UI Development Environments (UIDE) & Prototyping Tools

Lecture 9

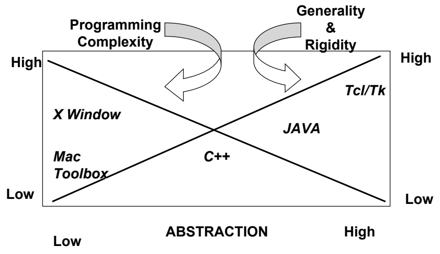
The Problem

- Interface Programming is
  - Complex
  - Time-consuming
  - Error-Prone
- Results in
  - Low programming productivity
  - Untested products
  - Limited number of expert programmers who can write UI code

The Solution

- Abstraction
  - High-level programming languages tailored to UI
  - Object-oriented languages supporting event processing
- Environment
  - Specialized GuiBuilders
- Automation
  - UIMS
  - Geometry managers
  - Knowledge-based systems
    - UI expert systems

Tradeoffs with Abstraction



User Interface Implementation

- Software Prototype only
  - Primarily for developing and testing prototype
  - Examples
    - » ???? (Macintosh)
    - » Visual Basic (Microsoft Windows)
    - » Tcl/Tk, QT, Suit, Garnet/Amulet (Cross-Platform)
  - Index of UI Tools for Prototyping
    - http://www.geocities.com/SiliconValley/Vista/7184/guitool.html
    - http://www-cgi.cs.cmu.edu/afs/cs/usr/bam/www/toolnames.html
- Target Software Implementation
  - Direct implementation in target language
  - Examples
    - » Java, C++

Supportive Programming Environment

- Strong support (GUI Builders)
  - Drag and drop of widgets from inventory
  - Visual display and positioning of objects
  - Interactive sizing of object geometry
  - Programming by Demonstration
    - » example: paths for animation
    - » example: interactive selection of objects
- Weak support
  - Dialog boxes, menus
  - Structure editor
  - Debugging?
- Specialized interface language
- Fast program/test cycle

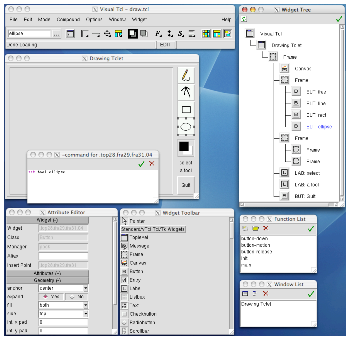
Demo of GUI Builder: Visual Tcl

- Download from
  - http://vtcl.sourceforge.net/
  - Available for numerous platforms

Attributes of a GUI Builder: Visual Tcl

- Strong Support
  - Selection of widgets from icon panel
  - Continuous display of constructed UI
  - Drag and drop to position widgets in window
  - Interactive sizing of object geometry
  - Display of constructed UI widget hierarchy
- Weak Support
  - Attribute editor for each widget
  - Click on widget to open -command editor to bind actions to widget
- Fast program/test cycle
  - Button to change from “Edit” to “Test”
- Generates Tcl/Tk code

Visual Tcl Screen: Draw

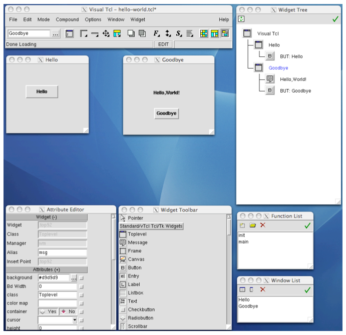


```
set tool ellipse
```

Widget .top28.fra29.fra31.04
Class Button
Manager pack
Alias
Insert Point .top28.fra29.fra31
anchor center
expand ◆ Yes ◇ No
fill both
side top
int. x pad 0
int. y pad 0

button-down
button-motion
button-release
init
main

Visual Tcl Screen: Hello world



Widget (-)
Widget .top92
Class Toplevel
Manager wm
Alias msg
Insert Point .top92
Attributes (+)
background #d9d9d9
Bd Width 0
class Toplevel
color map
container ◇ Yes ◆ No
cursor
height 0

Visual Tcl Program: Hello world

Goodbye window, button & message

```
proc vTclWindow.top92 {base} {
    if {$base == ""} {
        set base .top92
    }
    if {[winfo exists $base]} {
        wm deiconify $base; return
    }
    set top $base
    ###################
    # CREATING WIDGETS
    ###################
    vTcl:toplevel $top -class Toplevel
    wm withdraw $top
    wm focusmodel $top passive
    wm geometry $top 293x210+853+80; update
    wm maxsize $top 2865 994
    wm minsize $top 1 1
    wm overrideredirect $top 0
    wm resizable $top 1 1
    wm title $top "Goodbye"
    vTcl:DefineAlias "$top" "msg" vTcl:Toplevel:WidgetProc "" 1
    bindtags $top "$top Toplevel all _TopLevel"
    vTcl:FireEvent $top <<Create>>
    wm protocol $top WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"

    message $top.mes93 \
        -highlightcolor black -text Hello,World! -width 87
    vTcl:DefineAlias "$top.mes93" "Message1" vTcl:WidgetProc "msg" 1
    button $top.but94 \
        -command {Window hide $widget(msg)} -text Goodbye
    vTcl:DefineAlias "$top.but94" "Button1" vTcl:WidgetProc "msg" 1
    ###################
    # SETTING GEOMETRY
    ###################
    place $top.mes93 \
        -in $top -x 100 -y 50 -width 87 -height 52 -anchor nw \
        -bordermode ignore
    place $top.but94 \
        -in $top -x 124 -y 121 -width 67 -height 31 -anchor nw \
        -bordermode ignore

    vTcl:FireEvent $base <<Ready>>
}
```

Benefits and Limitations of GUI Builders

- Benefits
  - Improved productivity
  - Can often be used by non-programmers
  - Allow rapid program/test cycle to allow usability testing
- Limitations
  - Often rigid
    - » Support limited types of widgets
    - » Primitive semantics for actions
  - Single-pass
    - » Can't write new code into the program and run back through the GUI Builder
  - Generate very inefficient, undocumented code