From: pvhp@lns62.lns.cornell.edu[1] (Peter Prymmer)
Newsgroups: comp.lang.perl.tk[2], comp.lang.perl.announce[3], comp.answers[4], news.answers[5]
Subject: comp.lang.perl.tk FAQ[6]
Followup−To: comp.lang.perl.tk[7]
Date: Sat May 31 16:48:37 1997
Approved: news−answers−request@MIT.EDU
Summary: comp.lang.perl.tk Frequently Asked Questions.
Archive−name: perl−faq/ptk−faq
Posting−Frequency: monthly
Last−modified: Date: Sat May 31 16:48:37 1997
URL: http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html[8]
Version: 1.00_07

URL (Hypertext−split): http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html[9]
URL (Plaintext): http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt[10]
Image−supplement: http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html[11]
ftp−Archive: ftp://ftp.ccd.bnl.gov/pub/ptk/ptkFAQ.txt[12]
ftp−Archive: ftp://rtfm.mit.edu/pub/usenet/perl−faq/ptk−faq/[13]
e−mail−Archive: ptkfaq@pubweb.bnl.gov[14]

Perl/Tk FAQ

pTk

The Perl/Tk extension to the Perl programming language is copyrighted by its author Nick Ing−Simmons <nik@tiuk.ti.com>[15] whose `Tk400.202/COPYING` file reads as follows:

```
Copyright (c) 1995−1996 Nick Ing−Simmons. All rights reserved.
This package is free software; you can redistribute it and/or
modify it under the same terms as Perl itself, with the exception
of the files in the pTk sub−directory which have separate terms
derived from those of the orignal Tk4.0 sources and/or Tix.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY
FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY
DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES,
```

1 mailto:pvhp@lns62.lns.cornell.edu
2 news:comp.lang.perl.tk
3 news:comp.lang.perl.announce
4 news:comp.answers
5 news:news.answers
6 ptkFAQ.html
7 news:comp.lang.perl.tk
8 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html
9 http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html
10 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt
11 http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html
12 ftp://ftp.ccd.bnl.gov/pub/ptk/ptkFAQ.txt
13 ftp://rtfm.mit.edu/pub/usenet/perl−faq/ptk−faq/
14 mailto:ptkfaq@pubweb.bnl.gov
15 mailto:nik@tiuk.ti.com

This compilation of Frequently Asked Questions & answers (FAQ) is intended to answer several of the first (and largely more basic) questions posted to the comp.lang.perl.tk[18] newsgroup and the ptk mailing list[19].

This document concerns itself with the Perl/Tk programming language (or, if you prefer, the Tk extension to Perl). Please be aware that this is not the Perl FAQ[20], this is not the Tcl FAQ[21], nor is this the Tcl/Tk FAQ[22]. Each of those other documents is a useful source of information for Perl/Tk programmers but they are completely different from this one.

This is a dynamic document and contributions, clarifications, and corrections are most welcome![23] Please send e−mail to <pvhp@lns62.lns.cornell.edu>[24] or to <pvhp@forte.com>[25]. With your help this document will get better (−:

perl/Tk FAQ

---

16 misc/license.terms
17 misc/Tix.license
18 news:comp.lang.perl.tk
19 mailto:ptk@lists.stanford.edu
20 http://www.perl.com/perl/faq/
21 http://www.smartpages.com/faqs/tcl−faq/top.html
22 http://www.smartpages.com/faqs/tcl−faq/tk/top.html
23 mailto:pvhp@lns62.lns.cornell.edu
24 mailto:pvhp@lns62.lns.cornell.edu
25 mailto:pvhp@forte.com
26 #A1
27 #A2
28 #A3
29 #A4
30 #A5
31 #A6
32 #A7
33 #A8
34 #A9
35 #A10
36 #A10.1
37 #A10.2

38 #A10.3
39 #A10.4
40 #A10.5
41 #A10.6
42 #A10.7
43 #A10.8
44 #A10.9
45 #A10.10
46 #A10.11
47 #A10.12
48 #A10.13
49 #A10.14
50 #A10.15
51 #A10.16
52 #A10.17
53 #A10.18
54 #A10.19
55 #A10
56 #A11.1
57 #A11.2
58 #A11.3
59 #A11.4
60 #A11.5
61 #A11.6
62 #A11.7
63 #A11.8
64 #A11.9
65 #A11.10
66 #A12
67 #A12.1
68 #A12.2

---

69 #A12.3
70 #A12.4
71 #A12.5
72 #A12.6
73 #A12.7
74 #A12.8
75 #A12.9
76 #A12.10
77 #A12.11
78 #A13
79 #A14
80 #A15
81 #A16
82 #A16.1
83 #A16.2
84 #A16.3
85 #A16.4
86 #A16.5
87 #A16.6
88 #A17
89 #A17.1
90 #A17.2
91 #A17.3
92 #A17.4
93 #A18
94 #A18.1
95 #A18.2
96 #A18.3
97 #A18.4
98 #A18.5
99 #A19

Return to table of contents[110]

1. What is perl/Tk?

Perl/Tk (also known as pTk or ptk) is a collection of modules and code that attempts to wed the easily configured Tk 4 widget toolkit to the powerful lexigraphic, dynamic memory, I/O, and object−oriented capabilities of Perl 5. In other words, it is an interpreted scripting language for making widgets and programs with Graphical User Interfaces (GUI). (Examples of widget programs [not necessarily written in perl/Tk] include `xterm`, `xclock`, most web−browsers, *etc.*. They are programs with "GUI" interfaces of one sort or another and are subject to the effects of your window manager.)

The current release of Perl/Tk is based on "Tk 4.0p3" the widget **T**ool**k**it originally associated with the Tcl (**T**ool **c**ommand **l**anguage) scripting language. However, Perl/Tk does not require any of the lexical features/idiosynchrocies of Tcl. Perl/Tk uses perl 5 syntax, grammar, and data structures.

The ‘‘Tk400.202’’ package is the production release of perl/Tk (corresponding to Tcl/Tk−4.0p3) and was written primarily by Nick Ing−Simmons <Nick.Ing−Simmons@tiuk.ti.com>[111] at Texas Instruments in Northampton, England, to work with the latest version of Larry Wall’s ‘‘perl’’[112]. Nick Ing−Simmons is currently busy converting the Tcl/Tk−4.1 code to perl callable code as well. An initial alpha release of the effort is available from CPAN as ‘‘Tk402.000’’.

The pTk code proper is an externally callable Tk toolkit (*i.e.* a re−write of the Tk 4.0 code that allows easier external linking & calling, especially by perl). Ptk can then be called from Perl 5 via the Tk.pm *et al* perl glue modules. Hence "*ptk*" does not necessarily refer to *Perl Tk* but could be taken to mean *portable Tk* − given a glue package to another language. The stated goal of the pTk code is to have its library usable from perl, Tcl, LISP, C++, python, *etc.*. It just so happens that present work is concentrating on perl.

Historical note: "ptk" was known as "ntk" before about 11:30 EST 4 May 1995.

100 #A20
101 #A21
102 #A22
103 #A23
104 #A24
105 #A25
106 #A26
107 #A27
108 #A28
109 #A29
110 #TOC
111 mailto:Nick.Ing−Simmons@tiuk.ti.com
112 http://www.perlc.om/perl/

The perl/Tk language is itself further extensible via the standard perl 5 module mechanism. A number of composite widget and special character extensions to the language have been written using perl modules.

Return to table of contents[113]

---

2. What is the difference between perl/Tk and Tkperl?

TkPerl was originally the name of a (now unsupported) perl 4 package that Malcolm Beattie <mbeattie@sable.ox.ac.uk>[114] at Oxford University gave to his code to wed the Tk X toolkit with Perl. (He has since referred to that package as a different "*kettle of fish*" from perl/Tk.)

Since that beginning Malcolm has also come up with a Tcl[115] module for perl 5 that has a Tcl::Tk[116] module extension. That module allows the use of Tcl within a Perl script (*i.e.* you must know both languages to get your widgets to work.) If you are interested in that package instead, see the necessary kits for Malcolm Beattie's Tcl/Tk extensions to Perl, which have been distrubuted as `Tcl-b#.tar.gz` and `TclTk-b#.tar.gz` files in the `authors/id/MICB/`[117] directory at CPAN sites (locations given in a separate question in this FAQ)[118].

The name "*tkperl*" is sometimes applied to the "perl/Tk" or "ptk" package that is the subject of this FAQ. Nick Ing–Simmons prefers "perl/Tk" as the name of the package, with "pTk" or "ptk" as contractions of that name as well as referring to something technically distinct: given the extensibility of the pTk code the "p" could also be taken to mean 'portable' or 'pure' (not to be confused with either the Helsinki University of Technology portTk[119], nor with Brian Warkentine's Rivet[120]). In this document the code of interest is either referred to as "perl/Tk", "pTk", or "ptk" though the primary focus is on perl/Tk.

Warning: do not attempt to install both perl/Tk and Tcl/Tkperl in the same perl installation. The names in the respective modules overlap. In particular the Tcl::Tk module is declared in a `Tk.pm` file – so a statement like:

```
use Tk;
```

will probably confuse your perl. If you cannot live without either module then install make & maintain separate perls for each and arrange your script writing accordingly (this will not be easy).

A more extensive comparison of the differences between the Tkperl and the perl/Tk code is given in the `Tcl-perl.pod`[121] file that is distributed with perl/Tk (see the following questions for locations).

Lastly, it should be mentioned that if you build your perl/Tk statically rather than dynamically it will make a new perl interpreter called `tkperl` (confusing isn't it? :–).

Return to table of contents[122]

---

113 #TOC1
114 mailto:mbeattie@sable.ox.ac.uk
115 http://sable.ox.ac.uk/~mbeattie/perl.html
116 http://sable.ox.ac.uk/~mbeattie/perl.html
117 http://perl.com/CPAN/authors/id/MICB/
118 #CPAN
119 http://www.cs.hut.fi/~kjk/porttk.html
120 http://www.veritas.com/~brianw/rivet/index.html
121 pod/Tcl–perl.html
122 #TOC2

3. Do I need Tcl/Tk in order to build Perl/Tk?

Short answer: No not at all. Perl/Tk is *completely* independent of Tcl/Tk.

Longer answer: In order to build Perl/Tk from source code you do need a recent version of perl, the perl/Tk source code kit, a graphical user interface library such as Xlib, a C or C++ compiler, and a make utility. In some rare cases Perl/Tk binaries are distributed for some platforms but that is more the exception than a general rule (see below).

If you will be attempting to port Perl/Tk to your platfrom then you might want to consult the document at:

`http://w4.lns.cornell.edu/~pvhp/ptk/ptkPORT.html`[123]

where each of these necessities is discussed in a bit more detail.

Return to table of contents[124]

---

4. Where is it?

Source code

Tk400.202 & Tk402.00[125] the latest production and alpha releases are available from the `modules/by-authors/Nick_Ing-Simmons/">` [126] directory on the CPAN[127]. You will need a made and installed perl (**Perl 5.004** being an excellent choice), a recent **MakeMaker** and the **Tk4*** kit. To obtain all of these (as well as several other modules that sophisticated Tk programs now rely on) visit a CPAN ftp site[128]. CPAN[129] (the Comprehensive Perl Archive Network) and what you need to get from it, is discussed in more detail in the next question.[130]

(The rest of this question remains for historical reasons, as well as to point out some non CPAN resources.)

Tk-b8: The Tk-b8 kit remains on CPAN since it was compatible with the widely distributed and installed **Perl (5.001m)**

Binaries

A pre-compiled binary distribution of Perl5.001m with Tk-b8 for Linux is available from:

```
Australia (please be patient and only try during off hours)
    ftp://syd.dit.csiro.au/pub/perl5/local/perl5.001m+Tk-b8-Linux-ELF.tar.gz
```
[131]

123 http://w4.lns.cornell.edu/~pvhp/ptk/ptkPORT.html
124 #TOC3
125 http://perl.com/CPAN/modules/by-authors/Nick_Ing-Simmons/
126 http://perl.com/CPAN/modules/by-authors/Nick_Ing-Simmons/
127 http://perl.com/CPAN
128 #CPAN
129 #CPAN
130 #CPAN
131 ftp://syd.dit.csiro.au/pub/perl5/local/perl5.001m+Tk-b8-Linux-ELF.tar.gz

It unpacks into `/usr/local`. You need to have ELF running and to have the ELF X11 libraries (**please** be patient and only try during **off hours**).

Binaries for the old Perl 5 & Tk−b6 are available for a number of UNIX platforms courtesy of Thomas Schlagel and Alan Stange of Brookhaven Lab at:

USA
: http://pubweb.bnl.gov/~ptk/[132]

Thomas and Alan have recently (winter 1995−1996) announced that they will update the Tk module version number of the many binaries they distribute.

Physical media (mostly source code)

With traffic jams on today's information superhighway more and more common it is often convenient to be able to snail mail a CD rather than suffer with `.tar.gz` files that are corrupted by network spottiness. Here is a very brief list of some folks who distribute perl (and hopefully Tk too!) on physical media. This list is not intended to be complete, nor an endorsement of any vendor (I personally do not have the time to check out any of these but have seen that some tend to be out of date by a few months with respect to CPAN[133] so please be careful). See the hypertext version of this document for hyperlinks to the following vendors:

Walnut Creek[134] Perl CD[135]
: This CD specifies a release date. $39.95

Cosmos Engineering Company[136]
: Offers Linux plus perl for sale on a 1 Gigabyte IDE hard drive for PC−like computers. $279.00 (Fall 1996)

Unix Review System Administration[137]
: A CD that contains "Perl 5.0" (and much other stuff including Tcl/Tk and Expect) for $49.95. Telephone: (800) 444−4881.

InfoMagic[138] Mother of Perl[139]
: This 2 CD set contains perl 5.001 and sells for $35.00

Ready to Run[140]
: Perl (unknown version) available for sale for many types of Unix and other operating systems.

Return to table of contents[141]

---

5. What/Where is CPAN?

"CPAN"[142] = Comprehensive Perl Archive Network[143] a worldwide collection of anonymous ftp sites for **Perl** *et al* (not to be confused with CTAN which is for TeX, nor CSPAN which rarely deals with computer software). The files of particular interest to a **perl/Tk** programmer would include:

132 http://pubweb.bnl.gov/~ptk/
133 http://www.perl.com/perl/CPAN
134 http://www.cdrom.com/
135 http://www.cdrom.com/titles/prog/perl.htm
136 mailto:cjc@earthlink.net
137 mailto:orders@mfi.com
138 http://www.infomagic.com/
139 http://www.infomagic.com/infomagic/2mperl.html
140 http://www.rtr.com/
141 #TOC4
142 http://www.perl.com/perl/CPAN/README.html
143 http://www.perl.com/perl/CPAN/CPAN.html

- The latest Tk release should be in the modules/by-authors/Nick_Ing-Simmons/[144] directory at any CPAN ftp site (listed below).
- The latest official Perl release should be in the src/ directory in a file called latest.tar.gz[145] at any CPAN ftp site (listed below).
- If you need something older or newer than the "latest supported release" try the src/5.0/[146] directory at any CPAN ftp site (listed below).

There are a number of modules available for use with Tk. Among them:

- Alan Scheinine's SelFile.pm is in the modules/by-authors/Alan_Scheinine/[147] directory at any CPAN ftp site (listed below).
- Guy Decoux's BLT_Table geometry manager is in the modules/by-authors/id/GUYDX/[148] directory at any CPAN ftp site (listed below).
- Ilya Zakharevich's eText[149] plug in replacement for the Text widget is in the modules/by-authors/id/ILYAZ/etext/[150] directory at any CPAN ftp site (listed below).
- Brent B. Powers' Tk-FileDialog and Tk-WaitBox are available from the modules/by-authors/id/BPOWERS/[151] directory at any CPAN ftp site (listed below).

Some of the fancier perl/Tk scripts (news readers and web browsers *e.g.*) make use of other perl modules/module-bundles. Among those that you ought to consider installing are:

- To run the ptknews script you will need Mail/Internet.pm from the Mailtools module kit, available from the modules/by-authors/id/GBARR/[152] directory at any CPAN ftp/http site (listed below). While there pick up the latest libnet-* module bundle too.
- The build of recent versions of perl/Tk requests that you have the URI::URL and HTML::Parse modules already installed. These modules are part of the libwww-perl-*.tar.gz kit, available from the modules/by-authors/id/GAAS/[153] at any CPAN ftp/http site (listed below).

Documentation is available from CPAN:

- Assorted documentation for **perl** is in the doc/ directory at any CPAN ftp site (listed below).
- Fairly up-to-date versions of some Perl/Tk external documentation (such as this FAQ) is in the modules/by-authors/id/PVHP/[154] directory at any CPAN ftp/http site (listed below).
  (for **Perl/Tk** official documentation check the Tk/doc/*.htm files that are built with Nick's Tk kit on your computer.)

Bringing it all together one can look at:

144 http://www.perl.com/CPAN/modules/by-authors/Nick_Ing-Simmons/
145 http://www.perl.com/CPAN/src/latest.tar.gz
146 http://www.perl.com/CPAN/src/5.0/
147 http://www.perl.com/CPAN/modules/by-authors/Alan_Scheinine/
148 http://www.perl.com/CPAN/modules/by-authors/id/GUYDX/
149 ftp://ftp.math.ohio-state.edu/pub/users/ilya/etext.html
150 http://www.perl.com/CPAN/modules/by-authors/id/ILYAZ/etext/
151 http://www.perl.com/CPAN/modules/by-authors/id/BPOWERS/
152 http://www.perl.com/CPAN/modules/by-authors/id/GBARR/
153 http://www.perl.com/CPAN/modules/by-authors/id/GAAS/
154 http://www.perl.com/CPAN/modules/by-authors/id/PVHP/

- Most things perl/Tk (with certain exceptions) are also linked to a `modules/by-module/Tk/`[155] directory at any CPAN ftp/http site (listed below).
- The long version of Tim Bunce and Andreas Koenig's module list helps you sort out things like "which kit is HTML::Parse a part of?". It is in `modules/00modlist.long.html`[156] and is also posted periodically to newsgroups.

Here are the 52 CPAN sites/directories (with dotted quads [IP numbers] given on the right for those without name−servers):

Updated: Sun Dec 8 17:12:55 EST 1996

```
Africa
  South Africa
    ftp://ftp.is.co.za/programming/perl/CPAN/[157]          196.4.160.12[158]
Asia
  Hong Kong
    ftp://ftp.hkstar.com/pub/CPAN/[159]                     202.82.7.4[160]
  Japan
    ftp://ftp.jaist.ac.jp/pub/lang/perl/CPAN/[161]          150.65.7.5[162]
    ftp://ftp.lab.kdd.co.jp/lang/perl/CPAN/[163]            192.26.91.6[164]
  South Korea
    ftp://ftp.nuri.net/pub/CPAN/[165]                       203.255.112.6[166]
  Taiwan
    ftp://dongpo.math.ncu.edu.tw/perl/CPAN/[167]            140.115.25.3[168]
Australasia
  Australia
    ftp://coombs.anu.edu.au/pub/perl/CPAN/[169]             150.203.76.2[170]
    ftp://ftp.mame.mu.oz.au/pub/perl/CPAN/[171]             128.250.209.2[172]
  New Zealand
    ftp://ftp.tekotago.ac.nz/pub/perl/CPAN/[173]            202.49.6.24[174]
Europe
  Austria
    ftp://ftp.tuwien.ac.at/pub/languages/perl/CPAN/[175]    128.130.34.160[176]
  Belgium
    ftp://ftp.kulnet.kuleuven.ac.be/pub/mirror/CPAN/[177]   134.58.127.2[178]
  Czech Republic
    ftp://sunsite.mff.cuni.cz/Languages/Perl/CPAN/[179]     194.50.23.220[180]
  Denmark
    ftp://sunsite.auc.dk/pub/languages/perl/CPAN/[181]      130.225.51.30[182]
```

155 http://www.perl.com/CPAN/modules/by−module/Tk/
156 http://www.perl.com/CPAN/modules/00modlist.long.html
157 ftp://ftp.is.co.za/programming/perl/CPAN/
158 ftp://196.4.160.12/programming/perl/CPAN/
159 ftp://ftp.hkstar.com/pub/CPAN/
160 ftp://202.82.7.4/pub/CPAN/
161 ftp://ftp.jaist.ac.jp/pub/lang/perl/CPAN/
162 ftp://150.65.7.5/pub/lang/perl/CPAN/
163 ftp://ftp.lab.kdd.co.jp/lang/perl/CPAN/
164 ftp://192.26.91.6/lang/perl/CPAN/
165 ftp://ftp.nuri.net/pub/CPAN/
166 ftp://203.255.112.6/pub/CPAN/
167 ftp://dongpo.math.ncu.edu.tw/perl/CPAN/
168 ftp://140.115.25.3/perl/CPAN/
169 ftp://coombs.anu.edu.au/pub/perl/CPAN/
170 ftp://150.203.76.2/pub/perl/CPAN/
171 ftp://ftp.mame.mu.oz.au/pub/perl/CPAN/
172 ftp://128.250.209.2/pub/perl/CPAN/
173 ftp://ftp.tekotago.ac.nz/pub/perl/CPAN/
174 ftp://202.49.6.24/pub/perl/CPAN/
175 ftp://ftp.tuwien.ac.at/pub/languages/perl/CPAN/
176 ftp://128.130.34.160/pub/languages/perl/CPAN/
177 ftp://ftp.kulnet.kuleuven.ac.be/pub/mirror/CPAN/
178 ftp://134.58.127.2/pub/mirror/CPAN/
179 ftp://sunsite.mff.cuni.cz/Languages/Perl/CPAN/
180 ftp://194.50.23.220/Languages/Perl/CPAN/
181 ftp://sunsite.auc.dk/pub/languages/perl/CPAN/
182 ftp://130.225.51.30/pub/languages/perl/CPAN/

```
Finland
  ftp://ftp.funet.fi/pub/languages/perl/CPAN/[183]        128.214.248.6[184]
France
  ftp://ftp.pasteur.fr/pub/computing/unix/perl/CPAN/[185] 157.99.64.12[186]
Germany
  ftp://ftp.leo.org/pub/comp/programming/languages/perl/CPAN/[187] 131.159.0.252[188]
  ftp://ftp.rz.ruhr-uni-bochum.de/pub/CPAN/[189]          134.147.32.42[190]
  ftp://ftp.uni-hamburg.de/pub/soft/lang/perl/CPAN/[191]  134.100.32.54[192]
Greece
  ftp://ftp.ntua.gr/pub/lang/perl/[193]
Hungary
  ftp://ftp.kfki.hu/pub/packages/perl/CPAN/[194]          148.6.0.5[195]
Italy
  ftp://cis.utovrm.it/CPAN/[196]                          160.80.22.17[197]
the Netherlands
  ftp://ftp.cs.ruu.nl/pub/PERL/CPAN/[198]                 131.211.80.17[199]
  ftp://ftp.EU.net/packages/cpan/[200]                    134.222.91.7[201]
Norway
  ftp://ftp.uit.no/pub/languages/perl/cpan/[202]          129.242.4.34[203]
Poland
  ftp://ftp.pk.edu.pl/pub/lang/perl/CPAN/[204]            149.156.132.152[205]
  ftp://sunsite.icm.edu.pl/pub/CPAN/[206]                 148.81.209.3[207]
Portugal
  ftp://ftp.ci.uminho.pt/pub/lang/perl/[208]              193.136.16.247[209]
  ftp://ftp.telepac.pt/pub/CPAN/[210]                     194.65.5.98[211]
Russia
  ftp://ftp.sai.msu.su/pub/lang/perl/CPAN/[212]           158.250.29.1[213]
Slovenia
  ftp://ftp.arnes.si/software/perl/CPAN/[214]             193.2.1.72[215]
Spain
  ftp://ftp.etse.urv.es/pub/mirror/perl/[216]             193.144.20.6[217]
  ftp://ftp.rediris.es/mirror/CPAN/[218]                  130.206.1.2[219]
Sweden
  ftp://ftp.sunet.se/pub/lang/perl/CPAN/[220]             130.238.253.4[221]
```

183 ftp://ftp.funet.fi/pub/languages/perl/CPAN/
184 ftp://128.214.248.6/pub/languages/perl/CPAN/
185 ftp://ftp.pasteur.fr/pub/computing/unix/perl/CPAN/
186 ftp://157.99.64.12/pub/computing/unix/perl/CPAN/
187 ftp://ftp.leo.org/pub/comp/programming/languages/perl/CPAN/
188 ftp://131.159.0.252/pub/comp/programming/languages/perl/CPAN/
189 ftp://ftp.rz.ruhr−uni−bochum.de/pub/CPAN/
190 ftp://134.147.32.42/pub/CPAN/
191 ftp://ftp.uni−hamburg.de/pub/soft/lang/perl/CPAN/
192 ftp://134.100.32.54/pub/soft/lang/perl/CPAN/
193 ftp://ftp.ntua.gr/pub/lang/perl/
194 ftp://ftp.kfki.hu/pub/packages/perl/CPAN/
195 ftp://148.6.0.5/pub/packages/perl/CPAN/
196 ftp://cis.utovrm.it/CPAN/
197 ftp://160.80.22.17/CPAN/
198 ftp://ftp.cs.ruu.nl/pub/PERL/CPAN/
199 ftp://131.211.80.17/pub/PERL/CPAN/
200 ftp://ftp.EU.net/packages/cpan/
201 ftp://134.222.91.7/packages/cpan/
202 ftp://ftp.uit.no/pub/languages/perl/cpan/
203 ftp://129.242.4.34/pub/languages/perl/cpan/
204 ftp://ftp.pk.edu.pl/pub/lang/perl/CPAN/
205 ftp://149.156.132.152/pub/lang/perl/CPAN/
206 ftp://sunsite.icm.edu.pl/pub/CPAN/
207 ftp://148.81.209.3/pub/CPAN/
208 ftp://ftp.ci.uminho.pt/pub/lang/perl/
209 ftp://193.136.16.247/pub/lang/perl/
210 ftp://ftp.telepac.pt/pub/CPAN/
211 ftp://194.65.5.98/pub/CPAN/
212 ftp://ftp.sai.msu.su/pub/lang/perl/CPAN/
213 ftp://158.250.29.1/pub/lang/perl/CPAN/
214 ftp://ftp.arnes.si/software/perl/CPAN/
215 ftp://193.2.1.72/software/perl/CPAN/
216 ftp://ftp.etse.urv.es/pub/mirror/perl/
217 ftp://193.144.20.6/pub/mirror/perl/
218 ftp://ftp.rediris.es/mirror/CPAN/
219 ftp://130.206.1.2/mirror/CPAN/
220 ftp://ftp.sunet.se/pub/lang/perl/CPAN/
221 ftp://130.238.253.4/pub/lang/perl/CPAN/

```
  Switzerland
    ftp://sunsite.cnlab-switch.ch/mirror/CPAN/[222]          193.5.24.1[223]
  UK
    ftp://ftp.demon.co.uk/pub/mirrors/perl/CPAN/[224]        158.152.1.44[225]
    ftp://sunsite.doc.ic.ac.uk/packages/CPAN/[226]           193.63.255.1[227]
    ftp://unix.hensa.ac.uk/mirrors/perl-CPAN/[228]           129.12.200.129[229]
North America
  Ontario
    ftp://ftp.utilis.com/public/CPAN/[230]                   207.34.209.49[231]
    ftp://enterprise.ic.gc.ca/pub/perl/CPAN/[232]            192.197.182.100[233]
  California
    ftp://ftp.digital.com/pub/plan/perl/CPAN/[234]           204.123.2.4[235]
    ftp://ftp.cdrom.com/pub/perl/CPAN/[236]                  165.113.58.253[237]
  Colorado
    ftp://ftp.cs.colorado.edu/pub/perl/CPAN/[238]            128.138.243.20[239]
  Florida
    ftp://ftp.cis.ufl.edu/pub/perl/CPAN/[240]                128.227.205.206[241]
  Illinois
    ftp://uiarchive.cso.uiuc.edu/pub/lang/perl/CPAN/[242]    128.174.5.14[243]
  Massachusetts
    ftp://ftp.iguide.com/pub/mirrors/packages/perl/CPAN/[244] 206.15.105.99[245]
  New York
    ftp://ftp.rge.com/pub/languages/perl/[246]               157.225.178.12[247]
  North Carolina
    ftp://ftp.duke.edu/pub/perl/[248]                        152.3.233.7[249]
  Oklahoma
    ftp://ftp.ou.edu/mirrors/CPAN/[250]                      129.15.2.40[251]
  Oregon
    ftp://ftp.orst.edu/pub/packages/CPAN/[252]               128.193.4.12[253]
  Texas
    ftp://ftp.sedl.org/pub/mirrors/CPAN/[254]                198.213.9.194[255]
    ftp://ftp.metronet.com/pub/perl/[256]                    192.245.137.6[257]
South America
  Chile
```

222 ftp://sunsite.cnlab-switch.ch/mirror/CPAN/
223 ftp://193.5.24.1/mirror/CPAN/
224 ftp://ftp.demon.co.uk/pub/mirrors/perl/CPAN/
225 ftp://158.152.1.44/pub/mirrors/perl/CPAN/
226 ftp://sunsite.doc.ic.ac.uk/packages/CPAN/
227 ftp://193.63.255.1/packages/CPAN/
228 ftp://unix.hensa.ac.uk/mirrors/perl-CPAN/
229 ftp://129.12.200.129/mirrors/perl-CPAN/
230 ftp://ftp.utilis.com/public/CPAN/
231 ftp://207.34.209.49/public/CPAN/
232 ftp://enterprise.ic.gc.ca/pub/perl/CPAN/
233 ftp://192.197.182.100/pub/perl/CPAN/
234 ftp://ftp.digital.com/pub/plan/perl/CPAN/
235 ftp://204.123.2.4/pub/plan/perl/CPAN/
236 ftp://ftp.cdrom.com/pub/perl/CPAN/
237 ftp://165.113.58.253/pub/perl/CPAN/
238 ftp://ftp.cs.colorado.edu/pub/perl/CPAN/
239 ftp://128.138.243.20/pub/perl/CPAN/
240 ftp://ftp.cis.ufl.edu/pub/perl/CPAN/
241 ftp://128.227.205.206/pub/perl/CPAN/
242 ftp://uiarchive.cso.uiuc.edu/pub/lang/perl/CPAN/
243 ftp://128.174.5.14/pub/lang/perl/CPAN/
244 ftp://ftp.iguide.com/pub/mirrors/packages/perl/CPAN/
245 ftp://206.15.105.99/pub/mirrors/packages/perl/CPAN/
246 ftp://ftp.rge.com/pub/languages/perl/
247 ftp://157.225.178.12/pub/languages/perl/
248 ftp://ftp.duke.edu/pub/perl/
249 ftp://152.3.233.7/pub/perl/
250 ftp://ftp.ou.edu/mirrors/CPAN/
251 ftp://129.15.2.40/mirrors/CPAN/
252 ftp://ftp.orst.edu/pub/packages/CPAN/
253 ftp://128.193.4.12/pub/packages/CPAN/
254 ftp://ftp.sedl.org/pub/mirrors/CPAN/
255 ftp://198.213.9.194/pub/mirrors/CPAN/
256 ftp://ftp.metronet.com/pub/perl/
257 ftp://192.245.137.6/pub/perl/

ftp://sunsite.dcc.uchile.cl/pub/Lang/perl/CPAN/[258] 146.83.5.204[259]

For those equipped with multi−protocol browsers you might pay a visit to Tom Christiansen's[260] CPAN multiplexer[261] whose relevant Tk URLs are (the second one is not active since it violates the HTML−2.0 spec according to nsgmls):

```
http://perl.com/cgi-bin/cpan_mod?module=Tk
http://perl.com/cgi-bin/cpan_mod?module=Tk&readme=1
```
[262]

According to Stephen P. Potter some of the CPAN sites have decompression on the fly for people who do not have programs like `gunzip`. For example, at the ufl site (Florida USA) type this into your ftp session to download a gunzipped version of Tk:

```
ftp> get Tk400.202.tar.gz Tk400.202.tar
```

If you have the appropriate CPAN and FTP modules (yes there is a CPAN module for retreiving CPAN modules and its name is CPAN oddly enough) already installed you can retrieve a module from CPAN[263] and carry out a complete installation with a perl one−liner like this:

```
perl -MCPAN -e 'install "Tk"'
```

For more information on CPAN you can send e−mail to the CPAN administrators, <cpan−adm@ftp.funet.fi>.[264] If you know of some Perl resources that seem not to be in the CPAN (you did check the contents listings in `indices/`, didn't you?) please tell the CPAN administrators.[265] If you have some modules/scripts/documentation yourself that you would like to contribute to CPAN, please read the file `authors/00upload.howto` and let the CPAN administrators[266] know about it.

Return to table of contents[267]

---

6. How do I build it?

In general, building perl/Tk requires:

1. A made & installed perl (requires a C language compiler). You may need different versions of perl depending on which version of Tk you wish to run.
2. A C language compiler for the Tk code itself.
3. A linkable Xlib[268] (.o, .so, .a, *etc.*) for X−windows.

Perl/Tk has been successfully built using various vendors' `cc` compilers, as well as with the free GNU `gcc` compiler. A `make` utility of some sort (`make`/`gmake`) will be extremely helpful.

Step − by − step the commands to build the Tk extension to Perl are (for the dynamically linked

258 ftp://sunsite.dcc.uchile.cl/pub/Lang/perl/CPAN/
259 ftp://146.83.5.204/pub/Lang/perl/CPAN/
260 http://www.perl.com/
261 http://www.perl.com/perl/news/cpan−mux.html
262 http://perl.com/cgi−bin/cpan_mod?module=Tk
263 #CPAN
264 mailto:cpan−adm@ftp.funet.fi
265 mailto:cpan−adm@ftp.funet.fi
266 mailto:cpan−adm@ftp.funet.fi
267 #TOC5
268 http://www.x.org/

version) roughly as follows:

1. `make install` # the appropriate version of perl.
2. `uninstall` # prior versions of the Tk extension to perl.
3. `gunzip -c Tk400.202.tar.gz | tar xvf -` (options to tar may vary esp. on SysV)
4. `cd Tk400.200`
5. read `INSTALL`
6. `perl Makefile.PL`
7. `make`
8. `make test`
9. `make install`

For the statically linked version you would `` `make tkperl` `` just after executing the `` `make` `` step and before the `` `make test` `` step.

Note carefully that this process leaves behind a large amount of documentation and examples in various sub−directories. You are strongly encouraged to look carefully through your build tree for docs, examples, *etc.* and keep those valuable files in a safe place. You might consider tar−ing them off and installing in a webserver directory tree.

A relatively easy way to determine if the perl on your system allows for dynamic linking was mentioned by Kenneth Albanowski <kjahds@kjahds.com>. If the following does not say `"dl_none.xs"` then you probably do have dynamically linked perl (or perhaps a very non−Unixy perl):

```
perl -e 'use Config; print $Config{dlsrc},"\n"'
```

(thanks to Paul Lussier <plussier@isd.3com.com> for the correction!).

Here is a little more detailed discussion of each the steps just given:

- **Install Perl** Read the Tk files (Tk*/README, *etc.*) for info on which version of perl is required for the perl/Tk kit you obtained.) For code locations see a CPAN site (separate question in this FAQ),[269] the actual installation instructions come bundled in the perl***.tar.gz distribution file. (Perl `Configure` & `make` troubles are beyond the scope of this FAQ − please see the Perl FAQ[270] itself or the `INSTALL` file for more help with this critical step.) You can install perl almost anywhere you like by specifying the `-Dprefix=`*`/path`* argument to `sh Configure`
- Unpack perl/Tk **outside** the Perl distribution
(*i.e.* outside the perl build, perl install, or perl lib areas).
`gunzip Tk400.202.tar.gz`
`tar -xvf Tk400.202.tar`
(Your tar program may not take `-xvf`. The resultant `Tk400.202/` area will be referred to as your ''Tk build'' directory throughout this document.)
- Read `INSTALL`[271] carefully
`cd Tk400.202`
`pager` `INSTALL`
where *`pager`* is the program you use to scroll through text files more or less. Be sure to read it

[269] #CPAN
[270] http://www.perl.com/perl/faq/
[271] etc/INSTALL−Tk.html

and don't just pound away on the spacebar.

- If necessary remove any previously installed version of perl/Tk
  If you had a previously working version of Tk installed, you may need to resurrect the Makefile for it and execute:
  `make uninstall`
  `make realclean`
  before you unpack the new version. (The uninstall target of MakeMaker is relatively new so please be careful here.)
- Have perl generate a custom `Makefile`.
  `perl Makefile.PL`
  (see below for more on this step.)
- Compile.
  `make`
  (if and only if building static: `make tkperl`
- Test.
  `make test`
- Install.
  `make install`
- Play with it.
  `basic_demo`
  (modify `#!` line if necessary, or specify `/`*`path`*`/`*`to`*`/perl ./basic_demo`)
  (*warning* if you build Tk−b9.01 with perl5.002gamma then change the line in `basic_demo` from
  `use lib ./blib;`
  to
  `use lib qw(blib/arch blib/lib);`)
- Save the documentation and examples in a safe accessible place.
  use `tar`, `cp`, `mv`, `chmod` or whatever you prefer to save the valuable ancillary files from your Tk build tree.

On the `perl Makefile.PL` step it may be necessary to give explicit locations of the required X11 libraries and/or include headers. For example:

```
perl Makefile.PL X11=/usr/local/X11R5
```

or perhaps different directory tree specification is necessary with your X installation:

```
perl Makefile.PL X11INC=/usr/local/share/X11R5/include \
                 X11LIB=/usr/local/arch/X11R5/lib
```

There are system and site dependencies in all of the above steps. However, the largest single source of build trouble comes from not using the latest versions of the various utilities (`C compiler`, `make`, *etc.*). In particular ensure that when you say `perl Makefile.PL` that the `perl` that gets invoked is up to date − use `which perl` (or `whence perl`) and `perl -v` to determine this. If necessary specify the full path name to your perl5 interpreter/compiler. (Some people do not `rm` their older perl interpreters when upgrading to a more recent version − beware.)

If you still run into trouble take a look at the INSTALL[272], the README[273] and the `README` file for your specific system (*e.g.* `README.AIX`, `README.OSF`, *etc.*). You might also find your system mentioned in the ptk hyper−mail archive at:

[272] etc/INSTALL−Tk.html
[273] etc/README

`http://www.rosat.mpe-garching.mpg.de/mailing-lists/ptk/`[274]

or

`http://pubweb.bnl.gov/~ptk/`[275]

or

`ftp://ftp.ccd.bnl.gov/pub/ptk/archives/`[276]

or the Perl 5 Porters page at one of the following URLs:

`http://www.rosat.mpe-garching.mpg.de/mailing-lists/Perl5-Porters/`[277]
`http://www.hut.fi/~jhi/perl5-porters.html`[278]

If you wish to discuss your Tk build problems with others run and save the output from the `myConfig` script in the Tk build directory (the output may already be in the `myConfig.out` file from your perl/Tk build directory), as well as the `myconfig` script in your perl build directory (or the output of `perl -V` with a capitol V). It is often helpful to include the output of either (or both) of these scripts in your discussion.

Presented here are the *beginnings* of a list of problems associated with building perl/Tk on various platforms (for help building perl itself please refer to the Perl FAQ[279]). This list is in no way complete nor authoritative (nor is it necessarily even up–to–date!) but simply lists problems people have reported. Keep in mind that your *installation* may differ (*e.g.* location differences such as `/usr/bin/perl` *vs.* `/usr/local/bin/perl`) even if its the same platform listed here:

A Sampling of Perl/Tk Platforms:[280]

AIX:[281]

As of perl5.002b & Tk–b9.01 `README.AIX` says no patching is necessary.

For Tk–b8: modifying the `perl.exp` file may be necessary. There is a patch in `Tk-b8/README.AIX`. It may be necessary to `make regen_headers` after the patch.

FreeBSD:

Nate Patwardhan <nvp@nfic.com> reports no trouble at all with Tk400.200 on FreeBSD–2.1.5 or FreeBSD–2.2.1.

HPUX:[282]

For Tk–b11: One person reports a need to add `#define TIMEOFDAY_TZ` to the `tkConfig.h` header file in order to compile on HPUX 9.05.

Previous versions: Most people seem to prefer the dynamic linking afforded by a *recent* version of the gcc compiler on this system.

Linux:[283]

John C. Wingenbach indicates that should you encounter an error message like `Cannot find -lX11 anywhere at ./myConfig line 184` when running your `perl Makefile.PL` (under Slakware 3.0) that you should be more specific about `-l/`*`path/to`*`/libX11.a`.

Adam Wasserman <awasser@hermes.sgc.com> has graciously provided a compilation of

[274] http://www.rosat.mpe–garching.mpg.de/mailing–lists/ptk/
[275] http://pubweb.bnl.gov/~ptk/
[276] ftp://ftp.ccd.bnl.gov/pub/ptk/archives/
[277] http://www.rosat.mpe–garching.mpg.de/mailing–lists/Perl5–Porters/
[278] http://www.hut.fi/~jhi/perl5–porters.html
[279] http://www.perl.com/perl/faq/
[280] ptkPORT.html
[281] hints/README.AIX
[282] hints/README.HPUX
[283] hints/README.linux

Linux compilation trials & tribulations. It is an (as yet un–edited) document available at: `http://w4.lns.cornell.edu/~pvhp/ptk/hints/linux_compile.txt`[284]

MachTen:

Mark Pease <pease@act.sps.mot.com>[285] mentions that:

I was able to get Tk−b11.02 running under MachTen 2.2 perl5.002_01. I did need to make one change to get a round a MachTen problem. In pTk/tclUnix.h, pwd.h is included, but it is also included in pTk/tkPort.h (which is included in Lang.h, which is use by tclUnixUtil.c, whew!)

MachTen's pwd.h can't be included more that once or you get an error.

It looked to me like tclUnix.h was only used in tclUnixUtil.c, so I commented out the #include <pwd.h> in tclUnix.h.

NetBSD:

Jesus M. Gonzalez <jgb@gsyc.inf.uc3m.es> mentions success with:

Tk−b11.01 compiles, installs and runs just out of the box in NetBSD−1.1/i386. I just followed the INSTALL instructions.

NeXTSTEP[286]:

Gerd Knops recently posted a discussion of the steps to get perl running on several NeXTSTEPs[287] to p5p.

OS/2:

Ilya Zakharevich <ilya@math.ohio–state.edu>[288] has compiled a modified form of Tk–b11.02 to work with the Xfree86 client/server package, as well more advanced versions working with the Open32 PM package.

OSF/1[289]:

As of perl5.002b & Tk–b9.01 you will probably be able to follow the usual instructions. John Stoffel <john@wpi.edu> reports that if you use gcc (rather than cc) you should use at least version 2.7.2

For Tk–b8: `make` is reputedly not up to the task on this system. `Tk-b8/README.OSF` recommends `gmake` instead.

Stephane Bortzmeyer <bortzmeyer@pasteur.fr>[290] reports a successful build with **Perl 5.001m**, **xsubpp 1.922**, **MakeMaker 4.23**. He points out that it was necessary for him to upgrade the xsubpp and MakeMaker that he received with his copy of Perl5.001m.

SCO[291]:

For Tk–b8: Eric J. Bohm <bohm@cs.Buffalo.EDU>[292] reported a need to comment out line(s) from `myConfig` and `GNUMakefiles` using GNU make 3.67. (See `Tk-b8/README.SCO` for specifics.)

SGI (Irix):

For Tk–b11.02: Phillip Moore <wpm@morgan.com> reports a clean build on IRIX 5.3.

Matthew Black <black@csulb.edu>[293] recently mentioned a need to apply "patchSG0000596" to get perl sockets to work. His message was copyrighted and is not included here. Send e–mail to him[294] to find out where the get "patchSG0000596".

---

284 hints/linux_compile.txt
285 mailto:pease@act.sps.mot.com
286 hints/next.steps.txt
287 etc/next.steps.txt
288 mailto:ilya@math.ohio–state.edu
289 hints/README.OSF
290 mailto:bortzmeyer@pasteur.fr
291 hints/README.SCO
292 mailto:bohm@cs.Buffalo.EDU
293 mailto:black@csulb.edu
294 mailto:black@csulb.edu

Suns:

SunOS (BSD):

For Tk−b10 on SunOS 4.1.3_U1
using SparcWorks acc 3.0.1 Frederick L. Wagner <derf@ti.com> reports needing to use the perl malloc rather than the system malloc() when building perl.

For Tk−b8: Tom Tignor <tpt2@BBN.COM>[295] reports the following on SunOS (sun4m sparc):
Tue, 28 Nov 1995 13:19:42
In trying to make, I got a "write: argument mismatch" error for the file ptK/Lang.h. I looked at the file and found the offending function, Tcl_GetOpenFile, which has a third argument called "doWrite" (not "write") in tkGlue.c. I changed the argument from "write" to "doWrite" in Lang.h and it's compiling fine (for the moment. :)

Solaris[296] (System V):

For Tk−b8: There is trouble getting **perl** to use **Socket** routines (*i.e.* trouble with `make` perl itself not necessarily trouble with Tk−b8). See the perl FAQ[297] for more info or the `.shar`[298] file that Tom Christiansen occasionally posts to comp.lang.perl.misc.[299] Further information on perl **i** nter **p**rocess **c**ommunication can be found in the `perlipc*` files at:
`ftp://ftp.perl.com/perl/info/everything_to_know/`.[300]

SVR4:[301]

For Tk−b8: Martha G. Armour and Len Reed report on two separate hardware platforms running SVR4 − extensive details in `Tk-b8/README.SVR4`. Interestingly, they report no trouble at all on Linux.

Ultrix:[302]

Peter Prymmer reports that with Tk−b11 it was necessary to change the line in Makefile.PL that reads:
```
'LIBS' => ["$xlib -lX11 -lpt -lsocket -lnsl -lm"],
```
to read:
```
'LIBS' => ["$xlib -lX11 -lpt -lsocket -lnsl -lm -ldnet"],
```
because of a newer X11 in /usr/local that needed the DECnet protocol linking.

John Stoffel reports a successful build of static Tk−b10 on Ultrix 4.5.

Windows NT:

Nick Ing−Simmons reports success with the alpha Tk404.000 kit, perl5.004, and Visual C++.

non−Unix(ish)es:[303]

Information on non−Unix(ish) perl platforms may be obtained from newsgroups and email lists as well as a few world wide web sites. For example, try the Perl 5 Porters **(p5p)** [page|archives] at one of:

`http://www.rosat.mpe-garching.mpg.de/mailing-lists/Perl5-Porters/`[304]
`http://www.hut.fi/~jhi/perl5-porters.html`[305]

In general your non−Unix platform must be able to support perl 5 and Xlib[306] (a C compiler and a make utility are tremendously useful too). If you want to run perl/Tk on another computer and simply have

295 mailto:tpt2@BBN.COM
296 hints/README.Solaris
297 http://www.perl.com/perl/faq/Q4.21.html
298 hints/SolSock.shar
299 news:comp.lang.perl.misc
300 ftp://ftp.perl.com/perl/info/everything_to_know/
301 hints/README.SVR4
302 hints/README.ultrix
303 ../perl/nixnix.html
304 http://www.rosat.mpe−garching.mpg.de/mailing−lists/Perl5−Porters/
305 http://www.hut.fi/~jhi/perl5−porters.html
306 http://www.x.org/

the display show up on yours then all you need on your computer is an "X server"[307] The long list of UNIX and non−unix[308] perl 5 ports, Tcl/Tk ports, and Perl/Tk ports that used to appear in this FAQ has now moved to a separate web page at:

`http://w4.lns.cornell.edu/~pvhp/ptk/ptkPORT.html`[309]

Return to table of contents[310]

---

7. Where is the Documentation?

A great deal of Perl/Tk documentation gets installed as part of the Tk extension building process. Hence, a great deal of documentation is probably residing on your computer already.

More documentation is "in the works": there are several books dealing with perl/Tk in progress, an ongoing magazine column[311] and a growing FAQ (the document you are presently reading).

The additional material you may want to look at can be split into Perl/Tk,[312] Perl,[313] Tcl/Tk,[314] and X[315] documentation categories:

Perl/Tk Specific Documentation

The man pages

With up to date Tk build kits the various perl/Tk pod documents[316] are converted to your systems' helpfile format and installed as part of the perl/Tk "`make install`" process. If you have a recent verion of perl/Tk try something like `man 3 Tk::Tk` if this does not work check with you system administrator for the proper MANPATH.

In your Tk build directory there should be a `doc/`[317] sub−directory in which there are a number of `.htm` files (after you `make install`). These files were originally Tcl/Tk man pages (from the man* sub−directories), but the `*.htm` files have been converted to Perl syntax and rendered in HTML format. You can use the Perl/Tk web browser to view them locally with a command like:

```
tkweb index.html
```

or you may view them on the web itself by installing them in a web−server directory tree, or by pointing a browser at:

`http://w4.lns.cornell.edu/~pvhp/ptk/doc/`[318]

The newsgroup

---

307 http://www.x.org/
308 ../perl/nixnix.html
309 ptkPORT.html
310 #TOC6
311 http://orwant.www.media.mit.edu/the_perl_journal/
312 #PTKDOCS
313 #PERLDOCS
314 #TCL−TKDOCS
315 #XDOCS
316 pod/
317 doc/
318 http://w4.lns.cornell.edu/~pvhp/ptk/doc/

The newsgroup name is comp.lang.perl.tk[319] and this FAQ will be periodically posted to that group (as well as a few other newsgroups). The newsgroup and/or the ptk mailing list are the appropriate places to post questions – yes even simple ones! (Although answers may sometimes be long in coming ... :–(

The nTk/pTk mailing list

The mailing list is an excellent supplement and complement to the newsgroup comp.lang.perl.tk[320]. All messages mailed to the list are forwarded to the newsgroup. (But not all messages posted to the newsgroup are forwarded to the list.) Some Perl/Tk experts only have access to e–mail.

The nTk/pTk Mailing List Archive is a very useful source of information too, and is accesible at either

    http://pubweb.bnl.gov/~ptk/[321]

or via `ftp` at

    ftp://ftp.ccd.bnl.gov/pub/ptk/archives/[322]

(both in the USA). You may *search* the contents of another ptk mailing list hypertext archive thanks to a cgi–bin script written by Achim Bohnet in Germany at:

    http://www.rosat.mpe-garching.mpg.de/mailing-lists/ptk/[323]

You must subscribe to the mailing list to receive e–mail from the list. To subscribe to the mailing list you can send mail to majordomo@lists.stanford.edu[324] (*i.e.* <majordomo@lists.stanford.edu>[325]) with the following command in the body of your e–mail message:

```
subscribe ptk joe.user@somewhere (Joe D. User)
```

To send a message to *all* recipients of the mailing list send e–mail to <ptk@lists.stanford.edu>[326].

To remove yourself from the mailing list send e–mail to majordomo@lists.stanford.edu[327] (*i.e.* <majordomo@lists.stanford.edu>[328]) with the following command in the body of your e–mail message:

```
unsubscribe ptk joe.user@somewhere (Joe D. User)
```

Where instead of "joe.user@somewhere" you might very well like to substitute another string of characters.

(Please note: one does not send unsubscribe messages to the ptk list. One does send "unsubscribe ptk" messages to a special e–mail list administration program. In the case of the ptk list you send to majordomo@lists.stanford.edu[329]. You must of course do this from the account and computer from which you initially subscribed. In particular if you are viewing the hypertext version of this FAQ you may not be able to unsubscribe from majordomo@lists.stanford.edu[330] by following the mailto:

319 news:comp.lang.perl.tk
320 news:comp.lang.perl.tk
321 http://pubweb.bnl.gov/~ptk/
322 ftp://ftp.ccd.bnl.gov/pub/ptk/archives/
323 http://www.rosat.mpe–garching.mpg.de/mailing–lists/ptk/
324 mailto:majordomo@lists.stanford.edu
325 mailto:majordomo@lists.stanford.edu
326 mailto:ptk@lists.stanford.edu
327 mailto:majordomo@lists.stanford.edu
328 mailto:majordomo@lists.stanford.edu
329 mailto:majordomo@lists.stanford.edu
330 mailto:majordomo@lists.stanford.edu

hyperlinks – if your web–browser account/computer is different from your subscribe to e–mail–lists account/computer (the details of this might depend on which browser you use though). Thank you for your cooperation on this.)

The demo programs

Examine (and try running) the code in your `Tk#/` build directory tree. You might also be interested in test–running the code that gets installed:

- or `perl5/site_perl/Tk/demos/` or `perl5/site_perl/Tk/demos/widget_lib/` directories (recent Tk's).
- `Tk#/` or `perl5/Tk/demos/` or `perl5/Tk/demos/widget_lib/` directories (older versions *e.g.* Tk–b8).

In order to determine where on your system the `perl5/` directory is located type the following one–line perl command (at your shell prompt – this is not a line from a perl script):

```
perl -e 'print join("\n",@INC,"");'
```

If that command does not turn up a `perl5/` directory then make sure that you are running perl 5 with the following: `perl -v` (this too can simply be entered at the shell prompt).

More on the pod & man documentation

[As previously mentioned a great number of Tcl/Tk man pages are converted from *roff format to html format and are to be found within your Tk build directory tree in the doc/[331] sub–directory. These documents[332] form an authoritative and extensive reference set for Perl/Tk.]

The raw pod files that come with the Tk kit are examples of the perl "plain old documentation"[333] format and are just about human readable as they are (*e.g.* you may `more`, `cat`, or `less` them; or send them to a printer). Many (not all) of the perl/Tk pod documents[334] get are converted to *roff format and are installed in you perl man page area as part of the perl/Tk build process.

If you have a recent version of perl/Tk try something like `man 3 Tk::Tk`. If this does not work check your manual page path with

```
perl -MConfig -e 'print $Config{man1dir},"\n",$Config{man3dir},"\n"'
```

And if you still cannot find the manual pages check with your system administrator for the proper MANPATH and/or Tk installation version.

"Raw" `.pod` (such as `UserGuide.pod`[335]) can be viewed with the **tkpod** hypertext pod viewer. Raw `.pod`[336] may also be run through any one or more of a large numbers of re–formatting perl filters. Such programs include **pod2man**, **pod2text**, **pod2html**, **pod2latex**, *etc.* (these get installed when you install `perl`). Other translators `pod2texinfo`, `pod2fm`, `pod2pdf`, *etc.*, also exist. Check a CPAN site[337] for these scripts if you do not already have them.

331 doc/
332 doc/
333 http://www.metronet.com/0/perlinfo/perl5/manual/perlpod.html
334 pod/
335 pod/UserGuide.html
336 pod/
337 #CPAN

A command line *like* the following (but subject to local variations) should work for you:

```
tkpod site_perl/Tk/UserGuide.pod
```

or if you like Unix manual page style:

```
pod2man perl5/Tk/UserGuide.pod | nroff -man | more
```

(note that I am showing examples with almost full file path names – the alternative would be to *cd* into the appropriate directory then type:

```
pod2man UserGuide.pod | nroff -man | more
```

There should even be a perl script to run that above command for you. It is executed as:

```
perldoc UserGuide
```

Note that *if* there is pod like documentation in a perl module you may also execute **tkpod** (or `perldoc`) on it as in:

```
tkpod ColorEditor.pm
```

(please note that unfortunately, not *all* `.pm` mod files have pod[338] embedded.)

If you have misplaced your **tkpod** program but still want that GUI look and feel (like *xman*) make the appropriate changes to the following script:

```
#!/usr/bin/perl
use Tk;
use Tk::Pod;
my $m = new MainWindow;
$m -> Pod(-file => 'ColorEditor.pm');
# or use command line path/filename:
# $m -> Pod(-file => $ARGV[0]);
MainLoop;
```

A miscellany of internet perl/Tk resources includes:

```
World Wide Web - perl/Tk man pages
    http://pubweb.bnl.gov/~ptk/doc/index.html[339]
    http://w4.lns.cornell.edu/~pvhp/ptk/doc/[340]
    http://w4.lns.cornell.edu/~pvhp/ptk/pod/[341]
The Perl/Tk Newsgroup
    comp.lang.perl.tk[342]
Perl Newsgroups
    comp.lang.perl.misc[343]
    comp.lang.perl.anounce[344]
    comp.lang.perl.modules[345]
Tcl Newsgroups
    comp.lang.tcl[346]
    comp.lang.tcl.announce[347]
```

338 http://www.metronet.com/0/perlinfo/perl5/manual/perlpod.html
339 http://pubweb.bnl.gov/~ptk/doc/index.html
340 http://w4.lns.cornell.edu/~pvhp/ptk/doc/
341 http://w4.lns.cornell.edu/~pvhp/ptk/pod/
342 news:comp.lang.perl.tk
343 news:comp.lang.perl.misc
344 news:comp.lang.perl.announce
345 news:comp.lang.perl.modules
346 news:comp.lang.tcl
347 news:comp.lang.tcl

```
Miscellaneous Newsgroups
    comp.answers[348]
    news.answers[349]
Perl/Tk FAQ-Archives (ftp sites) [Note: FAQ may be many separate files]
 (see also CPAN sites[350])
    ftp://rtfm.mit.edu/pub/usenet/comp.lang.perl.tk[351]
    ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/comp/lang/perl/tk[352]
    ftp://ftp.uu.net/usenet/news.answers/perl-faq/ptk-faq[353]
    ftp://ftp.ccd.bnl.gov/pub/ptk/[354]                  130.199.54.188
    ftp://ftp.ccd.bnl.gov/pub/ptk/ptkFAQ.txt[355]        130.199.54.188
    ftp://ftp.wpi.edu/perl5/pTk-FAQ[356]                 130.215.24.209
    ftp://perl.com/pub/perl/doc/ptkFAQ.gz[357]           199.45.129.30
    ftp://perl.com/pub/perl/doc/ptkFAQ.ps.gz[358]        199.45.129.30
WWW-FAQ for perl/Tk
    http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html[359]
    http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html[360]
World Wide Web - perl/Tk info sites
    http://pubweb.bnl.gov/~ptk/[361]
    http://fxfx.com/kgr/compound/[362](Perl Tk Compound Widget Page)
    http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html[363](FAQ image supplement)
    http://w4.lns.cornell.edu/~pvhp/ptk/etc/[364]
    http://w4.lns.cornell.edu/~pvhp/ptk/misc/[365]
    http://w4.lns.cornell.edu/~pvhp/ptk/PNMTYAK/[366]
    http://www.mirai.com/wks/[367]
The Mailing list
    majordomo@lists.stanford.edu[368]
    ptk@lists.stanford.edu[369]
```

Perl Specific Documentation

There are a growing number Perl books[370] available. A more complete Perl–bibliographic discussion than that given here is available in the Perl FAQ[371] or at:

```
    http://www.perl.com/perl/info/books.html[372]
```

For Perl 5 there is (as of September 1996) a "New Camel" by Larry Wall, Tom Christiansen,[373] and Randal L. Schwartz,[374] with Stephen Potter.[375]

348 news:comp.answers
349 news:news.answers
350 #CPAN
351 ftp://rtfm.mit.edu/pub/usenet/comp.lang.perl.tk
352 ftp://rtfm.mit.edu/pub/usenet–by–hierarchy/comp/lang/perl/tk
353 ftp://ftp.uu.net/usenet/news.answers/perl–faq/ptk–faq
354 ftp://ftp.ccd.bnl.gov/pub/ptk/
355 ftp://ftp.ccd.bnl.gov/pub/ptk/ptkFAQ.txt
356 ftp://ftp.wpi.edu/perl5/pTk–FAQ
357 ftp://perl.com/pub/perl/doc/ptkFAQ.gz
358 ftp://perl.com/pub/perl/doc/ptkFAQ.ps.gz
359 ptkTOC.html
360 ptkFAQ.html
361 http://pubweb.bnl.gov/~ptk/
362 http://fxfx.com/kgr/compound/
363 ptkIMG.html
364 etc/
365 misc/
366 PNMTYAK/
367 http://www.mirai.com/wks/
368 mailto:majordomo@lists.stanford.edu
369 mailto:ptk@lists.stanford.edu
370 http://www.perl.com/perl/faq/Q2.2.html
371 http://www.perl.com/perl/faq/
372 http://www.perl.com/perl/info/books.html
373 mailto:tchrist@perl.com
374 http://www.teleport.com/~merlyn
375 mailto:spp@psa.pencom.com

Programming Perl 2nd Edition[376]
Larry Wall, Tom Christiansen, & Randal L. Schwartz with Stephen Potter [377]
(c) 1996 O'Reilly & Associates Inc.[378]
ISBN 1−56592−149−6 (English)

A second edition of the Llama is due out soon too:

Learning Perl, 2ndEdition[379]
Randal L. Schwartz[380]
June 1997 (est.) O'Reilly & Associates Inc.[381]
ISBN 1−56592−284−0 (English)

The two early Perl books by Schwartz and Wall are very helpful (even if they do pertain to perl 4 and not 5. Beware that perl/Tk makes extensive use of perl 5[382] object−oriented[383] features.):

Learning Perl[384] (The Llama)[385]
Randal L. Schwartz[386]
Copyright (c) 1993 O'Reilly & Associates Inc.[387]
ISBN 1−56592−042−2 (English)
ISBN 2−84177−005−2 (French)
ISBN 3−930673−08−8 (German)
ISBN 4−89502−678−1 (Japanese)

Programming Perl[388] (The Camel)[389]
Larry Wall and Randal L. Schwartz
Copyright (c) 1991 O'Reilly & Associates Inc.[390]
ISBN 0−937175−64−1 (English)[391]
ISBN 3−446−17257−2 (German) (Programmieren in Perl, translator: Hanser Verlag)
ISBN 4−89052−384−7 (Japanese)

There is also some Perl5 (book material) information at:

`http://www.metronet.com/1h/perlinfo/perl5/`[392]

Jon Orwant (the organizer of the comp.lang.perl.tk[393] newgroup) has a book on Perl 5 that has a chapter that discusses the Tk extension. (Please note that it is mostly about Perl 5, there is a some discussion of four simple Perl/Tk programs, but it is not a book wholly devoted to Perl/Tk.) It is nevertheless a good introduction to object−oriented Perl 5 programming. The relevant info:

376 http://amber.ora.com/gnn/bus/ora/item/pperl2.html
377 http://amber.ora.com/gnn/bus/ora/item/pperl2.author.html
378 http://gnn.com/gnn/bus/ora/aboutora.html
379 http://www.ora.com/catalog/lperl2/
380 http://www.teleport.com/~merlyn
381 http://www.ora.com/
382 http://www.perl.com/perl/info/perl5−brief.html
383 http://www.metronet.com/0/perlinfo/perl5/manual/perlobj.html
384 http://www.ora.com/gnn/bus/ora/item/lperl.html
385 http://www.ora.com/info/perl/
386 http://www.teleport.com/~merlyn
387 http://gnn.com/gnn/bus/ora/aboutora.html
388 http://www.metronet.com/perlinfo/perl5/manual/perlbook.html
389 http://www.ora.com/gnn/bus/ora/item/pperl.html
390 http://gnn.com/gnn/bus/ora/aboutora.html
391 http://nearnet.gnn.com/gnn/bus/ora/catalog/pperl.desc.html
392 http://www.metronet.com/1h/perlinfo/perl5/
393 news:comp.lang.perl.tk

Perl 5 Interactive Course[394]
Jon Orwant[395]
(c) 1996 The Waite Group Press[396]
A Division of SAMS Publishing, Corte Madera, CA USA
ISBN: 1−57169−064−6

The **Perl 5 Quick Reference Guide**[397] (may require **LaTeX** for installation) can be obtained from any CPAN ftp site[398]. Detailed location information is also available at the author's website:

`http://www.xs4all.nl/~jvromans/perlref.html`[399]

The **quick reference guide** has also been turned into a small Nutshell handbook:

Perl 5 Desktop Reference[400]
Johan[401] Vromans[402]
Copyright (c) February 1996 O'Reilly & Associates Inc.[403]
ISBN: 1−56592−187−9; Order number: 1879

Eric F. Johnson has a book that discusses many perl5 for Unix *vs.* perl5 for Windows NT issues. He includes a final chapter with extensive discussion of the Tk extension and the `->Text()` widget in particular. The information on that book is:

Cross−Platform Perl[404]
(c) 1996 Eric F. Johnson[405]
MIS:Press/M&T Books[406]
ISBN: 1−55851−483−X

Kamran Husain and Robert F. Breedlove have written a perl 5 book that includes a chapter on Tk with some discussion of `Menu()`s. That book is:

Perl 5 Unleashed[407]
Kamran Husain and Robert F. Breedlove
(c) 1996 Sams Publishing, Indianapolis, IN
ISBN: 0−672−30891−6

There is also a "Perl 5 How−To" book available that contains a great deal of erroneous information about perl/Tk. Among other things that book wrongly mentions that it is necessary to have a complete Tcl/Tk library installed on one's system to compile the Tk extension to perl. (They are incorrect − it is only necessary to have the appropriate perl version, libc and Xlib, the Tk extension is otherwise "self−contained").

There is also a book on perl web client. It features a a chapter on perl/Tk that was written by Nancy Walsh:

394 http://www.mcp.com/waite/ezone/ezone/catalog/picdesc.htm
395 mailto:orwant@media.mit.edu
396 http://www.waite.com/waite/
397 http://www.xs4all.nl/~jvromans/perlref.html
398 #CPAN
399 http://www.xs4all.nl/~jvromans/perlref.html
400 http://www.ora.com/gnn/bus/ora/item/perlqr.html
401 http://www.ora.com/gnn/bus/ora/item/perlqr.author.html
402 mailto:jv@squirrel.nl
403 http://www.ora.com/
404 http://www.pconline.com/~erc/perlbook.htm
405 http://www.pconline.com/~erc/
406 http://www.mandt.com/
407 http://www.perl.com/perl/critiques/p5−unleashed.html

Web Client Programming with Perl[408]
Clinton Wong[409]
1st Edition March 1997
O'Reilly & Associates Inc.[410]
ISBN: 1−56592−214−X; Order number: 214X

Additional book information may be found at Tom Christiansen's perl & cgi books page,[411] or at his Perl−related Book Reviews[412] page.

The multi−part perl 5 manual pages are available (assuming they have been installed in your `MANPATH`, type `man perl`, `man perlmod` *etc.*).

The perl 5 man pages are also available on the web at a number of locations. In general the more recent the documentation the more helpful it is.

In addition to the CPAN ftp source sites,[413] a miscellany of internet perl resources includes:

```
Newsgroups
    comp.lang.perl.misc[414]
    comp.lang.perl.announce[415]
    comp.lang.perl.modules[416]
    comp.infosystems.www.authoring.cgi[417]
    comp.answers[418]
    news.answers[419]
Perl FAQ−Archives (ftp sites) [Note: FAQ may be many separate files]
(as of 5.004 the FAQ ships in pod format with perl)
    (see also the CPAN sites[420])
  North America
    ftp://rtfm.mit.edu/pub/usenet/news.answers/perl−faq/[421]
    ftp://ftp.uu.net/usenet/news.answers/perl−faq[422] 192.48.96.9
    ftp://ftp.khoros.unm.edu/pub/perl/faq.gz[423]      198.59.155.28
  Europe
    ftp://ftp.cs.ruu.nl/pub/NEWS.ANSWERS/perl−faq/[424]131.211.80.17
    ftp://src.doc.ic.ac.uk/packages/perl/FAQ[425]      146.169.2.10
Gopher Perl FAQ
    gopher://gopher.metronet.com/11/perlinfo/faq[426]
WWW−FAQ for Perl
    http://www.perl.com/perl/faq/[427]
    http://www.smartpages.com/bngfaqs/comp/lang/perl/top.html[428]
    http://www.smartpages.com/bngfaqs/comp/lang/perl/misc/top.html[429]
    http://www.smartpages.com/bngfaqs/comp/lang/perl/announce/top.html[430]
```

408 http://www.ora.com/catalog/webclient/noframes.html
409 http://www.ora.com/catalog/webclient/author.html
410 http://www.ora.com/
411 http://www.perl.com/perl/info/books.html
412 http://www.perl.com/perl/critiques/index.html
413 #CPAN
414 news:comp.lang.perl.misc
415 news:comp.lang.perl.announce
416 news:comp.lang.perl.modules
417 news:comp.infosystems.www.authoring.cgi
418 news:comp.answers
419 news:news.answers
420 #CPAN
421 ftp://rtfm.mit.edu/pub/usenet/news.answers/perl−faq/
422 ftp://ftp.uu.net/usenet/news.answers/perl−faq/
423 ftp://ftp.khoros.unm.edu/pub/perl/faq.gz
424 ftp://ftp.cs.ruu.nl/pub/NEWS.ANSWERS/perl−faq/
425 ftp://src.doc.ic.ac.uk/packages/perl/FAQ
426 gopher://gopher.metronet.com/11/perlinfo/faq
427 http://www.perl.com/perl/faq/
428 http://www.smartpages.com/bngfaqs/comp/lang/perl/top.html
429 http://www.smartpages.com/bngfaqs/comp/lang/perl/misc/top.html
430 http://www.smartpages.com/bngfaqs/comp/lang/perl/announce/top.html

```
    http://www.cis.ohio-state.edu/hypertext/faq/usenet/perl-faq/top.html[431]
Perl for Win32 FAQ  (discusses Win95)
    http://www.perl.org/CPAN/doc/FAQs/win32/Perl_for_Win32_FAQ.html[432]
```

Perl info sites

```
Gopher (gopher:70) perl info sites
  USA
    gopher://gopher.metronet.com/11h/perlinfo[433]
World Wide Web (http:80) perl info sites
  USA
    http://www.yahoo.com/Computers_and_Internet/Languages/Perl/index.html[434]
    http://www.perl.com/[435]
    http://www.khoros.unm.edu/staff/neilb/perl/home.html[436]
    http://www.khoros.unm.edu:80/staff/neilb/perl/metaFAQ/[437]
    http://www.metronet.com/perlinfo/[438]
    http://www.metronet.com/perlinfo/perl5.html[439](Perl 5)
    http://www.eecs.nwu.edu/perl/perl.html[440]
    http://cesr39.lns.cornell.edu/public/perl/[441]
    http://www.virtualschool.edu/mon/Perl.html[442]
    http://www.hermetica.com/technologia/unexec/[443]
    http://www.oac.uci.edu/indiv/ehood/perlWWW/[444]
    http://web.sau.edu/~mkruse/www/scripts/[445]
    http://orwant.www.media.mit.edu/the_perl_journal/[446]
    http://www.perl.com/Architext/AT-allperl.html[447]
    http://www.mispress.com/introcgi/[448]
    http://www.walrus.com/~smithj/webcan/[449]
    http://web.syr.edu/~chsiao05/cps600_project.html[450]
    http://www.iftech.com/classes/webdev/webdev_perl.htm[451]
    http://www.cc.iastate.edu/perlmenu/[452]
    http://www.ora.com/www/item/cgi_prog.html[453]
  UK
    http://pubweb.nexor.co.uk/public/perl/perl.html[454]
    http://www.bio.cam.ac.uk/web/form.html[455]
Web references to Perl mailing lists
    http://www.perl.com/perl/info/mailing-lists.html[456]
    http://www.nicoh.com/cgi-bin/lwgate/PERL5-PORTERS/[457]
    http://www.hut.fi/~jhi/perl5-porters.html[458]
    http://www.rosat.mpe-garching.mpg.de/mailing-lists/[459]
```

Tcl/Tk Specific Documentation

[431] http://www.cis.ohio-state.edu/hypertext/faq/usenet/perl-faq/top.html
[432] http://www.perl.org/CPAN/doc/FAQs/win32/Perl_for_Win32_FAQ.html
[433] gopher://gopher.metronet.com/11h/perlinfo
[434] http://www.yahoo.com/Computers_and_Internet/Languages/Perl/index.html
[435] http://www.perl.com/
[436] http://www.khoros.unm.edu/staff/neilb/perl/home.html
[437] http://www.khoros.unm.edu:80/staff/neilb/perl/metaFAQ/
[438] http://www.metronet.com/perlinfo/
[439] http://www.metronet.com/perlinfo/perl5.html
[440] http://www.eecs.nwu.edu/perl/perl.html
[441] http://cesr39.lns.cornell.edu/public/perl/
[442] http://www.virtualschool.edu/mon/Perl.html
[443] http://www.hermetica.com/technologia/unexec/
[444] http://www.oac.uci.edu/indiv/ehood/perlWWW/
[445] http://web.sau.edu/~mkruse/www/scripts/
[446] http://orwant.www.media.mit.edu/the_perl_journal/
[447] http://www.perl.com/Architext/AT-allperl.html
[448] http://www.mispress.com/introcgi/
[449] http://www.walrus.com/~smithj/webcan/
[450] http://web.syr.edu/~chsiao05/cps600_project.html
[451] http://www.iftech.com/classes/webdev/webdev_perl.htm
[452] http://www.cc.iastate.edu/perlmenu/
[453] http://www.ora.com/www/item/cgi_prog.html
[454] http://pubweb.nexor.co.uk/public/perl/perl.html
[455] http://www.bio.cam.ac.uk/web/form.html
[456] http://www.perl.com/perl/info/mailing-lists.html
[457] http://www.nicoh.com/cgi-bin/lwgate/PERL5-PORTERS/
[458] http://www.hut.fi/~jhi/perl5-porters.html
[459] http://www.rosat.mpe-garching.mpg.de/mailing-lists/

The two Tcl/Tk books by Ousterhout and Welch are very good starting points (you must however, translate the tcl–isms to perl in the sample scripts):

Tcl and the Tk Toolkit
John K. Ousterhout
Copyright (c) 1994 Addison–Wesley Publishing Company
ISBN 0–201–63337–X (alk. paper)
LOC QA76.73.T44097 1994; 005.13'3––dc20

Practical Programming in Tcl and Tk[460]
Brent Welch[461]
Copyright (c) 1995 Prentice Hall[462]
ISBN 0–13–182007–9

Within the `tclsh` or `wish` shells your manpath includes the tcl/tk man pages (which may *not* be in your login MANPATH). Thus from the `%` prompt within either shell type commands like:

```
% man -k Tk
```

The Tcl/Tk Reference Guide is also a source of useful information. Although it's Tcl specific most perl/Tk commands can be, more or less, easily derived from it. [Note that in perl/Tk the names of some functions and some configuration options have changed slightly from their Tcl/Tk counterparts. With recent versions of perl/Tk a great many functions start with an upper case letter and continue with all lower case letters (*e.g.* there is a perl/Tk `Entry` widget but no `entry` widget), and many configuration options are all lower case (*e.g.* there is a perl/Tk `highlightthickness` option but no `highlightThickness` option).] You may fetch the Tcl/Tk Reference Guide (may require **LaTeX** for installation) from:

```
ftp://ftp.slac.stanford.edu/software/TkMail/tkref-4.0.1.tar.gz[463] 134.79.18.30
ftp://ftp.aud.alcatel.com/tcl/docs/tkref-4.0.1.tar.gz[464]          198.64.191.10
```

There are a number of other Tcl/Tk resources on the internet including:

```
Newsgroups
    comp.lang.tcl[465]
    comp.lang.tcl.announce[466]
    comp.infosystems.www.authoring.cgi[467]
    comp.answers[468]
    news.answers[469]
FAQ-Archive (ftp) [Note: Tcl FAQ may be many files, Tk FAQ is one file]
    ftp://ftp.aud.alcatel.com/tcl/docs/[470]                   198.64.191.10
    ftp://rtfm.mit.edu/pub/usenet/news.answers/tcl-faq[471]
    ftp://rtfm.mit.edu/pub/usenet/news.answers/tcl-faq/tk[472]
WWW-FAQ for Tcl/Tk
    http://www.teraform.com/%7Elvirden/tcl-faq/[473]
```

460 http://www.smli.com/~bwelch/book/index.html
461 mailto:welch@parc.xerox.com
462 http://www.prenhall.com/
463 ftp://ftp.slac.stanford.edu/software/TkMail/tkref–4.0.1.tar.gz
464 ftp://ftp.aud.alcatel.com/tcl/docs/tkref–4.0.1.tar.gz
465 news:comp.lang.tcl
466 news:comp.lang.tcl
467 news:comp.infosystems.www.authoring.cgi
468 news:comp.answers
469 news:news.answers
470 ftp://ftp.aud.alcatel.com/tcl/docs/
471 ftp://rtfm.mit.edu/pub/usenet/news.answers/tcl–faq
472 ftp://rtfm.mit.edu/pub/usenet/news.answers/tcl–faq/tk
473 http://www.teraform.com/%7Elvirden/tcl–faq/

```
    http://www.smartpages.com/faqs/tcl-faq/top.html[474]
    http://www.smartpages.com/bngfaqs/comp/lang/tcl/top.html[475]
    http://www.cis.ohio-state.edu/hypertext/faq/usenet/tcl-faq/top.html[476]
  http://www.cis.ohio-state.edu/hypertext/faq/bngusenet/comp/lang/tcl/top.html[477]
    http://www.sco.com/Technology/tcl/Tcl.html[478]
World Wide Web - Tcl/Tk info sites
  Canada
    http://web.cs.ualberta.ca/~wade/Auto/Tcl.html[479]
  UK
    http://http2.brunel.ac.uk:8080/~csstddm/TCL2/TCL2.html[480]
    http://www.cis.rl.ac.uk/proj/TclTk/[481]
  USA
    http://www.yahoo.com/Computers_and_Internet/Languages/Tcl_Tk/index.html[482]
    http://www.sunlabs.com/research/tcl/docs.html[483]
    http://www.sunlabs.com/research/tcl/4.0.html[484]
    http://www.sco.com/Technology/tcl/Tcl.html[485]
    http://www.neosoft.com/tcl/[486]
    http://www.elf.org/tcltk-man-html/contents.html[487]
Tcl/Tk - miscellaneous extensions
    ftp://ftp.cme.nist.gov/pub/expect/[488]
    http://www.cs.hut.fi/~kjk/porttk.html[489]
    http://www.cis.upenn.edu/~ioi/tix/tix.html[490]
    http://www.ece.cmu.edu/afs/ece/usr/svoboda/www/th/homepage.html[491]
    http://www.tcltk.com/ [incr Tcl][492]
    http://www.neosoft.com/tcl/TclX.html[493]
    http://www.eolas.com/eolas/webrouse/tcl.htm [WebWish][494]
    http://www.se.cuhk.hk/~hkng2/big5tk/big5tk.html[495]
    http://www.cs.uoregon.edu/~jhobbs/work/ [BLT etc.][496]
```

X Documentation

Tk certainly makes the generation of GUI code a lot easier than hard coding things in traditional compiled languages such as C, C++, or Lisp. Nevertheless there is a very large body of X documentation out there that will assist all widget and GUI builders with issues of design, implementation, *etc.* Hence it is good practice to be informed of the general design goals of X itself[497] as well as the other toolkits that have been built on top of X.

There are a number of X resources on the internet including:

```
Newsgroups
    comp.windows.x[498]
```

474 http://www.smartpages.com/faqs/tcl−faq/top.html
475 http://www.smartpages.com/bngfaqs/comp/lang/tcl/top.html
476 http://www.cis.ohio−state.edu/hypertext/faq/usenet/tcl−faq/top.html
477 http://www.cis.ohio−state.edu/hypertext/faq/bngusenet/comp/lang/tcl/top.html
478 http://www.sco.com/Technology/tcl/Tcl.html
479 http://web.cs.ualberta.ca/~wade/Auto/Tcl.html
480 http://http2.brunel.ac.uk:8080/~csstddm/TCL2/TCL2.html
481 http://www.cis.rl.ac.uk/proj/TclTk/
482 http://www.yahoo.com/Computers_and_Internet/Languages/Tcl_Tk/index.html
483 http://www.sunlabs.com/research/tcl/docs.html
484 http://www.sunlabs.com/research/tcl/4.0.html
485 http://www.sco.com/Technology/tcl/Tcl.html
486 http://www.neosoft.com/tcl/
487 http://www.elf.org/tcltk−man−html/contents.html
488 ftp://ftp.cme.nist.gov/pub/expect/
489 http://www.cs.hut.fi/~kjk/porttk.html
490 http://www.cis.upenn.edu/~ioi/tix/tix.html
491 http://www.ece.cmu.edu/afs/ece/usr/svoboda/www/th/homepage.html
492 http://www.tcltk.com/
493 http://www.neosoft.com/tcl/TclX.html
494 http://www.eolas.com/eolas/webrouse/tcl.htm
495 http://www.se.cuhk.hk/~hkng2/big5tk/big5tk.html
496 http://www.cs.uoregon.edu/~jhobbs/work/
497 http://www.x.org/
498 news:comp.windows.x

```
    comp.windows.x.announce[499]
    comp.windows.x.apps[500]
X FAQs:
    ftp://ftp.x.org/contrib/faqs/[501]
X FAQ on the World Wide Web:
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-1[502]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-2[503]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-3[504]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-4[505]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-5[506]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-6[507]
    http://www.x.org/ftp/contrib/faqs/x-faq-multipart/x-faq-7[508]
X Window System book info on the Web:
    http://www.x.org/ftp/contrib/docs/Xbibliography.OReilly[509]
    http://www.ora.com/catalog/v1/[510]
    http://www.ora.com/catalog/v2/[511]
    http://www.ora.com/catalog/v3/[512]
    http://www.ora.com/catalog/v4/[513]
    http://www.ora.com/catalog/v5/[514]
    http://www.ora.com/catalog/v6a/[515]
    http://www.ora.com/catalog/v6b/[516]
    http://www.ora.com/catalog/v6c/[517]
    http://www.ora.com/catalog/r6/noframes.html[518]
    http://www.ora.com/oracom/prog/flanart.html[519]
World Wide Web - X Window System info sites
    http://www.x.org/[520]
    http://www.x.org/consortium/GettingX.html[521]
    http://www.x.org/consortium/x_info.html[522]
    http://www.x.org/consortium/R6.1doc/man/X11/[523]
    http://www.wolfram.com/~cwikla/widget/[524]
    http://www.zeta.org.au/~rosko/pigui.htm[525]
    http://www.rahul.net/kenton/xsites.html[526]
    http://www.unx.com/DD/txaCurrent.shtml[527]
```

Return to table of contents[528]

---

8. How do I write scripts in perl/Tk?

499 news:comp.windows.x.announce
500 news:comp.windows.x.apps
501 ftp://ftp.x.org/contrib/faqs/
502 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−1
503 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−2
504 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−3
505 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−4
506 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−5
507 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−6
508 http://www.x.org/ftp/contrib/faqs/x−faq−multipart/x−faq−7
509 http://www.x.org/ftp/contrib/docs/Xbibliography.OReilly
510 http://www.ora.com/catalog/v1/
511 http://www.ora.com/catalog/v2/
512 http://www.ora.com/catalog/v3/
513 http://www.ora.com/catalog/v4/
514 http://www.ora.com/catalog/v5/
515 http://www.ora.com/catalog/v6a/
516 http://www.ora.com/catalog/v6b/
517 http://www.ora.com/catalog/v6c/
518 http://www.ora.com/catalog/r6/noframes.html
519 http://www.ora.com/oracom/prog/flanart.html
520 http://www.x.org/
521 http://www.x.org/consortium/GettingX.html
522 http://www.x.org/consortium/x_info.html
523 http://www.x.org/consortium/R6.1doc/man/X11/
524 http://www.wolfram.com/~cwikla/widget/
525 http://www.zeta.org.au/~rosko/pigui.htm
526 http://www.rahul.net/kenton/xsites.html
527 http://www.unx.com/DD/txaCurrent.shtml
528 #TOC7

Start your script as you would any perl script (*e.g.* `#!/usr/bin/perl`, `#!/usr/local/bin/perl`, `#!/opt/bin/perl`, [built static? then `#!/usr/bin/tkperl`], *whatever*, see the `perlrun(1)`[529] man page for more information).
Throwing the `-w` warning switch is recommended.
The use of the statement `use strict;` is recommended.
Use of the statement `use Tk;` is required.

A simple "Hello World!" widget script could be written as follows:

```
#!/usr/local/bin/perl -w

use strict;
use Tk;

my $main = new MainWindow;
$main->Label(-text => 'Hello World!'
             )->pack;
$main->Button(-text => 'Quit',
              -command => sub{exit}
              )->pack;
MainLoop;
```

The `MainLoop;` statement is the main widget event handler loop and is usually found in perl/Tk scripts (usually near the end of the main procedure after the widgets have been declared and `pack` ed). `MainLoop;` is actually a function call and you may see it written as `MainLoop();`, `&Tk::MainLoop;`, `&Tk::MainLoop();`, *etc.*

Note the use of the `->` infix dereference operator. Most things in calls to perl/Tk routines are passed by reference.

Note also the use of the => operator which is simply a synonym for the comma operator (well it is a bit more than that :-). In other words, the arguments that get passed to `Label` and `Button` in the above example are good old perl *associative arrays* (perl 5 people prefer to call them "hashes" however). Indeed, we might have written the above as:

```
#!/usr/local/bin/perl -w

use strict;
use Tk;

my $main = new MainWindow;
$main->Label(-text , 'Hello World!'
             )->pack;
$main->Button(-text , 'Quit',
              -command , sub{exit}
              )->pack;
MainLoop;
```

Or even as:

```
#!/usr/local/bin/perl -w
use strict;
use Tk;
my $main = new MainWindow;
```

[529] http://www.metronet.com/perlinfo/perl5/manual/perlrun.html

```
    my %hello = ('-text','Hello World!');
    my %quit_com = ('-text' => 'Quit', '-command' => sub{exit});

    $main->Label(%hello)->pack;
    $main->Button(%quit_com)->pack;
    MainLoop;
```

Note however, that the use of the => in the first method of writing this script makes it look more "Tcl–ish" :–).

Lastly, we note the extensive use of the `my` function in most perl/Tk programs. `my` is roughly equivalent to `local` in Perl 4 – but is purported to be "faster and safer" as well as much more strictly local in scope. See `perlfunc(1)` manpage[530] for more information on `my`.

Other examples of code may be found in the `perl5/Tk/demos/` directory and in `perl5/Tk/demos/widget_lib/`.

(A variant on this scipt called `hello` is available in the file `perl5/Tk/demos/hello` in your own pTk distribution. Also, Source code for this[531] and other examples from UserGuide.pod[532] may be found at http://w4.lns.cornell.edu/~pvhp/ptk/pod/.[533] To load code from the web save as a local *filename*, edit the first line to point to your perl interpreter, then: `chmod u+x` *filename*, then execute: *filename*.)

Return to table of contents[534]

---

9. What widget types are available under perl/Tk?

The following Tk widget primitives are available under perl/Tk:

- **Button**[535]
- **Canvas**[536]
- **Checkbutton**[537]
- **Entry**[538]
- **Frame**[539]
- **Label**[540]
- **Listbox**[541]
- **Menu**[542]
- **Menubutton**[543]
- **Message**[544]
- **Radiobutton**[545]

530 http://www.metronet.com/0/perlinfo/perl5/manual/perlfunc.html
531 pod/ex0.pl
532 pod/UserGuide.html
533 pod/
534 #TOC8
535 doc/button.htm
536 doc/canvas.htm
537 doc/checkbut.htm
538 doc/entry.htm
539 doc/frame.htm
540 doc/label.htm
541 doc/listbox.htm
542 doc/menu.htm
543 doc/menubutton.htm
544 doc/message.htm
545 doc/radiobut.htm

- **Scale**[546]
- **Scrollbar**[547]
- **Text**[548]
- **Toplevel**[549]

The following are Tix widget primitives available under perl/Tk:

- HList[550]
- InputOnly[551]

There are (a lot of) other [compound|composite|constructs] available too. You can also synthesize new widgets out of these primitives using perl5's object−oriented multiple inheritance features. You can even build entirely new widget primitives from raw C (XS) code then use and re−use that. (Perl 5 is extremely configurable.)

A good introduction to the primitives and how they may be used in conjunction with each other may be found in the `widget` demo script. Note that all the widget demos have a "Show Code" button. To help figure out what is happening in the script you may, when the window appears, edit the text and instrument the code with `print` statements and then simply press "Rerun Demo". Another place to see examples of the primitives (on the web) is at the image supplement to this FAQ at the following URL:

`http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html`[552]

Return to table of contents[553]

---

10. How do I get widget *X* to do *Y* ?

There are a number of tasks that can be accomplished with perl/Tk widgets, configurations, and bindings (a few that can't and a few that require specific tricks). Beginners are encouraged to work through the examples in `UserGuide.pod`[554]. Some examples from `UserGuide.pod`[555] are addressed in this document among those that follow.

Basically a widget can be "created" by simply calling the sub of the same name:

```
my $main = new MainWindow;
```

will set aside the necessary system memory *etc.* for a new `MainWindow` widget (it does not *appear* until after the `MainLoop;` call). The object "created" is then callable via the variable `$main`. So, for example, if you wanted a `Button` in your `MainWindow`, then this:

```
$main−>Button();
```

would be a very basic example of a widget command. If you wanted to later call this button widget

546 doc/scale.htm
547 doc/scrollbar.htm
548 doc/text.htm
549 doc/toplevel.htm
550 doc/HList.htm
551 doc/InpOnly.htm
552 ptkIMG.html
553 #TOC9
554 pod/UserGuide.html
555 pod/UserGuide.html

you would need a "widget tag or ID" to "get a handle on it". Instead of the above call try something like:

```
my $button = $main->Button();
```

The variable `$button` is how you refer to the Button widget in subsequent calls, such as when we call the `pack` routine:

```
$button -> pack;
```

A complete script that incorporates these ideas to make a very plain button would look like:

```
#!/usr/bin/perl -w
use Tk;
use strict;
my $main = new MainWindow;
my $button = $main -> Button();
$button -> pack;
MainLoop;
```

But who wants such a plain looking button? You can provide a number of different widget configurations via calls to the `configure` routine as in:

```
#!/usr/bin/perl -w
use Tk;
use strict;
my $main = new MainWindow;
my $button = $main->Button();
$button -> configure(-text => 'Press me!');
$button -> pack;
MainLoop;
```

The Perl motto is "there is more than one way to do it." – perl/Tk remains quite true to this motto as well. Note that the above script could have been written quite succinctly without the use of either the `$main` or `$button` variables as:

```
#!/usr/bin/perl -w
use Tk;
use strict;
new MainWindow -> Button(-text => 'Press me!') -> pack;
MainLoop;
```

But if you want your widgets to actually *do* things then you must set up callback procedures as discussed later...

Do not overlook the `-` sign in front of some options (like `-text` in the above example) Another commonly overlooked problem is that elements in a hash are supposed to be strings hence a configuration option like `-length +> 5,` really ought to be specified as either `'-length' +> 5,` or `"-length" +> 5,` *etc.*, rather than perl's builtin `length()` function.

Return to table of contents[556]

10.1. How do I get a `Button` to call a Perl subroutine?

[556] #TOC10

You may specify the `-command` option in the call to create & pack the button as in:

```
$main->Button(-text => 'Print',
               -command => sub{do_print($filename, $font)}
               )->pack;
```

Where `sub do_print { }` is a subroutine that handles two arguments and is declared elsewhere in the script. A full script example of the use of the above code is presented in the second example(s) in UserGuide.pod[557]

(Full source code for this[558] and other examples from UserGuide.pod[559] may be found at http://w4.lns.cornell.edu/~pvhp/ptk/pod/.[560] To load code from the web save as a local file say `ex1.pl`, edit the first line to point to your perl interpreter, then change permission: `%chmod u+x ex1.pl`, then execute the script: `%ex1.pl`.)

The above method is called the "anonymous subroutine (closure)" method. As discussed in `Callback.pod`[561] one might have re–written that statement to use the "reference to a sub" method thusly:

```
$main->Button(-text => 'Print',
               -command => [ \&do_print , $filename, $font ]
               )->pack;
```

Note the backslash in front of `\&do_print`. This causes perl to generate a reference to `sub do_print` rather than call it. (thanks Jim Stern :–)

Return to table of contents[562]

10.2. How do I get a `Button` to actively change under my mouse pointer?

You should specify both an `'-image'` and an `'-activeimage'` configuration option either when calling the `->Button()` method or in a later separate call to the `->configure()` method.

Here is an example[563] excerpted from the `basic_demo` script that comes with the Tk kit:

```
#!/usr/local/bin/perl -w

use Tk;

$main = MainWindow->new;

$QPBFile  = "demos/images/QuitPB.xpm";
$QPBaFile = "demos/images/QuitPBa.xpm";

$QuitPB  = $main->Pixmap('-file' => Tk->findINC("$QPBFile"));
$QuitPBa = $main->Pixmap('-file' => Tk->findINC("$QPBaFile"));
```

557 pod/UserGuide.html
558 pod/ex1.pl
559 pod/UserGuide.html
560 pod/
561 pod/Callback.html
562 #TOC10.1
563 etc/but–wave–bye.pl

```
my $but  = $main->Button('-image'       => $QuitPB,
                         '-activeimage' => $QuitPBa,
                         '-command'     => sub { $main->destroy }
                        ) -> pack;

MainLoop;

__END__
```

Return to table of contents[564]

10.3. How do I arrange the layout of my widgets?

To control the layout and appearance of widgets in a window one makes use of a *geometry manager*, as well as −*pad*ding, −*fill*, −*expand*, and −*anchor* options of individual widgets.

A *geometry manager* is any Tk procedure for controlling the arrangement of widgets in your application window. The predominant geometry manager used in both Tcl/Tk and perl/Tk is `pack` also known informally as the "packer" (other geometry managers are the "placer" and the `canvas` widget itself but are much less popular. There is also Nick Ing−Simmon's Table widget[565] [discussed in a later question[566]] and BLT_Table[567] [which made it's way into perl/Tk thanks to Guy Decoux − but is also discussed in a later question[568]]. So far tixForm[569] is for Tcl/Tk only, but a perl/Tk version of Tix is in the works. You can invoke `pack` at the time of widget creation via calls like:

```
$widget->pack;
```

where *widget* can be any of the perl/Tk widget primitives. Widget option lists are usually passed as an associative array (hash) in parentheses thusly:

```
$widget(-option0 => value0,-option1 => value1)->pack;
```

`pack` is often used in conjunction with the `frame` container widget to arrange your widgets much like a hiearchically arranged set of window panes (ultimately in a rectangular "tiling" fashion of sorts). An example of this would be:

```
my $top2 = $main->Toplevel;
my $frame = $top2->Frame;
$frame->pack;
$frame->Label(-text => 'Left2')->pack(-side => 'left');
$frame->Label(-text => 'Right2')->pack(-side => 'right');
$top2->Label(-text => 'Bottom2')->pack(-side => 'bottom');
MainLoop;
```

Note that `pack` itself is given parameters in this example. The *default* behavior for `pack` is equivalent to specifying `-side => 'top'` which can be overridden as in the above example.

564 #TOC10.2
565 #A16.6
566 #A16.6
567 #A16.6
568 #A16.6
569 http://www.cis.upenn.edu/~ioi/tix/doc/man/Form.html

(Full source code for this[570] and other examples from UserGuide.pod[571] may be found at http://w4.lns.cornell.edu/~pvhp/ptk/pod/.[572] To load code from the web save as a local file say `ex2.pl`, edit the first line to point to your perl interpreter, change permission using: `chmod u+x ex2.pl`, then type the name of your script: `ex2.pl`.)

One of the more helpful options to pass to `pack` when trying to get a given widget layout "just right" is through *padding*: either `-padx` or `-pady`. The details of the use of `pad` depend on which specific widget you are trying to `pack`. In fact you can often add the `-pad` in the call to create the widget rather than in the call to `pack`.

There is also the `-anchor` configuration option for widgets. A good introduction to the 9 possible `-anchor` (and `-overanchor`) values is given by the `popup` demo in your perl/Tk build directory.

When setting a widget within a `frame` next to another widget one may wish to make use of the `-fill => '`*style*`'` (where *style* = `none | x | y | both`) options of either `pack` or the widget itself. A typical situation where this is used is in setting up the `Scrollbar` next to a `Canvas` or `Text` widget.

Another aspect to consider when laying out your widgets is their behavior under resize operations (grabbing a part of the window frame and making it bigger or smaller – details depend on your window manager). This may be controlled by the `-expand` option of either `pack` or the widget itself.

Return to table of contents[573]

10.4. How do I get a Popup to popup?

For things like a simple "are you sure?" dialog box you might want to take a look at `Dialog.pm` which is discussed in a later question within this FAQ [16.1].[574]

If you don't wish to require Tk::Dialog, you need something more complicated, or you simply want to create your own independent window with widgets; you must first setup a `Toplevel` in perl/Tk. The fourth example in UserGuide.pod[575] gives a simple example of how to call `Toplevel`. Quoting from that script[576]:

```
my $main = new MainWindow;
fill_window($main, 'Main');
my $top1 = $main->Toplevel;
```

Where `sub fill_window` is declared after the call to `MainLoop;`. When running that script[577] take careful note of which window pops up first, which window has *grab*bed the active attention of your input device(s), and which widget within the active window has the keyboard/mouse *focus* when all three windows are open.

570 pod/ex2.pl
571 pod/UserGuide.html
572 pod/
573 #TOC10.3
574 #A16.1
575 pod/UserGuide.html
576 pod/ex3.pl
577 pod/ex3.pl

The use of `Toplevels` brings up the issue of *grab* – or which independent window is presently "active" and which are activatable. To make a `Toplevel` window active call `grab` thusly:

```
$Top_widget->grab(grab_option);
```

where `$Top_widget` identifies the desired `Toplevel` (it would be either `$top1` or `$top2` in the sample script[578] referred to above). *`grab_option`* could be `-global` – but this is discouraged as a sign of "desparate programming style". To give a `Toplevel` "local grab" you may simply say:

```
$Top_widget->grab;
```

That is, without an argument.

The use of `Toplevels` may also bring up the issue of *focus* – or which window – even which widget within a window – is presently "hot". You may call `focus` on an entire `Toplevel`:

```
$Top_widget->focus;
```

However, `focus` is most often used with individual widgets rather than a whole `Toplevel`.

To *de−iconify* a widget there is in fact a `Popup` function that may be called thusly:

```
$Top_widget->Popup();
```

Return to table of contents[579]

10.5. How do I `bind` keyboard keys?

There are many default key bindings built in to the widgets of perl/Tk. Making proper use of them often involves setting up the right callback. (You may wish to consult the examples in `BindTable.pod`[580] for help with this subject.)

The basic idea is:

```
$widget -> bind('<keyname>' => action);
```

Where `$widget` is the tag or ID of the widget for which the bindings are to hold (note for global bindings you have to `bind` to `<All>`, for semi−global bindings you need to `bind` to all the relevant widgets in your application), '<*keyname*>' can be things like:

```
<Key> or <KeyPress> or <Any-KeyPress>
<KeyRelease>
<Button> or <ButtonPress>
<ButtonRelease>
<Button-1> or <B1>
<Double-1>
<Enter>
<Leave>
<Motion>
```

578 pod/ex3.pl
579 #TOC10.4
580 pod/BindTable.html

To figure out what names perl/Tk uses for such <bindings> use the "binder−finder" on a widget's .pm file. For example, you could find bindings hidden inside of `Button.pm` by typing this at your shell prompt:

```
perl -ne 'print if s/.*(<[^>]*>).*/$1/g;' Button.pm
```

while in the directory where `Button.pm` is located (and if you are not there then simply specify the `/path/to/Button.pm`). Note that due to inheritance (*e.g.*the type of script bindings that are being discussed here) what the binder−finder turns up may not be the last word on a given widget's behaviour. This may be especially true for a widget inside of a compound/composite widget. Note also that the binder−finder will turn up things like <FILEHANDLES> as well as honest <Bindings>. Discrimination in its use is called for (and while your at it you could have just as easily used an editor and actually examined the code directly now couldn't you?).

To get an idea of what the code is for a *key* that you are interested in try running the `xlib_demo` that comes in your perl/Tk build directory. Hold your mouse pointer over the window that appears and simply type the key that you are interested in. The code should appear in the window. If you do not have perl/Tk up and running yet try "*xmodmap −pk*" or look directly at the `/usr/include/X11/keysymdef.h` file where keysym names are given with an `XK_` pre−pended. Do not try things like the Tcl/Tk `%k` symbols in perl scripts. `%K`s will be mis−interpreted as non−existant perl hashes. Instead look at the `Xevent` function.

Ali Corbin <corbin@adsw.fteil.ca.boeing.com>[581] recently posted a great little script for determining keyboard key bindings on a `MainWindow`:

```
#!/usr/local/bin/perl -w
use Tk;
$top = MainWindow->new();
$frame = $top->Frame( -height => '6c', -width => '6c',
                        -background => 'black', -cursor => 'gobbler' );
$frame->pack;
$top->bind( '<Any-KeyPress>' => sub
{
    my($c) = @_;
    my $e = $c->XEvent;
    my( $x, $y, $W, $K, $A ) = ( $e->x, $e->y, $e->K, $e->W, $e->A );

    print "A key was pressed:\n";
    print "  x = $x\n";
    print "  y = $y\n";
    print "  W = $K\n";
    print "  K = $W\n";
    print "  A = $A\n";
} );
MainLoop();
```

To `bind` the action of one widget to that of another try taking a look at the .pm file for the widget of interest − is there a `binding` function already defined? If so you may use it. An example would be the use of "Up" & "Down" `Button`s for a `Listbox`: one could `bind` the `Button`s to call `Tk::Listbox::UpDown`, however, Guy Decoux describes a much more clever way to use the `<Up>` and `<Down>` already defined in `Listbox.pm` (this does not work with Tk−b9.01):

```
#!/usr/local/bin/perl
use Tk;
$top = MainWindow->new;
```

581 mailto:corbin@adsw.fteil.ca.boeing.com

```
$lb = $top->Listbox(-height => 10);
for($i=0; $i < 120; $i++) {
  $lb->insert('end', $i);
}
$f = $top->Frame;
$up = $f->Button(
       -text => "Up",
       -command => [ $lb->bind(ref $lb, '<Up>'), $lb]
       );
$down = $f->Button(
         -text => "Down",
         -command =>sub {&{$lb->bind(ref $lb, '<Down>')}($lb)}
         );
$up->pack(-side => 'left');
$down->pack;
$f->pack;
$lb->pack;
MainLoop;
```

Return to table of contents[582]

10.6. How do I add `bind`ings?

On Fri, 15 Sep 95 10:30:56 BST Nick Ing−Simmons <Nick.Ing−Simmons@tiuk.ti.com> writes:

```
Re: Multiple binds to a single widget?
**************************************

On Thu, 14 Sep 1995 14:57:54 -0400
Alain St <astdenis@cmc.doe.CA> writes:
!In the tcl/tk doc I have, they say that prepending the script
!with '+' appends the new binding to the current one.
!
!How do I do that in perlTk?
!
```

You cannot do that that way (yet?) – one issue is what it would mean to prepend '+' to a perl/Tk callback :

```
$widget->bind('<A>','+',[\&subname,$arg]);
# did not look right to me
```

Other issue is that I would need to manage a list−of−callbacks in glue code.

Bind your new command to a new tag:

```
$widget->bind('Extra',....);
```

And add `Extra` to the widgets bindtags:

```
$widget->bindtags([ref($widget),$widget,'Extra',
                    $widget->toplevel,'all']);
```

Return to table of contents[583]

[582] #TOC10.5
[583] #TOC10.6

10.7. How do I bind the action of a slider (*sic*) to ... ?

Technically speaking they are called `Scrollbars` (not sliders) and one must `configure` the action of the desired widget to call the `Scrollbars` (*i.e.* `bind` is not involved here)

A common task using `Scrollbars` is to `configure` things like `Canvas`, `Listbox`, or a `Text` widgets to be updated (change appearance) when the slider of the acompanying `Scrollbar` is moved by the user.

As an example consider the code that sets up a twelve element `Listbox` and an accompanying vertical `Scrollbar`:

```
my $main = new MainWindow;
my $box = $main->Listbox(-relief => 'sunken',
                         -width => -1, # Shrink to fit
                         -height => 5,
                         -setgrid => 'yes');
my @items = qw(One Two Three Four Five Six Seven
               Eight Nine Ten Eleven Twelve);
foreach (@items) {
   $box->insert('end', $_);
}
my $scroll = $main->Scrollbar(-command => ['yview', $box]);
```

So far so good. But merely setting them up does not mean that the `Listbox` even knows that the `Scrollbar` is lying next to it. Note that the scalar variable `$scroll` is how we refer to the `Scrollbar`, thus, hooking the `$box` up to handle `$scroll` events is a matter of `configur` ation:

```
$box->configure(-yscrollcommand => ['set', $scroll]);
```

A complete script that makes use of this code (and adds the necessary calls to `pack` and `MainLoop;`) is given as the fifth example[584] in UserGuide.pod[585] (and may be found at http://w4.lns.cornell.edu/~pvhp/ptk/pod/.[586])

There was an old Perl/Tk tendency to have a bunch of `Scrl`*`Foo`* widgets (such as `ScrlListbox`). The use of such widgets is now deprecated in favor of a new `Scrolled` class, as in:

```
$w = $patent->Scrolled('Text',...);
```

The widgets that can be `->Scrolled()` include:

- `Canvas (::Axis)`
- `Entry`
- `Ghostview`
- `HList`
- `HTML (::Web)`
- `Listbox`

584 pod/ex4.pl
585 pod/UserGuide.html
586 pod/

- Pod
- Text (::ROText) (::TextUndo)
- Tiler

Return to table of contents[587]

10.8. How do I `configure` a `Scrollbar` to scroll multiple widgets?

Note that the widget type that you wish to scroll can be important as a scroll "unit" on a `Text` or `Listbox` may be a character (several pixels − depending on font) whereas it would be an X "units" on a `Canvas` (could be pixel − but you may also specify other units).

A concrete answer for scrolling 3 `Listbox`es comes courtesy of Frederick L. Wagner <derf@ti.com>:

From a working example of multi−xscrolling:

```
sub multiscrollx
{  # multiscrollx
 my ($sb,$wigs,@args) = @ARG;
 my $w;
 foreach $w (@$wigs)
 {
   $w->xview(@args);
 }
}  # multiscrollx

# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  $sh->configure( -command => [ \&multiscrollx, $sh,
                      [$scratchrule,$ruleheader,$ruletable]]);
  $ruletable->configure(  -xscrollcommand => [ 'set', $sh]);
  $ruleheader->configure( -xscrollcommand => [ 'set', $sh]);
  $scratchrule->configure(-xscrollcommand => [ 'set', $sh]);
```

In this case,
`$sh` is a horizontal `Scrollbar`,
`$ruletable` and `$scratchrule` are `Table`s
`$ruleheader` is an `Entry`

However, this approach is good for any widget with X−scrolling capability, I think. So the Y counterpart should be:

```
sub multiscrolly
{  # multiscrolly
 my ($sb,$wigs,@args) = @ARG;
 my $w;
 foreach $w (@$wigs)
 {
   $w->yview(@args);
 }
}  # multiscrolly

# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

[587] #TOC10.7

```
    $sv->configure( -command => [ \&multiscrolly, $sv,
                                [$l1,$l2,$l3]]);
    $l1->configure( -yscrollcommand => [ 'set', $sv]);
    $l2->configure( -yscrollcommand => [ 'set', $sv]);
    $l3->configure( -yscrollcommand => [ 'set', $sv]);
```

Hope that helps.

Greg VanSickle [<vansickl@bnr.ca>](mailto:vansickl@bnr.ca)[588] points out that this little script snippet does not provide for the binding of `'<Button-2<'` that he is accustomed to. He wrote a package called DSListbox[589] to address this binding issue.

Conversely, Jong Park asked how to setup multiple `Scrollbars` to scroll the same widget. Nick Ing−Simmon's reply makes use of an anonymous `sub` and can be summed up in a little script that scrolls a `Text` widget (to see the scrolling in action type more than 20 lines of text into the widget):

```
#!/usr/local/bin/perl -w

use Tk;
my $mw = MainWindow->new();

my $s1 = $mw->Scrollbar(-orient => 'vertical');
my $s2 = $mw->Scrollbar(-orient => 'vertical');

$s1->pack(-side => 'left', -fill => 'y');
my $t = $mw->Text(
    -yscrollcommand =>  sub{$s1->set(@_), $s2->set(@_)},
    -wrap           => 'word',
    -width          => 70,
    -height         => 20,
    -font           => $font,
    -setgrid        => 1,
)->pack(-side => 'left');
$s2->pack(-side => 'right', -fill => 'y');
$s1->configure(-command => [$t => 'yview']);
$s2->configure(-command => [$t => 'yview']);

MainLoop;

__END__
```

Return to table of contents[590]

10.9. How do I display a bitmap?

You can display X bitmaps on your widgets with the `-bitmap` configuration option. Typically `-bitmaps` are configured into `Label`, `Frame`, `Button`, *etc.* widgets (`Canvas` widgets are another story however see question [11.1] below[591]). In order to emphasize the bitmap option itself let us assume we were specifying a bitmap for a `Label` with a call like:

```
$main->Label(-bitmap => 'bitmap-name')->pack;
```

588 mailto:vansickl@bnr.ca
589 etc/DSListbox.tar.gz
590 #TOC10.8
591 #A11.1

Where *`bitmap-name`* could be any of the built in Tk bitmaps: `error`, `gray25`, `gray50`, `hourglass`, `info`, `question`, `questhead`, `warning` (see the `widget` demo for a full list).

In order to use some of the bitmaps in the `perl5/Tk/demos/images/` directory you would specify a fuller path name like:

```
    $main->Label(-bitmap => "\@$tk_library/demos/images/face")->pack;
```

Note the escaped `"\@"` on the directory specification (as well as the use of the `$tk_library` variable imported by `use Tk;`). If you wanted to specify a file called `foobar.xbm` in the directory where you were running the script then either:

```
    $main->Label(-bitmap => '@foobar.xbm')->pack;
#or
    $main->Label(-bitmap => "\@foobar.xbm")->pack;
```

should work just fine. In another directory however that would be a problem. So something like:

```
    $main->Label(-bitmap => "\@$ENV{'HOME'}/img/foobar.xbm")->pack;
```

will help someone who has an `img/foobar.xbm` file in their $HOME directory. If you don't mind the non−portability then hard−wiring in the full path name will help as well. (Or if you have write access then put your files in `Tk/demos/images/` *e.g.*)

Return to table of contents[592]

10.10. How do I display an image?

You will want to get a "`Photo`" handle on the file as in the following example where `'imggif'` is the Photo handle for a gif file that is distributed with perl/Tk:

```
    #!/usr/bin/perl -w
    use strict;
    use Tk;
    my $main = new MainWindow;

    $main ->Label(-text => 'Main')->pack;
    $main -> Photo('imggif',
                   -file => "$Tk::tk_library/demos/images/earth.gif");
    my $l = $main->Label('-image' => 'imggif')->pack;

    $main->Button(-text => 'close',
                  -command => sub{destroy $main}
                  )->pack(-side => 'left');
    $main->Button(-text => 'exit',
                  -command => [sub{exit}]
                  )->pack(-side => 'right');
    MainLoop;
```

(`Canvas` widgets are another story however see question a later question within this FAQ[593]).

Return to table of contents[594]

---

[592] #TOC10.9
[593] #A11.3
[594] #TOC10.10

10.11. What Image types are available?

In addition to the **Tk builtin bitmaps** there is support for reading images from files in formats such as: X11 Bitmaps (.xbm), X Pixmaps (.xpm), and Graphics Inline Format (.gif). See the `CrtPhImgFmt` man page for more info (if you have Tk 4.*X* installed). (In order to support other formats you might also consider running through a netpbm filter.)

For perl generation of images see the question (later in this FAQ) on graphics modules.[595]

Return to table of contents[596]

10.12. Is there any way to have more than one `Listbox` contain a selection?

To allow more than one `Listbox` to contain a "selection", (or at least a highlighted item – which need not be the actual selection) specify the configuration option:

```
-exportselection => 0
```

which will dis–associate `Listbox`'s selection from X selection (only one window can have X selection at a time).

Here[597] is a rather simple script that illustrates what happens when only one `Listbox` has `-exportselection => 0` specified:

```
#!/usr/bin/perl -w

use Tk;

my $main = MainWindow->new;

my @fruits = ('Apple','Banana','Cherry','Date','Elderberry','Fig');
my @nuts   = qw(Almond Brazil Chestnut Doughnut Elmnut Filbert);

my $fruit_list = $main->Listbox();
for (@fruits) { $fruit_list -> insert('end',$_); }
$fruit_list->pack();
my $fruitprint_button = $main->Button(
                          -text => "print selection",
                          -command => sub{ printthem($fruit_list) }
                                      )->pack;

my $nut_list = $main->Listbox(
                              -selectmode => 'multiple',
                              -exportselection => 0,
                             )->pack;
for (@nuts) { $nut_list -> insert('end',$_); }
my $nutprint_button = $main->Button(
                          -text => "print selection(s)",
                          -command => sub{ printthem($nut_list) }
                                      )->pack;
```

595 #A24
596 #TOC10.11
597 etc/lb–basic

```
my $quit_button = $main->Button(-text => "quit program",
                                -command => sub{exit},
                                )->pack();

MainLoop;

sub printthem {
    my $list = shift;
    my @entries = $list->curselection;
    for (@entries) { print $list -> get($_),"\n";}
}
```

For a more extensive example of `Listbox` usage combined with some perl data structure exploitation see the script at:

http://w4.lns.cornell.edu/~pvhp/ptk/etc/lb-constructor[598]

Return to table of contents[599]

10.13. How do I select a range of tags in a `Text` widget?

A question arose concerning getting a range of selections from a `Text` widget. Nick Ing−Simmons'[600] answer mentions several possibilities including:

Keyboard Copy/Paste 'is' implemented of course...

```
Subj:   RE: $Text->tag('ranges', 'sel') - does this work?

In <199512291957.OAA02609@ohm.nrl.navy.mil>
On Fri, 29 Dec 1995 14:57:42 -0500
Charles J Williams <chas@ohm.nrl.navy.mil> writes:
!I was writing a little tk perl today, and i decided to try to
!implement a copy/paste using the 'sel' tag
!
!I enabled exportselection, and then try to probe the select
!region with:
!
!    $buffer = $text->tag('ranges', 'sel');
!
!$buffer comes back with one entry, the end of the selection.
```

That is to be expected − the scalar gets assigned the last element of the list.

```
!I tried:
!
!    @buffer = $text->tag('ranges', 'sel');
!
!same difference.
```

This seems to work for me:

```
($start,$end) = $text->tagRanges('sel');
```

598 etc/lb−constructor
599 #TOC10.12
600 mailto:nik@tiuk.ti.com

In perl/Tk `->tagRanges(...)` is an alias for `->tag('ranges',...)`

The following subroutine can also probe and print the tagRanges:

```
sub showsel
{
 my $text = @_;
 my @info = $text->tagRanges('sel');
 if (@info)
  {
   print "start=$info[0] end=$info[1]\n"
  }
}
```

Return to table of contents[601]

10.14. How do I group `Radiobutton`s together?

Specify the `-variable` option on each one. Here is an example pulled from the `icon.pl` demo script:

```
 $letters = '';
 my $w_frame_left_b3 = $w_frame_left->Radiobutton(
     -bitmap   => "\@$tk_library/demos/images/letters",
     -variable => \$letters,
     -value    => 'full',
 );
 my $w_frame_left_b4 = $w_frame_left->Radiobutton(
     -bitmap   => "\@$tk_library/demos/images/noletters",
     -variable => \$letters,
     -value    => 'empty',
 );
```

Return to table of contents[602]

10.15. How do I specify fonts?

The quick answer is to specify the `font` configuration option of your widget as in:

```
#!/usr/local/bin/perl -w
use Tk;
$main = MainWindow->new();
$labl = $main -> Label('-text' => "Foo", '-font' => "fixed");
$labl -> pack;
MainLoop;
```

The long answer involves figuring out what fonts you have access to locally. The Unix programs *xlsfonts* and *xfontsel* are useful in this regard.

The perl/Tk version of *xfontsel* was distributed as the `font_test` script in the Tk build directory.

601 #TOC10.13
602 #TOC10.14

See also the later question (within this FAQ) on international fonts.[603]

Return to table of contents[604]

10.16. How do I `get` the entry in an `Entry`?

You want to call `get` on the return value of the widget itself. Here is how it may be used in a simplified version of example 1.1 from the Tk::UserGuide[605] where a `Button` is set up to call a `sub` where the call to `get` lies:

```
#!/usr/bin/perl -w
use strict;
use Tk;

my $main = MainWindow -> new();
my $entry = $main -> Entry();
$entry -> pack;
$main->Button(-text => 'Print',
              -command => sub{do_print($entry)}
              )->pack;
MainLoop;

sub do_print {
    my ($widget) = @_;
    my $entered = $widget -> get();
    print "The string \"$entered\" was entered.\n";
}
```

Return to table of contents[606]

10.17. How do I hide a password `Entry`?

Set the `-show` option to zero, as in this example:

```
$entry = $form->Entry(-textvariable => \$user_entry,
                      -show => 0);
```

Return to table of contents[607]

10.18. How do I limit an `Entry`'s insertion width?

Nick Ing-Simmons recommends writing a new `Entry` widget with the `insert` method appropriately overridden by one that does limit the width. His code is avaialable as a separate package from:

    http://w4.lns.cornell.edu/~pvhp/ptk/etc/LEntry-0_00.tar.gz[608]

603 #A22
604 #TOC10.15
605 pod/UserGuide.html
606 #TOC10.16
607 #TOC10.17
608 etc/LEntry-0_00.tar.gz

Now Brent Powers points out a possible problem with that approach and recommends an `insert()` method as follows:

```
Date: Thu, 22 Aug 1996 10:32:44 -0400
From: "Brent B. Powers" <powers@ml.com>
Subject: Re: How to set max characters for Entry widget
In-reply-to: <199608211445.PAA09248@pluto>

Ummm, before we set this into the distribution or FAQ, maybe we should
make it work properly.  An example:  Imagine maxwidth configured to 8,
the user fills in ABCDEFGH, moves the cursor back 4 places, and types
I.  The SUPER::insert call sets the string to ABCDIEFGH, which this
code then modifies to ABCDIEFG.

Hmmm, how about

sub insert {
  my($w, @args) = @_;
  my($max) = $w->cget(-maxwidth);
  my($sval) = $w->get;
  if (length($sval) >= $max) {
     $w->SUPER::insert(@args);
     if (length($w->get) > length($sval) {
    ## Reject it;
    my($idx) = $w->index('insert'); #  get current cursor position
        $w->delete(0, 'end');
        $w->insert(0, $sval);
        $w->icursor($idx);
    $w->bell;
  } else {
     $w->SUPER::insert(@args);
  }
}

Of course, that still doesn't deal with the selection, but ...
```

To which Nick Ing−Simmons responded (Thu Aug 22 1996):

```
'paste' and <ButtonRelease-2> call insert method, what other selection
issues are there?
```

Return to table of contents[609]

10.19. How do I obtain `Menu`s that do not tear off?

Nick Ing−Simmons[610] outlined a couple of ways to achieve this result. The critical feature being the `-tearoff => 0` configuration option of the `Menu`. In Nick's[611] words:

```
my $mb = $parent->Menubutton(...);     # The button
my $menu = $mb->Menu(-tearoff => 0);  # Create a non-tearoff menu
$mb->configure(-menu => $menu);       # Tell button to use it.
$mb->command(....);
```

Above is for clarity − you can loose `$menu` variable:

[609] #TOC10.18
[610] mailto:nik@tiuk.ti.com
[611] mailto:nik@tiuk.ti.com

```
my $mb = $parent->Menubutton(...);
$mb->configure(-menu => $mb->Menu(-tearoff => 0));
$mb->command(....);
```

Return to table of contents[612]

---

11. How do I get a Canvas to ... ?

The Canvas widget is the most configurable and versatile. With versatility comes complication however, and it is certainly deserving of its own special section within this FAQ...

You might also see the examples in the widget demo especially the "canvas item types" selection (which runs the items.pl demo script).

Return to table of contents[613]

11.1. Display a bitmap?

Unlike other widgets the Canvas does not take the -bitmap configuration option. One of the ways to place things – including bitmaps – onto a Canvas is to call create on it. To emphasize how a Canvas handles bitmaps differently from the configurable widgets let me assume that you wanted to specify the 'hourglass' built in bitmap in the following. (For more on xbm file specification see a previous question [10.9] within this FAQ[614].) Here is a way to combine the Canvas; and create; calls:

```
my($canvar) = $main->Canvas();
my($bittag) = $canvar->create('bitmap',10,10, -bitmap=>'hourglass');
$canvar->pack;
```

You can also create an image that will display a bitmap (plus a whole lot more):

```
my($canvar) = $main->Canvas();
my($bitmap) = $main->Bitmap(-data => $data);
my($bittag) = $canvar->create(qw(image 10 10), -image => $bitmap);
$canvar->pack;
MainLoop;
```

Return to table of contents[615]

11.2. Erase a display?

To erase something like a bitmap call delete on the item. Assuming your Canvas tag is $canvar and your item tag it $bittag (as in the previous [11.1] question[616]) then the call proceeds like:

[612] #TOC10.19
[613] #TOC11
[614] #A10.9
[615] #TOC11.1
[616] #A11.1

```
$canvar -> delete($bittag);
```

This is of course useful in a callback. For example to configure a `Button` to do your deletion for you you could say something like:

```
$main->Button(-text   => 'clear',
              -command=>sub{$canvar -> delete($bittag)}
             )->pack;
```

To remove an entire `MainWindow()` call the `withdraw()` method:

```
$main -> withdraw;
```

Return to table of contents[617]

11.3. Display an Image?

Just as for the other widget types there is a two step process of first getting a "`Photo`" handle on the file of interest. For the `Canvas` (unlike the other widgets) one then makes a call to `create` an `image` as in the following example where `'IMG'` is the Photo handle for a GIF file that comes distributed with the Tk kit (it just happens to be handled in this example *via* the scalar variable `$img`):

```
#!/usr/bin/perl -w
use strict;
use Tk;
my $main = new MainWindow;
my $canvar = $main ->Canvas;
$canvar->pack;
my $file = 'demos/images/earth.gif';
my $img =
 $canvar->Photo( 'IMG',
                 -file => Tk->findINC($file) );

$canvar->create( 'image',0,0,
                 '-anchor' => 'nw',
                 '-image'  => $img );

MainLoop;
__END__
```

Return to table of contents[618]

11.4. What things can be `created` on a `Canvas`?

The following types can be used in `$canvar->create` calls:

```
arc        sections of circle
bitmap     for X11 bitmap files/builtins
image      for Photo image types (gif, xpm, xbm, ...)
line
oval       includes circles
```

617 #TOC11.2
618 #TOC11.3

```
polygon    may be -filled
rectangle  may also be -filled
text       similar to Text widget primitive
window     allows embeddding of other widgets
```

Return to table of contents[619]

11.5. How do I redraw a `line` on a `Canvas`?

By calling the `->coord` method on the item as in the following example:

```
#!/usr/bin/perl
use Tk;
$m = MainWindow->new;
$c = $m -> Canvas;
$i = $c -> create('line', 0,0 => 50,50 );
$c -> pack;
$b = $m -> Button('-text' => 'extend',
                  '-command' => sub{push_it($c,$i)},
                 )->pack;
MainLoop;

sub push_it {
    my ($canvas, $line) = @_;
    $canvas -> coords($line, 0,0 => 100,100 );
}
```

Thanks to Christopher Dunn and Harry Bochner <bochner@das.harvard.edu> for providing this question and answer.

Return to table of contents[620]

11.6. How do I use the `Canvas` as a geometry manager?

In a call to `create` a `window` (or anything) on your `Canvas` you need to specify its position – this is in part how a `Canvas` can be used as a geometry manager. *e.g.*:

```
my($bittag) = $canvar->create('bitmap',10,10, -bitmap=>'hourglass');
```

Specifies the x=10, y=10 screen pixel location (from the upper left). Other possible units are:

tag	unit	example
	pixels	25,50 # i.e. no unit tag at all
m	milliimeters	10c,20c
c	centimeters	1c,2c
p	points (1/72")	35p,70p

There can be a great deal more to it than just units, however. Note the following question posed and answered by Eric J. Bohm.

```
Eric J. Bohm <bohm@cs.buffalo.edu> wrote:
```

619 #TOC11.4
620 #TOC11.5

```
!I've got a row of entries packed side by side in a frame.
!These frames are packed on top of each other.
!So, when someone deletes a row, the lower ones bubble
!up automatically.  This works just fine and dandy, and let me
!extend my thanks to our brave and energetic pTk team.
!
!The trick here is what widget do I put this in so that
!it will be scrollable when I have too many rows to
!fit on the screen?
  [details and complaints]
```

Following up to my own message here.

All right, after several false leads, I spent 3 hours fighting a canvas widget and pounding my head against the canvas.html doc, until I finally understood how to include my entries in a frame in a window in the canvas and get things to scroll nicely.

Turns out that the whole thing isn't all that hard to do once I understood how canvas widgets work.

Not sure if its of general interest, but here's the snippet, which was stolen from the items demo inside the widget_lib and then brutally hacked.

Perhaps a simpler demo would have been easier to use as a guide, but I got there eventually, so my thanks for the widget demo.

```
#----------------------------------------
my $c = $w_frame->Canvas();
$c->configure(
     -height       => '300',
     -width        => '600',
     -relief       => 'sunken',
     -bd => 2,
 );
 my $w_frame_vscroll = $w_frame->Scrollbar(
                                    -command => ['yview', $c]
                                         );
 my $w_frame_hscroll = $w_frame->Scrollbar(
                                    -orient => 'horiz',
                                    -command => ['xview', $c]
                                          );
 $c->configure(-xscrollcommand => ['set', $w_frame_hscroll]);
 $c->configure(-yscrollcommand => ['set', $w_frame_vscroll]);
 $w_frame_hscroll->pack(-side => 'bottom', -fill => 'x');
 $w_frame_vscroll->pack(-side => 'right', -fill => 'y');
 $c->pack(-expand => 'yes', -fill => 'both',-side=>'top');
 my $entryframe=$c->Frame;
 my $c_win= create $c 'window','0','0',
                       -window=>$entryframe,
                       -anchor=>'nw';
 #----------------------------------------
```

Where `$c -> configure( -scrollregion => [$top, $left, $right, $bottom])` can be used to size things nicely once you find out how big it'll be.

And the widgets you want scrolled should be slaves of `$entryframe`.

Vastly more robust than anything I had running in the BLT Table.

EJB

Return to table of contents[621]

11.7. How do I get a `Canvas` to output PostScript(c)?

Many thanks to Tom Oelke <tpo9617@rit.edu>[622] for providing this question, answer & snippet of code:

> The following section of code gets the postscript code for the section of canvas that's top−left corner is at $min_x, $min_y, and has a width and height equivalent to the displayed region. This ps code is then piped out to lpr to be printed.

```
my $ps = $canvas->postscript( '-x' => $min_x,
                              '-y' => $min_y,
                              -width => $canv->Width,
                             -height => $canv->Height);
open (PS, "| lpr"); # customize with -Pname e.g.
print PS $ps;
close (PS);
```

Whereas you would use something like:

```
open (PS, ">file.ps"); # to output to a file
print PS $ps;
close (PS);
```

Return to table of contents[623]

11.8. How do I get a PostScript(c) output of a `Canvas` w/ widgets?

In general you don't. You can't do it in Tcl/Tk either (if that is any consolation). Nick Ing−Simmons[624] posted an explicit discussion of what is involved:

```
Subj: RE: Canvases and postscript output

On Tue, 28 Nov 95 14:37:09 PST
Davis <morry@dsg.tandem.com> writes:
! I have a canvas with text and some entry widgets that I want to create
!postscript from. I used the
!widget->postscript( -file => 'ld.ps', -colormode  => 'gray');
!the file gets created but its empty. Is there some other options I need?
```

> Core Tk cannot write postscript for embedded windows, the best it could do would be to grab a Pixmap of the window as displayed. This is fine if the window is visible, but if it is scrolled off screen or under another application there is no pixmap.
>
> Only complete fix is to have a `->postscript` method for every possible widget which can render un−mapped widgets. This is non−trivial task.

621 #TOC11.6
622 mailto:tpo9617@rit.edu
623 #TOC11.7
624 mailto:nik@tiuk.ti.com

```
!Also I have a scrollbar for this canvas and when I scroll  the entry widget
!actually scroll part way out of the frame the canvas is in. Why does this
!happen and can I fix it?
```

The `Entry` widgets need to be descendants of the canvas or they just get clipped to their parent.

Return to table of contents[625]

11.9. How do I get the size of a `Canvas`? After a re–size?

```
$canvas->cget(-width);
```

simply returns the size of the canvas when it was created, whereas

```
$canvas->Width;
```

will get the answer even after a re–size. Substitute `[Hh]eight` for `[Ww]idth` in the above if that is what you want.

Nick Ing–Simmons points out that if you want to have your `Canvas` be able to grow to arbitrarily large sizes be sure to specify the `-expand` or `-fill` options when you `->pack` the `Canvas`.

Return to table of contents[626]

11.10. How do I `bind` different actions to different areas of the same `Canvas`?

KOBAYASI Hiroaki <kobayasi@sowa.is.uec.ac.jp> recently posted an extraordinary little script that addresses this question quite succinctly:

How about this?
I don't know whether this is a good solution or not.
but it works under Tk–b9 + perl5.002b1f.

```
#!/usr/local/bin/perl -w
use Tk;

($c = MainWindow->new->Canvas)->
   pack(-fill => 'both', -expand => 1);
# to survive under Tk-b8.
# You don't need paren before pack in b9.

($pop1 = $c->Menu)->command(-label => "FOO");
($pop2 = $c->Menu)->command(-label => "BAR");

$c->create(oval => 0, 0, 100, 100,
           -fill => 'black',
```

625 #TOC11.8
626 #TOC11.9

```
            -tags => ['popup']);

$c->Tk::bind($c, '<3>', [\&PopupOnlyThis, $pop1]);
$c->bind('popup', '<3>', [\&PopupOnlyThis, $pop2]);

sub PopupOnlyThis {
    print "@_\n";
    my($c, $pop) = @_;

    # to prevent multiple popup.
    Tk->break if defined $Tk::popup;

    my $e = $c->XEvent;
    $pop->Popup($e->X, $e->Y);
    # Tk::Menu::Popup sets $Tk::popup.

}
MainLoop;

$Tk::popup = undef; # to kill warning.

__END__
```

Return to table of contents[627]

---

12. Common Problems.

Everything in Tk−land is a reference. When defining callbacks take care to pass variables by reference. Callbacks are closures and to ensure a variable gets its current value, as opposed to its value when the callback is defined, pass by reference, *e.g.*:

```
$frog = 123;
$b = $mw->Button(
    -text    => 'Push Me',
    -command => [
        sub {
           my($widget, $frog) = @ARG;
           print STDERR "widget=$widget!\n";
           print STDERR "frog=$$frog!\n";
        }, $mw, \$frog,
     ],
); # end Button definition
```

If `$frog` is not passed by reference the `print` statement will always output "`123`" (actually, the `print` as it exists will print nothing since it's trying to dereference `$frog`, which presumably is now not a reference). Note that by definition all perl/Tk widgets are already references, since they're simply Perl objects, and that's why you do not have to print `$$widget`!

A good "reference" for handling references and dereferencing are the `perlref(1)`[628] and `perlobj(1)`[629] man pages. A good "reference" for the various data types you will encounter in this kind of perl programming is Tom Christiansen's[630] Perl Data Structures Cookbook[631] which is now available as the perldsc(1)[632] man page.

627 #TOC11.10
628 http://www.metronet.com/0/perlinfo/perl5/manual/perlref.html
629 http://www.metronet.com/0/perlinfo/perl5/manual/perlobj.html
630 mailto:tchrist@perl.com
631 http://www.perl.com/perl/pdsc/index.html
632 http://www.metronet.com/0/perlinfo/perl5/manual/perldsc.html

Also beware the traps[633] that befall perl4 programmers in making the move to perl 5. References for this include the new perltrap(1)[634] man page as well as William Middleton's perl425 trap document at:

`http://www.perl.com/perl/all_about/perl425.html`[635]

or

`http://w4.lns.cornell.edu/~pvhp/ptk/misc/perl425.html`[636]

Return to table of contents[637]

12.1. What do the `->`, => and `::` symbols mean?

The `->` is the "infix dereference operator". In other words it is the means by which one calls a `sub` with a pass by reference (among other things you can do with `->`). As stated above most things in calls to perl/Tk routines are passed by reference. The `->` is used in perl just as in C or C++. (Most of the widget primitives are elements of the `Tk::` "perl class".) A simple example of dereferencing would be:

```
$x = { def => bar };  # $x is a reference to an anon. hash
print $x->{def},"\n"; # prints ``bar''
```

Note that in the case of calling perl/Tk `subs` there may be more than one way to call by reference. Compare

```
my($top) = MainWindow->new;
```

with

```
my($top) = new MainWindow;
```

But in general you will be making extensive use of calls like:

```
$top -> Widge-type;
```

There is a clear and succint discussion of references, dereferences, and even closures in `man perlref(1)`[638] or see the perl 5 info page at:

`http://www.metronet.com/perlinfo/perl5.html`[639]

The use of the => operator is quite common in perl/Tk scripts. Quoting from `man perlop(1)`[640]:

> The => digraph is simply a synonym for the comma operator. It's useful for documenting arguments that come in pairs.

You could say that => is used for aesthetic or organizational reasons. Note in the following how hard it is to keep track of whether or not every `-option` has an `argument`:

[633] http://www.metronet.com/0/perlinfo/perl5/manual/perltrap.html
[634] http://www.metronet.com/0/perlinfo/perl5/manual/perltrap.html
[635] http://www.perl.com/perl/all_about/perl425.html
[636] misc/perl425.html
[637] #TOC12
[638] http://www.metronet.com/0/perlinfo/perl5/manual/perlref.html
[639] http://www.metronet.com/perlinfo/perl5.html
[640] http://www.metronet.com/0/perlinfo/perl5/manual/perlop.html

```
$query -> Button(-in,\$reply,-side,'left',-padx,2m,-pady,
 2m,-ipadx,2m,-ipady,1m)->pack(-side,'bottom');
```

As opposed to:

```
$query ->Button( -in => \$reply,
                 -side => 'left',
                 -padx => 2m,
                 -pady => 2m,
                 -ipadx => 2m,
                 -ipady => 1m
                )->pack(-side => 'bottom');
```

By the way if you wanted the numeric "greater than or equal" you would use >= not =>.

While the `::` symbol can be thought of as similar to the period in a C struct, it is much more akin to the `::` class scope operator in C++:

```
a.b.c;       /* something in C */
a::b::c();   // function in C++
$a::b::c;    # a scalar in Perl 5
@a::b::c;    # a list in Perl 5
%a::b::c;    # an associative array or "hash" in Perl 5
&a::b::c;    # a function in Perl 5
```

It is also analogous to the single forward quotation mark in perl 4:

```
$main'foo;   # a $foo scalar in perl 4
$main::foo;  # a $foo scalar in Perl 5
```

For backward compatibility perl 5 allows you to refer to `$main'foo` but `$main::foo` is recommended.

Return to table of contents[641]

12.2. What happened to the ampersands `&`?

Perl 4 programmers especially may be surprised to find that as of Perl 5.0 the ampersand `&` may be omitted in a call to a subroutine if the subroutine has been declared before being used. Actually you can even get around the declare before omit ampersand rule by using the `subs.pm` pragma, or by pre−declaring (without defining) as in a script like:

```
#!/usr/bin/perl -w
use strict;
use Tk;
sub Mysub;  #pre-declare allows calling Mysub()

...Other main/Tk stuff -
        including call to Mysub() sans &...

sub Mysub {

    ...Mysub stuff...

}
```

[641] #TOC12.1

Note however that one place the `\&` *reference* is sometimes used in perl/Tk in the setting up a callback for a widget. Other references are possible: *e.g.* `\$foo` is a reference to the scalar variable `$foo` (this was true even under perl 4).

Return to table of contents[642]

12.3. What happened to the quotation marks?

Perl 4 programmers especially may be surprised to find a serious dearth of quotation marks around strings in perl 5 scripts such as in perl/Tk. The "rules have been relaxed" somewhat for the use of quotation marks. Basically it is OK to leave them out if the context of the string in question is unambiguous. However, it never hurts to leave them in and may help readability.

Here is Larry Wall's synopsis of the string situation:

Newsgroups:
: comp.lang.perl.misc[643]

Subject:
: Re: To string or not to string...

```
In article <4e49fv$j0u@panix3.panix.com>,
Andy Finkenstadt <genie@panix.com> wrote:
! Back when I was learning perl (after receiving a review copy of
! learning perl, and buying the real perl book, each from ORA),
! I always got bit by when I needed to use "strings" and when
! I could get away with bare_words within braces for associative
! arrays.  (Yes, this is under 4.036 if it matters.)
!
! the most typical example would be:
!
! When must I use $assoc{"trailer"} and when can I get away with
! $assoc{trailer}?   Similarly, $ENV{CONTENT_LENGTH} versus
! $ENV{"CONTENT_LENGTH"}?  Unfortunately sometimes my strings
! end up being numbers in their own right, i.e.:  $message{"0"}
! or $msg=0; $message{$msg}.  Which is more appropriate,
! which are merely stylistic, and which are stricly perl5
! features now that I'm upgrading most of my installations
! of perl.
```

> Perl 4 let you use a "bareword" for a string if it had no other interpretation. It would warn you under −w if you used a word consisting entirely of lower−case characters, since such a word might gain an interpretation someday as a keyword.
>
> Perl 5 still works the same way, but with several twists.
>
> 1.) Since you can now call predeclared subroutines as though they were builtins, you have to worry about collisions with subroutine names too. However...
> 2.) You can completely disallow the default interpretation of barewords by saying "use strict subs", which requires any such bareword to be a predeclared subroutine. But...
> 3.) Overriding all that, Perl 5 (in recent versions) will FORCE string interpretation of any bare

642 #TOC12.2
643 news:comp.lang.perl.misc

identifier used where a single hash subscript is expected, either within curlies or before a =>. (Those are the places you might usually want the old barewords anyway.)

The upshot of these rules is that you can write Perl 5 with much less punctuation than Perl 4, yet also with less ambiguity. If you so choose.

Larry

Tcl programmers should note that in Perl the single quotation marks `''` act much as the curly brace `{}` enclosure does in Tcl (no escaping special characters `$@\` *etc.*). Whereas the double quotation marks `""` allow for substitution of `$variables` (the rules are a little different between Tcl and Perl however).

Note also that a frequently seen short hand in perl5/Tk scripts is the @list returned by qw():

```
@list = qw(zoom schwartz bufigliano);
```

which is equivalent to:

```
@list = split(' ','zoom schwartz bufigliano');
```

or more simply:

```
@list = ('zoom','schwartz','bufigliano');
```

i.e. the `qw/STRING/` @list is not equivalent to the quotation marks provided by `q/STRING/`, `qq/STRING/`, or `qq(STRING)` ...

There are, ironically enough, situations in perl/Tk where one needs to use quotation marks as in the following by post by <a904209@pluto.tiuk.ti.com>:

```
 Paul Wickman wrote in article <4b4o0fINNlu8@CS.UTK.EDU>:
!
!    Why does the following statement work fine:
!
!$day->pack(-before => $year, -side => 'left');
!
!    But the below generates the given error:
!
!$day->pack(-after => $year, -side => 'left');
!
!Ambiguous use of after => resolved to "after" => at line 191.
!
```

Because there is a sub `after` in scope, probably imported from Tk via `use Tk;`.

There are two workrounds:

```
use Tk qw(MainLoop exit ...); # just ones you use
```

or

```
$day->pack('-after' => $year, -side => 'left');
```

12.4. Must I use "my" on all my variables?

If you `use strict;` (as recommended) the answer is "probably". This confines the variables names to your namespace – so your variable does not conflict with one in the module(s) your are using (you are at the least `useing Tk;`). `my` does "lexical scoping" on a variable rather than the "dynamic scoping" done by `local` (like `auto` variables in C). The difference between these two is that the scope of `my $var` is confined to the block (`sub`, `if`, `foreach`, *etc.*) in which it is declared and used, as opposedto `local $iable` which can propogate to all blocks called by the block in which it is declared. In general the confined scope of `my $var` means that its use will proceed quicker and more efficiently than `local $iable`.

If you give a *fully qualified* variable name such as

```
$main::var = 1;  # No "my" needed
```

Then no `my $var` is needed. However, the lexical scoping of `my $var` makes it preferable.

If you choose to use `my` (as recommended) then beware that you should declare a variable `my` only at the first use (instantiation) of a variable. Consider yet another way to re–write the "Hello World!" script:

```
#!/usr/local/bin/perl -w
use strict;
use Tk;
my $main = new MainWindow;
my $label = $main->Label(-text => 'Hello World!');
my $button = $main->Button(-text => 'Quit',
                           -command => sub{exit});
$label->pack;  #no "my" necessary here
$button->pack; #or here
MainLoop;
```

Considering the finite number of names (in particular the high probability that a variable named $label or $button was used in one or more of the extensions to perl that you may be using) it helps one's programming to `use strict;` and declare variables yours alone with `my`.

James M. Stern points out that redundant `my` declarations are not simply useless they can be dangerous as in the following script which will not work:

```
#!/usr/local/bin/perl -w
use strict;
use Tk;
my $main = new MainWindow;
my $label = $main->Label(-text => 'Hello World!');
my $main;   #WRONG: this $main overrides previous
my $button = $main->Button(-text => 'Quit', #will now fail
                           -command => sub{exit});
$label->pack;
$button->pack;
MainLoop;
```

644 #TOC12.3

12.5. Is there a way to find out what is in my perl/Tk "PATH"?

Presuming this question is asking for a little more than the answer you get when you type:

```
ls perl5/lib/Tk/*.pm
```

there are ways to find out what gets EXPORTED by Tk.pm. Use a script like:

```
#!/usr/bin/perl

use Tk;
require 'dumpvar.pl';

dumpvar('Tk');
```

or more succintly at the shell command prompt:

```
perl -e 'use Tk; require "dumpvar.pl"; dumpvar("Tk");'
```

The advantage of using `dumpvar` over *ls* is that it gives you a *brief* summary of all the arguments your widgets want. Note that the output is many lines and you may wish to pipe through *more* or *less*.

If you wish to determine the Configuration options a given widget accepts (and what the values are at a given point in a script) you may use the `->configure` method with no arguments to retrieve the list of lists, as in this example:

```
#!/usr/bin/perl

use Tk;
my $main = MainWindow -> new;
my $scrl = $main -> Scrollbar('-orient' => 'horizontal');

@scrollconfig = $scrl -> configure;
for (@scrollconfig) {
    print "@$_\n";
}

etc.
```

Such code is useful for development but is probably best left out, commented out, or switched out of "production line" code.

Return to table of contents[646]

12.6. What is the difference between `use` and `require`?

The short answer is that something like:

```
use Tk;
```

[645] #TOC12.4
[646] #TOC12.5

is equivalent to:

```
BEGIN { require "Tk.pm"; import Tk; }
```

Hence the essential difference is that a mere `require Tk;` does not achieve the `import` of function/method names. The significance of this is that it allows one to call `->Button` rather than having to call the fully qualified `->Tk::Button` *e.g.*. For further details on this subject see `man perlmod(1)`[647] or see Tom Christiansen's[648] document at:

ftp://ftp.perl.com/perl/info/everything_to_know/use_vs_require[649]

Return to table of contents[650]

12.7. How do I change the cursor/color?

Nick Ing–Simmons <nik@tiuk.ti.com>[651] and others posted a series of answers to this type of question. In summary what they said was:

Basically

```
$mw->configure(-cursor => ... );
```

Unless you use one of built–in cursors it gets messy.

Here copy of what `Tk/demos/color_editor` does:

```
#!/usr/local/bin/perl -w
use Tk;
my $mw = MainWindow->new;
$mw->configure(-cursor => ['@' . Tk->findINC('demos/images/cursor.xbm'),
                                 Tk->findINC('demos/images/cursor.mask'),
                                  'red', 'green']);
MainLoop;
```

That says that argument to `-cursor` is a list of 4 things:

1. . Pathname to bitmap with `'@'` prepended to say it isn't a built in name (Using `findINC` to locate file relative to Tk install location.)
2. . Pathname to mask bitmap (no `@` required)
3. . Foreground colour
4. . Background colour

```
! I want to remap it for the MainWindow
! and will be using a pixmap.
```

You won't be using a Pixmap with normal X11. X11 allows *bitmap* with optional mask (another bitmap), and two colours.

647 http://www.metronet.com/perlinfo/perl5/manual/perlmod.html
648 mailto:tchrist@mox.perl.com
649 ftp://ftp.perl.com/perl/info/everything_to_know/use_vs_require
650 #TOC12.6
651 mailto:nik@tiuk.ti.com

The optional nature of the mask means that a simple call with a list reference like:

```
$mw->configure(-cursor => ['watch', 'red', 'blue']);
```

should work alright.

You may also obtain the value of the default cursor for a widget using something like `->optionGet`.

Return to table of contents[652]

12.8. How do I ring the bell?

The short answer is

```
$widget -> bell;
```

A slightly longer answer might include a fully functioning script:

```
#!/usr/bin/perl
use Tk;
$main = MainWindow -> new;
$butn = $main->Button(-text => 'bell')
$butn->configure(-command => sub{ $butn->bell; });
$butn->pack();
MainLoop;
```

An even longer answer would be a fully functioning script with a callback[653]:

```
#!/usr/bin/perl
use Tk;
$main = MainWindow -> new;
$but = $main->Button(-text => 'bell',
                     -command => sub{ringit($main)})->pack;
MainLoop;

sub ringit {
    my $m = shift;
    $m->bell;
}
```

Simon Galton <galtons@candu.aecl.ca>[654] reminds us to be careful in that

> some systems remap this [the "console bell"] to anything from a digital sound to a flash on the screen.

Return to table of contents[655]

12.9. How do I determine the version of perl/Tk that I am running?

652 #TOC12.7
653 etc/bellit
654 mailto:galtons@candu.aecl.ca
655 #TOC12.8

With an up to date perl installation one may query the local perl setup and all extensions via the command:

```
perldoc perllocal
```

For the Tk extension: version numbering has changed recently and determining the version of perl/Tk that you are running now depends on what version you are running:

Tk−b10++:
Tk−b10 (and higher) has changed to `$Tk::VERSION` (rather than the older "`$Tk:Version`") to be consistent with other packages. Hence a short succinct way to tell which version you have installed (that works with Tk−b11 and Tk400.200) is:

```
perl -MTk -e 'print $Tk::VERSION."\n"'
```

Tk−b9.01:
The version numbers as of Tk−b9.01 are stored in the following variables:

```
Core Tk version : $Tk::version
Tk patchLevel :   $Tk::patchLevel
library :         $Tk::library
perl/Tk Version : $Tk::Version
```

At your shell prompt you could say something like the following to determine you perl/Tk Version:

```
perl -e 'use Tk; print "$Tk::Version\n";'
```

The switch to Tk−b9.01 from previous versions included a large number of method name changes. Nick was kind enough to include a `b9names` script in the distribution that assists with the job of updating your older scripts. See the `b9names` script for a rather complete discussion of the name changes. Geoffroy Ville also posted a notice of some of the changes. Here is a brief (and very incomplete!) summary:

```
older                          Tk-b9.01++
packslaves                     pack('slaves')
packpropagate                  pack('propagate')
packForget                     pack('forget')
                               pack('info')

$w->delete if ($w);             $w->destroy if ($w);
```

Tk−b8(−−):
A little script (Tk_module)[656] can tell you and return the value:

```
#!/usr/bin/perl
use Tk;
local(*Tk_m) = \$Tk::Tk_module;
print "$Tk_m\n";
```

Or more succintly say something like the following (at your shell prompt):

```
perl -e 'use Tk; print "$Tk::Tk_module\n";'
```

[656] etc/Tk_module

You can obtain the version of **Tk** in use with the following (at your shell prompt):

```
perl -e 'use Tk; print "$Tk::tk_version\n";'
```

where this command returned "4.0" when the previous one (or Tk_module[657]) returned "b8".

All Tk versions:
Don't forget that you can always determine your Perl version/patchlevel/*etc.* with:

```
perl -v
```

(at the shell prompt – it's actually a little harder to get as much information from within a #!script.) As of perl 5.002 you can use `perl -V` to determine your perl `Config`uration.

OZAWA Sakuro <ozawa@prince.pe.u–tokyo.ac.jp>[658] points out some ways to do it in a script:

1. `'$]'` holds the version number.
2. In Perl5, `'require NUMBER;'` will complain if version is younger than NUMBER. (*e.g.* `require 5.001;`)
3. Of course, newly imported (and incompatible) features in newer scripts will bailout before execution if parsed by an old interpreter.

Note that if you `use English;` then `$PERL_VERSION` holds the version number.

To determine your MakeMaker version number try something like this (5.002):

```
perl -MExtUtils::MakeMaker -e 'print "$ExtUtils::MakeMaker::VERSION\n";'
```

or this (5.001m ok):

```
perl -e 'use ExtUtils::MakeMaker;print"$ExtUtils::MakeMaker::VERSION\n";'
```

or even this (older perls and MakeMakers):

```
perl -e 'use ExtUtils::MakeMaker;print"$ExtUtils::MakeMaker::Version\n";'
```

Please note that thoughout this FAQ document there are references to things like **Tk–b10(++)** or **Tk–b10++** which roughly translated to `use English;` means something like "I think this will work with this version of Tk and (maybe) higher versions...". You might also see **Tk–b8(––)** which means something like "it worked with that old version and probably worked with prior versions and if you are stuck with an old Tk version you might have to do it this way...".

Return to table of contents[659]

12.10. How do I call perl from C?

You need to see a recent copy of the `perlembed(1)`[660] pod page. By "recent" it needs to be up to date with at least perl5.002.

---

657 etc/Tk_module
658 mailtp:ozawa@prince.pe.u–tokyo.ac.jp
659 #TOC12.9
660 etc/perlembed.html

Borrowing from Jon Orwant's preamble to that document:

Do you want to:

Use C from Perl?
: Read (at least) the `perlcall(1)`[661], the `perlapi(1)`[662], the `perlxs(1)`[663], the `perlxstut(1)`[664], and the `perlguts(1)`[665] manpages.

Use C++ from Perl?
: Recent changes to MakeMaker will make this easier. Be sure you are familiar with the `perlcall(1)`[666], the `perlapi(1)`[667], the `perlxs(1)`[668], the `perlxstut(1)`[669], and the `perlguts(1)`[670] manpages.

Use an executable program from Perl?
: Read about backquotes ``` `` ```, `system()`, and `exec()` built in perl functions. Try reading the `perlfunc(1)`[671] manpage.

Use Perl from Perl?
: Read about `do`, `eval`, `use` and `require`. The `perlfunc(1)`[672] manpage discusses these. For complete scripts you may also make use of the backquotes ``` `` ```, `system()`, or `exec()` built in perl functions, but you may take a performance hit in doing so (see `perlfunc(1)`[673] for information).

Use C from C?
: Rethink your design.

Use C++ from C++?
: See previous.

Use Perl from C?
: Read (at least) the `perlembed(1)`[674] and the `perlguts(1)`[675] manpages.

Use Perl from C++?
: Read (at least) the `perlembed(1)`[676] and the `perlguts(1)`[677] manpages.

There is also an Doug MacEachern's <dougm@osf.org>[678] embedder's development kit on CPAN and at a URL of the following form:

```
    http://www.osf.org/~dougm/perl/Devel-embed-*.tar.gz
or
    http://www.perl.com/cgi-bin/cpan_mod?module=ExtUtils::embed
```
[679] [680]

Return to table of contents[681]

661 http://www.metronet.com/0/perlinfo/perl5/manual/perlcall.html
662 http://www.metronet.com/0/perlinfo/perl5/manual/perlapi.html
663 http://www.metronet.com/0/perlinfo/perl5/manual/perlxs.html
664 http://www.metronet.com/0/perlinfo/perl5/manual/perlxstut.html
665 http://www.metronet.com/0/perlinfo/perl5/manual/perlguts.html
666 http://www.metronet.com/0/perlinfo/perl5/manual/perlcall.html
667 http://www.metronet.com/0/perlinfo/perl5/manual/perlapi.html
668 http://www.metronet.com/0/perlinfo/perl5/manual/perlxs.html
669 http://www.metronet.com/0/perlinfo/perl5/manual/perlxstut.html
670 http://www.metronet.com/0/perlinfo/perl5/manual/perlguts.html
671 http://www.metronet.com/0/perlinfo/perl5/manual/perlfunc.html
672 http://www.metronet.com/0/perlinfo/perl5/manual/perlfunc.html
673 http://www.metronet.com/0/perlinfo/perl5/manual/perlfunc.html
674 etc/perlembed.html
675 http://www.metronet.com/0/perlinfo/perl5/manual/perlguts.html
676 etc/perlembed.html
677 http://www.metronet.com/0/perlinfo/perl5/manual/perlguts.html
678 mailto:dougm@osf.org
679 http://www.osf.org/~dougm/perl/
680 http://www.perl.com/cgi-bin/cpan_mod?module=ExtUtils::embed
681 #TOC12.10

12.11. How do I call Tcl code from perl/Tk?

Assuming that you have a pressing need to call Tcl from perl/Tk then one "official way" to so would be via the `->send()`[682] and the `->Receive()`[683] methods.

It is also worth noting that you can still have access to a complete Tcl script from perl via the perl `system`, or ` `` ` (backtick), or even `exec` mechanisms. Just be careful with I/O waits and return values if you try one of these approaches. Further suggestions may be found in the various `perlipc` files at:

ftp://ftp.perl.com/perl/info/everything_to_know/[684]

A more satisfactory Tcl/Tk-`wish`-like behavior can be embedded in perl by making appropriate modifications to Dov Grobgeld's perl script[685] that uses sockets for perl<->wish communication:

```
#!/usr/local/bin/perl
#####################################################################
#  An example of calling wish as a subshell under Perl and
#  interactively communicating with it through sockets.
#
#  The script is directly based on Gustaf Neumann's perlwafe script.
#
#  Dov Grobgeld dov@menora.weizmann.ac.il[686]
#  1993-05-17
#####################################################################

    $wishbin = "/usr/local/bin/wish";

    die "socketpair unsuccessful: $!!\n" unless socketpair(W0,WISH,1,1,0);
    if ($pid=fork) {
            select(WISH); $| = 1;
            select(STDOUT);

        # Create some TCL procedures
            print WISH 'proc echo {s} {puts stdout $s; flush stdout}',"\n";

        # Create the widgets
        print WISH <<TCL;
        # This is a comment "inside" wish

        frame .f -relief raised -border 1 -bg green
        pack append . .f {top fill expand}

        button .f.button-pressme -text "Press me" -command {
            echo "That's nice."
        }
        button .f.button-quit -text quit -command {
            echo "quit"
        }
        pack append .f .f.button-pressme {top fill expand} \\
                       .f.button-quit {top expand}
TCL
        # Here is the main loop which receives and sends commands
        # to wish.
```

682 doc/send.htm
683 doc/send.htm
684 ftp://ftp.perl.com/perl/info/everything_to_know/
685 etc/perl_wish
686 mailto:dov@menora.weizmann.ac.il

```
        while (<WISH>) {
            chop;
            print "Wish sais: <$_>\n";
            if (/^quit/) { print WISH "destroy .\n"; last; }
        }
            wait;
    } elsif (defined $pid) {
        open(STDOUT, ">&W0");
        open(STDIN, ">&W0");
        close(W0);
        select(STDOUT); $| = 1;
        exec "$wishbin --";
    } else {
        die "fork error: $!\n";
    }
```

Ilya Zakharevich <ilya@math.ohio−state.edu>[687] has a "ptcl.h" header file for the construction of tcl bindings from pTk (there are limitations to this approach). It was posted to the mailing list archive at:

    http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0057.html[688]

If you absolutely must pass large amounts of pre−parsed data between Tcl and perl then perhaps you should look into Malcolm Beattie's Tcl/Tk extensions to Perl instead. Those modules are distrubuted at CPAN sites.[689] As mentioned above running Tcl/Tk/perl is incompatible with running perl/Tk.

Return to table of contents[690]

---

13. What are some of the primary differences between Tcl/Tk and Perl/Tk?

Considering that both interpreters/(compiler) for Tcl and Perl were written in C for original use on Unix computers it is not surprising that there are some similarities between the two languages.

Nevertheless, there are a large number of differences between the Tcl language and the Perl language. One thing to keep in mind is that to build, install, and use Perl/Tk one does not need to have Tcl/Tk on hand at all. Perl/Tk is completely independent of Tcl/Tk.

Tom Christiansen (a definite perl proponent) has put up a web page that elucidates some critical technical differences between Tcl and Perl at:

    http://www.perl.com/perl/versus/tcl−discussion.html[691]

Within each language there is Tk − a widget **T**ool**k**it. One must be careful that some of the Tcl/Tk widget names and options have been modified slightly in the perl/Tk language.

With Tk−b9.01 (and higher) a great many functions (method calls actually) start with an upper case letter and continue with all lower case letters (*e.g.* there is a perl/Tk `Entry` widget but no `entry` widget), and many configuration options are all lower case (*e.g.* there is a perl/Tk `highlightthickness` option but no `highlightThickness` option). Thus if you are having trouble converting a script check your typing. (there is a script `b9names` to help). There is also a

687 mailto:ilya@math.ohio−state.edu
688 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0057.html
689 #CPAN
690 #TOC12.11
691 http://www.perl.com/perl/versus/tcl−discussion.html

`tcl2perl` script (discussed later).

The html docs that get created during the build of perl/Tk ought to help clarify most any language difference. While the following table does not cover all the differences it is hoped that it will prove useful, especially to those people coming from a primarily Tcl/Tk programming background. These are some of the common Tcl−>Perl stumbling points:

```
what               Tcl/Tk                 Perl/Tk
variable           set a 123              $a = 123; or $a = '123';
 initialization
re−assignment      set b $a               $b = $a;

lists/arrays       set a {1 2 fred 7.8}   @a = (1,2,'fred',7.8);
re−assignment      list set b $a          @b = @a;

associative        set a(Jan) 456.02      %a = ('Jan',456.02,'Feb',534.96);
 arrays            set a(Feb) 534.96
re−assignment      foreach i \            %b = %a;
                    [array names a] {
                    set b($i) = $a($i) }

Note on the above examples:
In Tcl the scalar, list, and array variable 'a' will overwrite each
previous assignment.
In Perl $a, @a, %a are all distinct (occupy separate namespaces).

expressions        set a [expr $b+$c]     $a = $b+$c;

increment          incr i                 $i++; or ++$i;

declare            proc plus {a b} {      sub plus { my($a,$b) = @_;
 subroutines        expr $a + $b }         $a+$b; }

variable scope     local default          global default
                   override w/ "global"   override w/ "my" (or "local")

call               plus 1 2               &plus(1,2); #or
 subroutines                              plus(1,2);  #OK after sub plus

statement sep      newline or at ";"      ";" required

statement          "\" − newline          none required
 continuation

verbatim strings   {}                     ''
 e.g.              {a \ lot@ of $stuff}   'a \ lot@ of $stuff'

escaped strings    ""                     ""
 e.g.              "Who\nWhat\nIdunno"    "Who\nWhat\nIdunno"

STDOUT             puts "Hello World!"    print "Hello World!\n"
                   puts stdout "Hello!"   print STDOUT "Hello!\n"
```

Note also that Tcl/Tk has a built−in abbreviation completion mechanism that lets you specify short hand, *e.g.*

```
canvas .frame.canvas −yscrollcommand ".frame.scroll set" ; #Tcl/Tk OK
canvas .frame.canvas −yscroll ".frame.scroll set" ;        #Tcl/Tk also OK
$canvas=$main−>Canvas(−yscroll => ['set',$scroll]);  #ERROR perl/Tk
$canvas=$main−>Canvas(−yscrollcommand => ['set',$scroll]); #perl/Tk OK
```

You may get around this with the perl `abbrev.pl` package in certain circumstances. For example:

```
   require 'abbrev.pl';
   %foo = ();
   &abbrev(*foo,'-yscrollcommand');
 ...
   $canvas=$main->Canvas($foo{'-yscroll'} => ['set',$scroll]); #perl/Tk OK
```

In Perl you can emulate the Tcl `unknown` proc (through the perl `AUTOLOAD` mechanism) as follows:

```
    use Shell;
    print($p = man(-k => bitmap));
```

Which is equivalent to what you would get if you typed:

```
    man -k bitmap
```

From within `tclsh` or `wish`. (Thanks to Ilya Zakharevich <ilya@math.ohio−state.edu>[692] for pointing out this feature. ;−)

Return to table of contents[693]

---

14. How do I install new scripts | modules | extensions?

(Thanks to Ilya Zakharevich <ilya@math.ohio−state.edu>[694] for pointing out that perl code comes in a variety of flavors and some code requires more work than others to install. Hence I have expanded this topic and will refer to three distinct categories here: **Scripts Modules** and **Extensions**:)

Scripts

A "self−contained" script needs little modification (in principle!) to run. It is a good idea to check the `#!` line at the very top of the file to reflect your local perl setup (*e.g.* `#!/usr/bin/perl -w` (*change to*) `#!/usr/gnu/local/perl -w` or what have you). There are allegedly "more portable" ways to invoke the perl interpretor as well − they are more fully documented in the perl FAQ[695] and the `perlrun(1)`[696] man page, however.

Other things you do not want to forget when trying to run a perl script include giving yourself permission to do so, *e.g.*:

```
    chmod u+x newscriptname
```

You also want to be sure your `DISPLAY` environment variable is set up properly when attempting to run a perl/Tk script. You may also need to look at the `xhost(1)` or the `xauth(1)` man pages for setting up your X−display properly.

If you are still experiencing difficulty check to be sure that extraneous /newsgroup|e−mail|HTML

692 mailto:ilya@math.ohio−state.edu
693 #TOC13
694 mailto:ilya@math.ohio−state.edu
695 http://www.perl.com/perl/faq/
696 http://www.metronet.com/perlinfo/perl5/manual/perlrun.html

headers|footers|markup//; are not in the file and that you have on hand all that is `require`ed or `use` ed by the script (if not you may need to install a module – or even a perl4 style *lib*`.pl` file).

Modules

Check out the module – make sure it is OK and will run on your system – does it require a specific location? For testing purposes (always a good idea) or if you do not have root priveleges set the file in a directory that you do have write access to and try to include it in a test script. Assuming you have a module to test called "`Foo.pm`" and are simply running the test script in the same directory as the module begin by adding to the @INC array like so:

```
#!/usr/bin/perl -w
 BEGIN { @INC = ("$ENV{'PWD'}",@INC); }
 use Tk;
 use Foo;
```

or

```
#!/usr/bin/perl -w
 use lib $ENV{PWD};
 use Tk;
 use Foo;
```

Another approach is to set either the PERLLIB or PERL5LIB environment variable from your shell. This method allows invoking the test script from within a number of different directories without having to edit a hard coded `use lib` or `push(@INC,".")` kind of statement within the script. Yet another way to do it is with the `-I` switch on the command line like so:

```
perl -Ipath/to/Foo -e fooscriptname
```

After a successful test; if you are a system administrator, or have root priveleges, or are modifying your own copy of **perl**; then copy it to the `perl5/Tk` directory. Depending on how the module was written it should be possible to use it either with the `use Tk;` statement itself or with an explicit `use Tk::Foo;` (for module `perl5/Tk/Foo.pm`).

Extensions (Overgrown Modules)

These may come as a multi–file kit (tape archive usually) and may require a C compiler for part of the installation (perl/Tk itself falls into this category). You know you have an **Overgrown Module (Extension)** when there is one or more files with an `.xs` extension (perl–>C meta code) and a `Makefile.PL` (perl–>make meta code). One invokes the perl `MakeMaker` on the file called `Makefile.PL` in order to create a `Makefile` via:

```
perl Makefile.PL
```

You may now run `make` on the resultant `Makefile` – but the details of this process are module dependent and should be documented in a `README` or an `INSTALL` file. A very standard perl extension requires 4 (or 5 if making static) standard commands to make and install:

```
perl Makefile.PL
make
make test
make install
```

If you have the appropriate CPAN and FTP modules already installed you can retrieve a module from CPAN[697] and carry out all of the above steps with a perl one–liner like this:

```
perl −MCPAN −e 'install "Foo"'
```

Return to table of contents[698]

---

15. How do I write new modules?

You might want to start by poking around your perl/Tk build directory. Is there something there that already does what you want? Is there something that is reasonably close – but only requires minor modification?

Next go through the various perl documents – including the FAQ[699] as well as the various relevant `man` pages: `perlmod(1)`[700], `perlobj(1)`[701], `perlbot(1)`[702], (and please don't forget: `perlpod(1)`[703]!)

Post your idea to comp.lang.perl.tk[704] and discuss it with others – there might very well be someone working on an approach already. A clear explanation of all the stuff that gets put into a module was posted to the mailing list and can be found in the archive at:

`http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.10/0012.html`[705]

Also, be sure to check out a recent version of the official Module List[706] that Tim Bunce <Tim.Bunce@ig.co.uk>[707] and Andreas Koenig <a.koenig@franz.ww.TU–Berlin.DE>[708] maintain and post to comp.lang.perl.announce[709] periodically. The list is also available at any CPAN ftp site[710] as well as:

`http://www.perl.com/CPAN/modules/00modlist.long.html`[711]
`ftp://rtfm.mit.edu/pub/usenet/news.answers/perl−faq/module−list`[712]
`ftp://ftp.demon.co.uk/pub/perl/db/mod/module−list.txt`[713]
`ftp://ftp.wpi.edu/perl5/Modules/module_list.txt`[714]

Finally ready to ship? Small (*perl/Tk*) modules have been posted directly to comp.lang.perl.tk[715]. Big modules may require ftp distribution (see upload info at one of the CPAN sites[716]) then make your announcement to comp.lang.perl.tk[717] and possibly to comp.lang.perl.announce[718].

Return to table of contents[719]

---

697 #CPAN
698 #TOC14
699 http://www.perl.com/perl/faq/
700 http://www.metronet.com/0/perlinfo/perl5/manual/perlmod.html
701 http://www.metronet.com/0/perlinfo/perl5/manual/perlobj.html
702 http://www.metronet.com/0/perlinfo/perl5/manual/perlbot.html
703 http://www.metronet.com/0/perlinfo/perl5/manual/perlpod.html
704 news:comp.lang.perl.tk
705 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.10/0012.html
706 http://www.perl.com/CPAN/modules/00modlist.long.html
707 mailto:Tim.Bunce@ig.co.uk
708 mailto:a.koenig@franz.ww.TU−Berlin.DE
709 news:comp.lang.perl.announce
710 #CPAN
711 http://www.perl.com/CPAN/modules/00modlist.long.html
712 ftp://rtfm.mit.edu/pub/usenet/news.answers/perl−faq/module−list
713 ftp://ftp.demon.co.uk/pub/perl/db/mod/module−list.txt
714 ftp://ftp.wpi.edu/perl5/Modules/module_list.txt
715 news:comp.lang.perl.tk
716 #CPAN
717 news:comp.lang.perl.tk
718 news:comp.lang.perl.announce
719 #TOC15

16. Composite Widgets.

Composite widgets[720] combine the functions of two or more widget primitives into something that is not quite a stand alone program but is something that may prove very useful for inclusion in your own scripts. A variety of composite widgets have been written and many are still being worked on. Many come bundled with your perl/Tk distribution kit, and some are simply posted to comp.lang.perl.tk[721]. It is quite common to have composite widgets written in perl modules – usually in terms of the Tk widget primitives. Graphical examples of some of the composites discussed here can be seen by GUI browsers at:

`http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html`[722]

Return to table of contents[723]

16.1. How do I get a `Dialog` box?

For things like a simple "are you sure?" dialog box you might want to take a look at `perl5/Tk/Dialog.pm`. This module may be invoked with `require Tk::Dialog;` *etc.* – there are much more extensive directions inside the comment fields at the top of the `Dialog.pm` file itself. The module has a lot of options and has a tutorial driver script in `perl5/Tk/demos/dialog`. Dialog.pm is also used by the `perl5/Tk/demos/widget` demo. In particular look at `perl5/Tk/demos/widget_lib/dialog1.pl` and `dialog2.pl` for examples of how one makes use of Tk::Dialog. A snippet of a script that uses this module could look like:

```
    require Tk::Dialog;

    my $mw = MainWindow->new;
    my $D = $mw->Dialog(
                 -title => 'Are you sure?',
                 -text  => "You have requested rm \*\nAre you sure?",
                 -default_button => 'No',
                 -buttons        => ['No','yes']
                       );
    my $choice = $D->Show;  # use Show for Tk-b9.01
# if using Tk-b8:    my $choice = $D->show;

    print " you chose $choice \n";
```

A question concerning configuration of the Subwidgets on the Dialogs came up recently:

```
<Greg_Cockerham@avanticorp.com> wrote:
! I want to reconfigure the colors of the Dialog and
! ErrorDialog buttons.  How do I do this?
! Thanks in advance.

   $dialog_widget->configure(-background => 'purple');
```

[720] pod/Composite.html
[721] news:comp.lang.perl.tk
[722] ptkIMG.html
[723] #TOC16

Since these two widgets are composites you manage them like any 'ol widget. If the default delegate subwidget(s) aren't to your liking you can always get to individual component widgets of the composite via the `->Subwidget()` method.

I see these subwidgets:

Dialog
: 'message' is the label subwidget with the dialog text, and 'bitmap' is the label subwidget showing the dialog bitmap

ErrorDialog
: 'error_dialog' is Dialog subwidget, 'text' is text subwidget

You can even do things like this:

```
$error_dialog->Subwidget('error_dialog')->
                    Subwidget('bitmap')->configure(..);
```

to "get to" the label widget of the dialog component of the error_dialog widget.....

Be sure to also check out the "`dialog`" demo.

Return to table of contents[724]

16.2. Is there a file selector?

Yes, there may be several eventually...

One distributed with the perl/Tk code kit itself is called FileSelect.pm[725] and was written by Frederick L. Wagner − (based on an original by Klaus Lichtenwalder).

Another module called SelFile.pm was adapted by Alan Louis Scheinine from Wagner's FileSelect.pm. It is available from:

`http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0122.html`[726]
or
`http://w4.lns.cornell.edu/~pvhp/ptk/etc/SelFile.pm`[727]

A module called FileSave.pm allows one to type in a new (non−existant) filename for "Save as..." type operations. It was posted by Mark Elston on 12 Oct 1995 to the mailing list and is available from:

`http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.10/0093.html`[728]
or
`http://w4.lns.cornell.edu/~pvhp/ptk/etc/FileSave.pm`[729]

A slightly different behaviour is to be had with Brent B. Powers' `FileDialog.pm` that was posted to the mailing list on 12 Jan 1996 and available from:

724 #TOC16.1
725 etc/FileSelect.html
726 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.11/0122.html
727 etc/SelFile.pm
728 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.10/0093.html
729 etc/FileSave.pm

http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.12/0201.html[730]

or

http://w4.lns.cornell.edu/~pvhp/ptk/etc/FileDialog.pm[731]

Harry Bochner chimed in with `SaveAs.pm`. It is available from:

http://w4.lns.cornell.edu/~pvhp/ptk/etc/SaveAs.pm[732]

In general, if there is a feature that you want missing from one of these, or some behaviour that you would like to see modified then by all means `cp` the source code to your area and start hacking ;–)

Return to table of contents[733]

16.3. Is there a color editor?

There is. Please see

```
perldoc ColorEditor.pm
```

or run the `Tk/demos/color_editor` demo script for more information.

Return to table of contents[734]

16.4. Is there a round Scale?

It is not quite a "round Scale" but Roy Johnson has written "Dial.pm" for round dial (or speedometer) –like settable widgets. It is available from:

http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.08/0431.html[735]

or

http://w4.lns.cornell.edu/~pvhp/ptk/etc/Dial.pm[736]

As well as from the `Contrib/` sub–directory of your perl/Tk build directory.

Return to table of contents[737]

16.5. Is there something equivalent to `tkerror`?

Yes there is. Please see the `Tk/ErrorDialog.pm` module for further information.

Return to table of contents[738]

730 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.12/0201.html
731 etc/FileDialog.pm
732 etc/SaveAs.pm
733 #TOC16.2
734 #TOC16.3
735 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.08/0431.html
736 etc/Dial.pm
737 #TOC16.4
738 #TOC16.5

16.6. Are there Tables?

There are least two:

Nick's Table

Nick Ing−Simmons has distributed his own Table widget package with Tk−b9.01 (and higher). It is used through a `use TK::Table;` and `$top->Table();` calls. A rather detailed demo of this widget/geometry manager's capabilities can be found in the `table_demo` script (in your `Tk-b9.01/` build directory). There is also pod in the `perl5/Tk/Table.pm` file. You may also browse the perl Tk::Table man page on the web at

http://w4.lns.cornell.edu/~pvhp/ptk/etc/Table.pm.html[739]

Guy Decoux's BLT_Table

Guy Decoux <decoux@moulon.inra.fr>[740] has ported the popular BLT_Table Tcl/Tk tabular geometry manager to perl/Tk. It was known to work with Tk−b8. You may obtain the latest version of it either from

ftp://moulon.inra.fr/pub/pTk/[741]

or from a CPAN site[742] in the authors/id/GUYDX/[743] directory. You may also browse the perl BLT_Table man page on the web at

http://w4.lns.cornell.edu/~pvhp/ptk/etc/Table.html[744]

Return to table of contents[745]

---

17. Programming/development tools.

There are a number of tools and methods to help you with your perl/Tk scripting and development. It is worthwhile to note here that the `-w` switch is recommended as is the `use strict;` statement near the top of your script/program. If it dies and you still cannot decrypt the error message that these generate take a look though `man perldiag(1)`[746].

Return to table of contents[747]

17.1 Is there a Tcl/Tk to perl/Tk translator?

Nick Ing−Simmons has written a (rather lengthy) `tcl2perl` script. It is distributed with the perl/Tk

739 etc/Table.pm.html
740 mailto:decoux@moulon.inra.fr
741 ftp://moulon.inra.fr/pub/pTk/
742 #CPAN
743 http://www.perl.com/CPAN/authors/id/GUYDX/
744 etc/Table.html
745 #TOC16.6
746 http://www.metronet.com/0/perlinfo/perl5/manual/perldiag.html
747 #TOC17

build kit. Please handle carefully! (translation: do not expect it to translate arbitrary tcl code accurately nor even into the most efficient perl/Tk equivalent. Do go over the converted script with care – and do not forget `-w` and `use strict;`.) Thanks Nick :–)

Return to table of contents[748]

17.2 Is there something equivalent to `wish` in perl/Tk?

The answer is yes.

The idea of `wish` is that you read from `<STDIN>` and evaluate each statement. The standard way to do this in perl/Tk is to use the `tkpsh` script that comes in your perl/Tk build directory. Another elegant way to get `wish` like behavior in perl/Tk is to use **rmt** which you can find in `perl5/Tk/demos` in your perl/Tk distribution. When you run **rmt** you already have Tk.pm set up for you so you can start typing things like `$mmm = new MainWindow;` *etc.* at the `rmt:` prompt. (This use belies the power of **rmt** which is derived from Ousterhout's Tcl/Tk version of **rmt** [see section 27.2 of his book]. **rmt** is capable of "inserting Tk code" into simultaneously running Tk applications.)

A cruder way to get `wish`–like behaviour with perl/Tk is to run a "perl shell" and type in your usual commands, including `use Tk;` *etc.* There is a script distributed with perl called `perlsh` which is written quite simply as:

```
#!/usr/bin/perl
 $/ = '';        # set paragraph mode
 $SHlinesep = "\n";
 while ($SHcmd = <>) {
     $/ = $SHlinesep;
     eval $SHcmd; print $@ || "\n";
     $SHlinesep = $/; $/ = '';
 }
```

You can use this during code development to test out little snippets of code. It helps to be an accurate typist and the `use strict;` is optional here :–)

KOBAYASI Hiroaki has a more sophisticated `wish` like perl/Tk "shell" that is called EVA. It is available from:

> ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/eva-*.tar.gz[749]

Return to table of contents[750]

17.3. Is there a debugger specifically for perl/Tk?

Not for the latest version – but the `-w` switch and `use strict;` are always helpful with debugging as they provide informative error messages.

748 #TOC17.1
749 ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/
750 #TOC17.2

You can, of course, run under the standard perl debugger using the `-d` switch like so:

```
perl -d myscript
```

But it is recommended that you set you breakpoints carefully since just the calls to `ManWindow->new` require many steps.

(Older information): Gurusamy Sarathy <gsar@engin.umich.edu>[751] had built a PERL5DB file called Tkperldb (which despite the name is for pTk not Tk/perl). One must install an early de−bugger then apply a patch to bring the debugger up to date. The early debugger kit was available from:

ftp://ftp.perl.com/pub/perl/ext/TK/Tkperldb-*.tar.gz[752]

And Gurusamy Sarathy notes that the patch to bring the debugger up to date is available at:

```
You need a post 5.001m perl that has support for debugging closures.
Or you can simply apply:
```

http://www-personal.umich.edu/~gsar/perl5.001m-bugs.patch[753]

```
to 5.001m. (5.002beta includes all the fixes in the above patch).
```

Note that a perl debugger may be invoked within your script with a line like:

```
$ENV{'PERL5DB'} = 'BEGIN { require Tkperldb }';
```

See `man perldebug(1)`[754] for more help.

Keep in mind that you are programming in perl after all. The perl debug line mode is available to you through executing the following from your shell:

```
perl -de 0
```

Whereupon you must enter all the lines of a script including `use Tk;`. (Fancier file reads & `evals` are possible − but if you are getting that sophisticated why not create your own custom PERL5DB file?) When using `perl -dwe 0` beware of the *emacs* like line editing under this debugger, and be forewarned that as soon as you type in the `MainLoop;` statement perl will no longer read from <STDIN>.

Ilya Zakharevich <ilya@math.ohio−state.edu> points out that very recent perldb versions will allow for simultaneous X and STDIN reads. He also points out:

```
Note that you may use

    sub myLoop {
      if (defined &DB::DB) {
        while (1) {             # MainWindow->Count
          Tk::DoOneEvent(0);
        }
      } else {
        MainLoop;
      }
    }
```

751 mailto:gsar@engin.umich.edu
752 ftp://ftp.perl.com/pub/perl/ext/TK/
753 http://www−personal.umich.edu/~gsar/
754 http://www.metronet.com/0/perlinfo/perl5/manual/perldebug.html

```
(and I hope the analogous provision will be in MainLoop in
 tk-b9 - hi, Nick ;-)
```

Return to table of contents[755]

17.4. Is there a GUI builder in perl/Tk?

Work has reputedly (January 1996) started on porting a Tcl/Tk GUI builder known as SpecTcl[756] for use with perl/Tk. For the Tcl/Tk SpecTcl kit see:

`ftp://ftp.sunlabs.com/pub/tcl/SpecTcl-*.tar.[gz|Z]`[757]

and address questions about SpecTcl to <spectcl@tcl.eng.sun.com>.[758]

In <news:ANDREAS.96Mar24234521@marvin.berlin.de> Andreas Koschinsky <marvin@logware.de>[759] announced a perl script for use with SpecTcl that has some interesting capabilies:

> 24 Mar 1996 22:45:21 GMT
> ... So i wrote a perl−script that can convert project−file (.ui−files) which spectcl writes. The script reads the .ui−file and generates an equivalent perl−module.

The URL for **ui2perl** should be something like:

`ftp://susan.logware.de/pub/incoming/ui2perl*.tar.gz`[760]

Somewhat more removed from SpecTcl there is also SWIG.[761]

Return to table of contents[762]

---

18. Processes & Inter−Process Communication under Perl/Tk.

Inter−Process Communication (IPC) is the subject of spawning and controlling other programs or "processes" from within perl (sometimes using sockets to do so). The subject is briefly discussed in the `perlipc(1)` man page,[763] and was addressed towards the end of Chapter 6 of The Camel. The subject is also discussed in the perl FAQ[764] and at Tom Christiansen's ftp site (in the various `perlipc*` files) at:

`ftp://ftp.perl.com/perl/info/everything_to_know/`[765] `199.45.129.30`

as well as the web site at:

755 #TOC17.3
756 http://www.sco.com/Technology/tcl/README/UsenixTclBOF96.html
757 ftp://ftp.sunlabs.com/pub/tcl
758 mailto:spectcl@tcl.eng.sun.com
759 mailto:marvin@logware.de
760 ftp://susan.logware.de/pub/incoming/
761 http://www.cs.utah.edu/~beazley/SWIG/
762 #TOC17.4
763 http://www.metronet.com/0/perlinfo/perl5/manual/perlipc.html
764 http://www.perl.com/perl/faq/
765 ftp://ftp.perl.com/perl/info/everything_to_know/

`http://www.perl.com/perl/everything_to_know/ipc/index.html`[766]

In addition to the usual perl IPC routines Tk allows (at least) three more special functions: `fileevent` (for handling I/O events), `send` (for inter−widget communication), and `after` (for time control like a `sleep` expressly for widgets).

Remember:

> If a packet hits a pocket on a socket on a port,
> And the bus is interrupted as a very last resort,
> And the address of the memory makes your floppy disk abort,
> Then the socket packet pocket has an error to report!
> *−Ken Burchill(?)*[767]

Return to table of contents[768]

18.1. How does one get Perl/Tk to act on events that are not coming from X?

On 22 Nov 1995 (Yaniv Bargury) bargury@milcse.cig.mot.com wrote:

> *I need to write a GUI monitor, that displays the status and controls a set of processes running in the background. The idea is to have the GUI application start a few child processes, command the children through pipes from the GUI to the children, and display the children status coming on pipes from the children to the GUI in real time.*
>
> *The GUI must not be busy waiting, because the CPU resources are limited. This excludes using the* `Tk_DoWhenIdle` *as explained in the manual.*
>
> *The usual way to do this is to for the GUI process to have one* `select()` *in its main loop. That* `select()` *should wait for X events or input from the pipes leading from the children.*
>
> *How do you do this?*

To which Nick Ing−Simmons <nik@tiuk.ti.com> replied:

> `fileevent` − it is the hook into the `select()` in the mainloop.

In addition Avi Deitcher <avi@morgan.com> replied with:

> I wrote something similar to effectively do a `tail -f` on multiple hosts, displaying the result on separate text widgets. Do the following:

```
parent
 child
 child
 child
 ..
```

766 http://www.perl.com/perl/everything_to_know/ipc/index.html
767 http://www.infi.net/~cashman/humor/misc/drsuess_tech.html
768 #TOC18

with a one–way pipe from each child to the parent. Set up the following:

```
$main->fileevent(FILEHANDLE,status,subroutine);
```

for each pipe that you have. This will cause pTk to monitor the `FILEHANDLE` and call `'subroutine'` when an event happens on that handle. In this case: `FILEHANDLE = pipename` `status = 'readable'` or `'writable'` or `'exception'` and `subroutine` = any subroutine that you want.

To provide a concrete example of `fileevent` usage Stephen O. Lidie wrote a wonderful little GUI tail monitor he calls `tktail`:

```
#!/usr/local/bin/perl -w
#
# tktail pathname

use English;
use Tk;

open(H, "tail -f -n 25 $ARGV[0]|") or die "Nope: $OS_ERROR";

$mw = MainWindow->new;
$t = $mw->Text(-width => 80, -height => 25, -wrap => 'none');
$t->pack(-expand => 1);
$mw->fileevent(H, 'readable', [\&fill_text_widget, $t]);
MainLoop;

sub fill_text_widget {

    my($widget) = @ARG;

    $ARG = <H>;
    $widget->insert('end', $ARG);
    $widget->yview('end');

} # end fill_text_widget
```

An example of how one might use such a script would be to create and monitor a file `foo` like so:

```
echo Hello from foo! > foo
tktail foo &
echo \"A ship then new they built for him/of mithril and of elven glass\" --Bilbo \
 >> foo
```

Return to table of contents[769]

18.2. Is there a `send` and do I need `xauth`?

There is a `Tk::send`, but to use it own must write one's own version of `Tk::receive`. An example of this may be found in the **rmt** program distributed with perl/Tk. Note that as of Tk–b12 (including the released version Tk400.200) the script that receives from a Tk::send must run with taint chcecking turned on (*i.e.* with the `-T` switch thrown) and it must untaint all commands received from the other process.

[769] #TOC18.1

The `Tk::send <-> Tk::receive` process will work under `xhost +` authority. The security this affords comes from the fact that anyone who would want to exploit it would have to know how to write a `Tk::receive` custom tailored to your application (in addition to all the other protocol hacking).

Please note that while you may not *need* `xauth` authorization it is nevertheless *always* a good idea.

Return to table of contents[770]

18.3. How can I do animations using `after`?

There is a "toggling button" demo script supplied with Tk called `after_demo` that makes effective use of `after()`.

Terry Greenlaw <terry@encompass.is.net>[771] of Encompass Technologies[772] posted a character cell animator for the really bored. Here it is in a slightly modified form that allows string input from the command line (note too the recursive call that doesn't sop up system memory):

```
#!/usr/bin/perl

=head1 NAME

From: z50816@mip.lasc.lockheed.com "Terry Greenlaw"  Thu Feb 1 12:02:24 EST 1996
To: ptk@guest.WPI.EDU
Subj: A code sample for the REALLY bored

For everyone with a case of Browser envy after using Microsoft's Internet
Explorer, here's a perl/tk script only slightly more useful than a script
to do <BLINK>. Don't know why I wrote it. Don't know why you'd run it.
Maybe if you were writing a ticker tape application. Or had a weird thing
for Times Square. Anyway....

tog
Terry Greenlaw (on-site @ Lockheed Martin)      Encompass Technologies
z50816@mip.lasc.lockheed.com                    terry@encompass.is.net

###################################################################

=cut

    #!/usr/bin/perl

    #use strict;
    use Tk;

    $message=join(' ',@ARGV,'');
    if (!$message) {
        $message="THIS IS A VERY LONG SCROLLING MESSAGE...       ";
        $topmssg="This is the top of the screen";
        $botmssg="This is the bottom of the screen";
    }
    else {
        $topmssg=$message;
        $botmssg=$message;
```

770 #TOC18.2
771 mailto:terry@encompass.is.net
772 http://www.encompass.is.net

```
}
$top = MainWindow->new;
$l1 = $top->Label(-fg => 'White', -text => $topmssg);
$l1->pack(-fill => 'both', -expand => 1 );
$m1 = $top->Label(-fg=>'Red', -bg=>'black',
                  -textvariable => \$message,
                  -width => 15
                  );
$m1->pack();
$m2 = $top->Label(-wrap=>1,
                  -fg=>'Green', -bg=>'black',
                  -textvariable => \$message2,
                  -width=>1, -height=>8
                  );
$m2->pack(-anchor=>'w');
$l2 = $top->Label(-fg => 'White', -text => $botmssg);
$l2->pack(-fill => 'both', -expand => 1 );

after(100, \&scroll_it);

$top->MainLoop;

sub scroll_it {
    $message =~ /(.)(.*)/;
    $message="$2$1";
    ($message2 = $message) =~ s/ /  /g;
    after(100, \&scroll_it);
}
__END__
```

(Please note that a script like this is now distributed as `"TickerTape"` in your `Tk*/Contrib/` directory.)

Return to table of contents[773]

18.4. How do I `update` widgets while waiting for other processes to complete?

The short answer is either

```
    $widget -> update;
or
    $widget -> DoOneEvent;
```

Here is a script that makes use of the first of these methods. Note that instead of actually doing something useful the "long running process" is simply a call to the perl `sleep()` function for illustrative purposes:

```
#!/usr/bin/perl -w

use Tk;

my $m = MainWindow->new();
my $l = $m -> Listbox();
$l -> bind('<Double-1>' => sub{sleepy($l)} );
my @nuts   = qw(Almond Brazil Chestnut Doughnut Elmnut Filbert);
for (@nuts) { $l -> insert('end',$_); }
$l -> pack;
```

773 #TOC18.3

```
    MainLoop;

    sub sleepy {
        my $widget = shift;
        print "before 1st sleep \n";
        sleep(10);
        print "after 1st sleep before delete \n";
        $widget -> delete('active');
        $widget -> update;             # try [un]*commenting this
        print "after delete before 2nd sleep \n";
        sleep(10);
        print "after 2nd sleep \n";
    }
    __END__
```

Return to table of contents[774]

18.5. How do you `fork` on System V (HP)?

Kraegeloh Martin <mkr@dm−server.cv.com> originally asked[775]:

```
! Subj: signal handling difference on HP vs. SUN
!
! the following code will fork an xterm with vi in it, and it
! will refuse to do so while the first xterm is still running.
! works fine on my sun.
! On HP however, the second time an xterm is started, NO handler
! is called when the child dies.
!
! the code:
! ===================== 8< ===============================
! $SIG{CHLD}=\&w;
!
! sub w{
!    $pid=wait;
!    print STDERR "died: $pid\n";
!    if ( $have == $pid ) { $have = 0; }
! }
```

To which a part of Nick Ing−Simmons' response[776] was:

> I suspect HPUX is SysV−ish not BSD or POSIX. So every time a signal fires, it removes the handler − you need to reset it in the handler:
>
> ```
>  sub w{
>      $SIG{CHLD}=\&w;
>      $pid=wait;
>      print STDERR "died: $pid\n";
>      if ( $have == $pid ) { $have = 0; }
>   }
> ```
>
> Whether you reset it before/after the wait may be very important ...

774 #TOC18.4
775 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0046.html
776 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0048.html

Then Bjarne Steinsbo <bjarne@hsr.no> followed up with[777]:

> That's not the whole story... Another problem is that `SIGCLD` interrupts the read system call on SysV−ish (I like that word! :−) systems. This means that you have to test why `""` fails, and act accodingly. A program that works on both Sun and HP is:

```
$SIG{CHLD}=\&w;
while(1){
   $_ = ;
   $! = 0, next if $! =~ /Interrupted/;
   last if $! or !defined $_;
   if($have){
        print STDERR "child still alive\n";
   }
   else{
        if(($pid=fork()) != 0){
           $have=$pid;
           print STDERR "forked $pid\n";
        }
        else {
           exec("xterm −e vi")
        }
   }
}

sub w{
   $pid=wait;
   print STDERR "died: $pid\n";
   if ( $have == $pid ) { $have = 0; }
   $SIG{CHLD}=\&w;
}
```

Return to table of contents[778]

---

19. How do I "clear the screen"?

What screen are you trying to clear?

If you are trying to clear a `tty` (*e.g.* `xterm`) screen then try either of the following within your script:

```
    system "clear";
or
    print `clear`;
```

(where the choice between these two depends on the rest of the script: the first is fast − but proceeds via `fork` and may not occur at exactly the time that you want it to in the script).

David and Rachel Weintraub <davidw@cnj.digex.net> recommend using the old `termcap.pl` p4 library. You might also consider the perl 5 equivalents: `Term.pm` (especially the `Term::Control` module), `Curses.pm`, `Perlmenu.pm`, `PV`.

Returning to X−windows and perl/Tk: if you are trying to eliminate a `TopLevel` or a `MainWindow` then try:

[777] http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0052.html
[778] #TOC18.5

```
$main -> destroy;
```

If you would rather not destroy then try:

```
$main->withdraw;    # remove

$main->deiconify;   # put back
```

If `$w` is a sub−window (sub−widget) then

```
$w->pack('forget'); # remove if packed (newer Tk-b9.01++)
$w->packForget;     # remove if packed (older versions)
$w->pack(...);      # put back
```

There are also ways to call low level C−ish versions:

```
$w->UnmapWindow;
```

but that is for special purposes only....

If you are trying to erase an `$item` on a `Canvas` then try:

```
delete($item);
```

(Thanks to the post by <a904209@pluto.tiuk.ti.com> which extended this answer considerably.)

Return to table of contents[779]

---

20. Is there a way to have an X application draw inside a perl/Tk window?

(*This question was posed by Andrew Allen March 19 1997*):

No not yet. But according to Nick Ing−Simmons:

```
If app. can take XID of window to draw on it should be doable now,
but if Tk has to pro-actively watch for creation of app's 'top level',
and "capture" it is more tricky.
```

Return to table of contents[780]

---

21. Is there a version for Microsoft Windows(tm)?

Yes. Tk402.000 is in alpha release at this time (Spring 1997) and builds with perl 5.004, nmake, and Visual C++ on at least Windows NT.

Return to table of contents[781]

---

[779] #TOC19
[780] #TOC20
[781] #TOC21

22. Are there any international font packages for perl/Tk?

In principle you may specify the `-font` configuration option on all your `Button`, `Entry`, `Label`, `Message`, *etc.* widgets. In addition to the Unix programs *xfontsel* and *xlsfonts* you may find *xset* useful for determining and/or specifying fonts – especially international ones.

KOBAYASI Hiroaki <kobayasi@sowa.is.uec.ac.jp> has converted the Tcl/Tk "japanization" by <nisinaka@sra.co.jp> so that it may be used with perl/Tk. It is presently available (the current README file is in Japanese) from the following ftp site:

> ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/[782]

From the author's own description:

```
Currently, the "japanization patch for perl/Tk" enables:

  [1] To show kanji & ASCII (by choosen kanji-font) in every widget.
  [2] To edit kanji (double width of ASCII) correctly in Text & Entry.
  [3] To support of Kanji Input method. (tkKinput.c)
  [4] Automatic kanji-code-detection & conversion with 'insert/get'.
      Supports: "JIS(Japanese Industrial Standard)", "MS-KANJI", "EUC".

& the patch lacks:

  [5] by manual Kanji-code conversion. (JIS <=> MS-KANJI <=> EUC)
  [6] 'Good' interface to specify kanji-code used in internal. (tkWStr.c)
  [7] Documentation in English about [1-6].
      # but, since interface-change is suspected in near future,
      # documenting them is ...

I thought that[5-7] was not enough for world-people, but already worth
for natives. So I announced it on "fj.lang.perl"[783].
```

Return to table of contents[784]

---

23. Are there any other ways to create event based interfaces from perl?

Yes. A short list would have to mention these for X:

For perl 4:
: WAFE, STDWIN

For perl 5:
: Sx[785] (uses Athena & Xlib), Motif (uses Motif & Xt), Fresco (post X11R6.1). This perl extension was originally written by Dominic Giampaolo, but was re–written for Sx by Frederic Chauveau. The code is available from CPAN in the `authors/Frederic_Chauveau/`[786] directory.

: There is an `X11::*` hierarchy of perl–>X modules under development and available from CPAN.

[782] ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/
[783] news:fj.lang.perl
[784] #TOC22
[785] http://funnelweb.utcc.utk.edu/~harp/gnu/perl5/Sx.html
[786] http://www.perl.com/CPAN/authors/Frederic_Chauveau

For hooking into the C++ Qt library there is Ashley Winters' PerlQt.[787]

The Perl 5 Module List[788] has a listing of User Interfaces (Character and Graphical).[789]

There is also Malcolm Beattie's Tkperl (which is largely incompatible with perl/Tk).

Further information on X interfaces to Perl is provided in the perl FAQ.[790]

For primarily graphics (with some user interface programming) there is also an OpenGL Perl Module[791] of particular interest[792] to anyone with either OpenGL or the MesaGL library.[793] This package allows for compact perl VRML scripting. If you would like to see the OpenGL network X demonstration go to:

`http://www.arc.ab.ca/vr/opengl/examples/`[794]

(to run that demo one needs only a forms capable web browser and a local X−server, hence running Lynx on Windows 95 with eXodus is perfectly OK.) If you would like to see the OpenGL netscape plugin go to:

`http://www.arc.ab.ca/vr/opengl/plugin.html`[795]

For perl generation of images see the question in this FAQ on graphics modules.[796]

The X window system is not the only means of doing event based user interface programming in perl. There is, for example, William Setzer's Curses[797] modules for perl 5 too.

For Active X scripting on Windows try the PerlScript package from

`http://www.activeware.com/`[798]

which will work with Internet Explorer, Internet Information Server, as well as the `wscript` command line interface.

Return to table of contents[799]

---

24. Where can I get more information on graphics (modules|scripts)?

To generate server side push animation in perl please see the perl cgi FAQ.[800] To generate GIF89a loops from a collection of stills visit the Gifloop[801] page at

`http://www.homecom.com/gifanim.html`[802]

787 http://www.perl.com/CPAN/modules/by−authors/Ashley_Winters/
788 http://www.perl.com/CPAN/modules/00modlist.long.html
789 http://www.perl.org/CPAN/modules/by−category/08_User_Interfaces/
790 http://www.perl.com/perl/faq/
791 http://www.arc.ab.ca/vr/opengl/
792 http://www.arc.ab.ca/vr/opengl/about.html
793 http://www.ssec.wisc.edu/~brianp/Mesa.html
794 http://www.arc.ab.ca/vr/opengl/examples/
795 http://www.arc.ab.ca/vr/opengl/plugin.html
796 #A24
797 http://www.perl.com/CPAN/modules/by−authors/William_Setzer/
798 http://www.activeware.com/
799 #TOC23
800 http://www.perl.com/perl/
801 http://www.homecom.com/gifanim.html
802 http://www.homecom.com/gifanim.html

For other animation from perl see the OpenGL module listed below (it does 3D VRML animation).

There are several graphics modules in perl already available − with several more still under development. Among them:

GD.pm[803]

The **GD.pm** perl module is a perl interface to the C code of a similar name and was written by Lincoln Stein. It allows for the generation of GIF (Graphics Inline Format) images from within a perl script. The module itself is available from any CPAN ftp site,[804] and Lincoln maintains an informational web page at:

`http://www−genome.wi.mit.edu/ftp/pub/software/WWW/GD.html`[805]

gd/GIF.pm[806]

The **gd/GIF.pm** module is similar to GD.pm (generates GIFs using gd.c) and was written by Roberto Cecchini who maintains a web page for his module at:

`http://www.fi.infn.it/pub/perl/GIF/`[807]

JPEG.pm[808]

The **JPEG.pm** module was written by Nick Ing−Simmons expressly for use with the Tk family of modules. It is distributed on CPAN[809] in the `authors/id/NI−S/`[810] directories as a `Tk−JPEG−*.tar.gz` file.

Open/Mesa−GL Perl Module[811]

This module can use the OpenGL lib on computers that have it (available for SGI, AIX, Linux), or with the MesaGL package[812] (which allegedly runs a little slower but does not require the X accelerator hardware that OpenGL does). This package supports relatively easy graphics programming, 3D shading, *etc.* Very cool stuff! The URL for the OpenGL perl module is:

`http://www.arc.ab.ca/vr/opengl/`[813]

PGPERL[814]

If you will be interfacing to the **PGPLOT**[815] FORTRAN language routines you might consider Karl Glazebrook's **PGPERL** which has an interactive web demo[816] as well a a web home page at:

`http://www.aao.gov.au/local/www/kgb/pgperl.html`[817]

A pgperl/tk script might look like this:

```
#!/usr/bin/perl
use PGPLOT;
```

803 http://www−genome.wi.mit.edu/ftp/pub/software/WWW/GD.html
804 #CPAN
805 http://www−genome.wi.mit.edu/ftp/pub/software/WWW/GD.html
806 http://www.fi.infn.it/pub/perl/GIF/
807 http://www.fi.infn.it/pub/perl/GIF/
808 http://www.perl.com/CPAN/authors/id/NI−S/
809 #CPAN
810 http://www.perl.com/CPAN/authors/id/NI−S/
811 http://www.arc.ab.ca/vr/opengl/
812 http://www.ssec.wisc.edu/~brianp/Mesa.html
813 http://www.arc.ab.ca/vr/opengl/
814 http://www.aao.gov.au/local/www/kgb/pgperl.html
815 http://astro.caltech.edu/~tjp/pgplot/
816 http://www.ast.cam.ac.uk/cgi−bin/pgperl_formdemo.pg
817 http://www.aao.gov.au/local/www/kgb/pgperl.html

```
use Tk;
open(IMG,"test.img");       # Read 128x128 image from binary file
read(IMG, $img, 4*128*128); # data - list of 4 byte float [C type]
close(IMG);                 # stored as perl string.
pgbegin(0,'test.gif/gif',1,1);
pggray($img,128,128,1,128,1,128,1,0,*tr); # pg-Plot it
my $main = new MainWindow;
$main ->Label(-text => 'Main')->pack;
$main -> Photo('IMG', -file => "test.gif");
my $l = $main->Label('-image' => 'IMG')->pack;
$main->Button(-text => 'close',
              -command => sub{destroy $main}
              )->pack(-side => 'left');
$main->Button(-text => 'exit',
              -command => [sub{exit}]
              )->pack(-side => 'right');
MainLoop;                    # pg-tk plot it
```

perlDL[818]

perlDL or **PDL** is the **P**erl **D**ata **L**anguage for fast manipulation of large arrays – such as those that often occur in image processing/analysis/creation (or scientific data sets). **PDL** is also by Karl Glazebrook[819]. This is a very hot module full of programmer convenience features such as overloaded operators *etc.* There is a mailing list and a home page for this module:

http://www.aao.gov.au/local/www/kgb/perldl/[820]
perldl-request@jach.hawaii.edu[821]
http://www.rosat.mpe-garching.mpg.de/mailing-lists/perldl/[822]

Another tip: it is quite useful to have the PBMPlus/NetPBM[823] set of graphics file interconversion programs if you will be dealing with many graphics file formats[824]. There is a Sunsite web server with many graphics utilities including netpbm at:

http://sunsite.unc.edu/pub/Linux/apps/graphics/convert/INDEX.short.html[825]

(Note there are linux binaries as well as source code kits there – also at sunsite mirrors.) You might also be interested in some of the graphics file format specifications[826].

Other programming tools/utilities are of help with graphics. The X11 release contains many such tools. There is also The GIMP[827]. Consider also the ImageMagick[828] program.

Return to table of contents[829]

---

25. Are there any major applications written in perl/Tk?

Yes. In fact, there are some interesting perl/Tk applications already available from:

818 http://www.aao.gov.au/local/www/kgb/perldl/
819 http://www.aao.gov.au/local/www/kgb/thisiskgb.html
820 http://www.aao.gov.au/local/www/kgb/perldl/
821 mailto:perldl−request@jach.hawaii.edu
822 http://www.rosat.mpe−garching.mpg.de/mailing−lists/perldl/
823 ftp://ftp.cs.columbia.edu/archives/X11R6/contrib/utilities/
824 http://www.dcs.ed.ac.uk/~mxr/gfx/faqs/FileFormats.faq
825 http://sunsite.unc.edu/pub/Linux/apps/graphics/convert/INDEX.short.html
826 http://www.dcs.ed.ac.uk/~mxr/gfx/2d−hi.html
827 http://www.XCF.Berkeley.edu/~gimp
828 http://www.wizards.dupont.com/cristy/www/formats.html
829 #TOC24

The following programs may be found in your own `Tk#/` directory (you already have these if you have the Tk extension to perl)*:

```
program        description
pfm            perl file manager - cute iconic front to emacs
ptknews        a GUI perl newsreader - a work in progress.
tkpsh          perl/Tk equivalent of the Tcl/Tk wish shell.
toyedit        a Text widget editor.
```

The following programs may be found either in your `demos` directory (consult the `README` file there for fuller descriptions) or in your perl/bin install directory:

```
program        description
browse         Simple file browser front end for emacs.
color_editor   Front end to Tk::ColorEditor[830]
               allows RGB, CMY, and HSV color cube manipulation
               (based on tcolor.tcl).
ixset          GUI front end to xset - for terminal settings.
pgs            Perl front end to Ghostscript (viewing PostScript(c) files).
rmt            perl/Tk
               "development shell/remote control application"
               You can launch or talk to other perl/Tk apps with rmt.
rolodex        Like the Tcl/Tk app of the same name.
               Requires editing for personal use.
timer          Stopwatch-like seconds timer.
tkpod          The perl gui pod viewer (like xman).
tkweb          The perl "Rough and Ready" web browser.
```

*Peter Prymmer recently posted a means by which one can integrate any or all of these GUI applications into one's own X−window environment. Here for terse illustration is the basic idea behind using an X11R6.1 `.mwmrc.m4` resource file for making a Menu (make sure the applications are in your PATH or specify a full path to each one as needed):

```
Menu Perl
{
    "Perl"          f.title
    "editor"        f.exec "toy_edit &"
    "tkemacs"       f.exec "browse &"
    "manual"        f.exec "tkpod perl.pod &"
    "the web"       f.exec "tkweb http://www.perl.com/perl/ &"
    "news"          f.exec "ptknews comp.lang.perl.tk &"
    "pgs"           f.exec "pgs &"
    "stop watch"    f.exec "timer &"
}

# We bind it so that left mouse button when pressed while
# over the root or background pops up with this Perl menu:
Buttons DefaultButtonBindings
{
    <Btn1Down>      root       f.menu  Commands
    # etc.
}
Buttons ExplicitButtonBindings
{
    <Btn1Down>      root       f.menu  Commands
```

830 etc/ColorEditor.html

```
    # etc.
}
```

Other perl/Tk application distributors:

ptkb.pl
: an *xbiff* like mailbox watcher. Available from ftp://ftp.wpi.edu/perl5/pTk-Modules/ptkb.pl[831]

bioTkperl
: Was announced[832] by Gregg Helt <gregg@fruitfly.berkeley.edu>[833] recently. See the home page at: http://www.cbil.upenn.edu/~dsearls/bioTk.html.[834] Source at: ftp://fruitfly.berkeley.edu/pub/bioTk/bioTkperl0.8.tar.gz[835]

EVA
: KOBAYASI Hiroaki's EVA is a sophisticated `wish` like perl/Tk "shell". It is available from: ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/eva-*.tar.gz[836]

plop[837]
: Stephen Lidie's very useful arbitrary function **P**lot **P**rogram, was featured in the premier issue of **The Perl Journal**[838] and is available on the web at: http://tpj.com/tpj/programs/Vol_1_Issue_1_Tk/plop[839]

modo[840]
: Yet another Lidie creation. This one hails from Issue 2[841] of **The Perl Journal**[842]

neko (*et al.*)[843]
: In the third issue of **The Perl Journal**[844] Stephen Lidie makes creative use of approaches to event timing from perl/Tk. `neko` is one axample.

Workspace[845]
: Is an integrated front end to the fvwm window manager that makes use of Perl/Tk. See the web page at http://www.mirai.com/wks/[846] for more information.

www
: The original 8 line wonder by Jon Orwant. Pick it up (and modify it) from: http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.08/0411.html.[847] (Please note: **www** is for amusement, the more serious perl/Tk browser – **tkweb** – is distributed with perl/Tk and it is "only" 60 lines long!.)

ptkclock & ptkmines
: From the generous software archive[848] of Jason Smith at RPI.

XCricket[849]
: This application was written by Nem Schlecht.[850]

Be sure to also check the newsgroups comp.lang.perl.tk,[851] comp.lang.perl.announce,[852]

---

831 ftp://ftp.wpi.edu/perl5/pTk-Modules/ptkb.pl
832 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.11/0034.html
833 mailto:gregg@fruitfly.berkeley.edu
834 http://www.cbil.upenn.edu/~dsearls/bioTk.html
835 ftp://fruitfly.berkeley.edu/pub/bioTk/bioTkperl0.8.tar.gz
836 ftp://ftp.sowa.is.uec.ac.jp/pub/Lang/perl5/Tk/
837 http://tpj.com/tpj/programs/Vol_1_Issue_1_Tk/plop
838 http://tpj.com/
839 http://tpj.com/tpj/programs/Vol_1_Issue_1_Tk/plop
840 http://tpj.com/tpj/programs/Vol_1_Issue_2_Tk/
841 http://tpj.com/tpj/programs/
842 http://tpj.com/tpj/
843 http://tpj.com/tpj/programs/Vol_1_Issue_3_Tk/
844 http://tpj.com/
845 http://www.mirai.com/wks/
846 http://www.mirai.com/wks/
847 http://sun20.ccd.bnl.gov/~ptk/archive/ptk.1995.08/0411.html
848 http://hadron4-en.phys.rpi.edu/~smithj4/software/
849 http://abattoir.cc.ndsu.nodak.edu/~nem/perl/cricket/
850 http://abattoir.cc.ndsu.nodak.edu/~nem/
851 news:comp.lang.perl.tk
852 news:comp.lang.perl.announce

comp.lang.perl.misc[853], and comp.lang.perl.modules[854], as well as the mailing list archive[855] for more Perl/Tk program and package announcements.

Return to table of contents[856]

---

26. What is the history of pTk and perl/Tk?

This list is only *slowly* coming together. Please forgive any absences.

- tkperl5a5 is announced[857] Thu, 20 Oct 1994 14:44:23 +0000 (BST)

```
NOTE
This project is unrelated to the one which is adding usersubs to
perl4 to get access to Tk via Tcl. Often, postings on comp.lang.perl
just say "tkperl" without saying which one is meant. They are two
totally different kettles of fish.

--Malcolm (Beattie)
```

- Fri, 25 Nov 94 14:29:53 GMT Nick Ing−Simmons is working on[858] what will be known as "nTk" eventually.
- Mon, 12 Dec 94 08:56:36 GMT, Nick Ing−Simmons reports[859]:

```
I have a re-port of ext/Tk nearly ready for alpha.
It builds its own "pTk" library from sources semi-automatically derived
from Tk3.6.  There is no Tcl library at all.

Would anyone like to assist me in testing it?
```

- nTk−a2 announced[860] Fri, 16 Dec 1994 10:59:36 −0500
- nTk−a3 announced[861] Mon, 19 Dec 1994 18:03:27 −0500
- nTk−a5 announced[862] Fri, 23 Dec 1994 10:18:16 −0500 (last to use Tk 3.6 ?)
- nTk−a6 first to use Tk 4.0 (?)
- nTk−a7 announced[863] Fri, 13 Jan 1995 10:55:27 −0500
- nTk−a8 has appeared[864] *before* Tue, 17 Jan 95 09:04:33 GMT
- nTk−a9 has appeared[865] *before* Wed, 18 Jan 95 19:25:10 GMT
- nTk−a10 announced[866] Tue, 24 Jan 1995 14:32:02 −0500
- nTk−a11 announced[867] Tue, 31 Jan 95 19:05:32 GMT
- Malcolm Beattie suggests[868] the nTk −> Tk name change, Larry Wall concurs[869]
- nTk−a12 announced[870] Thu, 16 Feb 1995 09:12:26 −0500

---

853 news:comp.lang.perl.misc
854 news:comp.lang.perl.modules
855 http://pubweb.bnl.gov/~ptk/
856 #TOC25
857 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9410.gz/date/article−844.html
858 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9411.gz/date/article−255.html
859 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9412.gz/date/article−137.html
860 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9412.gz/Date/article−240.html
861 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9412.gz/date/article−280.html
862 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9412.gz/date/article−378.html
863 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9501.gz/date/article−232.html
864 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9501.gz/date/article−295.html
865 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9501.gz/date/article−360.html
866 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9501.gz/date/article−496.html
867 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9501.gz/date/article−750.html
868 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9502.gz/date/article−109.html
869 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9502.gz/date/article−246.html
870 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9502.gz/date/article−447.html

- Nick Ing−Simmons calls[871] for a new mail list Thu, 16 Feb 95 14:13:55 GMT
- Tk−a13 announced[872] Wed, 1 Mar 1995 11:38:15 −0500 (Name has changed from "nTk")
- Tk−b1 announced[873] Tue, 14 Mar 95 16:58:40 GMT
- Tk−b2 announced[874] Wed, 29 Mar 95 15:52:44 BST
- Tk−b3 announced[875] Fri, 31 Mar 95 16:54:54 BST
- Tk−b4 announced[876] Fri, 12 May 1995 11:45:32 −0400 EST
- Tk−b5 announced[877] Mon, 26 Jun 95 17:14:06 BST
- Tk−b6 announced[878] Fri, 21 Jul 95 15:42:35 BST
- Tk−b7 announced[879] Fri, 28 Jul 95 15:16:02 BST
- Tk−b8 announced[880] Wed, 16 Aug 95 12:34:05 BST
- an RFD[881] (Request For Discussion) for a new usenet group comp.lang.perl.tk[882] is circulated by Jon Orwant Fri, 4 Aug 1995 08:29:46 −0400
- unmoderated newsgroup comp.lang.perl.tk[883] passes by a vote of 352 to 18 with 1 abstention in an announcement[884] made Mon, 9 Oct 1995 10:13:17 −0400 (EDT). The new group makes its appearance at news−servers roughly 18 October 1995.[885]
- Tk−b9.01 announced[886] Wed, 20 Dec 95 10:06:47 GMT.
- Tk−b10 announced[887] Sat, 23 Mar 96 17:16:27 GMT.
- Tk−b11 announced[888] Mon, 1 Apr 96 16:44:48 GMT.
- Tk−b11.01 announced[889] Wed, 3 Apr 96 17:48:09 GMT.
- Tk−b11.02 announced[890] 10 Apr 96 12:52:28 GMT.
- Tk−b12 announced 28 August 1996.
- Tk400.200 announced 6 September 1996.
- Tk400.201 announced Fall 1996.
- **Tk400.202** announced Late Fall 1996.
- Tk402.000 (alpha release for Windows) announced May 1997.

Return to table of contents[891]

---

27. What can we expect the future to hold?

With the production release of Tk400.202 and the alpha release of Tk402.000 the future of this code looks quite bright. (Hopefully the FAQ maintainer will manage to keep up :−).

Return to table of contents[892]

871 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9502.gz/date/article−448.html
872 http://www.nicoh.com/cgi−bin/lwgate/PERL5−PORTERS/archives/perl5−porters.archive.9503.gz/date/article−27.html
873 http://sun20.ccd.bnl.gov/%7Eptk/archive/ntk/0117.html
874 http://sun20.ccd.bnl.gov/%7Eptk/archive/ntk/0198.html
875 http://sun20.ccd.bnl.gov/%7Eptk/archive/ntk/0228.html
876 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.05/0055.html
877 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.06/0348.html
878 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.07/0211.html
879 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.07/0389.html
880 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.08/0303.html
881 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.08/0112.html
882 news:comp.lang.perl.tk
883 news:comp.lang.perl.tk
884 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.10/0031.html
885 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.10/0144.html
886 http://sun20.ccd.bnl.gov/%7Eptk/archive/ptk.1995.12/0086.html
887 history/Tk−b10.announced.html
888 history/Tk−b11.announced.html
889 history/Tk−b11.01.announced.html
890 history/Tk−b11.02.announced.html
891 #TOC26
892 #TOC27

28. How do I obtain the latest version of this FAQ?

On the world wide web

Hypertext (split by question):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html[893]
Hypertext (whole thing − may be too large for some browsers, but is amenable to searching):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html[894]
As far as I know any other source may be slightly out of date with respect to those two web URLs.
Nevertheless, there are many other places to retrieve this FAQ from, in a variety of formats, such as:

Plaintext (whole):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt[895]
Plaintext (multi−part):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ0.txt[896]
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ1.txt[897]
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ2.txt[898]
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ3.txt[899]
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ4.txt[900]
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ5.txt[901]
or gzipped PostScript(c) (about 60 US 8.5"x11" pages):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.US.ps.gz[902]
or gzipped PostScript(c) (about 60 A4 pages):
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.A4.ps.gz[903]

Return to table of contents[904]

---

29. Acknowledgements & maintainer.

The Perl/Tk extension to the Perl programming language is copyrighted by its author Nick Ing−Simmons <nik@tiuk.ti.com>[905] whose `Tk400.202/COPYING` file reads as follows:

```
Copyright (c) 1995-1996 Nick Ing-Simmons. All rights reserved.
This package is free software; you can redistribute it and/or
modify it under the same terms as Perl itself, with the exception
of the files in the pTk sub-directory which have separate terms
derived from those of the orignal Tk4.0 sources and/or Tix.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY
FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY
```

893 http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html
894 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html
895 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt
896 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ0.txt
897 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ1.txt
898 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ2.txt
899 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ3.txt
900 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ4.txt
901 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ5.txt
902 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.US.ps.gz
903 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.A4.ps.gz
904 #TOC28
905 mailto:nik@tiuk.ti.com

```
DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.  THIS SOFTWARE
IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE
NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS.

See pTk/license.terms[906] for details of this Tk license,
and pTk/Tix.license[907] for the Tix license.
```

Especial thanks to:
Nick Ing−Simmons for writing perl/Tk.
Dan Riley <dsr@mail.lns.cornell.edu>[908] for outstanding networking and countless other tough jobs.
Malcolm Beattie for tkperl.
An anonymous comp.lang.perl.tk poster for writing the initial "pseudo−FAQ" that got this started.
Larry Wall for writing extensible Perl 5 & John Ousterhout for writing Tk 4.
Tom Christiansen and Stephen P. Potter for writing and maintaining excellent perl documentation, and general doc help.
Jon Orwant <orwant@media.mit.edu>[909] for organizing the comp.lang.perl.tk[910] Usenet newsgroup.
Alan Stange & Tom Schlagel for the hypermail archive, the ftp & e−mail distribution of the FAQ, *etc*.
Achim Bohnet for an excellent searchable hypermail archive.
Ilya Zakharevich <ilya@math.ohio−state.edu> for great perl/Tk pod docs.
KOBAYASI Hiroaki <kobayasi@sowa.is.uec.ac.jp> for great perl/Tk scripts.
William J. Middleton <wjm@best.com> for archive help.
Ioi Kim Lam for Tix.
Larry Virden for cross−posting the Tcl FAQ, as well as editorial comments on this one.
Don Libes <libes@cme.nist.gov>[911] for lucid informative conversations on Tcl & Expect.
Terry Carroll <carroll@tjc.com> for valuable usage advice.
Nancy Walsh for great bibliographic help.

In addition, this FAQ has benefitted from the contributions of many people all over the net to whom I am quite grateful.

I am:
Peter Prymmer[912]
Wilson Synchrotron Laboratory[913]
Cornell University[914]
Ithaca,[915] *NY*[916] *14853*[917]

pvhp@lns62.lns.cornell.edu[918]

Return to table of contents[919]

906 misc/license.terms
907 misc/Tix.license
908 mailto:dsr@mail.lns.cornell.edu
909 mailto:orwant@media.mit.edu
910 news:comp.lang.perl.tk
911 mailto:libes@cme.nist.gov
912 http://w4.lns.cornell.edu/~pvhp/
913 http://w4.lns.cornell.edu/
914 http://www.cornell.edu/
915 http://www.ithaca.ny.us/
916 http://wings.buffalo.edu/world/nywww.html
917 http://www.mit.edu:8001/geo?Ithaca
918 mailto:pvhp@lns62.lns.cornell.edu
919 #TOC29

Hypertext (split by question) FAQ:
http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html[920]
Hypertext whole (very big) FAQ:
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html[921]
Plaintext FAQ:
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt[922]
Plaintext multi−part FAQ:
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ[0..5].txt[923]
Image−supplement:
http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html[924]
Reference manual:
http://w4.lns.cornell.edu/~pvhp/ptk/doc/[925]

920 http://w4.lns.cornell.edu/~pvhp/ptk/ptkTOC.html
921 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html
922 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.txt
923 http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ0.txt
924 http://w4.lns.cornell.edu/~pvhp/ptk/ptkIMG.html
925 http://w4.lns.cornell.edu/~pvhp/ptk/doc/