Human Computer Interface Design
Chapter 9 – User Interface Software and Tools

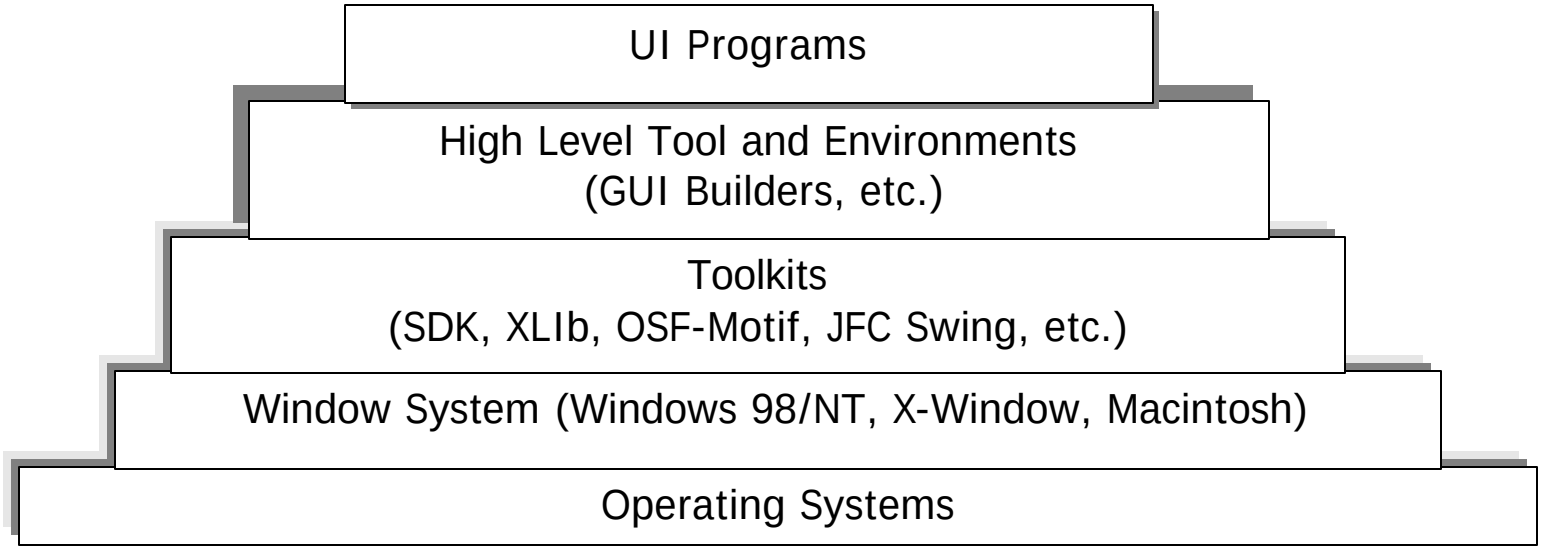
Objective

UI software and tools can improve the usability of UI and its code more economical to create and maintain. This lecture introduces a classification of UI and discusses the event-driven, object-oriented and visual programming style, which is commonly, used in modern UI development tools.

Outlines

1- Event-Driven, Object-Oriented and Visual Programming Styles and Principles
2- Structures of GUI interfaces programs
3- Window Systems, Toolkits, UIMS, GUI Builders

User Interface Software and Tools



Window Systems

- Windows Systems: manages and controls multiple contexts by separating them into different physical parts of the screen.
- Can be part of a program (Smalltalk), part of operating system (SunTools), or a separate program (X-Window)
- Window systems taxonomy include the X Window System (X), Microsoft (MS) Windows (including Windows'95, Windows NT, etc), the BeBox Interface Kit (Be Kit), and the Java Abstract Windowing Toolkit (AWT).

Window System is:
- Programming interface
- Provides output graphics operations to draw clipped to a window = Output Model
- Channels input from mouse and keyboard to appropriate window = Input Model

UI Toolkits

- A library of UI objects or interaction technique (commonly called Widgets) that can be called by application programs.
- A Widget is manipulated using a physical input device to input a certain type of value.
- Toolkits contain procedures to do menus, scroll bars, buttons, dialog boxes

  - Macintosh Toolbox
  - Windows Toolkits: MFC, OWL, Inprise Visual Component Library (VCL),
  - Motif and Open Look for X-window
  - Amulet
  - See The GUI Toolkit, Framework Page (http://www.theoffice.net/guitool) for a complete listing of toolkits

Advantages and Limitations

+ Consistent Look and Feel, Re-use of code
- Hard to use – usability and learnability - (very large libraries with very large manuals), no help with when and how to call what

The Two Perceptions of a Toolkit

1- A toolkit defines a set of widgets with a particular "look and feel"

- Different look-and-feels on same Intrinsics: Athena, Open Look and Motif on Xtk
- The same look-and-feel can be implemented on different Intrinsics: Motif on Xtk, Interviews, Amulet,

2- A toolkit defines how the widgets are implemented (intrinsics)

- Procedure-oriented
  - A toolkit is a library of procedures that can be called
  - Example: Macintosh Toolbox, SunTools library
- Object-oriented
  - A toolkit is a library defines standard classes (object-oriented programs)
  - Natural way to think about UI organization: widgets on screen "seem" like objects
  - Easier to make customizations: programmer can make specialized sub-classes
  - Requires special (single) programming language
  - Xtk, Interviews, Garnet

Higher Level Tools and Environments

Rapid Prototyping tools
- Quickly see how UI is going to look and act

GUI Builders
- Lay out widgets
- Create menus, dialog boxes

User Interface Development Environments
- Comprehensive support for UI Software

Advantages and Limitations

- Ease of use versus power
- Evidence that interactive tools 10 to 50 times faster than coding with toolkits

Event-Driven Programming Principles

- Anything that typically sits around waiting to be told what to do, and responding to external stimuli, **is event-driven**.

Event-Driven versus procedure-driven program

- Procedure-driven Program: In a traditional application, execution begins at the top of the program, then flows through function calls and control-flow statements, all in a fairly predictable manner. The program is in control, relegating the user to a subordinate role of entering keystrokes when requested.

- Event-driven Program: In an event-driven program, the user is in control, and decides which portion of the program gets executed. The user clicks the buttons or uses the keyboard to select menu options, tab through controls and so on. Each mouse click and key press generates an event to which the program must respond.

Events can be things that happen or take place like:

- Mouse Buttons
- Modifier Keys (Shift, Control, Option, etc.)
- Double Clicking, triple-clicking
- Function Keys
- Mouse Movement, Mouse-Enter and Exit
- Keyboard
- Windowing Events: Create, Destroy, Open, Close, Resize, [de] activation, [de] iconification,
- System-specific events
- Redrawing Events
- Pseudo-Events: communication between objects In fact, almost everything the user does generates an event

1. Events may be generated by the windowing system, devices, or define by users (custom events).
2. Events are delivered using messages; the operating system usually has a message system, which can send and receive messages containing event information.

Generic Structure of an vent-Driven Program

1- Setting up (creation, instantiation) the top-level widget (e.g. main window, dialog box, applet, etc.)
2- Setting up others widgets
3- Organizing and grouping widgets in containers
4- Making visible the top-level widget
5- Handling events

- Main loop that wait for events to occur (one or more specific event is the condition, e.g. exit item in menu file)
- Procedures that are executed when events occur. Two kinds of procedures are possible:
  a- Procedure (callback functions) = Response to a specific event addressed to a certain widget (
  b- Procedure (event handlers) = Response to a set of similar events addressed to different related widgets (e.g. close main window can close all intermediate windows)