the system would have to be extremely easy to use — we didn't want our traders to have to decipher a character-based interface. So we searched for a development tool that would allow us to build highest-quality graphical screens that we could initially deploy in the old PC/Windows environment, and then easily migrate to UNIX when the time came."

OPEN INTERFACE: THE PORTABLE SOLUTION

Aronson and his staff evaluated numerous screen builders and GUI development tools, but none met the application's portability requirements as well as Open Interface. "Open Interface offers great possibilities for modular applications that can be built and re-deployed on different platforms with minimal effort," he said.

Using Open Interface, Aronson and his staff delivered the first phase of the system in Windows in less than two months — building screens that had the look and feel of the Motif graphical environment. Four months later, Lehman Brothers put Sun Sparcstations on every desktop, and the trading system was easily ported to the new environment in a little under a month. "Because Open Interface let us build Motif-like screens initially, the traders were already familiar with Motif when the migration occurred. So they only changed computers — not the way they worked."

Without Open Interface, said Aronson, the PC version of the trading system might never have been developed; the project would have been delayed until the switch to UNIX. "This is a mission critical application; on any given day, each of our 35 traders has about $2 billion being traded through the system. Every day we delayed would have meant a significant loss of revenue through inadequate tracking methods."

VISUAL TRADING

In the Lehman Brothers Fixed Income division, traders are grouped on desks according to the products they are assigned to trade, such as mortgage backed securities or asset backed securities. The head of each desk is responsible for all the traders on the desk.

Through Open Imterface screens, traders perform such transactions as buying and selling, and track their positions in the market against data coming in through real-time feeds. When a transaction takes place, the system performs basic accounting functions and captures data for back office use. At the same time, the system provides important assistance in hedging. Whenever traders purchase a bond, they attempt to hedge or protect the investment by selling one or more securities of equal value. The trading system calculates what they need to buy of a particular bond to hedge that particular position.

Most important, the trading system allows supervisors to monitor trading activity in real time, from their desktops. "At any time, a supervisor can look in on a particular trader's activity and determine whether it's within the bounds of good trading practices," said Aronson. "Or, supervisors can look at traders' positions cumulatively to see how the group is doing, or how one trader's performance compares with another's. It's a very valuable management capability that wasn't possible with the old spreadsheet method."

THE OPEN INTERFACE DIFFERENCE

Since its deployment, the trading system has been well received by traders and managers alike, and has drawn the attention of other Lehman Brothers divisions.

"Traders can easily view large amounts of data in windows or groups of windows, and can flip through quickly to find the information they need. The system gives them a new perspective on their work," Aronson commented.

"This system has made a huge difference in the effective management of our division and the profitability of our traders," he added. "And Open Interface made it possible for us to realize its benefits much sooner."

OPEN INTERFACE AT LEHMAN BROTHERS

Application:

Mission-critical trading system for managing $30 billion a day in transactions.

Environment:

Initial deployment on Windows; later migration to UNIX/Motif.

ND
NEURON DATA
156 University Avenue
Palo Alto, CA 94301
415.321.4488
FAX 415.321.3648

OPEN INTERFACE

CASE STUDY

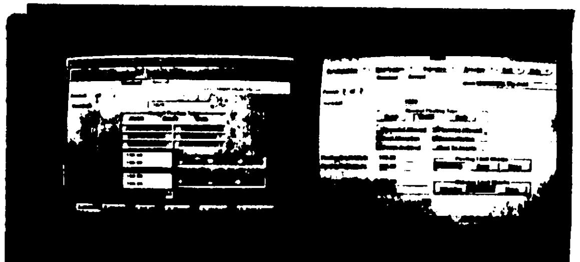
ND +

NEURON DATA OPEN INTERFACE™ PUTS THE UNIVERSAL INTERFACE ON UNIVERSAL OLAS

> "THE KEY TO OUR APPROACH IS GENERATING CODE FROM A SINGLE SOURCE, SO YOU CAN DEVELOP THE BUSINESS FUNCTIONALITY KNOWING THAT IT WILL RUN IN ALL TARGET ENVIRONMENTS."

For ten years Quality Software Products have dominated the market in online accounting systems for IBM mainframes, with major users such as British Airways, B&Q, and the banking and insurance community, including Pearl, Allied Dunbar, Nat West and the Bank of England itself. But with the rise of open systems, PC networking, client/server technology and other alternatives to centralized processing, the company realized it needed a new product strategy.

The result, Universal OLAS Financial and Procurement Management Systems will run on IBM mainframes, OS/2 personal computers and UNIX platforms, in conjunction with IBM's DB2, Oracle, INGRES, Sybase or almost any other relational database management system.



Open Interface allows you to create your user interface once, and have it work across all standard windowing environments.

The beauty of Universal OLAS is that, whatever processor the user chooses, it uses exactly the same Cobol code.

And yet the users can work with whatever screen environment they prefer. This universality of presentation is partly due to Quality Software's own user screen design tool, SOFTSCREEN, and partly to Neuron Data Open Interface.

"You can run the presentation services on an OS/2 PC, and offload the remaining functionality to a UNIX server or a mainframe," says Mark Eastman, Quality Software's Research Manager. "Or you could have them all running on the same platform. It doesn't matter. The key to our approach is that you generate the code from a single source, so you can develop the business functionality knowing that it will run in all target environments however it is split up."

One major problem for Quality Software was that the screen presentation techniques which the various environments use are all different. "We couldn't rely upon the environment supplied with the operating system to provide the screen presentation techniques. So we developed our own, SOFTSCREEN, which handles all the presentation requirements for character-based and GUI modes of presentation."

"But the world is moving to a multiplicity of GUIs. We wanted users to be able to work with the GUI of their choice. However, we wanted to keep to the principle behind Universal OLAS, which is that the Cobol applications never change, and SOFTSCREEN seamlessly provides the required front-end."

GIVING USERS THE GUI OF THEIR CHOICE

"We needed to be able to address all GUIs. For example, we had to be sure that the interface we offered to the managerial user was the same as for the other PC applications he was running, so that he wouldn't suffer a culture shock when he accessed OLAS."

"We also had to consider that people move around from office to office. Someone who usually works with a PC might find himself at a site where they have only UNIX systems. We wanted them to be able to work with Universal OLAS on whatever platform, without having to be trained to use several different interfaces."

But the timescales and resources involved in redeveloping SOFTSCREEN to address every GUI environment would have been horrendous. Although the environments appear basically the same to users, the underlying technology is significantly different. There is also a radical change in the ways in which character-based interfaces and GUIs interact with the underlying application. Generating GUIs is relatively easy; generating GUIs which are uncluttered, efficient and comfortable to work with is a lot harder. If the goal of a single, static set of business applications was to be reached, all these differences in approach would have to be looked after by SOFTSCREEN.

"We looked around to see if there were any tools available that would allow us to interact with all these interface technologies. There was a lot of talk and vapourware, but we began to feel it would be quicker to develop the technology ourselves than to wait for a third party to deliver it."

Until, that is, Quality Software saw a demonstration of Open Interface. "We attended a seminar, then took Open Interface for evaluation. We already know how we wanted to go about generating the different GUIs, and when we looked at the technical details of Open Interface, we discovered that they were already working in exactly this way. There was a great synergy in our technologies and our approaches to problem solving, which gave us tremendous confidence in Open Interface.

"We set to work to re-implement SOFTSCREEN to use Open Interface. However, we soon saw that Open Interface would not shorten the time it would take us to get our GUI product to market. The Open Interface toolkit is very similar to any of the GUI toolkits, and so it took us as long to link Open Interface to the library of software calls in SOFTSCREEN as it would have taken in Presentation Manager or Windows."

"But the benefit of Open Interface was that we only had to write the application once for it to be available under every Graphical User Interface. So we had reduced the timescales by a factor equivalent to all the target GUI environments."

Users will now be able to work with any interface they choose. If PC users would rather have an IBM mainframe-style, character-based interface than a graphical one, it is available. If a Presentation Manager user is seconded to a UNIX site, the X-terminals can display a Presentation Manager-style interface to OLAS. There are tremendous benefits in terms of ease of use, and great savings in retraining and support.

REDUCED MAINTENANCE AND ENHANCEMENT COSTS

There is also a saving in ongoing maintenance and enhancement of Universal OLAS, since a change has to be made only once to be available on every platform, with any style of interface.

"We can also go on to define our own custom widgets in Open Interface, such as pie charts and bar graphs. We shall be defining widgets to generate specific tasks within our ledgers. For example, we will be able to create widgets which are 'intelligent' as far as the General Ledger is concerned, which 'know' how to generate a balance if given a company name."

"We only have to implement these once in the Open Interface libraries and they are available across all the GUI environments."

"Without Open Interface, we would have had to stage our development of SOFTSCREEN to incorporate the different GUIs one at a time. We would also have needed to buy in or train specialist developers for each of the GUIs."

"Instead, we can provide Universal OLAS for any GUI environment without any additional investment on our part. For example, on the day Windows NT becomes available, Open Interface will be able to support it. This means that we will have a product that can run on Windows NT from day one, with no effort on our part apart from recompiling. Neuron Data will have looked after all the porting issues for us."

Universal OLAS incorporating Neuron Data's Open Interface has already been enthusiastically previewed to Quality Software Products customers and is scheduled for public launch in early 1993.

Neuron Data Open Interface is a trademark of Neuron Data, Inc.
SOFTSCREEN is a trademark of Quality Software Products.

QUALITY SOFTWARE PRODUCTS' UNIVERSAL ONLINE ACCOUNTING SYSTEM

Application:
Open Interface puts a universal interface on Online Accounting Software

Environment:
IBM mainframes, OS/2 personal computers and UNIX platforms.

ND
NEURON DATA
156 University Avenue
Palo Alto, CA 94301
415 321 4488
FAX 415.321.3648