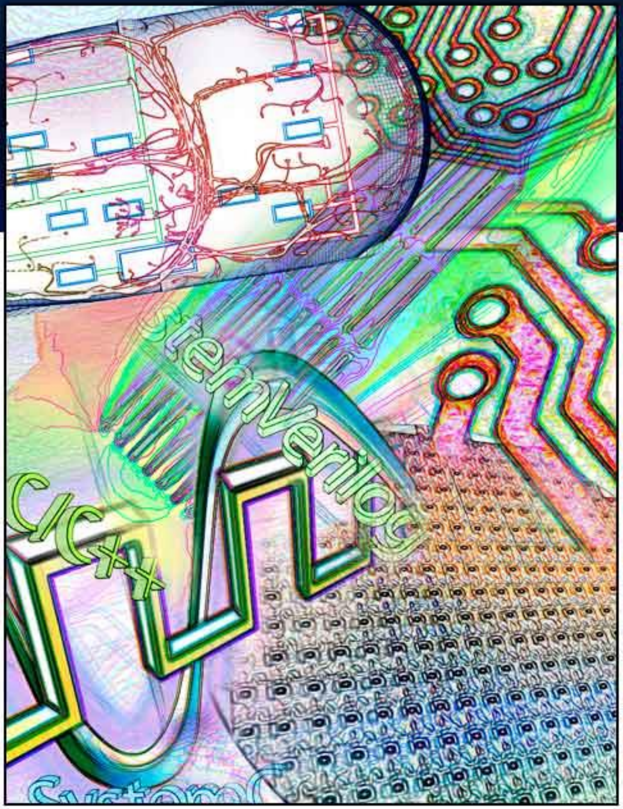

Wish2Wish An IDE For Your Tcl/Tk Application

Trey Jackson

Software Engineer, RVE Group

November, 2014

Mentor Graphics®

Wish2Wish

- Problem: You want to debug/enhance your Tcl/Tk app
- Solution: Tk's `[send]` functionality
- Bonus: Integrate with Emacs' inferior process support

The Problem

- You have a running Tcl/Tk application
- You don’t have an interpreter exposed, or
- You have an exposed command line, but who wants the crummy command line?
  - Still using cut/paste? Are you an animal?

Additional twists

- You want to introspect/modify multiple apps simultaneously

Solution: Tk `[send]`

- But, that's laborious to use bare. Let's make it nicer: (pseudo code):

```
while (1) {
   set cmd [w2w::get_next_command]
   if {[catch {send $::app $cmd} res]} {
      puts "Error: $res"
   } else {
      puts $res
   }
}
```

`[send]` continued

- The `[get_next_command]` handles incomplete commands and changing the prompt
- Also handles passing through commands directly to the `wish2wish` interpreter (for `w2w` development)
- The implementation has been muddied up by an initial push to introduce line-based debugging (not finished)

`[send]` continued

- Now we've got an "interperter", but we need our debug statements. Overload `[puts]` to send information back.
- When switching between applications, wish2wish automatically makes the window dance to indicate which is now connected.
- Automatically responds to application name changes (`tk appname new_name`)
- Hook into application startup and automatically connect to running `wish2wish` process
- Truncate long `puts` statements
- Automatically resends connection code if application (of same name) is brand-new

Emacs inferior process

- Here’s where I lose ¾ of you… “Emacs”
- Background: Emacs supports “inferior processes” – which means it will create a process and interact with it. Examples:
  - Shell
  - sql interpreter
  - Tcl interpreter
  - Lisp interpreter (of course)
- For a given mode (e.g. Tcl), you get convenient shortcuts like:
  - `switch-to-tcl (C-c C-s)`
  - `tcl-eval-defun (C-c C-v) (`aka `tcl-eval-proc)`
  - `tcl-eval-region (C-c C-x)`
  - `tcl-load-file (C-c C-f)`

Other Emacs tweaks

- Connect completion through interpreter

  I use hippie-expand, but can be done with any.
  MyComm `M-/` and it expands to MyCommand
- Stack traces (via `bgerror` (yes, I use Tcl/Tk 8.4)) automatically displayed in dedicated buffer
  - Additionally, enable stack trace navigation via keyboard shortcuts and using TAGS infrastructure
- Determine if process `[pwd]` is local to specific machine, and update process buffer appropriately (via `tramp`)
- Cycle through available applications with keystroke
- Use Emacs' `screen` for quick launching of application from shell with keystroke

Implementation

- One .tcl file
  - Implements read-eval-print interpreter
  - Also sourced into application
- One .el file
  - Implements command completion, stack trace display/navigation, etc.

Results

- I never leave Emacs, so I win
- Integration into other editors left as an exercise for the user

Mentor Graphics®

www.mentor.com