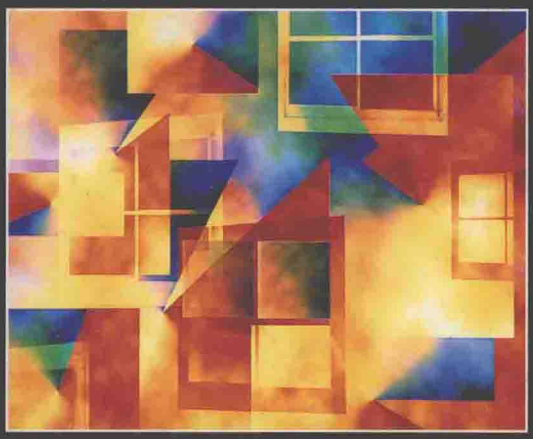
David H. Young



The Visual Tcl Handbook

CD-ROM
INCLUDED

The Visual Tcl Handbook

David H. Young

Prentice Hall P T R
Upper Saddle River, New Jersey 07458
http://www.prenhall.com

Library of Congress Cataloging-in-Publication Data

Young, David (David H.), 1952-
The Visual Tcl Handbook / David Young.
p. cm.
Includes index.
ISBN 0-13-461674-X (paper)
1.Graphical user interfaces (Computer systems) 2. Visual Tcl (Computer program language).
I. Title.
QA76.9.U83Y68 1997
005.13'3—dc20 96–17226
CIP

Editorial/Production Supervision: *Joe Czerwinski*
Acquisitions Editor: *Mark L. Taub*
Manufacturing Buyer: *Alexis R. Heydt*
Cover Design Director: *Jerry Votta*
Cover Illustration: *Marjory Dressler*
Cover Design: *Design Source*
Composition: *Thurn & Taxis*

Prentice-Hall, Inc.
A Division of Simon & Schuster
Upper Saddle River, NJ 07458

The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact:
Corporate Sales Department
Prentice Hall P T R
One Lake Street
Upper Saddle River, NJ 07458
Phone: 800-382-3419
Fax: 201-236-7141, e-mail: corpsales@prenhall.com

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-461674-X

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To my daughter Amanda, the real reason I wrote this book.

Preface

Just add Tcl...

By 1992, I had successfully weaned myself of programming in C and was ready to start acting like a real engineering manager. Managers should not do programming. They should spend their time giving reviews, fighting for resources, and enabling their folks to accomplish great feats (and meet a deadline or two!). At the time, we at SCO (The Santa Cruz Operation, Inc.) were implementing a graphical user interface (GUI) development technology we affectionately called the *widget server*. The widget server was the brainchild of Andy Schloss, Louie Boczek, and Susan DeTar, designed with the purpose of enshrouding SCO UNIX systems management with an OSF/Motif interface for the X environment. The widget server supported Motif programming by extending the conventional UNIX shell. As cool as it was to generate "instant" Motif "applets" around UNIX commands, few engineers were interested in doing it with less-than-glamorous shell scripting.

Then Mark Diekhans entered the picture. As codeveloper of the UNIX environment-specific TclX extensions, along with Karl Lehenbauer, Mark eventually made the suggestion that we try this cool scripting command language from UC Berkeley, developed by a professor whose name is one of the most mispronounced in the industry (next to Linux). John Ousterhout's name is pronounced /OH-stir-howt/ (according to his personal WWW page). So, to make a very long story short, we gave Tcl a shot and now, four years later, the technology formerly known as the *widget server* has been rechristened *Visual Tcl* and is available on ten UNIX platforms.

As the new manager of the widget server development group back in 1992, I thought I should develop at least a general notion of what it was like to program with this new language. My first, rather lofty goal was to create a "Motif wrapper" around the old BSD network utility `rwho` ("remote who"). A few hours later, I had a very useful Motif application up and running, monitoring virtually the entire SCO network of logged-in

users. My status as a nonprogramming engineering manager had evaporated within only a few hours. With this book, I hope you will learn what I did—that with Tcl I could take on projects for which I felt I had neither the time nor the bandwidth, and be effective.

As evidenced by their demonstration of a new graphical "GUI Builder" at the SCO Forum conference in August 1995, SCO continues to enhance the feature set of Visual Tcl. To monitor the newsworthy events of the evolution of Visual Tcl, be sure to check out the official SCO Visual Tcl home page at:

```
http://www.sco.com/Products/vtcl/vtcl.html
```

SCO has trademarked this new language as *SCO Visual Tcl*. In this book, we will refer to it simply as *Visual Tcl*.

Strategy and style

Although Visual Tcl is ideal for a wide range of application development, many of the examples and discussions in this book are targeted at people who manage computers, such as system administrators and self-empowered users. UNIX administrators are usually the last folks to get access to tools, such as GUI scripting, that represent current technology trends and could make their lives easier (and more fun!). With that in mind, in addition to including a CD-ROM of Visual Tcl ported to ten flavors of UNIX (attached at the back of this book), I've made extensive use of UNIX administration-related examples to convey how Visual Tcl can be used to implement solutions in short order. Many of these examples represent simple tasks carried out by UNIX system administrators or experienced users who know their way around UNIX.

The goals behind the design of Visual Tcl were motivated by the demand for the rapid development of graphical systems management. As we will discuss in Chapter 1, *Modern Scripting*, much of the reason for the rise in scripting popularity is being driven by the need for flexibility and rapid development in the systems management environment.

This book focuses on examples that illustrate the most commonly used commands. Hoping to give novice readers the "biggest bang for their buck," I have focused on the basic value of each command as it is introduced. I avoid documenting every available option for the command. Instead, I try to use a narrative writing style to walk you through the introductory process of learning a new language. When you are ready to dive into greater detail regarding specific commands, the complete Visual Tcl command language and associated options are available in Part IV, *Command Pages*. This section is based on the man pages acquired from different contributors, including UC Berkeley, SCO, and the creators of TclX—the Tcl library that gives Visual Tcl its UNIX flavor.

This book was written with the assumption that the reader is familiar with at least one programming language, ranging from simple UNIX shell scripting to C, or just about any high-level language such as Pascal.

One last note about style. Dan Heller's style of writing, as found in *the Motif Programming Manual,* volume 6 (O'Reilly & Associates, Inc.), has had a major influence

on me. He was the first author I encountered who took something as dry as the X/Motif programming language and explained it in a book that I enjoy cracking open on a regular basis. I hope I've come close to making *The Visual Tcl Handbook* as enjoyable to read.

Why Visual Tcl?

The success of the Tcl language is largely due to the power of its X window development extension called the *Tk toolkit*. Tk is wonderfully crafted and highly diversified in its features, capabilities, and adaptability. Tcl/Tk is one of the languages I love to work with. So, with the availability of Tk, why should you and I be interested in Visual Tcl?

A complete UNIX product

Visual Tcl is a fully supported product, highly focused on leveraging the capabilities of the UNIX server environment. The base language supports a satisfyingly full range of UNIX access commands that control processes and manipulate the UNIX file system. Standard commands supporting enhanced debugging capabilities and built-in TCP/IP access have been added to make it a valid development option for conducting a wide range of projects.

Visual Tcl is fully supported by SCO as part of its cross-platform SCO Premier Motif product, as well as its SCO OpenServer Release 5 operating system. In fact, Visual Tcl is the core implementation language of SCO's graphical and character-oriented systems management environment. Visual Tcl is the only graphical widget-rendering Tcl-based scripting language that ships fully supported and exposed in a mainstream UNIX operating system. Soon, SCO will ship Visual Tcl as the native GUI scripting language for UnixWare.

Designed for the system administrator profile

The Visual Tcl language was expressly designed to be a powerful, easy-to-use GUI development tool for the system administrator "in the trenches," as well as for the developers of graphical systems administration solutions. The widget-building commands of Visual Tcl support default behavior that requires the shortest of learning curves. By taking advantage of standard Motif dialogs, in many cases only one command is required to build, for instance, a file selection box. Conversely, the large number of command options leaves lots of room for customization and greater control over design issues.

Based on the standard OSF/Motif language

Although Tk has become more compliant with the Motif look and feel, it's still very different under the covers. The Visual Tcl display server is built on top of the OSF/Motif library. The Visual Tcl language leverages OSF/Motif conventions and semantics, such as the control of Motif widget resources. The Visual Tcl language is designed to present a single layer of control that insulates the developer from the details of the underlying X library.

For Motif developers, Visual Tcl is ideal for complementing their larger C/Motif applications with the "glue" of Visual Tcl scripts. The Motif developer can apply the same geometry management principles to both Visual Tcl scripts and compiled Motif applications.

Standard Motif Widgets

One of SCO's goals for Visual Tcl is to open up the display server API so that third-party widgets can be added. Supporting the Motif API will make it possible to extend Visual Tcl with the large base of commercially available widget sets.

The use of standard Motif widgets reduces programming effort. Instead of having to craft commonly needed complex dialogs from scratch, novice developers can gain a consistent look and feel by taking advantage of standard Motif dialogs.

Policy and widgets for rapid development

By building on top of a moderate amount of built-in policy in the form of default behavior and the leveraging of OSF/Motif widgets, Visual Tcl supports a level of rapid development that is faster than developing with Tk. There are extensions to Tk that provide "mega-widgets," such as file selection boxes; however, as these are non-Tk features, the burden is on you, the developer, to link these extensions with your development system. Also, if control at the X11 level is key to your development project, Tk may be the appropriate choice.

The Drawn List widget for high-volume data representation

There are a number of widgets that SCO added to the base OSF Motif 1.2 widget set. In addition to combo box and spin button widgets, SCO added the "Drawn List" widget that features the ability to incorporate pixmaps in a column-based list widget, similar to the hierarchical file manager widget in Microsoft's Windows 3.1.

The Drawn List widget gives Visual Tcl the ability to represent a tremendous amount of data in a modest amount of graphical real estate. Arranged hierarchically, the user can view data by expanding or collapsing list items.

An architecture for multiple GUI development...

As you will discover in the Chapter 3, *Run-Time Environment*, Visual Tcl incorporates a client-server architecture, separating the Motif-rendering display server from the interpreter. There are a number of benefits to this, including enhanced performance and support for "GUI independence." In fact, the SCO OpenServer version of Visual Tcl supports two types of user interface "look and feel," namely, X-based Motif and a curses-based character version of Motif called *Charm*. Support for a Microsoft Windows display server is key to SCO's future systems management plans as well.

... and thin client Windows development

SCO is planning to release a Microsoft Windows-based Visual Tcl interpreter in order to support management of SCO server products from the Windows environment. Visual Tcl's client-server architecture will be linked over the network, supporting Windows-based graphical display servers driven by Visual Tcl scripts executing on UNIX servers. This design strategy supports the growing trend toward "thin client" application design, reducing the "application footprint" on desktop environments by shifting the functional part of the application to the remote server. This topic is discussed in more detail in Chapter 1, *Modern Scripting*.

Visual Tcl was not designed to compete with Tk. Instead, it was designed to put the benefits of Motif development in the hands of people who want to take advantage of GUI development but who, until now, have not been able to handle the cost and time-consuming impact of learning GUI development.

If you are a Tcl novice or Tk expert

The Visual Tcl Handbook focuses primarily on the TclX and Vt extensions provided in Visual Tcl. This is done somewhat at the expense of going into extensive detail about the base Tcl language itself. Most of the fundamental Tcl topics are addressed, so that if you are new to Tcl, you will do just fine when you start some serious development with the overall language. If you want to learn more about the fine points of the base Tcl language, you will probably want to consider acquiring one or two books that focus more painstakingly on the base Tcl language itself, such as Brent Welch's *Introduction to Tcl/Tk Programming*. Keep in mind that the entire Tcl language, version 7.3, is documented in the *Command Pages* section of this book.

If you are already a Tk/Tcl programmer and have not worked with the extensions of the TclX library, this book provides an introduction to many of the key features.

Organization

The organization of this book is based on major sections that focus on background and architecture, Tcl and TclX basics, GUI concepts, programming with Visual Tcl, and, finally, the entire Visual Tcl language in man page format. Here are some details about each part and the chapters they contain.

Part I: Introductions

These three chapters address the role of scripting today, introducing Visual Tcl. Its architecture and Tcl components are reviewed in detail. Finally, a step-by-step exercise that builds a "graphical who" application is provided to give you a flavor of Visual Tcl programming before diving into the details.

Part II: Essential Tcl

Essential Tcl covers the broad spectrum on non-GUI building commands contributed by the base Tcl language and the TclX command set that gives Visual Tcl its UNIX server flavor. Special attention is given to the parsing principles and syntax rules of the Tcl interpreter, intended to help the reader avoid the typical pitfalls of those new to Tcl development.

Part III: Visual Tcl

After a brief overview of OSF/Motif, this section covers Visual Tcl concepts such as how callback procedures receive user input information from the graphical display server. The five Visual Tcl option classes are discussed as major Vt commands are reviewed, including those for building custom dialogs, graphical hierarchical lists, and pulldown menus. A Rolodex application is described and used to illustrate common concepts of Visual Tcl programming. The complete Rolodex application is made available on the CD-ROM. This section is wrapped up with a discussion of how to apply a design strategy to the development of interfaces for the configuration of your application.

Part IV: Command Pages

This part is divided into three sections, providing man pages for the nongraphical Tcl and TclX commands, the graphical Vt commands, and the Vx extensions, which add another level of ease of use to the Visual Tcl language.

Conventions

Code Fragments

There are many code fragments in this book, many of which are contained in the CD-ROM at the back of this book.

```
# Code fragments will look like this:
set bookTitle "The Visual Tcl Handbook"
```

Visual Tcl commands that appear in code fragments or normal paragraph text will be identified with the typeface Courier bold, such as the variable-assigning **set** command above.

Output

Text generated by the execution of a Visual Tcl command is prefaced with the ➨ symbol, followed by the output text in Courier fixed font. In the book, we make extensive use of the **echo** command to write the contents of a variable to `stdout`. For Tcl programmers, this is a TclX command that is roughly equivalent to the **puts** `stdout` command. The

statement below combines the **echo** command with the string stored in the variable `bookTitle` from the statement above.

```
echo "This book is called $bookTitle"
```

results in the following output.

```
➡ This book is called The Visual Tcl Handbook
```

Tasks

Tasks are identified like this. Depending on the context, they are either suggested exercises for the reader or attempts to illustrate a concept or use of a command by following a suggested task.

TASK

Hints and tips boxes

> **Hint, tips, and clarification boxes ...**
>
> ... are bordered boxes that look like this. Their purpose is to point out some subtle uses or characteristics of Visual Tcl, to pass along neat tricks, or to highlight uses of particularly useful commands. The font used here is Helvetica-Narrow, ideal for squeezing a lot of information into a tiny little box.

Font conventions

• **bold Courier**	Visual Tcl commands
• *italicized Courier*	Variables or arguments passed to Visual Tcl commands
• plain Courier	Procedures and callbacks
• plain Times	Normal book text
• Helvetica-Narrow	Hints and tips boxes

Command syntax

The ***conventions*** listed in Table 1 are used to identify required and optional arguments to Visual Tcl commands. We have attempted to use self-descriptive argument names wherever possible (such as *fileName* to indicate that the argument requires the name of a file).

Table 1 Conventions for specifying command arguments

★	Indicates that the command is contributed by the TclX library. This is provided for those who want to write portable code for environments where Visual Tcl and TclX are not available.
varName	A user-defined variable name for pass-by-reference. Applies to any variable of the form *<object>Name*. Examples of other self-describing variable names are *fileName*, *listName*, and *arrayName*.
var	A generic name for a user-defined variable.
arg	A catch-all reference to an argument.
body	A Tcl script of one or more Tcl commands.
expr	An expression that combines values with one or more operators, such as `$varName == "done" or "$amount > 1.00"`
test	An expression used to test a condition, as in a flow of control command.
list	A list of items, such as `{apples oranges peaches olallieberries}` When the command is invoked, *list* may also be represented by a variable where $ substitution is performed, such as **llength** `$myList`
[*arg*]	Square brackets indicate that the argument is optional. *arg* may be either a user-created value (italicized) or a predefined command option (nonitalicized)
string	A user-provided string in quotes, such as *"The UNIX System"*. May be a variable, such as *$title*, containing a quoted string.

Command arguments that are italicized imply that the user determines the argument's value. Arguments like *test* and *body* will always be italicized, since the actual argument is created by the user. Nonitalicized arguments imply that a keyword known to the command is to be provided as an option, such as `compare` in the following command:

string compare *string1 string2*

UNIX-isms and other details

All of the examples used throughout the book were created with SCO OpenServer Release 5 version of SCO Visual Tcl 1.0. Most of the examples were tested on the Solaris port of SCO Visual Tcl to verify their portability. The Rolodex example that is used to

illustrate the use of most of the Visual Tcl graphical Vt commands has been written to be highly portable. The examples scattered throughout the book that utilize UNIX system commands are based on SCO OpenServer Release 5 and may, therefore, require "tweaking" to run on your favorite UNIX platform.

Visual Tcl on CD-ROM

SCO Visual Tcl 1.0 is a core component of the SCO OpenServer Release 5 family of operating systems. If you own or have access to these servers, then you already have access to Visual Tcl. In fact, you also have access to many, many scripts that were written in Visual Tcl as part of the SCO OpenServer native systems management environment called *SCOadmin*. Many of the SCOadmin application scripts are located in

```
/opt/K/SCO/Unix/5.0.0Cl/sa
```

SCO also provides a kit of SCOadmin tools and demonstration Visual Tcl scripting that enable you to write applications even faster than with raw Visual Tcl. It is downloadable via anonymous ftp from

```
ftp.sco.com:/TLS/tls575.custom    (tools and doc)
ftp.sco.com:/TLS/tls575.ltr       (cover letter, install info)
```

and should be useful for you even if you are not an SCO OpenServer customer. This kit has not been ported to non-SCO platforms, so treat it simply as a learning tool.

The CD-ROM enclosed in the back of this book contains the complete binary ports of SCO Visual Tcl 1.0 to ten UNIX platforms, including SCO Open Desktop. SCO requires that you purchase SCO Premier Motif in order to enjoy full support of Visual Tcl. SCO Visual Tcl 1.0 for the following platforms is provided on the CD-ROM.

- ✔ SunSoft **SunOS** 4.1.2 (For SPARC)
- ✔ SunSoft **Solaris** 2.1 (For SPARC)
- ✔ SunSoft **Solaris** 2.1 (For Intel)
- ✔ Digital Unix (**OSF/1**) 3.0 (For Alpha)
- ✔ SCO **UnixWare** 2.01
- ✔ HP **HP-UX** 9.0.1
- ✔ IBM **AIX** 3.2.5
- ✔ SGI **IRIX** 5.0.2
- ✔ SCO **Open Desktop** 3.0
- ✔ Sequent **DYNIX**/ptx 4.0.3

Visual Tcl Handbook Web Page

The capability of the Web, and Prentice Hall's support. has given me the opportunity to continue the evolution of this book with *The Visual Tcl Handbook* Home Page. This page can be accessed at the URL:

```
http://www/prenhall.com/young
```

My goal is to make this home page a place where I can further explore topics that I was able to only graze over in the book, as well as offer timely information of Visual Tcl-related developments. Topics areas will include:

Hints and Tips

e.g., Development issues with Charm
Expanded examples of passing variable information to callbacks

Errata

List of errors encountered in the book.

Enhancements of book discussions

Topics that might require further explanation are covered here. These topics go beyond the simple hints and tips.

Examples

A collection of examples sent to me from Visual Tcl users.

Rolodex Demo

My hope is to evolve the book's rolodex example with enhanced functionality as well as bug fixes.

Visual Tcl News and Thoughts

Late-breaking news related to Visual Tcl will appear here, as well as any help announcements that might point to recently-identified bugs.

If there is a topic that isn't addressed in this book, be sure to check the home page. If it isn't there either, send e-mail to me at david@inforef.com and we will do what we can to add it to the page.

Credits

Heller, Dan. 1991. *Motif Programming Manual.* Sebastopol, CA: O'Reilly & Associates, Inc.

Ousterhout, John K. 1994. *Tcl and the Tk toolkit.* Reading, MA: Addison-Wesley Publishing Company.

Welch, Brent. 1995. *Practical Programming in Tcl and Tk.* Upper Saddle River, NJ: Prentice Hall PTR.

Acknowledgments

Having the opportunity to publicly thank the people who support you is the best part of writing a book. Besides introducing Tcl to SCO, Mark Diekhans gave me the support and upbeat attention I needed as he patiently explained to me the subtleties of Tcl. People along the way who gave me the vital support I needed to keep the faith were led by Mike Shelton, who was the most supportive executive I've ever known. Other SCO management folks like Ron Rasmussen and Lorie Goudie have always supported me as friend and colleague.

A very special thank you is reserved for Ralf Holighaus of NetCS Informationstechnik GmbH. Ralf inspired the entirety of Chapter 22, *Design issues for configuration scripts*. No matter how wonderful a GUI-building language may be, it isn't worth much without good advice on user interface design. Ralf's wonderful Visual Tcl presentation at SCO Forum 95 inspired me to include his observations in this book.

The folks of the SCO Cambridge engineering team, particularly Olaf von Bremen and Zibi Perlin, made Visual Tcl a multiplatform reality as a component of the SCO Premier Motif CD-ROM, much of it through their own time and perseverance. Thanks Olaf! I hope this makes your efforts even more rewarding. I owe Olaf and Zibi, and the management team of Chris Scheybeler and Michelle Fearn, a great deal for supporting someone on the other side of the world who wanted to write a book.

Thanks to Michael "Hops" Hopkirk for his technical advice and inspiring vision for the next generation of Visual Tcl; also his SCO engineering partners, Shawn McMurdo, Mary Toscano, Susan DeTar, Bob Davis, and Donna Moore, representing the Visual Tcl development team, past and present. Thanks also to Wing Eng, Hops's predecessor, for making that great midcourse redesign that turned Visual Tcl language into a truly unique solution for those who have been reticent to move into graphical development. The entire engineering team poured their hearts into a project that had them swimming against the popular current.

Thanks to other former SCO colleagues Ron Record (of Skunkware fame), Chris Ratcliffe, Dion Johnson, and Brett Matesen. I thank them for going out of their way to help me when I, and Visual Tcl, needed it.

Thanks to Mark Taub of Prentice Hall, the editor who gave me just enough rope to write this book, I appreciate his faith, confidence, and patience.

Thanks to Michael O'Brien of Go Ahead Software. Michael's satisfying experience with using Tcl in his systems management product was particularly inspiring and gave me a number of ideas for examples in the book. Michael's inspiring comment to me was that he knew that Tcl was the way to go when he saw that just about every Can I do this? question posted to comp.lang.tcl was answered with a "yes."

Thanks to the gang of seven, Jean-Pierre Radley, Jerry Heyman, Fulko Hew, Bob Stockler, John 'tms' Navarra, Bueds Marc, and Tom Podnar, for their detailed reviews of the Tcl/TclX chapters. I appreciate their patience as I slid down the learning curve.

Thanks to Joe Moss for providing his *Tcl Language Usage Questions and Answers* Web page, http://route.psg.com/tcl.html, which gave me lots of ideas for example code.

Thanks to my start-up brothers of The Information Refinery, Inc., Paul Morgan, Michael Browder, and John Marco. Imagine trying to write a book and start a new company at the same time.

Thanks to my life friend Gale Frances for supporting a friend through the long haul; and to Lynne Hughes for her inspirational e-mail and miraculous love that pulled me through that last few months. And to Janet Young, thanks for being so supportive and helping me to find the time to wrap this book up.

Finally, like all fathers who have dreams, I did this book for my daughter Amanda and hope that it will inspire her to latch onto opportunities that turn dreams into rewarding accomplishments.

Contents