Course Overview

- Introduction
- Understanding Users and Their Tasks
- Principles and Guidelines
- Interacting with Devices
- Interaction Styles
- UI Design Elements
- Visual Design Guidelines
- **UI Development Tools**
- Iterative Design and Usability Testing
- Project Presentations and Selected Topics
- Case Studies
- Recent Developments in HCID
- Conclusions

Chapter Overview UI Development Tools

- Motivation
- Objectives
- UI Development Tools
- Tool Categories
- User Interface Management Systems
- Evaluation of UI Development Tools
- Important Concepts and Terms
- Chapter Summary

Agenda

- HW 10 Demos
- hand in Project Documentation Part 2
- hand in HW 5: CLUES evaluation

Why UI Development Tools?

- Write down three important reasons to use UI development tools!
  -
  -
  -

Motivation

- a substantial part of the development effort for software goes into the user interface
- current technology allows complex user interfaces:
  - graphics, windows, multi-tasking, drag and drop, etc
  - differences in I/O devices, customization of user interfaces
- competitive pressure
  - user interfaces are the most visible part of a system
  - fast development cycles
- UI development tools are business opportunities
  - savings in development time, more attractive appearance,

Objectives

- to be familiar with different types of UI development tools
- to know important selection criteria for UI development tools
- to be able to select a UI development tool for a specific task
- to have pointers for finding further information on UI development tools

UI Development Tools

- characterization
- requirements
- benefits
- drawbacks
- selection of tools
- UI development tool categories

Characterization

- software packages that provide automated support for any part of the UI development process
- UI development tools make it easier to prototype systems
- support many of the activities of a UI developer
- not necessarily complex
  - e.g. a simple routine that controls the movement of a mouse , or
  - a complex software system that oversees the operations, administration, and maintenance of an entire system
    - e.g., telephone network

Requirements for Tools

- help with the design of the UI
- translation of the end user's task specification
- prototype the UI given the design specification
- help create easy-to-use UIs
- permit designers to rapidly evaluate different prototype designs and alternatives
- permit non-programmers to design and prototype UIs
- portability across different platforms and devices
- easy to use

Benefits

- simple GUI
  - for designing windows and screens
- greater access and use
  - permit non-programmers to contribute their expertise
- prototype development
  - can be used to build prototypes throughout the development cycle
  - some tools provide a mode for testing displays without compiling and executing the entire application
- user input
  - users can evaluate each prototype and provide feedback
  - users are more satisfied with the final product

Benefits (cont.)

- automatic code generation
  - some tools can generate part of the interface code
    - less code to write
    - generated code is more consistent
    - coding standards like naming conventions and standard headers and comments are maintained automatically
    - generated graphics code contains fewer bugs when isolated from the functional code
    - tool-generated display designs, components, and generated code can sometimes be reused
      - ? saves development time
      - ? provides consistency across different Uis
      - ? different applications are more likely to have consistent UIs if they are built using the same tool

Drawbacks

- steep learning curve
  - learning to use a tool may be slow
    - especially if a tool has a complicated UI and is difficult to use
    - can make the development process less efficient
- resistance to use
  - developers may resist using a tool
  - especially if they believe it makes work harder
- hindrance to development
  - tool that automatically generates code can be helpful
  - it may also hinder development if poorly implemented

Drawbacks (cont.)

- increased task difficulties
  - the automatically generated code can be extremely difficult to read and modify
  - integration with the rest of the application can be difficult
- impact development schedule
  - can affect the UI development schedule
    - tool may be complex, slow, not robust, inconsistent, buggy
- limited functionality
  - a tool may not take advantage of the hardware and native platform capabilities
    - often occurs with multi-platform tools
    - may implement only a set of display functionality that is common to all of the supported platforms

Selection of Tools

- decisions
  - whether to use or not to use a UI development tool
  - if yes, which one?
- considerations
  - benefits/drawbacks
  - application needs, development environment, time, resources, money, ...
- goals
  - simplify the development process
  - develop a better product

UI Development Tool Categories

- GUI builders
- User Interface Management Systems (UIMS)
- graphical user environments
- specialized widgets
- plotting and analysis tools
- 3D visualization tools
- cross-platform development tools
- GUI porting tools
- application development tools
- other development environments

GUI Builders

- simple interface development environment
- developers can create and manipulate GUIs in a WYSIWYG environment
- examples:
  - Builder Xcessory (Integrated Computer Solutions)
    - $3,200
    - http://www.ics.com/Products/BX40/BX40Datasheet/Welcome.html
  - X-Designer (Imperial Software Technology)
    - $3,500
    - http://www.dvcorp.com/mktg/xd.html

User Interface Management Systems

- provide the functionality of GUI builder, plus
- the capability to define and execute the functionality of the displays from within the tool
- can generate code to build a version of the UI that will execute independently of the tool
- high-level and interactive
  - to free designers from low-level details
  - to provide them with a comprehensive set of tools
  - separation of responsibility between UIMS and application
    - the application responsible for carrying out the work
    - the UIMS handles all communication details with the end user

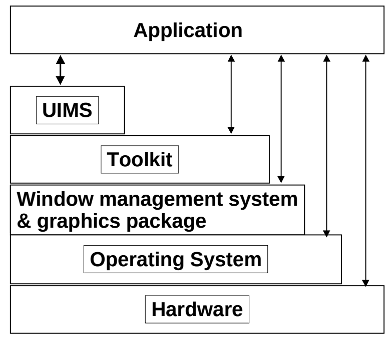
UIMS (cont.)

- typically consist of three complementary parts:
  - a graphics or text environment
  - a linkage function that defines the operational aspects of a dialog, and couples the design of an interface to the functionality of its underlying system
  - a management function that controls interaction during run time (session tools)
- underlying any UIMS is an interaction library
  - toolkit of routines for building UIs
- UIMS take care of low-level things
  - like the drawing of dialog boxes, and the tracking of the cursor as the user moves the mouse

UIMS (cont.)

- the applications programmer must deal with higher-level abstractions
  - for example “request an answer to this question from the user” reduces the amount of coding required for a GUI
- also separates the UI specification from the underlying functionality
  - changes in one do not always require changes in the other
- UIMS provide a way for a designer to specify the interface in a high-level language
  - UIMS then translate that specification into a working interface, managing both the details of the display and its associated input and output, and the interaction with the rest of the program

UIMS Architecture



UIMS Benefits

- UI code is easier and more economical to create and maintain
  - low-level and repetitive tasks can be eliminated
    - e.g. menu handling, button construction
    - more time to concentrate on UI design aspects
  - UI code can take up to 80% of the total code
  - Use of UIMS can reduce development time and costs by a factor of 4X or more.
  - UI specifications can be represented, validated, and evaluated more easily.
  - less code to write
    - much of it is supplied by the UIMS

UIMS Benefits (cont.)

- better modularization due to the separation of the UI component from the application
  - UI can change without affecting the application
  - Changes in the application are possible without affecting the UI
- level of programming expertise of the interface designers and implementers can be lower
  - hides much of the complexities of the underlying system
- reliability of the UI will be higher
  - the code of the UI is created automatically from a higher-level specification

UIMS Benefits (cont.)

- promote the creation of higher-quality UIs
  - designs can be rapidly prototyped and implemented
    - possibly even before the application code is written
  - incorporation changes discovered during user testing
  - there can be multiple UIs for the same application
  - more effort can be spent on the tool than may be practical on any single UI
    - the tool will be used with many different applications
  - consistent UIs for different applications
    - if they are created using the same UIMS
  - involvement of domain experts
    - rather than having the UI created entirely by programmers

UIMS Drawbacks

- too much emphasis on control flow and screen design
  - neglecting other key elements of UIs
    - such as context-sensitive menus, direct manipulation of interactively created objects, macros, undo, etc.
- level of abstraction is sometimes too low
  - forces the UI designer to deal with too many details
- for a good reference on UIMS, see Myers, B.A. (1996): *UIMS, Toolkits, Interface Builders*.
  - available at http://www.cs.cmu.edu/afs/cs/user/bam/www/toolnames/index.html

UIMS Examples

- commercial, industrial-strength examples:
  - UIM/X (Visual Edge Software, Montreal)
    - $5,000
    - available in the CS dept.
    - http://www.vedge.com/prods/uimx.html
  - TeleUSE (Aonix)
    - $7,500
    - http://www.aonix.com/Products/teleuse/teleuse.html

Graphical User Environments

- also referred to as data visualization tools
- provide dynamic data representations and visualization
- developers can define and animate objects or scenes of objects
  - design static displays with standard or custom objects.
  - attach dynamic behavior to objects
    - useful for real-time applications

Examples

- DataViews (Dataviews Corporation)
  - $17.7K
  - http://www.dvcorp.com/mktg/dv.html
- SL-GMS (SL Corp.)
  - $12,500
  - http://www.sl.com

Specialized Widgets

- provide specific functionalities
  - beyond the basic capabilities of a typical widget set like X/Motif
- customization
  - individual widgets or libraries of widgets from different vendors can be integrated

Examples

- GroupKit (University of Calgary)
  - free
  - http://www.cpsc.ucalgary.ca/projects/grouplab/projects/groupkit/groupkit.html
- INT Widgets (INT Corp.)
  - $1,750 - $3,000
  - http://www.int.com/products

Plotting and Analysis Tools

- let designers display technical data
  - via a variety of generic &/or customized graphs
- technical analysis of data
  - ability provided by some tools
- example
  - PV-Wave (Visual Numerics)
    - $?
    - http://www.vni.com/products/wave/newoverview.html

3D Visualization Tools

- specialized category of tools
- generally most practical on hardware platforms that support the extensive calculations required in 3-D graphics
- examples:
  - AVS/Express (Advanced Visual Systems)
    - $?
    - http://www.avs.com/products/index.htm
  - IDL (Research Systems)
    - $Free?
    - http://www.rsinc.com/idl/index.html

Cross-Platform Development Tools

- essential for multi-platform application
- common "look-&-feel" across different platforms
  - layered Application Programming Interface (API) or a simulated API
- API provides a common programming interface
  - across all the available platforms
  - adds a layer between the application and the native windowing system

Examples:

- XVT Development Solution for C and C++ (XVT Software)
  - $1,950 -$6,300
  - http://www.xvt.com/docs/dsc.html for C
  - http://www.xvt.com/docs/dscpp.html for C++
- Open Interface Element (Neuron Data)
  - $10,000
  - http://www.neurondata.com/Products

GUI Porting Tools

- automatically port UI code to a different platform.
- native calls from the original platform are replaced with native calls of the new platform
- example
  - Wind/U (Bristol Technology) ports Visual C++ UI by replacing Windows 95 or Windows NT with native platforms such as X/Motif
    - $? License
    - http://www.bristol.com/Products/windu.html

Application Development Tools

- miscellaneous capabilities beyond UI design and development
  - distributed application communications, real-time system capabilities, database access, expert system development
- examples:
  - Galaxy (Ambiencia Information Systems, Brazil)
    - $?
    - http://www.ambiencia.com
  - RTworks (Talarian Corp.)
    - $?
    - http://www.talarian.com/rtworks.html

UI Development Environments

- Visual BASIC
  - GUI builder with access to the Windows interface tools
  - can also be considered a visual programming language
- Visual C++
  - example of a cross-platform development tool
    - standard development environment for both Windows 95 and Windows NT platforms
  - programming language
  - C/C++ compiler
  - Windows host integrated development environment (IDE)
    - supports design, development and debugging, and Microsoft Foundation Class (MFC) applications

Amulet

- Automatic Manufacture of Usable and Learnable Editors and Toolkits (Brad Myers CMU)
  - UI development environment for C++
    - supports 3D, gestures, speech, multimedia, multiple users at the same time, WWW access
    - extensive end-user customization
  - set of flexible widgets
    - buttons, check boxes, radio buttons, option pop-up windows, menus, menu bars, scroll bars, scrolling windows, text input fields, selection handles
  - portable across X11, Microsoft Windows 95 and Windows NT, and the Mac
  - http://www.cs.cmu.edu/~amulet

Other UI Development Environments

- Java
  - see below
- Virtual Reality Modeling Language (VRML)
  - for 3D environments
- prototyping tools
  - MacroMind Director
    - powerful demo tool

Java Swing

- GUI toolkit for developing windowing components
  - part of a new class library called the Java Foundation Classes (JFC)
    - swing makes up the bulk of JFC
    - evolution of the Abstract Windowing Toolkit (AWT)
  - platform independent
  - Pluggable Look and Feel (PL&F)
    - lets developers choose the appearance and behavior of the windowing components
      - ? Java, Mac, Motif, Windows
  - accessibility
    - compatibility with h/w and s/w designed for people with special needs (e.g., limited sight, motor problems, etc.)

Tools for Other Modalities

- Speech Development Kits (SDKs), and Speech APIs (SAPIs)
  - are beginning to appear from companies like Microsoft and JavaSoft (Java Speech API)
- dedicated toolkits for developing speech UIs and applications are emerging

Examples

- SpeechWorks 4.0
  - Applied Language Technologies (Altech)
  - DialogModules (high-level application building blocks), SpeechBuilder (set of tools for customizing DialogModules, and for creating other apps)
- CSLUrp
  - Center for Spoken Language Understanding at the Oregon Graduate Institute. rp = rapid prototyper
- Nuance Developers' Toolkit
  - open architecture
  - Java and ActiveX APIs, C and C++ interfaces
- OpenSpeech Recognizer Developer Kit
  - Nortel Networks
- Speech Assistant, Natural Language Assistant
  - Unisys

Evaluation of Tools

- usability
  - should be easy to install, easy to learn, and easy to use
- functionality
  - should match the requirements of the UI
  - should provide the widgets you need
  - should provide a convenient way to extend the widget set
- flexibility
  - should be responsive to changing application requirements

Evaluation of Tools (cont.)

- portability
- support
  - should include documentation, training, and other information provided by the vendor, as well as on-line technical help
- cost
  - consider non-recurring costs (e.g., purchase price) as well as recurring costs (e.g., licensing, technical support, maintenance, updates, training, etc.)

© 1999 Franz Kurfess

Important Concepts and Terms

- 3D visualization tools
- application development tools
- cross-platform development tools
- graphical user environments
- graphical user interface (GUI)
- GUI builders
- GUI porting tools
- human factors engineering
- human-machine interface
- input/output devices
- Java Swing
- plotting and analysis tools
- rapid prototyping
- user interface design
- UI development tools
- UI management systems
- user requirements
- Visual Basic
- What You See Is What You Get" (WYSIWYG)
- widget
- window

© 1999 Franz Kurfess

Chapter Summary

- UI development tools facilitate the efficient design and implementation of consistent user interfaces
- re-use of components and libraries with frequently used functions and objects ("widgets")
- some tools provide support for the development of cross-platform systems with consistent, but platform-specific interfaces
- currently the main emphasis is on tools for visual user interface design (GUIs)

© 1999 Franz Kurfess

© 1999 Franz Kurfess