Tcl Widget Objects

Version 0.1.0

Alex Baker

https://github.com/ambaker1/wob

February 27, 2023

Abstract

Due a conflict between the OpenSees and Tcl *load* commands, Tcl binary packages cannot be loaded in after *model* is called. Most notably, this restricts the use of Tk widgets. However, the Tcl *interp* command can be used to create a fresh Tcl interpreter within the main OpenSees interpreter, and widgets can be built there instead. The "widget" module formalizes this, allowing for creation of Tk widget objects, each with their own Tcl interpreter.

Note that this package is simply a framework to build widgets with. Knowledge of Tk and event-driven programming is critical to build widgets. Also, although this package was developed specifically for OpenSees, it is still applicable to other Tcl applications.

Creating Widget Objects

Widget objects are created from the *widget* class using the standard methods *new* or *create*. When a widget is created, it also creates a unique Tcl interpreter and loads in the Tk package, binding the "close window" event to destroy the widget object and interpreter. Once created, *widget* objects act as commands with an ensemble of subcommands, or methods. These objects can be deleted with the method *destroy*.

```
widget new <$title>
widget create $objectName <$title>
```

`$objectName`	Explicit name for object.
`$title`	Title of main widget window (default "Widget").

Example 1: Creating a widget object

Code:

```
set widgetObj [widget new]
```

Removing Widget Objects

The standard method *destroy* removes a widget object from the main OpenSees interpreter, destroying the object, widget window, and widget interpreter. This is also called if the window is closed or an "exit" statement is evaluated in the widget interpreter.

```
$widgetObj destroy
```

The Widget Interpreter

All interfacing with the widget is done through its corresponding interpreter. The widget's interpreter command can be accessed with the method *interp*, for advanced introspection.

```
$widgetObj interp
```

Building a Widget

The main method for building a widget is *eval*, which evaluates Tcl/Tk code within the widget interpreter. The method behaves the same as the Tcl *eval* command, but within the widget interpreter.

```
$widgetObj eval $arg1 $arg2 ...
```

`$arg1 $arg2 ...`	Arguments to be concatenated into a Tcl script to evaluate.

Widget Variable Access

For convenience, variable values may be passed to the widget interpreter with the method *set*, and retrieved with *get*.

```
$widgetObj set $varName $value
```

```
$widgetObj get $varName
```

`$varName`	Name of variable in widget interpreter.
`$value`	Value to set.

Widget Command Aliases

By default, the widget interpreter does not interface directly with the main OpenSees interpreter. The method *alias* creates an alias command in the widget interpreter to access a command in the main interpreter. This is identical to the Tcl *interp* method.

```
$widgetObj alias $srcCmd $targetCmd <$arg1 $arg2 ...>
```

`$srcCmd`	Command in widget interpreter (creates the command).
`$targetCmd`	Command to link to in the main interpreter (does not create the command).
`$arg1 $arg2 ...`	Optional, prefix arguments to `$targetCmd`.

Entering the Event Loop

In order for widget components to display and be interactive, the Tk event loop must be entered. Some Tk commands automatically enter the event loop, like *tk_getOpenFile*, but for the most part, the event loop must be entered with a call to *vwait*, *tkwait*, or *update* (it is generally bad practice to use *update* though, for a variety of reasons).

The command *mainLoop* is provided as a method to enter the event loop for all widgets, while also taking interactive input from the command line, similar to the "wish.exe" Tcl/Tk program. To exit the event loop and continue with a script, simply enter "return" on the command line.

```
mainLoop <$onBlank>
```

`$onBlank`	What to do after user enters a blank line: "continue" will continue the interactive event loop, and "break" will exit the interactive event loop. Default "continue".

Basic Applications

The example below demonstrates how the widget module can be used to create and manipulate Tk widgets.

Example 2: Filename dialog

Code:

```
set widget [widget new]
set filename [$widget eval tk_getOpenFile]
$widget destroy
```

Example 3: Option selection

Code:

```
set widget [widget new]
$widget eval {
        label .label -text "Choose analysis type:"
        tk_optionMenu .options AnalysisType "" Pushover Dynamic
        pack .label -side top -fill x
        pack .options -side bottom -fill x
        vwait AnalysisType
}
puts [$widget get AnalysisType]
$widget destroy
```

Example 4: Access clipboard

Code:

```
set widget [widget new]
$widget set text "hello world"
$widget eval {
        clipboard clear
        clipboard append $text
}
$widget destroy
```