Tcl and Tk:

A Programming System

for X11 User Interfaces

John Ousterhout

Computer Science Division
Department of EECS

University of California at Berkeley

Overview

What I've built:

- Tcl: embeddable command language.
- Tk: X11 toolkit and widgets based on Tcl.

The principle: single interpretive language controls all aspects of all interactive applications.

- Function of application.
- Interface of application.
- Composing pieces of application.
- Communication between applications.

Results:

- Raise the level of X programming (simpler, 5-10x faster application development).
- Greater power (more things programmable, program applications to work together).

Outline

1. **The Tcl language.**
2. **The Tk toolkit.**
3. **Tk applications.**
4. **Composing applications: hypertools.**
5. **Status and conclusions.**

Tcl: Tool Command Language

Problem:

- Interactive programs need command languages.
- Traditionally redone for each application.
- Result: weak, quirky.
- Emacs and csh nice, but can't reuse.

Solution: Tcl

- Command language = embeddable C library.
- Powerful features: procedures, variables, lists, expressions, loops, etc.
- Extensible by applications.

Language Philosophy

Classes of language:

- Large application implementation (structure, performance important). } C
- Scripting, extensions.
- Interactive commands (structure bad, performance not critical). } **Tcl**

One language can’t meet all three needs?

Tcl goals:

- Simple syntax (for humans).
- Programmable.
- Easy to interpret.
- Simple interface to C procedures.

Tcl Syntax

Basic syntax like shells:

- Words separated by spaces:

```
cmd arg arg arg ...
```

- Commands separated by newlines, semi-colons.
- Commands return string results.

Simple substitution rules:

- Variables:

```
set a $b
```

- Command results:

```
set a [expr $b+2]
```

- Complex arguments:

```
if $a<0 {
  puts stdout "a is negative"
}
```

More on the Tcl Language

Rich set of built-in commands:

- Variables, associative arrays, lists.
- Arithmetic expressions.
- Conditionals, looping.
- Procedures.
- Access to UNIX files, commands.

Only datatype is strings:

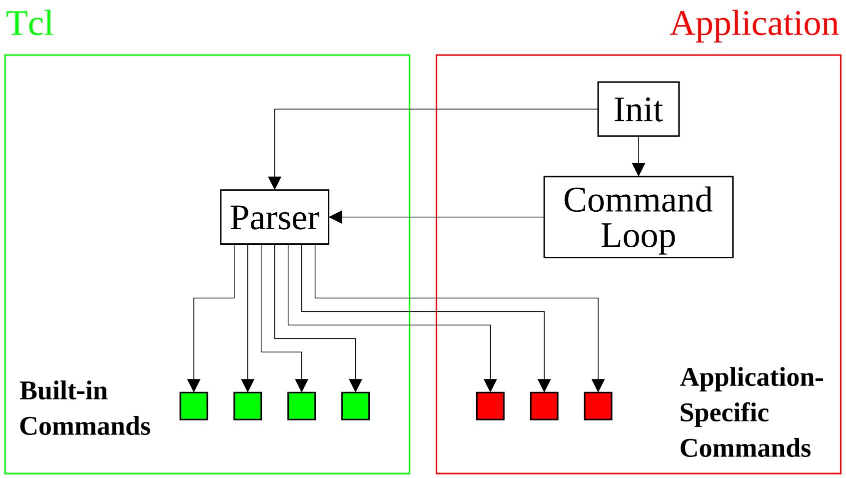
- Easy access from C.
- Programs and data interchangeable.

Factorial Procedure

```
proc fac x {
   if $x==1 {return 1}
   expr $x*[fac [expr $x-1]]
}
```

`fac 4` **returns** `24`

Embedding Tcl in Applications



- Application generates Tcl scripts.
- Tcl parses, substitutes, passes argc/argv to command procedures.
- Application extends built-in command set.

The Tk Toolkit

The problem:

- Too hard to build applications with nice user interfaces.

The wrong solution:

- C++, object-oriented toolkits.
- Only small improvement (10-20%?): must still program at a low level.

The right solution:

- Raise the level of programming.
- Create interfaces by writing Tcl scripts.

Creating Interfaces with Tk

Widgets/windows have path names:

```
.dlg.quit
```

Create widget with command named after class:

```
button .dlg.quit -text Quit \
    -foreground red -command exit
```

Tell geometry manager where to display widget:

```
place .dlg.quit -x 0 -y 0
pack .dlg.quit -side bottom
```


Other Tk Features

Manipulate widgets with widget commands:

```
.dlg.quit flash
.dlg.quit configure -relief sunken
```

Use Tcl for interconnection:

- Buttons, menu entries invoke Tcl commands.
- Scrollbars and listboxes communicate with Tcl.
- Can define new event bindings in Tcl.
- Selection, focus accessible via Tcl.

Tk also provides C interfaces:

- Create new widget classes.
- Create new geometry managers.

What's a Tk-based application?

1. **The Tcl interpreter.**

2. **The Tk toolkit.**

3. **Application-specific C code:**
   - New object types.
   - New widgets.

   Tcl commands

4. **Tcl scripts:**
   - Build user interface.
   - Compose application primitives into useful functions.

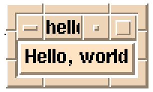
The Simplest Tk Application: Wish

No C code except command-line reader.

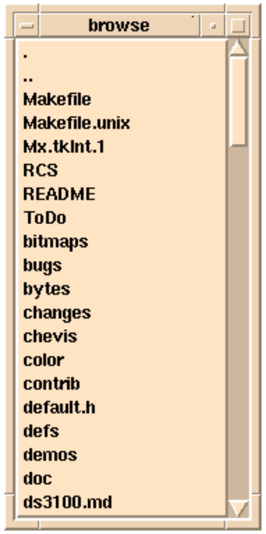
Can build many applications as `wish` scripts:

- Hello, world:

```
label .hello -text "Hello, world"
pack .hello
```



- Simple directory browser: 30 lines.



Browser Wish Script

```
listbox .list -yscroll ".scroll set" \
   -relief raised -geometry 20x20
pack .list -side left
scrollbar .scroll -command ".list yview"
pack .scroll -side right -fill y
if {$argc > 0} {
   set dir [lindex $argv 0]
} else {
   set dir "."
}
foreach i [exec ls -a $dir] {
   .list insert end $i
}
bind .list <Double-Button-1> {
   browse $dir [selection get]
}
bind .list <Control-c> {destroy .}
focus .list
proc browse {dir file} {
   global env
   if {$dir != "."} {set file $dir/$file}
   if [file isdirectory $file] {
       exec browse $file &
   } else {
       if [file isfile $file] {
           exec xedit $file &
       } else {
           puts stdout "can't browse $file"
       }
   }
}
```


Perspecta Presents!

Commercial presentation package:

- Presentation = sequence of slides.
- Text, graphics, images.
- Backgrounds, slides, notes.
- Postscript output, on-line slide shows.

Implemented using Tcl and Tk:

- 29000 lines of new C code.
- 1 new widget for displaying slides.
- ~30 other Tcl commands for manipulating presentations.
- 11000 lines of Tcl scripts.

Uses of Tcl in Perspecta Presents!

1. **Powertext: text created by Tcl script, not typed by user.**
   - Slide numbers.
   - Bullet numbers.
   - Update values from database?
2. **File format = Tcl script. To load, just execute file.**
3. **Selection exchanged as Tcl script (selectively copy backgrounds, looks, etc.)**
4. **Undo/redo:**
   - Undo/redo script pairs saved in log file.
   - Infinite-level undo/redo.
   - Recovery after crashes.
5. **Slide shows, etc. etc.**

Composing Applications

The problem:

- Only communication between applications is via selection.
- Result: monolithic applications.

The solution: `send` command

- `send appName command`
- Implemented using X11 properties.
- Any Tk application can invoke anything in any other Tk application: interface or actions.
- Result: powerful communication.

Composing Applications, cont'd

Examples:

- Debugger sends command to editor: highlight line of execution.
- User-interface editor sends commands to modify interface of live application.
- Multi-media: send `record`, `play` commands to audio and video applications.
- Spreadsheets: cell sends commands to database to fetch current value.

Revolutionary results:

- Build complex systems as collections of specialized but reusable hypertools.
- Easy to create active objects: embedded Tcl commands. Hypertext, hypermedia easy.

Status

Tcl:

- 20000 lines C code.
- First released January 1990.

Tk:

- Intrinsics: 21500 lines C code.
- Motif-like widgets: 34000 lines C code.

Buttons	Labels	Scales
Canvases	Listboxes	Scrollbars
Entries	Menus	Texts
Frames	Messages	

- First released March 1991.

User community:

- 5000-10000 (as of January 1993).

Conclusions

Power from programming:

- High-level programming for power, flexibility.
- Extensibility.
- One language for many things.

Power from composition:

- Widgets within an application.
- Send between applications.

Tcl + Tk = shell of 1990's?

Wanted: application developers.