**Review: Classic Mac OS**

- Designed for the user, not the developer
  - First commercially successful GUI system
  - Technically few advances
  - One address space, one process, “no” OS
  - But revolutionary approach to UI consistency (HI Guidelines)
- Macintosh Toolbox
  - Pascal procedures grouped into Managers, ROM+RAM
  - Extended as technology advanced (color, multiprocessing,...), but architecture was showing its age by late 90s
- Inspiration for other GUs, esp. MS Windows

**The X Window System (“X”)**

- Asente, Reid (Stanford): \(W\) window system for \(V\) OS, (1982)
  - \(W\) moved BWS&GEL to remote machine, replaced local library calls with synch. communication
  - Simplified porting to new architectures, but slow under Unix
- MIT: \(X\) as improvement over \(W\) (1984)
  - Asynchronous calls: much-improved performance
  - Application = client, calls \(X\) Library (Xlib) which packages and sends GEL calls to the \(X\) Server and receiving events using the \(X\) Protocol.
  - Similar to Andrew, but window manager separate
  - X10 first public release, X11 cross-platform redesigned

**X: Architecture**

- \(X\) is close to our 4-layer architecture model

\[\text{Application} \rightarrow \text{UITK} \quad \text{Network} \quad \text{BWS+GEL} \]

\[\text{X Server} \quad \text{Xlib} \quad \text{WM} \quad \text{X lib} \quad \text{Widget Set} \]
X Server

- X11 ISO standard, but limited since static protocol
- X server process combines GEL and BWS
  - Responsible for one keyboard (one EL), but n physical screens (GLs)
  - One machine can run several servers
- Applications (with UITK) and WM are clients
- GEL: Direct drawing, raster model, rectangular clip.
  - X-Server layers: Device-dependent X (DDX), device-independent X (DIX)
  - BWS can optionally buffer output regions

X Protocol

- Between X server process and X clients (incl. WM)
- Asynchronous, bidirectional byte stream, order guaranteed by transport layer
  - Implemented in TCP, but also others (DECnet,...)
  - Creates about 20% time overhead with apps over network
- Four packet types
  - Request, (Client→Server)
  - Reply, Event, Error (Server→Client)
- Packets contain opcode, length, and sequence of resource IDs or numbers

Typical Xlib application (pseudocode)

```c
#include Xlib.h, Xutil.h
Display *d; int screen; GC gc; Window w; XEvent e;
main () {
    d = XOpenDisplay("171.64.77.1:0");
    screen = DefaultScreen(d);
    w = XCreateSimpleWindow(d, DefaultRootWindow(d), x,y,w,h,
        border, BlackPixel(d), WhitePixel(d)); // foreground &
        background
    XMapWindow(d, w);
    gc = XCreateGC(d, w, mask, attributes); // Graphics Context
    setup left out here
    XSelectInput(d, w, ExposureMask|ButtonPressMask);
    while (TRUE) {
        XNextEvent(d, &e);
        switch (e.type) {
            case Expose: XDrawLine (d, w, gc, x,y,w,h); break;
            case ButtonPress: exit(0);
        }
    }
}
```

X: Resources

- Logical: pixmap, window, graphic context, color map, visual (graphics capabilities), font, cursor
- Real: setup (connection), screen (several), client
- All resources identified via RIDs
- Events: as in ref. model, from user, BWS, and apps, piped into appropriate connection
- X Server is simple single-entrance server (round-robin), user-level process
Window Manager

• Ordinary client to the BWS
• Communicates with apps via hints in X Server
• Look&Feel Mechanisms are separated from Look&Feel Policy
• Late refinement (session, user, application, call)

Window Manager

• Dynamically exchangeable, even during session
  • twm, ctwm, gwm, mwm (Motif), olwm (OpenLook), rtl (Tiling), ...
  • Implement different policies for window & icon placement, appearance, all without static menu bar, mostly pop-ups, flexible listener modes
• No desktop functionality (separate app)
• Only manages windows directly on background (root) window, rest managed by applications (since they don’t own root window space)

X: UITK

• X programming support consists of 3 layers
• Xlib:
  • Lowest level, implements X protocol client, procedural (C)
  • Programming on the level of the BWS
  • Hides networking, but not X server differences (see “Visual”)
  • Packages requests, usually not waiting for reply (async.)
  • At each Xlib call, checks for events from server and creates queue on client (access with XGetNextEvent())
  • Extensions require changing Xlib & Xserver source & protocol

X: UITK

• Xlib offers functions to create, delete, and modify server resources (pixmaps, windows, graphic contexts, color maps, visuals, fonts), but app has to do resource composition
• Display (server connection) is parameter in most calls
• X Toolkit Intrinsics (Xt)
  • Functions to implement an OO widget set class (static) hierarchy
  • Programming library and runtime system handling widgets
  • Exchangeable (InterViews/C++), but standard is in C
  • Each widget defined as set of “resources” (attributes) (XtNborderColor,...)
**X: UITK**

- **X Toolkit Intrinsics**
  - Just abstract meta widget classes (Simple, Container, Shell)
  - At runtime, widgets have 4 states
    - Created (data structure exists, linked into widget tree, no window)
    - Managed (size and position have been determined—policy)
    - Realized (window has been allocated in server; happens automatically for all children of a container)
    - Mapped (rendered on screen)—may still be covered by other window!

**UITK**

- **X Toolkit Intrinsics**
  - Xt Functions (XtRealizeWidget(), ...) are generic to work with all widget classes
  - Event dispatch:
    - Defined for most events in translation tables (I→A) in Xt
    - Widgets handle events alone (no event loop in app!)
    - App logic in callback functions registered with widgets

**Widget Sets**

- Collection of user interface components
- Together with WM, define look&feel of system
- Several different ones available for X
  - Athena (original, simple widget set, ca. 20 widgets, 2-D, no strong associated style guide) — Xaw... prefix
  - Motif (Open Software Foundation, commercial, 2.5-D widget set, >40 widgets, industry standard for X, comes with style guide and UIL)—Xm... prefix
- Programming model already given in Intrinsics
  - Motif just offers convenience functions

**Athena Widget Set**

- Original, free, extensible
- Ugly, simple
- Class hierarchy:
  - **Simple** — Base class for all other Athena widgets. Does nothing, but adds new resources such as cursor and border pixmap.
- Standard widgets:
  - **Label** Draws text and/or a bitmap.
  - **Command** Momentary push-button
  - **ToggleButton** Push-button with two states.
  - **MenuButton** Push-button that brings up a menu.
  - **Grip** Small widget used to adjust borders in a Paned widget.
  - **List** Widget to allow user to select one string from a list.
  - **Scrollbar** Widget to allow user to set a value; typically to scroll another widget.
  - **Box** Composite widget which simply lays children out left-to-right.
  - **Form** Constraint widget which positions children relative to each other.
  - **Dialog** Form widget for dialog boxes.
  - **Paned** Constraint widget letting user adjust borders between child widgets.
  - **Text** Base class for all other text classes.
  - **TextSink** Base class for other text sinks.
  - **TextSrc** Base class for other text sources (subclasses for ASCII and multi-byte text)
  - **SimpleMenu** Shell which manages a simple menu.
  - **Sme** RectObj which contains a simple menu entry (blank).
  - **SmeBSB** Menu entry with a string and optional left & right bitmaps.
  - **SmeLine** Menu entry that draws a separator line.

- Special widgets:
  - **Repeater** Command that repeatedly calls its associated callback function for as long as it’s held.
  - **Panner** Widget to allow user to scroll in two dimensions.
  - **StripChart** Widget to display a scrolling graph.
  - **Porthole** Composite widget which allows a larger widget to be windowed within a smaller window. Often controlled by Panners.
  - **Viewport** Constraint widget, like a Porthole with scrollbars.
  - **Tree** Constraint widget, lays its children out in a tree.