very preliminary (please comment)...

WavesWorld TCL Kit

Library:	/LocalDeveloper/Libraries/libWWTCLKit.a
IB Palette:	/LocalDeveloper/Palettes/WWTCLKit.palette
Header File Directory:	/LocalDeveloper/Headers/WavesWorld
Import:	WWTCLKit.h

Introduction

The WavesWorld Tcl Kit provides an Objective-C interface to the popular embeddable extension language tcl (the tool command language). In addition, these objects have been designed to seamlessly extend the power of NEXTSTEP's superior development environment by complete integration with InterfaceBuilder.

Tcl as a database

When most people talk about tcl, they usually concentrate on the fact that it is an extensible, embeddable scripting language. There is another perspective, though; tcl as a database. Tcl maintains a database of variables and procedure definitions. The database types aren't very flexible (there's only one - strings) and it's not the fastest around by a long shot, but it is amazingly flexible, and for many tasks, flexibility is much more important. Also, tcl *is* a scripting language, which allows for building up powerful queries on the database. Since tcl is both extensible and embeddable, there is ample opportunity to tie tcl's database to some other objects' database. For example, an extended version of tcl is used by the WW3DKit to do sophisticated modeling and rendering, where the tcl database is tied together with a 3D scene database.

Tracing variables

Like any good database, tcl allows you to perform certain activities when records in the database are read or written. I use this capability to allow user interface elements which conform to a particular (informal) protocol to automatically reflect any changes made to the tcl database that affects them. This can get very, very tricky when the UI objects that tcl is tracing can wink out of existence (say by a user closing the window that they're on) without telling tcl to untrace them. Since tcl is written in C, and is not object oriented (although it's certainly very well written), and I do not want to hack the core tcl library to integrate it into IB for my stuff, I've had to adopt a strict convention whereby the WWTCLInterp object opens up all nib files containing objects which will be tracing variables in the tcl database. In other words, "WWTCLInterp" is the File's Owner of any nib files you want to open. This allows me to become the delegate of windows and panels inside the nib that contain UI objects that we want traced. Unfortunately, given the lack of access to the contents of a nib file in any way other than through

outlet's to the File's Owner prevents me from automatically finding all Windows and Panels in the nib file, but it does allow me to ensure that any Windows or Panels I do find can be searched for UI objects that have tcl variables they want traced. This is all a bit confusing, right now, I'm sure, but it will get clearer soon...

The WWTCL Kit

If we consider tcl as a database, we obviously want to be able to both manipulate data in the database and visualize it. The WWTCLKit palette provides objects which are manipulators and visualizers; some objects can do both. For example, a button might execute some code (setting a variable to some known value, perhaps), but it's on-screen representation might not reflect anything. A slider, on the other hand, might both manipulate some value and also reflect that value in its on-screen representation.

Visualizers

Given the model of tcl as a database, the obv

A

Manipulators

fff.

The WWTCLInterp

While the WWTCLKit UI objects can certainly be used programmatically, by writing Objective-C code with a text editor and compiling it, they are intended to be used be used from inside of IB by dragging and dropping. There

is one object, though, that it is intended to be used programmatically: the WWTCLInterp.