DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF TORONTO

CSC428F/2514F

HUMAN-COMPUTER INTERACTION

Lecture 12

USER INTERFACE DEVELOPMENT TOOLS

Ronald Baecker
Professor of Computer Science,
Electrical and Computer Engineering, and Management
University of Toronto

12.1 Overview

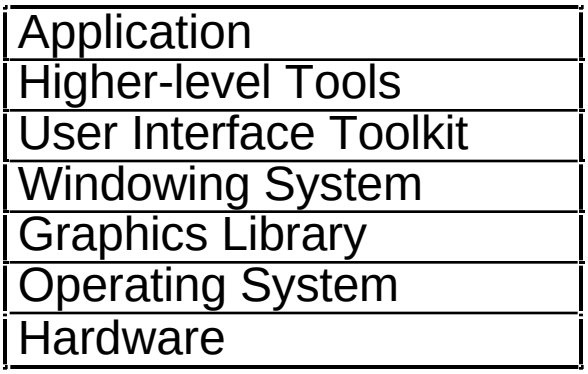
User interface software tools
- Software tools that assist in the implementation of highly interactive programs and systems

Why user interfaces are hard to build
- Need for iterative design and development
- Complexities of multiprocessing
- Need for real-time programming
- Need for robustness under great variety of input
- Difficulties in testing for “correctness”
- Desire for modularization, application-interface separation
- Complexity of tools
- Poor language support for interactive graph. interfaces

Goals for user interface software tools
- Rapid development of systems and interfaces
- Rapid modification of interfaces
- High-level primitives for design and programming
- Clarity, “readability” of interface definitions
- “Good” (easy to learn, easy to use, ...) interfaces
- Inclusion of undo, help, other user aids
- Consistency of interface style
- Participation of graphic designers, human factors specialists, and other non-programmers
- Possible weaknesses:
  - Sacrifices in terms of efficiency of code
  - Restrictions on interface style

History
- Sketchpad (1963) written in assembler
- Graphics packages embedded in high-level languages
- Interface toolkits, windowing systems
- Prototyping tools, user interface management systems

12.2 Architecture of user interface software



Hardware, e.g., Intel 386, Motorola 68030

Operating system, e.g., MacOS+Finder, Windows 95, UNIX+shell

Graphics library
- Provides the upper layers with an *imaging model* and a set of operations that can be used to paint the screen, and implements them in terms of display hardware primitives, e.g., Postscript as an imaging model, often included as part of the OS

Windowing system (see also 12.4)
- Window system (window kernel, base layer)
  - Provides upper layers with abstractions (e.g., windows) of physical screen resources and assigns real physical resources to these abstract objects, also
    controls use of graphics library and event processing to support window manager
- Window manager (upper layer)
  - Manages the windows and the mechanisms for controlling them with mouse and keyboard

User interface toolkit
- Implements user interface building blocks (*widgets*) such as menus, buttons, sliders, & dialog boxes

High-level tools (often separately packaged)
- Language-based tools
- Application frameworks
- Automatic model-based generation of interfaces
- Interactive graphical specification (of prototypes, systems)

Approaches we shall discuss:
- Windowing system plus toolkit for a personal computer — theMacintosh Toolbox (12.3)
- Windowing system architecture (12.4)
- Windowing system for networked machines — X (12.5)
- Toolkits and graphical user interfaces (GUI's) (12.6)
- Interactive graphical specification of prototypes — Director, HyperCard (12.7)
- Other higher-level tools —
  - tcl/tk (language-based, Lecture 13)
  - Garnet (application framework, Lecture 13)
  - ITS (model-based, Lecture 14)
- Interface style and style guidelines (Lecture 14)

See Myers, Brad A., State of the Art in User Interface Software Tools, 1993, in [BGBG].

12.3 The Macintosh Toolbox for interface construction

The Macintosh User Interface Toolbox “provides a simple means of constructing application programs that conform to the standard Macintosh user interface.” (*Inside Macintosh*, Apple Computer, Addison-Wesley, 1985)

The Toolbox and the Macintosh operating system (Fig. 12.1)

Figure 12.1 Macintosh programming environment (Inside Mac, p. 1-10)

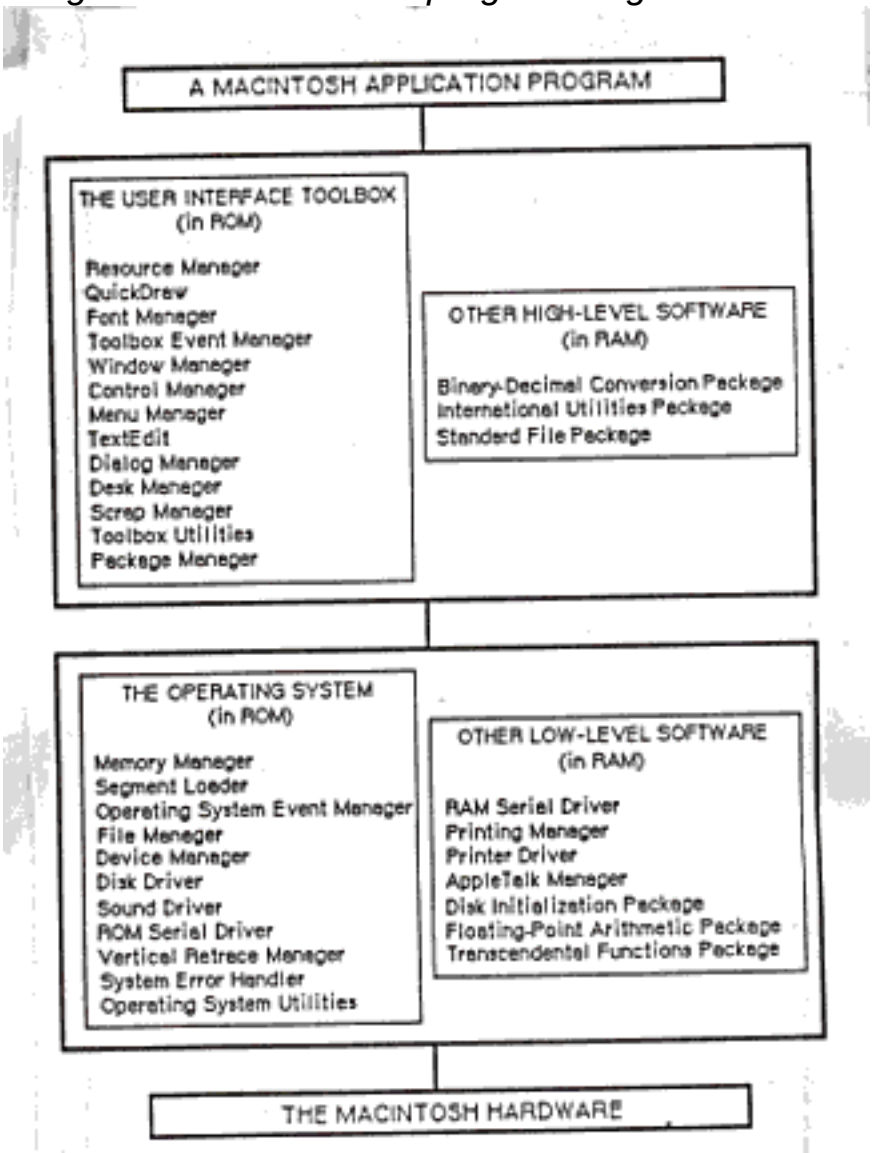
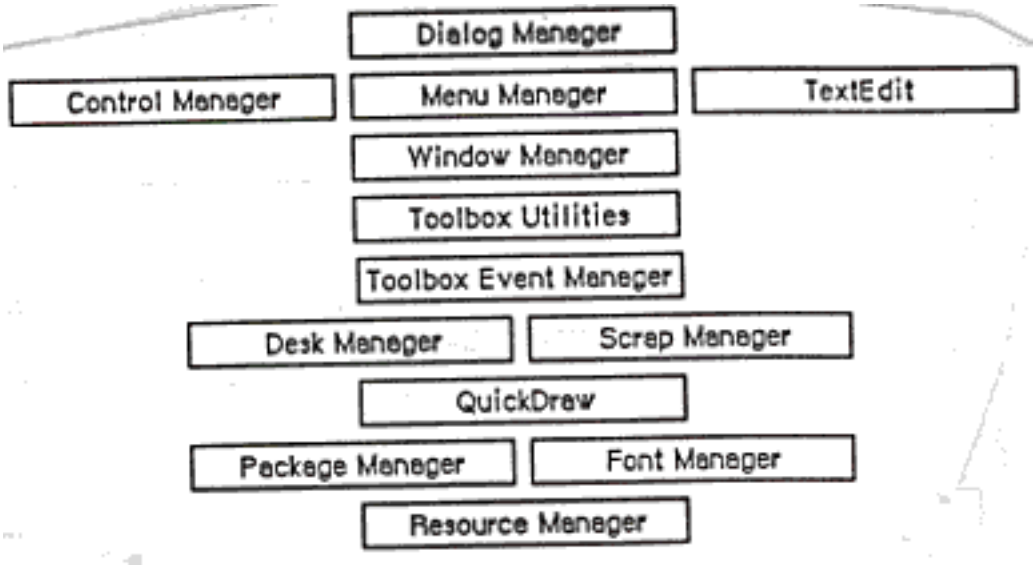


Figure 12.2 Hierarchic organization of parts of the Toolbox (Inside Mac, p. 1-11)



Font manager
: Implements typography: variety of fonts, styles, & sizes

QuickDraw
: "Graphics package" for drawing text, straight lines, rectangles, ovals, rounded rectangles, wedges, polygons, and regions, while also dealing with windows and viewports, clipping, and off-screen drawing

Toolbox event manager
: Monitors input events from keyboard and mouse such as key presses and mouse movements, as well as window events such as the exposure of a window that was overlapped, and decides what to do

Desk manager
: Manages the use of desk accessories

Window manager
: Creates, activates, moves, resizes, and closes windows, and manages the use of multiple overlapping windows

Control manager
: Creates, manipulates, and monitors controls such as buttons, check boxes, and scroll bars

Menu manager
: Creates, manipulates, and monitors menus

TextEdit
: Accepts text typed by user and provides standard editing capabilities, word wraparound, and justification

Dialog manager
: Creates, manipulates, and monitors dialog boxes and alerts

The Macintosh Toolbox in terms of the architecture

Application	
User Interface Toolkit	Menu manager, dialog manager, control manager...
Windowing System	Toolbox event manager, window manager, ...
Graphics Library	QuickDraw, font manager
Operating System	MacOS
Hardware	Motorola 68000,020,030

Higher-level tools
- Application framework — MacApp
  - An object-oriented class library that implements a standard set of user interface objects and interaction styles
- Prototyping tool via graphical specification — HyperCard

12.4 Windowing system architecture

Windowing system = window manager + window system
(Fig. 12.3)

Window system (base layer) supports a procedural interface known as an *application programmer interface* (API)

Figure 12.3 Architecture of window systems (Myers, 1993, from BGBG, p. 327)

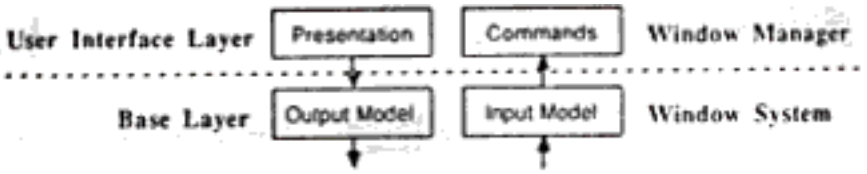


Figure 2. The windowing system can be divided into two layers, called the *base* or *window system* layer, and the *user interface* or *window manager* layer. Each of these can be divided into parts that handle output and input.

Base layer output model
- Windowing and clipping
- Drawing text, lines, etc.
- BitBlt (Rasterop), Postscript imaging models

Base layer input model handles the queueing and processing of input events, e.g., keystrokes, mouse clicks, mouse movements

User interface layer presentation component
- Controlling the creation, presentation, movement, sizing, layering, and deletion of windows
- Tiled versus overlapping windows

User interface layer command component accepts commands that control window presentation

12.5 A networked windowing system — X

History and motivation
- Developed at M.I.T.
- Device independence
- Windowing system for UNIX
- Windows over networks

Principles
- Client-server model
  - “Client” is “host”, could be arbitrary machine
  - “Server” is user's workstation
- Implementation
  - Windowing system mostly on server
  - User interface toolkit, application on client
- Multiple clients — one server possible

Application	On client
User Interface Toolkit	On client
Windowing System	Partially on client
Graphics Library	
Operating System	
Hardware	

..... across network.....

Application	
User Interface Toolkit	
Windowing System	Mostly on server
Graphics Library	On server
Operating System	
Hardware	

12.6 Toolkits and Graphical user interfaces (GUI's)

What is a graphical user interface?
- A highly interactive interface involving pointing and drawing and the display of pictures
- A WIMP (Windows, Icons, Menus, Pointers) interface
- The style of the interface seen by the user
- The look and feel of the interface
- The style defined by the user interface *toolkit* and by how the application is programmed

Features in GUIs
- Windows
- *Widgets* such as menus, buttons, scroll bars
- Controls and control panels
- Query and message boxes
- Mouse/keyboard interface

Toolkit options in GUIs
- Appearance of widgets
- Relationship of widgets
- Control of widgets
- Defaults and shortcuts
- Control by mouse and keyboard

Widget sets and intrinsics
- Determine look and feel, e.g., Motif, OpenLook, Garnet, tk
- *Call-back procedures* defined by application programmer are called when widget is operated by end user

The standards battle in GUI's
- Motivation: Ease of learning, transfer of user expertise
- The legal battleground (“look and feel” copyrights)

12.7 The Tk toolkit

Originally developed for X, but ported to Mac and Windows

Tk commands create and manipulate *widgets,* X windows with a particular appearance and behaviour

For example,
`button .b -text "Press me" -foreground red`
creates a button that displays the text "`Press me`" in red

Examples of Tk widget types
- Buttons
- Scrollbars
- Menus
- Text windows
- General purpose drawing widget called a *canvas*, which allows one to create lines, boxes, and bitmaps

Tk widgets organized into a hierarchy (Fig. 12.4)

Widgets under control of a *geometry manager* that controls their size and location, and that facilitates semi-automatic screen layout

Tk-based application has an *event-driven* control flow, with one widget at a time having the *focus*, with keyboard events being directed to it

Next lecture: higher-level tools, including the Tcl language which facilitates use of the Tk toolkit

Figure 12.4 Typical Tk widget hierarchy (Osterhout, 1994, p. 151)

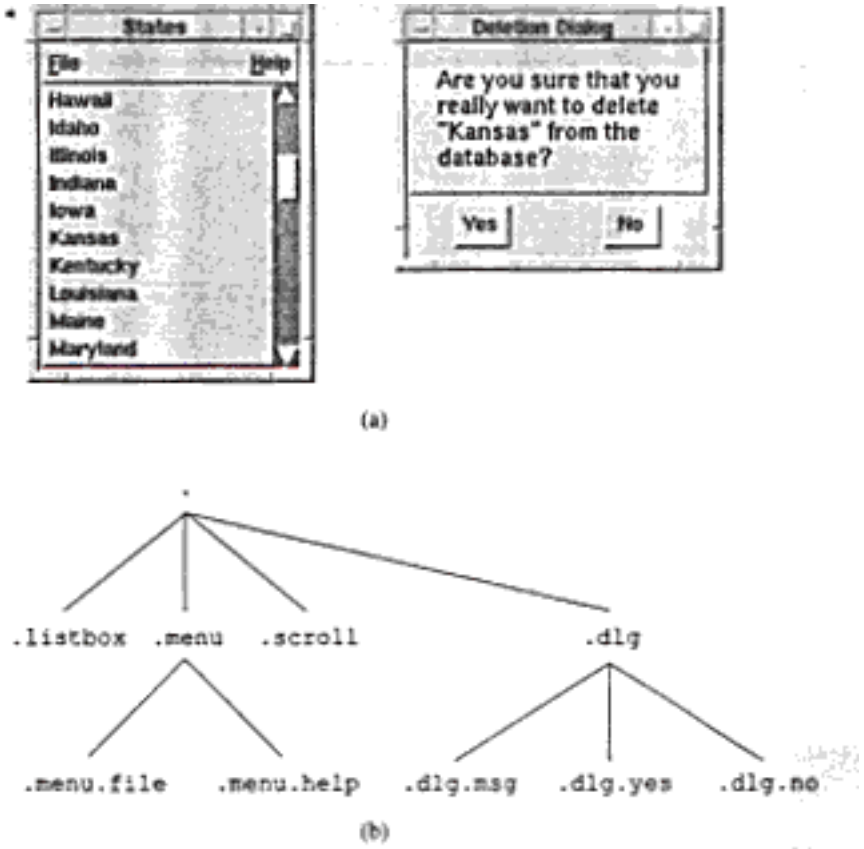


Figure 15.4. Top-level widgets appear in the Tk widget hierarchy just like internal widgets, but they are positioned on the screen independently from their parents in the hierarchy. In this example the dialog box `.dlg` is a top-level widget, as is the main widget. Figure (a) shows how the widgets appear on the screen and (b) shows Tk's widget hierarchy for the application.

12.8 References

Osterhout, John K., *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.