Tcl/Tk

Tcl/Tk
A Developer's Guide

Third Edition

Clif Flynt

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

Acquiring Editor: Todd Green
Development Editor: Robyn Day
Project Manager: A. B. McGee
Designer: Eric DeCicco

Morgan Kaufmann is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods or professional practices, may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information or methods described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data

Flynt, Clif.
Tcl/Tk : a developer's guide / Clif Flynt. – 3rd ed.
p. cm. – (The Morgan Kaufmann series in software engineering and programming)
ISBN 978-0-12-384717-1 (pbk.)
1. Tcl (Computer program language) 2. Tk toolkit. I. Title.
QA76.73.T44F55 2012
005.2'762–dc23

2011038927

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

For information on all MK publications visit our website at *www.mkp.com*

Printed in the United States of America
12 13 14 15 16 10 9 8 7 6 5 4 3 2 1

Contents

Foreword xvii
Preface xix
Acknowledgments xxi
Introduction xxiii

CHAPTER 1 Tcl/Tk Features **1**
1.1 Tcl Overview 2
1.1.1 The Standard Tcl Distribution 3
1.1.2 Documentation 3
1.2 Tcl as a Glue Language 6
1.2.1 Tcl Scripts Compared with UNIX Shell Scripts 7
1.2.2 Tcl Scripts Compared with MS-DOS `.bat` Files 7
1.3 Tcl as a General-purpose Interpreter 8
1.3.1 Tcl/Tk Compared to Visual Basic 9
1.3.2 Tcl/Tk Compared to Perl 9
1.3.3 Tcl/Tk Compared to Python 10
1.3.4 Tcl/Tk Compared to Java 10
1.4 Tcl as an Extensible Interpreter 11
1.5 Tcl as an Embeddable Interpreter 11
1.6 Tcl as a Rapid Development Tool 12
1.7 GUI-based Programming 13
1.8 Shipping Products 13
1.9 Bottom Line 13
1.10 Problems 14

CHAPTER 2 The Mechanics of Using the Tcl and Tk Interpreters **17**
2.1 The `tclsh` and `wish` Interpreters 17
2.1.1 Starting the `tclsh` and `wish` Interpreters 17
2.1.2 Starting `tclsh` or `wish` under UNIX 18
2.1.3 Starting `tclsh` or `wish` under Microsoft Windows 19
2.1.4 Starting `tclsh` or `wish` on the Mac 20
2.1.5 Exiting `tclsh` or `wish` 22
2.2 Using `tclsh/wish` Interactively 22
2.2.1 Tclsh as a Command Shell 22
2.2.2 Tk Console (`tkcon`)—An Alternative Interactive `tclsh/wish` Shell 23
2.2.3 Evaluating Scripts Interactively 23

2.3 Evaluating Tcl Script Files 24
2.3.1 The Tcl Script File 24
2.3.2 Evaluating Tcl Script Files 25
2.3.3 Evaluating a Tcl Script File under UNIX 26
2.3.4 Evaluating a Tcl Script File under Microsoft Windows 27
2.3.5 Evaluating a Tcl Script on the Mac 32
2.4 Bottom Line 33
2.5 Problems 33

CHAPTER 3 Introduction to the Tcl Language **35**
3.1 Overview of the Basics 36
3.1.1 Syntax 36
3.1.2 Grouping Words 37
3.1.3 Comments 38
3.1.4 Data Representation 38
3.1.5 Command Results 39
3.1.6 Errors 40
3.2 Command Evaluation and Substitutions 40
3.2.1 Substitution 40
3.2.2 Controlling Substitutions with Quotes, Curly Braces, and the Backslash 41
3.2.3 Steps in Command Evaluation 43
3.3 Data Types 44
3.3.1 Assigning Values to Variables 44
3.3.2 Strings 45
3.3.3 String Processing Commands 47
3.3.4 Lists 54
3.3.5 List Processing Commands 55
3.3.6 Dictionaries 61
3.3.7 Associative Arrays 65
3.3.8 Associative Array Commands 65
3.3.9 Binary Data 67
3.3.10 Handles 70
3.4 Arithmetic and Boolean Operations 70
3.4.1 Math Operations 70
3.4.2 Conditionals 74
3.4.3 Looping 78
3.4.4 Exception Handling in Tcl 80
3.5 Modularization 83
3.5.1 Procedures 84

3.5.2 Loading Code from a Script File 84
3.5.3 Examining the State of the Tcl Interpreter 85
3.6 Bottom Line 86
3.7 Problems 88

CHAPTER 4 Navigating the File System, Basic I/O and Sockets 91
4.1 Navigating the File System 91
4.1.1 Constructing File Paths 93
4.2 Properties of File System Items 96
4.3 Removing Files 98
4.4 Input/Output in Tcl 99
4.4.1 Output 99
4.4.2 Input 100
4.4.3 Creating a Channel 101
4.4.4 Closing Channels 104
4.5 Sockets 104
4.5.1 Using a Client Socket 105
4.5.2 Controlling Data Flow 107
4.5.3 Server Sockets 109
4.6 Bottom Line 112
4.7 Problems 113

CHAPTER 5 Using Strings and Lists 117
5.1 Converting a String into a List 117
5.2 Examining the List with a `for` Loop 118
5.3 Using the `foreach` Command 121
5.4 Using `string match` Instead of `string first` 121
5.5 Using `lsearch` 122
5.6 The `regexp` Command 124
5.6.1 Regular Expression Matching Rules 124
5.6.2 Examples of Regular Expressions 126
5.6.3 Advanced and Extended Regular Expression Rules 126
5.6.4 Back to the Searching URLs 131
5.7 Creating a Procedure 133
5.7.1 The `proc` Command 133
5.7.2 A `findUrl` Procedure 134
5.7.3 Variable Scope 135
5.7.4 Global Information Variables 136
5.8 Making a Script 137
5.8.1 The Executable Script 137

5.9 Speed 139
5.9.1 Comparison of Execution Speeds (Linux P4 @ 1.6 GHz) 140
5.10 Bottom Line 140
5.11 Problems 141

CHAPTER 6 Complex Data Structures with Lists, Arrays and Dicts **145**
6.1 Using the Tcl List 146
6.1.1 Manipulating Ordered Data with Lists 146
6.1.2 Manipulating Data with Keyed Lists 148
6.2 Using the Dict 151
6.2.1 Grouping Related Values 152
6.3 Using the Associative Array 156
6.4 Trees in Tcl 157
6.5 Tcl and SQL 159
6.5.1 SQL Basics 159
6.5.2 Using `tdbc` 162
6.5.3 Using Referenced Tables 167
6.6 Performance 175
6.7 Bottom Line 177
6.8 Problems 178

CHAPTER 7 Procedure Techniques **181**
7.1 Arguments to Procedures 181
7.1.1 Variable Number of Arguments to a Procedure 182
7.1.2 Default Values for Procedure Arguments 182
7.2 Renaming or Deleting Commands 184
7.3 Getting Information about Procedures 185
7.4 Substitution and Evaluation of Strings 187
7.4.1 Performing Variable Substitution on a String 187
7.4.2 Evaluating a String as a Tcl Command 188
7.5 Working with Global and Local Scopes 190
7.5.1 Global and Local Scope 191
7.6 Making a Tcl Object 197
7.6.1 An Object Example 197
7.7 Bottom Line 200
7.8 Problems 201

CHAPTER 8 Namespaces, Packages and Modules **205**
8.1 Namespaces and Scoping Rules 205
8.1.1 Namespace Scope 206

8.1.2 Namespace Naming Rules 206
8.1.3 Accessing Namespace Entities 208
8.1.4 Why Use Namespaces? 208
8.1.5 The `namespace` and `variable` Commands 208
8.1.6 Creating and Populating a Namespace 212
8.1.7 Namespace Nesting 215
8.1.8 Namespace Ensembles 222
8.2 Packages 225
8.2.1 How Packages Work 226
8.2.2 Internal Details: Files and Variables Used with Packages 226
8.2.3 Package Commands 227
8.2.4 Version Numbers 228
8.2.5 Package Cookbook 229
8.3 Tcl Modules 230
8.4 Namespaces and Packages 231
8.5 Hanoi with a Stack Namespace and Package 234
8.6 Conventions and Caveats 236
8.7 Bottom Line 236
8.8 Problems 237

CHAPTER 9 Basic Object-Oriented Programming in Tcl 241
9.1 Creating a TclOO Class and Object 242
9.1.1 Constructor and Destructor 245
9.1.2 Methods 247
9.1.3 Inheritance 251
9.1.4 Filters 260
9.2 Bottom Line 263
9.3 Problems 264

CHAPTER 10 Advanced Object-Oriented Programming in Tcl 267
10.1 Modifying Classes and Objects 268
10.1.1 Modifying Classes 268
10.1.2 Modifying Inheritance 275
10.1.3 Modifying Class, Constructor, Variables and Destructor 277
10.1.4 Static Methods and Variables I 278
10.2 Modifying Objects 280
10.2.1 Changing an Object's Class 280
10.2.2 Defining Per-object Mixins 283
10.2.3 Adding a Method to an Object 284

10.3 Examining Classes and Objects 284
10.3.1 Evaluation of Chains 285
10.3.2 Examining Methods 292
10.3.3 Examining Inheritance 298
10.3.4 Getting a List of Base Classes 299
10.4 Examining Objects 301
10.5 Using TclOO with Callbacks 305
10.6 Adding New Functionality to TclOO 308
10.6.1 Static Variables II 308
10.6.2 Static Methods II 310
10.6.3 Aggregated Objects That Modify the Possessor 312
10.6.4 Objects That Grow and Change 315
10.7 Bottom Line 317
10.8 Problems 318

CHAPTER 11 Introduction to Tk Graphics 321
11.1 Creating a Widget 322
11.2 Conventions 323
11.2.1 Widget Naming Conventions 323
11.2.2 Color Naming Conventions 323
11.2.3 Dimension Conventions 323
11.3 Common Options 324
11.4 Determining and Setting Options 325
11.5 The Basic Widgets 327
11.6 Introducing Widgets: `label`, `button`, and `entry` 328
11.6.1 The `label` Widget 328
11.6.2 The `button` Widget 329
11.6.3 The `entry` Widget 330
11.6.4 Using Namespaces or TclOO with Widgets 333
11.7 Application Layout: Geometry Managers and Container Widgets 336
11.7.1 Container Widgets: `frame`, `labelframe`, `panedwindow` 336
11.7.2 Widget Layout: `place`, `pack`, and `grid` 344
11.8 Selection Widgets: `radiobutton`, `checkbutton`, `menu`, and `listbox` 352
11.8.1 `radiobutton` and `checkbutton` 353
11.8.2 Pull-down Menus: `menu`, `menubutton`, and Menubars 356
11.8.3 Selection Widgets: `listbox` 367
11.9 Scrollbar 372
11.9.1 The Basic `scrollbar` 372
11.9.2 `scrollbar` Details 373
11.9.3 Intercepting `scrollbar` Commands 376

11.10 The `scale` Widget 379
11.11 New Windows 381
11.12 Interacting with the Event Loop 382
11.13 Scheduling the Future: `after` 383
11.13.1 Canceling the Future 385
11.14 Bottom Line 386
11.15 Problems 388

CHAPTER 12 Using the `canvas` Widget **391**
12.1 Overview of the `canvas` Widget 391
12.1.1 Identifiers and Tags 391
12.1.2 Coordinates 392
12.1.3 Binding 392
12.2 Creating a `canvas` Widget 392
12.3 Creating Displayable `canvas` Items 394
12.3.1 The Line Item 394
12.3.2 The Arc Item 395
12.3.3 The Rectangle Item 395
12.3.4 The Oval Item 395
12.3.5 The Polygon Item 395
12.3.6 The Text Item 396
12.3.7 The Bitmap Item 396
12.3.8 The Image Item 397
12.3.9 An Example 397
12.4 More `canvas` Widget Subcommands 398
12.4.1 Modifying an Item 398
12.4.2 Changing the Display Coordinates of an Item 399
12.4.3 Moving an Item 401
12.4.4 Finding Items, and Raising and Lowering Items 403
12.4.5 Fonts and Text Items 407
12.4.6 Using a Canvas Larger than the View 410
12.5 The `bind` and `focus` Commands 412
12.5.1 The `bind` Command 412
12.5.2 The `canvas` Widget `bind` Subcommand 415
12.5.3 Focus 418
12.6 Creating a Widget 419
12.7 A Help Balloon: Interacting with the Window Manager 427
12.8 The `image` Object 436
12.8.1 The `image` Command 437
12.8.2 Bitmap Images 438

12.8.3 Photo Images 439
12.8.4 Revisiting the delayButton Widget 442
12.9 Bottom Line 447
12.10 Problems 448

CHAPTER 13 The text Widget and htmllib 451
13.1 Overview of the text Widget 451
13.1.1 Text Location in the text Widget 452
13.1.2 Tag Overview 454
13.1.3 Mark Overview 454
13.1.4 Image Overview 455
13.1.5 Window Overview 455
13.2 Creating a text Widget 455
13.3 Text Widget Subcommands 457
13.3.1 Inserting and Deleting Text 458
13.3.2 Searching Text 460
13.3.3 The mark Subcommands 463
13.3.4 Tags 466
13.3.5 Inserting Images and Widgets into a text Widget 474
13.4 HTML Display Package 478
13.4.1 Displaying HTML Text 478
13.4.2 Using html_library Callbacks: Loading Images and Hypertext Links 480
13.4.3 Interactive Help with the text Widget and htmllib 486
13.5 Bottom Line 490
13.6 Problems 492

CHAPTER 14 Tk Megawidgets 495
14.1 Standard Dialog Widgets 495
14.1.1 tk_optionMenu 496
14.1.2 tk_chooseColor 497
14.1.3 tk_getOpenFile 498
14.1.4 tk_getSaveFile 500
14.1.5 tk_messageBox 501
14.1.6 tk_dialog 502
14.1.7 tk_popup 503
14.2 Megawidget Building Philosophy 506
14.2.1 Display in Application Window or Main Display? 506
14.2.2 Modal versus Modeless Operation 507

14.2.3 Widget Access Conventions 507
14.2.4 Widget Frames 507
14.2.5 Configuration 507
14.2.6 Access to Subwidgets 508
14.2.7 Following Tk Conventions 509
14.3 Functionality That Makes Megawidgets Possible 509
14.3.1 The `rename` Command 509
14.3.2 The `option` Command 510
14.3.3 The `-class` Option 511
14.4 Building a Megawidget 511
14.5 A Scrolling `listbox` Megawidget 512
14.5.1 Scrolled listbox Description 512
14.5.2 Using the `scrolledLB` 514
14.5.3 Implementing the Scrollable `listbox` 515
14.5.4 The `scrolledLB` Code 517
14.6 Namespaces and Tk Widgets 523
14.6.1 Creating a Multiple Language Megawidget 524
14.7 Incorporating a Megawidget into a Larger Megawidget 533
14.8 Making a Modal Megawidget: The `grab` and `tkwait` Commands 542
14.8.1 The `grab` Command 543
14.8.2 The `tkwait` Command 544
14.8.3 The Modal Widget Code 545
14.9 Automating Megawidget Construction 550
14.9.1 Building Namespace Megawidgets Summary 550
14.10 Building Megawidgets with TclOO 550
14.10.1 Using a Wrapper Proc for TclOO-Based Widgets 551
14.10.2 Using a Class Name as a Widget Type 557
14.10.3 Callbacks into TclOO Widgets 560
14.10.4 Combining TclOO Compound Widgets 561
14.10.5 Adding New Functionality to a TclOO Widget 564
14.11 Bottom Line 567
14.12 Problems 568

CHAPTER 15 Extending Tcl 571
15.1 Functional View of a Tcl Extension 572
15.1.1 Overview 572
15.1.2 Initialize Any Persistent Data Structures 572
15.1.3 Register New Commands with the Interpreter 573
15.1.4 Accept Data from Tcl Interpreter 574
15.1.5 Returning Results 578

15.1.6 Returning Status to the Script 581
15.1.7 Dealing with Persistent Data 582
15.2 Building an Extension 586
15.2.1 Structural Overview of an Extension 586
15.2.2 Naming Conventions 586
15.3 An Example 589
15.3.1 `demoInt.h` 591
15.3.2 `demoInit.c` 593
15.3.3 `demoCmd.c` 595
15.3.4 `demoDemo.c` 602
15.4 Complex Data 616
15.5 Embedding the Tcl Interpreter 626
15.6 Building Tcl and Tk from Sources 633
15.7 Bottom Line 634
15.8 Problems 637

CHAPTER 16 Applications with Multiple Environments 639
16.1 Event Loop 640
16.1.1 Using `fileevent` 640
16.1.2 Using `trace` 644
16.1.3 Using `after` 646
16.1.4 Using `bind` 647
16.1.5 Tcl Interpreters 648
16.2 Threads 652
16.3 Embedded Tcl Interp and Threading 657
16.4 Bottom Line 662
16.5 Problems 663

CHAPTER 17 Extensions and Packages 665
17.1 `[incr Tcl]` 667
17.2 `expect` 670
17.3 `TclX` 675
17.4 `TDBC` 679
17.5 `Rivet` 686
17.6 `BWidgets` 696
17.7 Graphics Extensions: `Img` 702
17.8 Bottom Line 702

CHAPTER 18 Programming Tools 703
18.1 Code Formatter 705
18.1.1 `frink` 705

18.2 Code Checkers 709
18.2.1 `tclCheck` 709
18.2.2 `nagelfar` 710
18.3 Debuggers 714
18.3.1 `debug` 714
18.3.2 Graphic Debuggers 717
18.4 Exercising and Regression Testing 717
18.4.1 `TkTest` 717
18.5 Packaging Tools 721
18.5.1 `freewrap` 722
18.5.2 Starkit and Starpack 722
18.6 Tcl Extension Generators 724
18.6.1 `SWIG` 724
18.6.2 CriTcl 727
18.7 Integrated Development Environments 728
18.7.1 KomodoEdit 729
18.7.2 Komodo 730
18.7.3 MyrmecoX 731
18.8 Bottom Line 732

CHAPTER 19 Tips and Techniques 733
19.1 Debugging Techniques 733
19.1.1 Reading the Error Messages 733
19.1.2 Instrumenting Code to Generate Log Files 735
19.1.3 Run Script in Interactive Mode 739
19.1.4 Use `puts` to Print the Value of Variables or Lines to Be Evaluated 741
19.1.5 Extract Portions of Script for Unit Testing 743
19.1.6 Attach a `tkcon` Session to Application 743
19.1.7 Create a Console Window under Windows 744
19.1.8 Create a Command Window to Interact with Your Application 744
19.1.9 Use a Second `wish` Interpreter for Remote Debugging 747
19.2 Tcl as a Glue Language: The `exec` Command 748
19.2.1 Creating a G-zipped `tar` Archive under UNIX 749
19.2.2 Creating a Zip Archive under Windows 750
19.3 Common Mistakes 751
19.3.1 Problems Using the `exec` Command 751
19.3.2 Calculating the Time: Numbers in Tcl 752
19.3.3 `set`, `lset`, `lappend`, `append`, and `incr` Are the Only Tcl Commands That Modify an Argument 753
19.3.4 The `incr` Command Works Only with Integers 753

19.3.5 The `upvar` Command Takes a Name, Not a Value 753
19.3.6 Changes to Widgets Do Not Appear Until the Event Loop Is Processed 754
19.3.7 Be Aware of Possible `%` Sign Reduction 754
19.4 Coding Tips and Techniques 754
19.4.1 Use the Interpreter to Parse Input 754
19.4.2 Handling Platform-Specific Code 758
19.5 Optimization 759
19.6 Techniques for Improving Performance 762
19.7 Bottom Line 767

Index **769**

Foreword

It is a pleasure to introduce you to the first revision of Clif's opus on the popular Tool Command Language (Tcl). Tcl has become the Swiss Army knife for the busy programmer—you keep it within easy reach at all times simply because it has lots of uses in many different environments.

Clif's book is both a pleasure to read and a useful reference. Clif is a strong advocate for Tcl, explaining the many different scenarios where Tcl shines and the myriad ways the same facilities can be used over and over again.

In addition to the breadth that Clif brings to the subject, he also knows the details—and, to his credit, he doesn't overwhelm you with them. Instead, each chapter starts with the basics and builds to the fun stuff. There are even self-help questions at the end of each chapter to help you review the lessons Clif has taught.

Whether you're using Clif's book as an introduction to Tcl or as a comprehensive reference, I'm sure you'll find it a necessary addition to your professional toolkit.

Marshall T. Rose
Principal, Dover Beach Consulting, Inc.

Preface

TCL/TK: GUI PROGRAMMING IN A GOOEY WORLD

Alan Watts, the Episcopal priest who popularized Buddhist thought in 1960s California, once wrote that philosophical thinkers can be divided into two basic types, which he called "prickly" and "gooey." The prickly people, by his definition, "are tough-minded, rigorous, and precise, and like to stress differences and divisions between things. They prefer particles to waves, and discontinuity to continuity. The gooey people are tender-minded romanticists who love wide generalizations and grand syntheses. … Waves suit them much better than particles as the ultimate constituents of matter, and discontinuities jar their teeth like a compressed-air drill."[1]

Watts chose the terms *prickly* and *gooey* carefully, seeking to avoid making one term positive and the other pejorative, and he offered delightful caricatures of the strengths and weaknesses of each type of person. In his view, a constant tension between the two perspectives is healthy in promoting progress toward a more complete understanding of reality.

Western science has long made a similar distinction between two complementary approaches, theoretical and empirical. Prickly theoreticians seek to understand the world through the abstractions of thought, whereas gooey empiricists return ceaselessly to the real world for the ever-more-refined data that methodical experimentation can yield. The complementarity of the two approaches is widely recognized by scientists themselves. Although most scientists would place themselves squarely in either the theoretical or empirical camp, very few would go so far as to argue that the other approach is without merit. A constant dialectic between empiricism and theory is generally seen to promote the integrity and health of scientific inquiry.

More recently, the advent of computer programming has brought with it a new round in the prickly–gooey dialectic. By temperament, there seem to be two types of programmers, which I will call the planners and the doers. Although good programmers will of course plan their software before building it (and will actually build it after planning it), some programmers (the prickly planners) see planning and designing as the primary activity, after which system building is relatively automatic. Others (the gooey doers) perceive system design as a necessary and important preparatory stage before the real work of system building.

Both planners and doers can be excellent programmers, and the best of them excel at both design and execution. However, they may be categorized as planners or doers by their basic worldview, or primary orientation. They show different patterns of strengths and weaknesses, and they can with skilled management play complementary roles in large project teams. Still, certain tasks cry out for one type of programmer or the other. The design of software systems to control nuclear weapons, for example, is a task that demands exhaustive planning because the risk of unanticipated errors is too high. The design of a throwaway program that needs to be run just once to calculate the answer to a single question, on the other hand, should probably be built by the fastest methodology possible, robustness

[1] Watts, Alan, *The Book: On the Taboo Against Knowing Who You Are*, NY: Vintage Books, 1966, 1989.

be damned. Between these extremes, of course, lie nearly all real-world programming projects, which is why both approaches are useful. They are not, however, always equally well appreciated.

The modern managerial mentality thrives on careful organization, planning, and documentation. It should be no surprise that most managers prefer the planners to the doers; if they were programmers, they would probably be planners themselves. Since they are not programmers, they may easily fail to recognize the value of the skills doers bring to the table. Programmers, however, are less likely to make this mistake. In the meager (but growing) literature by and for serious programmers, the role of doers is more generally recognized. Indeed, planning-oriented programmers often still perceive themselves as the underdogs, and programmer humor shows a deeply rooted skepticism regarding overplanned projects.

Historically, the dialectic between planners and doers has been a driving force in the evolution of programming languages. As the planners moved from FORTRAN to Pascal, C, C++, and Java, the doers have moved from COBOL to Lisp, Basic, Perl, and Tcl. Each new generation of language reflected the innovations of both planners and doers in the previous generation, but each made most of its own innovations in a single realm, either planning or doing.

Historically, it has been surprisingly difficult to make money by inventing even the most wildly successful programming languages. Success is typically highly indirect. For example, being the birthplace of C is certainly a claim Bell Labs is proud to make, but it is difficult to argue that the language was spectacularly meaningful for AT&T (or Lucent's) bottom line. ParcPlace has perhaps come the closest to a genuine language-invention success story, but it is clearly no Microsoft. Imagine, then, the dilemma in which Sun Microsystems found itself, circa 1994, when its research labs were polishing up two new languages, one prickly and one gooey, and each a plausible contender for leadership in the next generation of programming languages. Could any corporation really afford two such high-status, low-profit "success" stories?

Sun's decision to promote Java more vigorously than Tcl is difficult to argue with. Because most decision makers are managers, and most managers are planners, Java has always been an easier product to sell. Sun's continued and parallel support for Tcl was correct, even noble, and probably wiser than one could expect from the average large corporation. Still, in the face of the Java juggernaut, the promotion of Tcl has been left, for the most part, to the gooey doers among us. Pragmatic programmers with dirt on their hands, many of us became Tcl advocates for the most practical of reasons: it helped us get our jobs done, and quickly.

One such gooey doer is the author, a veteran of countless programming languages who has fallen in love with Tcl for all the neat things it lets him do. Java advocates will often praise that language's abstractions, such as its virtual machine model. Tcl advocates such as the author prefer to grab on to the nuts and bolts and show you how quickly you can get real work done. That is what this book is intended to do, walking you all the way from the simplest basics to sophisticated and useful examples. If you are in a "doer" state of mind, you have picked up the right language and the right book. It can help you get a lot of jobs done.

Acknowledgments

You may like to think of the author sitting alone, creating a book in a few weeks. I'm afraid that this idea has as much to do with reality as the Easter Bunny and Santa Claus. The initial revision of this book took the better part of a year and a lot of help from my friends (some of whom I'd never heard of when I started the project). The second and now third editions have each taken over a year to write and introduced me to more new friends.

The first acknowledgment goes to my wife, Carol. For more than a year she's been: putting up with "I can't do any work around the house. I've got to finish this chapter," correcting the grammar on my rough drafts before I send them out for technical review, editing galleys, making sure that I ate occasionally, and keeping life (almost) sane around here. The book would not have happened without her help.

I also want to thank Ken Morton and his editorial assistants Samantha Libby and Jenn Vilaga. If they hadn't been willing to take a chance on an unproven author's ability to finish the first edition of this book, this edition would never exist.

Alex Safonov provided the technical review for the first edition. Larry Virden, and Leam Hall kept me honest in the second edition. Arnulf Wiedemann, Massimo Manghi, Richard Owlett, Brian Stretch, Virginia Gielincki, and Robert Clay provided much assistance with the third edition. Their comments greatly improved the book, and earned more thanks than I can give them.

My heartfelt and sincere thanks to the folks who read and commented on the early drafts. These folks pulled duty way above the call of friendship by reading the truly wretched first drafts. Their comments were all invaluable. My thanks to: Margaret Bumby, Clark Wierda, Terry Gliedt, Elizabeth Lehmann, Nick DiMasi, Hermann Boeken, Greg Martin, John Driestadt, Daniel Glasser, Theresa Arzadon, Lee Cave-Berry, William Roper, Phillip Johnson, Don Libes, Jeffrey Hobbs, John Stump, Laurent Demailly, Mark Diekhans, David Dougherty, Tod More, Hattie Schroeder, Michael Doyle, Sarah Daniels, Ray Johnson, Melissa Hirschl, Jan Nijtmans, Forest Rouse, Brion Sarachan, Lindsay F. Marshall, Greg Cronau and Steve Simmons.

And, finally, my thanks to John Ousterhout and his teams at UC Berkeley, Sun, Scriptics and Ajuba, and now to the Tcl Core Team and the host of maintainers. Programming in Tcl has been more productive and given me more pleasure than any tool I've used since I got my first chance to play Lunar Lander on an IBM 1130.

Introduction

Thanks for purchasing this copy of *Tcl/Tk: A Developer's Guide*. (You did buy this, didn't you?) This book will help you learn Tcl/Tk and show you how to use these tools to increase your programming productivity.

By the time you've finished reading this book, you'll have a good idea of what the strengths and weaknesses of Tcl/Tk are, how you can use the Tcl/Tk libraries and interpreters, and what constitutes an effective strategy for using Tcl/Tk.

This book is aimed at both computer professionals (from novice to expert level) and students. If you are experienced with computer programming languages, you will be able to skim through the first few chapters getting a feel for how Tcl/Tk works, and then go on to the chapters on techniques and extensions. If you are less experienced, you should read the first chapters fairly carefully to be sure you understand the Tcl/Tk syntax. Tcl has some features that may not be obvious.

If your primary goal is to learn the Tcl/Tk basics as quickly as possible, visit one of the websites listed below for a tutorial. There are a large number of Tcl/Tk tutorials available in a large number of languages.

Every language has some simple tricks and techniques that are specific to that language. Tcl/Tk is no exception. I'll discuss how to best accomplish certain tasks with Tcl/Tk: how to use the list and associative array data structures effectively, how to build modular code using Tcl's software engineering features, and what kinds of programming constructs will help you get your project from prototype to product with the minimal amount of rewriting.

Finally, there are plenty of code snippets, examples, and bits of packages to help you see how Tcl/Tk can be used. These examples are chosen to provide you with some boilerplate procedures that you can add to your programs right now, to help you get from scribbles on paper to a working program.

This should give you a good idea of what's coming up. Feel free to skip around the book. It's written to be an overview and reference, as well as a tutorial.

WHERE TO GET MORE INFORMATION

Much as I would like to cover all the aspects of Tcl/Tk in this book, it isn't possible. Had I but world enough and time, I still wouldn't have enough of either. The Tcl/Tk community and the Tcl/Tk tools are growing too quickly for a book to do more than cover the central core of the language.

You can learn more about Tcl/Tk from these web references:

- `http://www.tcl.tk/`
  - Primary tcl.tk website
  - The definitive site for Tcl/Tk information.
- `http://wiki.tcl.tk/`
  - The Tcler's Wiki.
  - A collection of brief articles and discussions of Tcl features, tricks, tips, and examples.
- `http://www.tclcommunityassociation.org`
  - Tcl Community Association homepage

  - Promotes the use and development of Tcl/Tk. They run the annual Tcl/Tk Conference in the US, administer the Tcl contribution to Google Summer of Code, host the wiki and source websites and more.
- *www.purl.org/NET/Tcl-FAQ/*
  - Tcl FAQ Launch page.
  - URLs for several Tcl/Tk FAQs and *comp.lang.tcl*
- *http://core.tcl.tk/*
  - Tcl/Tk Fossil repository
  - The official repository for Tcl/Tk distributions.
- *http://sourceforge.net/projects/tcl/*
  - Tcl/Tk Archives
  - The mirror repository for Tcl/Tk distributions.
- *http://www.activestate.html*
  - Active State.
  - Tcl/Tk maintenance, ActiveTcl, development tools, and more.
- *http://www.tcl-tk.de/*
  - Tcl/Tk information in German
- *http://www.geocities.co.jp/SiliconValley/4137/*
  - Tcl Scripting Laboratory
  - Tcl/Tk information in Japanese
- *http://www.cyut.edu.tw/~ckhung/olbook/*
  - Chao-Kuei Hung's pages
  - Tcl/Tk information in Chinese
- *http://www.noucorp.com*
  - Noumena Corporation Home Page
  - Errata and extensions for this book, updated TclTutor

For additional links and resources please visit the companion site at: *http://www.elsevierdirect.com/9780123847171*.