Perl/Tk Tutorial - Create GUI with Perl's Tk Module

Introduction

Perl/Tk (also known as pTk) is a collection of modules and code that attempts to wed the easily configured Tk 8 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)**

Perl or *Practical Extraction and Report Language* is described by Larry Wall, Perl's author, as follows: "Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information It's also a good language for any system management tasks The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal)" The perlintro man page has this to say

> Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more

Tk, the extension(or module) that makes GUI programming in perl possible, is taken from **Tcl/Tk** Tcl(Tool Command Language) and Tk(ToolKit) was created by Professor John Ousterhout of the University of California, Berkeley Tcl is a scripting language that runs on Windows, UNIX and Macintosh platforms Tk is a standard add-on to Tcl that provides commands to quickly and easily create user interfaces Later on Tk was used by a lot of other scripting languages like Perl, Python, Ruby etc

Applications

Perl has been used since the early days of the web to write CGI scripts, and is now a component of the popular LAMP (Linux/Apache/MySQL/Perl) platform for web development Perl has been called "the glue that holds the web together" Large systems written in Perl include Slashdot, and early implementations of Wikipedia and PHP

Perl finds many applications as a glue language, tying together systems and interfaces that were not specifically designed to interoperate Systems administrators use Perl as an all-purpose tool; short Perl programs can be entered and run on a single command line

Philosophy

Perl has several mottos that convey aspects of its design and use One is There's more than one way to do it (TMTOWTDI - usually pronounced 'Tim Toady') Another is Perl: the Swiss Army Chainsaw of Programming Languages A stated design goal of Perl is to "make easy tasks easy and difficult tasks possible"

Perl is free software, and may be distributed under either the Artistic or the GPL License It is available for most operating systems but is particularly prevalent on Unix and Unix-like systems (such as Linux, FreeBSD, and Mac OS X), and is growing in popularity on Microsoft Windows systems

Perl/Tk Requirements

Before starting with the tutorial, make sure you have the following things If some are missing you still can learn *perl* - but you will not be able to use it to its full power

1. ActivePerl from http://wwwactivestatecom/ActivePerl/ for windows - for programming in Windows Linux don't need any special outside interpreter because it already has it in most of the distributions
2. A good text editor I would recommend Crimson Editor(http://wwwcrimsoneditorcom/) for Windows and XEmacs for Linux

Installing/Using Perl

In **Unix/Linux** you can execute your perl scripts by typing "*perl <filename>*" at command prompt But before you do that make sure you have **both** Perl and its Tk module Most linux distributions have perl - but quite a few don't have the Tk module Make sure that the system you are using have the Tk module If you don't have it, go to http://wwwcpanorg and download the perl module Or you can use the perl's CPAN module to install the Tk module To do this, open a terminal and enter the following command

```
perl -MCPAN -e shell
cpan> install Bundle::CPAN
cpan> reload cpan
cpan> install Tk
```

Another(and a much easier) way to do this is to get a rpm of Perl/Tk and installing it with the command

```
rpm -ivh FILENAME
```

If you are using Ubuntu, a easy way of installing Perl/Tk is using this command

```
sudo apt-get install perl-tk
```

If you are using Windows, download ActivePerl and install it Then you can execute any perl file by double clicking it

Two more things before we begin the tutorial I will be teaching perl/tk and I expect you to know how to program in perl I may ignore some of the perl coding conventions like including `use strict;`, `-w` or `use warnings;` in my examples The examples have only one purpose - to demonstrate the feature that will be taught in that part of the tutorial Sorry about that - but I have to keep my tutorial's example scripts short and to the point

Finally, this is a *tutorial for Perl/Tk only* - I will not be teaching perl here So if you know perl, continue But if you are a beginner to perl, I would recommend that you read my **perl tutorial**

Hello World

Let us begin, as all other tutorials begin, with the "Hello World" program Create a file called "Hellopl" and enter the following into it

```
#!/usr/local/bin/perl
use Tk;
# Main Window
my $mw = new MainWindow;
my $label = $mw -> Label(-text=>"Hello World") -> pack();
my $button = $mw -> Button(-text => "Quit",
                -command => sub { exit })
        -> pack();
MainLoop;
```

The first line - `#!/usr/local/bin/perl` is not needed in windows In Linux, it tells the name of the script language processor In our case it is perl Don't understand what that means? Don't worry your gray cells over it Just put it at the top of the file

The second line - `use Tk;` tells the interpreter that our program will use the Tk module This line is an absolute must

in all GUI programs you make using perl When the interpreter encounters this line, it will load the Tk components that we will be using to create our program

The third line - This is a comment Any line that starts with a '#' char is a comment Comments are not of any use in the program It is used by programmer to talk to themselves A programmer cannot be expected to remember every thing a script does So he uses a comment to write it down Next time he edits the script, he can read the comment and understand what the program is for It is good practice to make as much comments as possible

The fourth line, `my $mw = new MainWindow;`, will create a window into which the GUI elements will be placed The variable $mw is a object of type 'MainWindow' We will have to use this element when we want to place any widget inside it

The fifth line - `$mw -> Label(-text=>"Hello World") -> pack();` makes a label and writes "Hello world" in it You can change the text to any thing you like Note the structure of the command -

`$label` - This variable assigned to that particular widget Ever widget must have a UNIQUE variable This name will be used when ever that widget must be accessed

`$mw ->` - $mw is the MainWindow's object We will be placing our label widget inside this window

`Label(-text=>"Hello World")` - 'Label' is the name of the widget A widget is a user interface object in X graphical user interfaces Confused? Lets just say that it is the name of the object that appears on screen There are many other widgets too If you want to display a button, you use the button widget For text, you use the text widget For entry, you guessed it, the entry widget If you want, you can see more about widgets

-text=>"Hello World" - The option for this widget This option says that this widget must be given the text "Hello World" Options change according to the widgets - a button widget will not have all the options of the label widget and vise versa But there will be many common ones
Please note that operator used here is '=>' as opposed to the one used earlier '->' in `$mw ->` One uses the minus(-) sign while the other uses the equals(=) sign Do not confuse between these two
You can keep writing other options can also be written here For example, let us make a label for showing the text "Hello World" The other lines are same as the Hello World program

```
$mw -> Label(-text=>"Hello World",-font=>"courierfont",-relief=>"raised") ->
pack();
```

In this example, a lot more options are used The font option is used to tell which font must be used to make the text and the relief option tells whether the text should appear raised, sunken, flat etc To know all the options for a particular widget, read the manual that comes with Perl It lists every widget and every option they have If you are going to program in Perl, you will find your self peeking into the manual every few minutes The most important and most commonly used options are listed here

All options must separated by a comma But as you have noted, this line is a little difficult to read As the number of options increase, the more difficult to read it So a more readable version is

```
$mw -> Label(-text=>"Hello World",
                -font=>"courierfont",
                -relief=>"raised")
        -> pack();
```

Next comes the `-> pack();` This will pack the widget '$label' into the window '$mw' 'pack' is a geometry manager Another geometry manager is 'grid' Personally, I like grid better Once again, putting all this in one line is an eye sore - so you can put this part in the next line

```
my $label = $mw -> Label(-text=>"Hello World")
        -> pack();
```

In this case, pack has no options within it But that is not always the case

```
my $label = $mw -> Label(-text=>"Hello World")
        -> pack(-side=>"left",
                -anchor=>'w');
```

You don't have to pack the widget in the same line of creating it - but it is convenient in small programs You can pack the widget later using the widget's variable For example

```
my $label = $mw -> Label(-text=>"Hello World"); #We created the widget
$label -> pack(-side=>"left", -anchor=>'w'); #We pack it in another line
```

So we have the final syntax of how to create and display a widget

```
my $WidgetVariable = $Window -> WidgetType(?Option 1=>Value 1, ?Option 2=>Value 2 ??) -> pack();
```

The next three lines

```
my $button = $mw -> Button(-text => "Quit",
                -command => sub { exit })
        -> pack();
```

will create and display a button Here the widget variable is '$button' When we look at the options, we will find two options - 'text' and 'command' The given text is Quit - so the button will have the text "Quit" on it The command option determines what should happen when the user click on the button You can specify a function to execute when the user clicks on the button In this case the program will exit when this button is pressed One can also call functions that you have created from here

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

use Tk;
# Main Window
my $mw = new MainWindow;
my $label = $mw -> Label(-text=>"Hello World") -> pack();
my $button = $mw -> Button(-text => "Quit",
                -command =>\&exitProgam)
        -> pack();
MainLoop;

sub exitProgam {
        $mw->messageBox(-message=>"Goodbye");
        exit;
}
```

The next line - `MainLoop;` is the Main Loop or the Event Loop Its job is to invoke callbacks in response to events such as button presses or timer expirations If this line is missing, the program will run and exit with out waiting for the user to do any thing This is another one of those 'absolute musts' of Perl/Tk programming

Now Perl puritans will raise a great hue and cry and say that this is not the way to print "Hello World" The "pure" method is the following

```
#!/usr/local/bin/perl
print "Hello World"
```

Putting things in perspective, I am teaching Perl/Tk - not Perl The above is the Perl method of doing it My method is the pTk method of doing it

Widgets 1 : Button, Entry, Label

A widget is a user interface object in X graphical user interfaces Confused? Lets just say that it is the name of the object that appears on screen There are many types widgets If you want to display a button, you use the button widget For text, you use the text widget For entry, you guessed it, the entry widget

Syntax:

my $WidgetVariable = $Window -> WidgetType(?Option 1=>Value 1, ?Option 2=>Value 2 ??) -> pack();

Three things need to be said about widgets First is the **widget variable** This I have explained earlier The widget variable of all widgets must be unique and will be used whenever that widget needs to be accessed Second is the **options** Each widget has some options which can be used to configure it This is usually done when the widget is declared, but it can be done afterward also The final thing is **commands** Each widget has some commands which also

can be used to configure it or make it do some thing

But before we begin, we need to know a little about the pack command I have explained this earlier but just doing it one more time so that you don't have to push the back button Pack is a geometry manager Another geometry manager is 'grid' - we will explore that latter Pack is much more simpler than grid
The line *$hello -> pack;* tells the interpreter to pack the widget called "$hello"

Button

This will make a button It can be configured to execute some code when the button is pushed This will usually refer to a function so when the button is pushed, the function will run An button is shown below This button is created using HTML input tag

Some Options

-text=>"TEXT"	TEXT will be the text displayed on the button
-command=>CALLBACK	CALLBACK will be the code that is called when the button is pushed

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

# Main Window
my $mw = new MainWindow;

my $but = $mw -> Button(-text => "Push Me",
                  -command =>\&push_button);
$but -> pack();

MainLoop;

#This is executed when the button is pressed
sub push_button {
         whatever
}
```

You may have noticed that I used a slash(\) in the command callback (`-command =>\&push_button);`) Make sure that the slash stays there - to see why, go to the Most common mistakes by Perl/Tk beginners

---

Entry

An entry is a widget that displays a one-line text string and allows the user to input and edit text in it When an entry has the input focus it displays an insertion cursor to indicate where new characters will be inserted An entry element is shown using HTML

Some Options

-width=>NUMBER	Width of the input field NUMBER should be an integer
-textvariable=>\$VARIABLE	The contents of the variable VARIABLE will be displayed in the widget If the text in the widget is edited, the variable will be edited automatically
-state=>STATE	The state of the input field It can be **normal**, **disabled**, or **readonly** If it is **readonly** the text can't be edited

Some Commands

Syntax	**Description**	**Example**
$widget -> get();	The text inside input field can be taken by this command	$name = $ent -> get();
$widget -> delete(FIRST?,LAST?);	Delete one or more elements of the entry FIRST is	$ent -> delete(0,'end');

	the index of the first character to delete, and LAST is the index of the character just after the last one to delete If last isn't specified it defaults to FIRST+1, ie a single character is deleted This command returns an empty string	
$widget -> insert(*index*,"STRING");	Insert the characters of STRING just before the character indicated by *index* Index is 0 for the first character The word "end" can be used for the last character	$ent -> insert('end',"Hello");

Example

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

# Main Window
my $mw = new MainWindow;

#GUI Building Area
my $ent = $mw -> Entry() -> pack();
my $but = $mw -> Button(-text => "Push Me",
                  -command =>\&push_button);
$but -> pack();

MainLoop;

#This is executed when the button is pressed
sub push_button {
        $ent -> insert('end',"Hello");
}
```

Label

This widget display text messages

Some Options

-text => "TEXT"	TEXT will be the text displayed on the button
-font => FONT	Specifies the font to use when drawing text inside the widget You can specify just the font or you can give it in this format "FONTNAME SIZE STYLE" The STYLE can be bold, normal etc

Example

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

my $mw = new MainWindow; # Main Window

my $lab = $mw -> Label(-text=>"Enter name:") -> pack();
my $ent = $mw -> Entry() -> pack();
my $but = $mw -> Button(-text => "Push Me",
                  -command =>\&push_button);
$but -> pack();

MainLoop;

#This is executed when the button is pressed
sub push_button {
        $ent -> insert(0,"Hello, ");
}
```

Widgets 2 : Frame, Text, Scrollbar, Scale

Frame

A frame is a simple widget Its primary purpose is to act as a spacer or container for complex window layouts The only features of a frame are its background color and an optional 3-D border to make the frame appear raised or sunken Frame can be created just like any other widget -

```
my $frm = $mw -> Frame();
```

To place other widgets in this frame, you should use the frame widget variable as its parent Normally the parent is '$mw' or the MainWindow But if we wish to put a widget inside a frame, use the frame variable('$frm' in this case) in place of '$mw' Like this

```
my $lab = $frm_name -> Label(-text=>"Name:") -> pack();
```

Some Options

-relief=>STYLE	Specifies the 3-D effect desired for the widget Acceptable values are **raised, sunken, flat, ridge, solid**, and **groove** The value indicates how the interior of the widget should appear relative to its exterior; for example, raised means the interior of the widget should appear to protrude from the screen, relative to the exterior of the widget

Example

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

my $mw = new MainWindow; # Main Window

my $frm_name = $mw -> Frame() -> pack(); #New Frame
my $lab = $frm_name -> Label(-text=>"Name:") -> pack();
my $ent = $frm_name -> Entry() -> pack();

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button) -> pack();

MainLoop;

#This function will be executed when the button is pushed
sub push_button {
        $ent -> insert(0,"Hello, ");
}
```

Text

A text widget displays one or more lines of text and allows that text to be edited Similar to the entry widget but a larger version of it

Some Options

-xscrollcommand => COMMAND	This is to enable communication between a text widget and a scroll bar widget There is a -yscrollcommand similler to this one
-font => FONTNAME	Specifies the font to use when drawing text inside the widget
-width => NUMBER	Specifies the width of the widget
-height => NUMBER	Specifies the, you guessed it, height of the widget

Syntax	Description	Example

$widget -> get(*index1, ?index2 ?*);	Return a range of characters from the text The return value will be all the characters in the text starting with the one whose index is *index1* and ending just before the one whose index is *index2* (the character at *index2* will not be returned) If *index2* is omitted then the single character at *index1* is returned Note that the index of text is different from that of the entry widget The index of text widget is in the form LINE_NOCHARECTER_NO This means that 10 means the first character in the first line	$contents = $txt -> get(10,'end');
$widget -> insert(*index*,DATA);	Inserts all of the chars arguments just before the character at index If index refers to the end of the text (the character after the last newline) then the new text is inserted just before the last newline instead	$txt -> inset('end',"Hello World");

Example

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

my $mw = new MainWindow; # Main Window

my $frm_name = $mw -> Frame() -> pack();
my $lab = $frm_name -> Label(-text=>"Name:") -> pack();
my $ent = $frm_name -> Entry() -> pack();

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button) -> pack();
#Text Area
my $txt = $mw -> Text(-width=>40, -height=>10) -> pack();

MainLoop;

#This function will be executed when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"Hello, $name");
}
```

Scrollbar

A scroll bar is a widget that displays two arrows, one at each end of the scroll bar, and a slider in the middle portion of the scroll bar It provides information about what is visible in an associated window that displays an document of some sort (such as a file being edited or a drawing) The position and size of the slider indicate which portion of the document is visible in the associated window For example, if the slider in a vertical scroll bar covers the top third of the area between the two arrows, it means that the associated window displays the top third of its document It is made to work with other widgets like text **Some Options**

-orient=>DIRECTION	For widgets that can lay themselves out with either a horizontal or vertical orientation, such as scroll bars, this option specifies which orientation should be used DIRECTION must be either horizontal or vertical or an abbreviation of one of these
-command => COMMAND	This command gets executed when the scroll bar is moved This option almost always has

	a value such as t xview or t yview, consisting of the name of a widget and either xview (if the scroll bar is for horizontal scrolling) or yview (for vertical scrolling) All scrollable widgets have xview and yview commands that take exactly the additional arguments appended by the scroll bar

Example

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

my $mw = new MainWindow; # Main Window

my $frm_name = $mw -> Frame();
my $lab = $frm_name -> Label(-text=>"Name:");
my $ent = $frm_name -> Entry();

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button);

my $textarea = $mw -> Frame(); #Creating Another Frame
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                  -xscrollcommand=>['set',$srl_x]);

$lab -> grid(-row=>1,-column=>1);
$ent -> grid(-row=>1,-column=>2);
$frm_name -> grid(-row=>1,-column=>1,-columnspan=>2);

$but -> grid(-row=>4,-column=>1,-columnspan=>2);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);

MainLoop;

#This function will be executed when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"Hello, $name");
}
```

grid

As you can see I have used 'grid' here Grid is NOT a widget It is a geometry manager like pack but more advanced Lets take a closer look at the commands -

`$widget -> grid(-row=>1, -column=>1);`

This line will tell the interpreter to put the widget called '$txt' in the first row of the first column of its parent widget The below digram will help you understand

	Column 1	Column 2
Row 1	'$txt' widget will be here	'$srl_y' widget's place
Row 2	'$srl_x' widget's position	

Some Options

-sticky => STYLE	This option may be used to position (or stretch) the widget within its cell STYLE is a string that contains zero or more of the characters n, s, e or w Each letter refers to a side (north, south, east, or west) that the slave will "stick" to If both n and s (or e and w) are specified, the slave will be stretched to fill the entire height (or width) of its cavity
-ipadx => AMOUNT	The AMOUNT specifies how much horizontal internal padding to leave on each side of the slave(s) This is space is added inside the slave(s) border
-ipady => AMOUNT	The AMOUNT specifies how much vertical internal padding to leave on each side of the slave(s) Options same as -ipadx
-padx => AMOUNT	The amount specifies how much horizontal external padding to leave on each side of the slave(s), in screen units AMOUNT may be a list of two values to specify padding for left and right separately
-pady => AMOUNT	The amount specifies how much vertical external padding to leave on the top and bottom of the slave(s), in screen units Options same as -padx
-row => N	Insert the slave so that it occupies the Nth row in the grid Row numbers start with 0 If this option is not supplied, then the slave is arranged on the same row as the previous slave specified on this call to grid, or the first unoccupied row if this is the first slave
-column => N	Insert the slave so that it occupies the N'th column in the grid Options same as -row
-rowspan => N	Insert the slave so that it occupies N rows in the grid The default is one row
-columnspan => N	Insert the slave so that it occupies N columns in the grid

Using grid requires a bit of experience - but if you know HTML it would help a lot The rows and columns are just like those in HTML tables - although the codes are very different

---

Scale

Makes a slider that can be adjusted by the user to input a variable

Some Options

-from => NUMBER	Starting Number
-to => NUMBER	Ending Number
-tickinterval => NUMBER	Determines the spacing between numerical tick marks displayed below or to the left of the slider
-varable => NAME	Specifies the name of a global variable to link to the scale Whenever the value of the variable changes, the scale will update to reflect this value Whenever the scale is manipulated interactively, the variable will be modified to reflect the scale's new value

Syntax	**Description**	**Example**
$widget -> get();	Get the current value of the scale	my $age = $scl -> get();
$widget -> set(*value*);	Give the scale a new value	$scl -> set(20);

Example

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

#Global Variables
my $age = 10;
```

```
# Main Window
my $mw = new MainWindow;

#GUI Building Area
my $frm_name = $mw -> Frame();
my $lab = $frm_name -> Label(-text=>"Name:");
my $ent = $frm_name -> Entry();
#Age
my $scl = $mw -> Scale(-label=>"Age :",
         -orient=>'v',          -digit=>1,
         -from=>10,             -to=>50,
         -variable=>\$age,      -tickinterval=>10);

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button);

#Text Area
my $textarea = $mw -> Frame();
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                -xscrollcommand=>['set',$srl_x]);

#Geometry Management
$lab -> grid(-row=>1,-column=>1);
$ent -> grid(-row=>1,-column=>2);
$scl -> grid(-row=>2,-column=>1);
$frm_name -> grid(-row=>1,-column=>1,-columnspan=>2);

$but -> grid(-row=>4,-column=>1,-columnspan=>2);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);

MainLoop;

## Functions
#This function will be exected when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"$name is $age years old");
}
```

Now our little example is becoming more and more like a program We have added the comments to it as it has grown big and is difficult to understand Now we have added a slider with which age can be inputed

Dialogs

Dialogs can be called the elements in a program that detaches itself from the main window This is a VERY general definition and has many problems But for the moment, it will do Tk provides many dialogs

messageBox

This procedure creates and displays a message window with an application-specified message, an icon and a set of buttons Each of the buttons in the message window is identified by a unique symbolic name (see the -type options) After the message window is popped up, messageBox waits for the user to select one of the buttons Push the below button to see an example of messageBox

Some Options

-default=>*name*	Name gives the symbolic name of the default button for this message window ('ok', 'cancel', and so on) See -type for a list of the symbolic names If this option is not specified, the first

	button in the dialog will be made the default
-icon>*iconImage*	Specifies an icon to display IconImage must be one of the following: error, info, question or warning If this option is not specified, then the info icon will be displayed
-message>*string*	Specifies the message to display in this message box
-title>*String*	Specifies a string to display as the title of the message box The default value is an empty string
-type>*predefinedType*	Arranges for a predefined set of buttons to be displayed The following values are possible for predefinedType: **abortretryignore** Displays three buttons whose symbolic names are abort, retry and ignore **ok** Displays one button whose symbolic name is ok **okcancel** Displays two buttons whose symbolic names are ok and cancel **retrycancel** Displays two buttons whose symbolic names are retry and cancel **yesno** Displays two buttons whose symbolic names are yes and no **yesnocancel** Displays three buttons whose symbolic names are yes, no and cancel

Example

```
#!/usr/local/bin/perl
use Tk;
use strict;

# Main Window
my $mw = new MainWindow;
my $button = $mw-<Button(-text=<"Show Quit Dailog", -command =< \&exitTheApp)-<pack();

sub exitTheApp {
        my $response = $mw -< messageBox(-message=<"Really quit?",-type=<'yesno',-
icon=<'question');

        if( $response eq "Yes" ) {
                exit
        } else {
                $mw -< messageBox(-type=<"ok", -message=<"I know you like this application!");
        }
}

MainLoop;
```

---

chooseColor

chooseColor pops up a dialog box for the user to select a color

Some Options

-initialcolor>COLOUR	Specifies the color to display in the color dialog when it pops up

---

getOpenFile

The procedures **getOpenFile** and **getSaveFile** pop up a dialog box for the user to select a file to open or save The getOpenFile command is usually associated with the Open command in the File menu Its purpose is for the user to select an existing file only If the user enters an non-existent file, the dialog box gives the user an error prompt and requires the user to give an alternative selection If an application allows the user to create new files, it should do so by

providing a separate New menu command

The getSaveFile command is usually associated with the Save as command in the File menu If the user enters a file that already exists, the dialog box prompts the user for confirmation whether the existing file should be overwritten or not

Some Options

-initialdir=>DIRNAME	Specifies that the directories in directory should be displayed when the dialog pops up If this parameter is not specified, then the directories in the current working directory are displayed If the parameter specifies a relative path, the return value will convert the relative path to an absolute path
-defaultextension=>EXTENSION	Specifies a string that will be appended to the filename if the user enters a filename without an extension The default value is the empty string, which means no extension will be appended to the filename in any case
-filetypes=>filePatternList	If a File types listbox exists in the file dialog on the particular platform, this option gives the filetypes in this listbox When the user choose a filetype in the listbox, only the files of that type are listed If this option is unspecified, or if it is set to the empty list, or if the File types listbox is not supported by the particular platform then all files are listed regardless of their types This is a little tricky - see manual for information
-initialfile=>FILENAME	Specifies a filename to be displayed in the dialog when it pops up
-multiple	Allows the user to choose multiple files from the Open dialog

---

Toplevel

toplevel is a widget This can be used to create custom dialog boxes A toplevel is similar to a frame except that it is created as a top-level window: its X parent is the root window of a screen rather than the logical parent from its path name The primary purpose of a toplevel is to serve as a container for dialog boxes and other collections of widgets The only visible features of a toplevel are its background color and an optional 3-D border to make the toplevel appear raised or sunken

One can use toplevel to create new windows The widgets can be packed inside it in the same way widgets are packed inside a frame An example

```
#!/usr/local/bin/perl
use Tk;
# Main Window
$mw = new MainWindow;

my $lab = $mw -> Label(-text=>"This is the root window",
                -font=>"ansi 12 bold") -> pack;
my $but = $mw -> Button(-text=>"Click to Create Toplevel",
                -command=>\&makeTop)  -> pack;

MainLoop;

#A function to make a toplevel window
sub makeTop {
        my $top = $mw -> Toplevel(); #Make the window
        #Put things in it
        my $top_lab = $top -> Label(-text=>"This is the Toplevel window",
                        -font=>"ansi 12 bold") -> pack;
        my $txt = $top -> Text() -> pack;
        $txt -> insert('end', "Widgets can be packed in this window");

        #An option to close the window
        my $but_close = $top -> Button(-text=>"Close",
                -command => sub { destroy $top; } ) -> pack;
}
```

Widgets 3 : Radiobutton, Checkbutton

Radiobutton

Radiobutton is an input where any one of many choices MUST be chosen If one is chosen and another button is clicked, the last chosen will lose its state and the clicked button will be chosen A graphic example(in HTML) is given below

Choices ⦿ 1 | ○ 2 | ○ 3

Some Options

-command=>COMMAND	Specifies a command to associate with the button This command is typically invoked when mouse button 1 is released over the button window
-variable => \$VARIABLE	Specifies name of global variable to set to indicate whether or not this button is selected
-value => VALUE	Specifies value to store in the button's associated variable whenever this button is selected

Syntax	Description	Example
$widget -> deselect();	Deselects the checkbutton and sets the associated variable to its "off" value	$rdb_m -> deselect();
$widget -> select()	Selects the checkbutton and sets the associated variable to its "on" value	$rdb_m -> select();

Example

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

#Global Variables
my $age = 10;
my $gender = "Male";

# Main Window
my $mw = new MainWindow;

#GUI Building Area
my $frm_name = $mw -> Frame();
my $lab = $frm_name -> Label(-text=>"Name:");
my $ent = $frm_name -> Entry();
#Age
my $scl = $mw -> Scale(-label=>"Age :",
         -orient=>'v',          -digit=>1,
         -from=>10,             -to=>50,
         -variable=>\$age,      -tickinterval=>10);

#Gender
my $frm_gender = $mw -> Frame();
my $lbl_gender = $frm_gender -> Label(-text=>"Sex ");
my $rdb_m = $frm_gender -> Radiobutton(-text=>"Male",
                -value=>"Male",  -variable=>\$gender);
my $rdb_f = $frm_gender -> Radiobutton(-text=>"Female",
                -value=>"Female",-variable=>\$gender);


my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button);

#Text Area
my $textarea = $mw -> Frame();
```

```
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                -xscrollcommand=>['set',$srl_x]);

#Geometry Management
$lab -> grid(-row=>1,-column=>1);
$ent -> grid(-row=>1,-column=>2);
$scl -> grid(-row=>2,-column=>1);
$frm_name -> grid(-row=>1,-column=>1,-columnspan=>2);

$lbl_gender -> grid(-row=>1,-column=>1);
$rdb_m -> grid(-row=>1,-column=>2);
$rdb_f -> grid(-row=>1,-column=>3);
$frm_gender -> grid(-row=>3,-column=>1,-columnspan=>2);

$but -> grid(-row=>4,-column=>1,-columnspan=>2);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);

MainLoop;

## Functions
#This function will be executed when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"$name\($gender\) is $age years old");
}
```

This time the program is subjected to even more change - the geometry manager is fully grid now There is no instances of pack You will find this necessary when the layout becomes more complicated I hope you can stay with me in such trying times

---

Checkbutton

Checkbotton is a input with two options - Off or On - it has to be either one The state can be changed by clicking on it An example is shown below

check box

Some Options

-offvalue=>VALUE	Specifies value to store in the button's associated variable whenever this button is deselected Defaults to ``0''
-onvalue=>VALUE	Specifies value to store in the button's associated variable whenever this button is selected Defaults to ``1''
-command=>CALLBACK	Specifies a command to associate with the button This command is typically invoked when mouse button 1 is released over the button window
-variable=>\$VARABLE	Specifies name of global variable to set to indicate whether or not this button is selected

Syntax	**Description**	**Example**
$widget -> deselect();	Deselects the checkbutton and sets the associated variable to its ``off'' value	$chk -> deselect();
$widget -> select();	Selects the checkbutton and sets the associated variable to its ``on'' value	$chk -> select();

$widget -> toggle();	Toggles the selection state of the button, redisplaying it and modifying its associated variable to reflect the new state	$chk -> toggle();

Example

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

#Global Variables
my $age = 10;
my $gender = "Male";
my $occupied = 1;

# Main Window
my $mw = new MainWindow;

#GUI Building Area
my $frm_name = $mw -> Frame();
my $lab = $frm_name -> Label(-text=>"Name:");
my $ent = $frm_name -> Entry();
#Age
my $scl = $mw -> Scale(-label=>"Age :",
         -orient=>'v',          -digit=>1,
         -from=>10,             -to=>50,
         -variable=>\$age,      -tickinterval=>10);

#Jobs
my $chk = $mw -> Checkbutton(-text=>"Occupied",
        -variable=>\$occupied);
$chk -> deselect();

#Gender
my $frm_gender = $mw -> Frame();
my $lbl_gender = $frm_gender -> Label(-text=>"Sex ");
my $rdb_m = $frm_gender -> Radiobutton(-text=>"Male",
                -value=>"Male",  -variable=>\$gender);
my $rdb_f = $frm_gender -> Radiobutton(-text=>"Female",
                -value=>"Female",-variable=>\$gender);

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button);

#Text Area
my $textarea = $mw -> Frame();
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                -xscrollcommand=>['set',$srl_x]);

#Geometry Management
$lab -> grid(-row=>1,-column=>1);
$ent -> grid(-row=>1,-column=>2);
$frm_name -> grid(-row=>1,-column=>1,-columnspan=>2);

$scl -> grid(-row=>2,-column=>1);
$chk -> grid(-row=>2,-column=>2,-sticky=>'w');

$lbl_gender -> grid(-row=>1,-column=>1);
$rdb_m -> grid(-row=>1,-column=>2);
$rdb_f -> grid(-row=>1,-column=>3);
$frm_gender -> grid(-row=>3,-column=>1,-columnspan=>2);

$but -> grid(-row=>4,-column=>1,-columnspan=>2);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
```

```
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);

MainLoop;

## Functions
#This function will be executed when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"$name\($gender\) is $age years old");
}
```

Widgets 4 : Listbox

Listbox

A listbox is a widget that displays a list of strings, one per line When first created, a new listbox has no elements Elements may be added or deleted using widget commands described below

Some Options

-selectmode => MODE	Specifies one of several styles for manipulating the selection The MODE may be arbitrary, but the default bindings expect it to be either **single, browse, multiple**, or **extended**; the default value is **browse**

Some Commands

Syntax	**Description**	**Example**
$widget -> curselection();	Returns a list containing the numerical indices of all of the elements in the listbox that are currently selected If there are no elements selected in the listbox then an empty string is returned	$sel = $lst -> curselection();
$widget -> delete(*first,?last?*);	Deletes one or more elements of the listbox First and last are indices specifying the first and last elements in the range to delete If last isn't specified it defaults to first, ie a single element is deleted	$lst -> delete(5);
$widget -> get(*first,?last?*);	If last is omitted, returns the contents of the listbox element indicated by first, or an empty string if first refers to a non-existent element If last is specified, the command returns a list whose elements are all of the listbox elements between first and last, inclusive	$lst -> get(5,end);
$widget -> index(*index*);	Returns the integer index value that corresponds to *index* If *index* is end the return value is a count of the number of elements in the listbox (not the index of the last element)	$lst -> index(5);
$widget -> insert(*index,? element element ?*);	Inserts zero or more new elements in the list just before the element given by *index* If *index* is specified as end then the new elements are added to the end of the list Returns an empty string	$lst -> insert('end',"me");
$widget -> size();	Returns a decimal string indicating the total number of elements in the listbox	$count = $lst -> size();

Example

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

#Global Variables
my $age = 10;
my $occupied = 1;
my $gender = "Male";

# Main Window
my $mw = new MainWindow;

#GUI Building Area
my $frm_name = $mw -> Frame();
my $lab = $frm_name -> Label(-text=>"Name:");
my $ent = $frm_name -> Entry();
#Age
my $scl = $mw -> Scale(-label=>"Age :",
         -orient=>'v',          -digit=>1,
         -from=>10,                     -to=>50,
         -variable=>\$age,      -tickinterval=>10);

#Jobs
my $frm_job = $mw -> Frame();
my $chk = $frm_job -> Checkbutton(-text=>"Occupied",
        -variable=>\$occupied);
$chk -> deselect();
my $lst = $frm_job -> Listbox(-selectmode=>'single');

#Adding jobs
$lst -> insert('end',"Student","Teacher","Clerk","Business Man",
        "Militry Personal","Computer Expert","Others");

#Gender
my $frm_gender = $mw -> Frame();
my $lbl_gender = $frm_gender -> Label(-text=>"Sex ");
my $rdb_m = $frm_gender -> Radiobutton(-text=>"Male",
                -value=>"Male",  -variable=>\$gender);
my $rdb_f = $frm_gender -> Radiobutton(-text=>"Female",
                -value=>"Female",-variable=>\$gender);

my $but = $mw -> Button(-text=>"Push Me", -command =>\&push_button);

#Text Area
my $textarea = $mw -> Frame();
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                -xscrollcommand=>['set',$srl_x]);

#Geometry Management
$lab -> grid(-row=>1,-column=>1);
$ent -> grid(-row=>1,-column=>2);
$scl -> grid(-row=>2,-column=>1);
$frm_name -> grid(-row=>1,-column=>1,-columnspan=>2);

$chk -> grid(-row=>1,-column=>1,-sticky=>'w');
$lst -> grid(-row=>2,-column=>1);
$frm_job -> grid(-row=>2,-column=>2);

$lbl_gender -> grid(-row=>1,-column=>1);
$rdb_m -> grid(-row=>1,-column=>2);
$rdb_f -> grid(-row=>1,-column=>3);
$frm_gender -> grid(-row=>3,-column=>1,-columnspan=>2);

$but -> grid(-row=>4,-column=>1,-columnspan=>2);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
```

```
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);

MainLoop;

## Functions
#This function will be executed when the button is pushed
sub push_button {
        my $name = $ent -> get();
        $txt -> insert('end',"$name\($gender\) is $age years old and is ");

        my $job = "";
        #See whether he is employed
        if ( $occupied == 1 ) {
                my $job_id = $lst -> curselection(); #Get the no of selected jobs
                if ( $job_id eq "" ) { #If there is no job
                        $job = "a Non worker";
                }
                else {
                        $job = $lst -> get($job_id) ;#Get the name of the job
                        $txt -> insert('end',"a $job");
                }
        }
        else {
                $txt -> insert('end',"unemployed");
        }
}
```

Wow! Our 'little' example is a big (and utterly pointless) program now I am going to stop 'exampling' from now on This is quite complicated isn't it? Why don't you run the script and see what a beautiful script we made Copy the above script and paste it in a file called "infopl" and double click the file Voila! We are Perl/Tk programmers

Widgets 5 : Menubutton, Menu, Optionmenu

Menubutton

A menubutton is a widget that displays a textual string, bitmap, or image and is associated with a menu widgetIn normal usage, pressing left-clicking the menubutton causes the associated menu to be posted just underneath the menubutton

Some Options

-direction => DIRECTION	Specifies where the menu is going to be popup up above tries to pop the menu above the menubutton below tries to pop the menu below the menubutton left tries to pop the menu to the left of the menubutton right tries to pop the menu to the right of the menu button flush pops the menu directly over the menubutton
-menu => NAME	Specifies the path name of the menu associated with this menubutton The menu must be a child of the menubutton

---

Menu

A menu is a widget that displays a collection of one-line entries arranged in one or more columns There exist several different types of entries, each with different properties Entries of different types may be combined in a single menu Menu entries are not the same as entry widgets In fact, menu entries are not even distinct widgets; the entire menu is one widget

Some Options

-tearoff => BOOLEAN	This option must have a proper boolean value, which specifies whether or not the menu should include a tear-off entry at the top If so, it will exist as entry 0 of the menu and the other entries will number starting at 1 The default menu bindings arrange for the menu to be torn off when the tear-off entry is invoked
-title => STRING	The string will be used to title the window created when this menu is torn off If the title is NULL, then the window will have the title of the menubutton or the text of the cascade item from which this menu was invoked
-type => OPTION	This option can be one of menubar, tearoff, or normal, and is set when the menu is created While the string returned by the configuration database will change if this option is changed, this does not affect the menu widget's behavior

Some Commands

<table>
<tr><th>Syntax</th><th>Description</th></tr>
<tr><td>$widget -> TYPE(?option=>value,option=>value,?);</td><td>Add a new entry to the bottom of the menu The new entry's type is given by TYPE and must be one of cascade, checkbutton, command, radiobutton, or separator, or a unique abbreviation of one of the above If additional arguments are present, they specify any of the following options:
-accelerator => VALUE — Specifies a string to display at the right side of the menu entry Normally describes an accelerator keystroke sequence that may be typed to invoke the same function as the menu entry This option is not available for separator or tear-off entries
-columnbreak => VALUE — When this option is zero, the appears below the previous entry When this option is one, the menu appears at the top of a new column in the menu
-label => VALUE — Specifies a string to display as an identifying label in the menu entry Not available for separator or tear-off entries
-compound => VALUE — Specifies whether the menu entry should display both an image and text, and if so, where the image should be placed relative to the text Valid values for this option are bottom, center, left, none, right and top
-image => VALUE — Specifies an image to display in the menu instead of a text string or bitmap The image must have been created by some previous invocation of image create This option overrides the -label and -bitmap options but may be reset to an empty string to enable a textual or bitmap label to be displayed This option is not available for separator or tear-off entries
-underline => VALUE — Specifies the integer index of a character to underline in the entry This option is used to make keyboard shortcuts 0 corresponds to the first character of the text displayed in the entry, 1 to the next character, and so on</td></tr>
<tr><td>$widget -> delete(index1, ?index2?);</td><td>Delete all of the menu entries between index1 and index2 inclusive If index2 is omitted then it defaults to index1 Attempts to delete a tear-off menu entry are ignored (instead, you should change the tearOff option to remove the tear-off entry)</td></tr>
<tr><td>$widget -> insert(index,type, ? option=>value ?);</td><td>Same as the add widget command except that it inserts the new entry just before the entry given by index, instead of appending to the end of the menu The type, option, and value arguments have the same interpretation as for the add widget command It is not possible to insert new menu entries before the tear-off entry, if the menu has one</td></tr>
</table>

Example

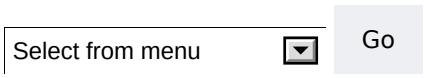
```
#!/usr/local/bin/perl
use Tk;
# Main Window
my $mw = new MainWindow;

#Making a text area
my $txt = $mw -> Scrolled('Text',-width => 50,-scrollbars=>'e') -> pack ();

#Declare that there is a menu
my $mbar = $mw -> Menu();
$mw -> configure(-menu => $mbar);

#The Main Buttons
my $file = $mbar -> cascade(-label=>"File", -underline=>0, -tearoff => 0);
my $others = $mbar -> cascade(-label =>"Others", -underline=>0, -tearoff => 0);
my $help = $mbar -> cascade(-label =>"Help", -underline=>0, -tearoff => 0);

## File Menu ##
$file -> command(-label => "New", -underline=>0,
                -command=>sub { $txt -> delete('10','end');} );
$file -> checkbutton(-label =>"Open", -underline => 0,
                -command => [\&menuClicked, "Open"]);
$file -> command(-label =>"Save", -underline => 0,
                -command => [\&menuClicked, "Save"]);
$file -> separator();
$file -> command(-label =>"Exit", -underline => 1,
                -command => sub { exit } );

## Others Menu ##
my $insert = $others -> cascade(-label =>"Insert", -underline => 0, -tearoff => 0);
$insert -> command(-label =>"Name",
        -command => sub { $txt->insert('end',"Name : Binny V A\n");});
$insert -> command(-label =>"Website", -command=>sub {
        $txt->insert('end',"Website : http://wwwgeocitiescom/binnyva/\n");});
$insert -> command(-label =>"Email",
        -command=> sub {$txt->insert('end',"E-Mail : binnyva\@hotmailcom\n");});
$others -> command(-label =>"Insert All", -underline => 7,
        -command => sub { $txt->insert('end',"Name : Binny V A
Website : http://wwwgeocitiescom/binnyva/
E-Mail : binnyva\@hotmailcom");
        });

## Help ##
$help -> command(-label =>"About", -command => sub {
        $txt->delete('10','end');
        $txt->insert('end',
        "About
----------
This script was created to make a menu for a\nPerl/Tk tutorial
Made by Binny V A
Website : http://wwwgeocitiescom/binnyva/code
E-Mail : binnyva\@hotmailcom"); });

MainLoop;

sub menuClicked {
        my ($opt) = @_;
        $mw->messageBox(-message=>"You have clicked $opt
This function is not implanted yet");
}
```

Create the main buttons as cascade menus and create the menus as their slaves For more information see the manual

---

Optionmenu

Makes a button, which when clicked on shows a list with available options Useful when user has to make one choice when multiple choices are given Below is a options menu in HTML A word of caution though - Perl/Tk's option menu has a very different appearance



Syntax

```
my $widget = $mw -> Optionmenu(?option=>value,option=>value,?);
```

Options

Syntax	Description
-options=>OPTIONS	(Re)sets the list of options presented
-command=>CALLBACK	Defines the callback that is invokes when a new option is selected
-variable=>\$VARIABLE	Reference to a scalar that contains the current value of the selected option

Methords

Syntax	Description	Example
$widget -> addOptions(*[Option1=>Value1], ? [Option2=>Value2]?*);	Adds newly given options to the already available options	$opt->addOptions([May=>5], [June=>6], [July=>7], [Augest=>8]);

Example

```
#!/usr/local/bin/perl
use Tk;
# Main Window
$mw = new MainWindow;

my $var;
my $opt = $mw -> Optionmenu(-options => [qw(January February March April)],
        -command => sub { print "got: ", shift, "\n" },
        -variable => \$var,
        )->pack;
$opt->addOptions([May=>5],[June=>6],[July=>7],[Augest=>8]);

$mw->Label(-textvariable=>\$var, -relief=>'groove')->pack;
$mw->Button(-text=>'Exit', -command=>sub{$mw->destroy})->pack;

MainLoop;
```

Some more Widgets - Canvas, Message, Adjuster, Scrolled

Canvas

The canvas widget is a very important widget as all points are addressable graphical drawing area Canvas widgets implement structured graphics A canvas displays any number of items, which may be things like rectangles, circles, lines, and text Items may be manipulated (eg moved or re-colored) and commands may be associated with items So if you don't like the paint program in windows, you can make your own program using this widget

The command *$widget* `-> create` *type options* is used to make different structures A few examples are given below For more information read the manual

Example

```
#!/usr/local/bin/perl
use Tk;
# Main Window
my $mw = new MainWindow;

my $cns = $mw -> Canvas(-relief=>"sunken", -background=>"blue");
$cns -> create('polygon',5,100,50,5,150,5,200,100,5,100,
        -joinstyle=>"bevel", -fill=>"red", -outline=>"white", -width=>5);
$cns -> create('oval',200,100,300,200, -fill=>"green");
$cns -> create('oval',100,150,300,100, -fill=>"white", -width=>0);
$cns -> create('rectangle',10,150,100,250, -dash=>[6,4,2,4,2,4]);
$cns -> pack;

MainLoop;
```

Message

A message is a widget that displays a textual string Much like the label widget but this can be used to make a multi-line text

The *-justify* option specifies how to justify lines of text Must be one of **left**, **center**, or **right** Defaults to **left** This option works together with the anchor, aspect, padX, padY, and width options to provide a variety of arrangements of the text within the window

---

Adjuster

An adjuster acts like the frame widget - with one notable exception The borders can be dragged and expended This widget contains any number of panes, arranged horizontally or vertically, according to the value of the *-orient* option Each pane contains one widget, and each pair of panes is separated by a movable sash Moving a sash can be done by dragging it This causes the widgets on either side of the sash to be resized

Some Options

-side=>DIRECTION	Specifies the side on which the managed widget lies relative to the Adjuster In conjunction with the pack geometry manager, this relates to the side of the master against which the managed widget and the Adjuster are packed Must be **left, right, top**, or **bottom** Defaults to **top**

Some Methods

$adjuster -> packAfter($widget, ?*pack_options*?)	This command configures the Adjuster's -widget and -side options respectively to '$widget' and the -side value specified in pack_options (top if not specified) It then packs the Adjuster after '$widget', with -fill set to **x** or **y** as appropriate

Example

```
use Tk;
use Tk::Adjuster;
my $mw = new MainWindow;

my $adj = $mw -> Adjuster();
my $lst = $mw -> Listbox();
```

```
$lst -> insert('end', "Item 1");
$lst -> insert('end', "Item 2");
$lst -> insert('end', "Item 3");
$lst -> insert('end', "Item 4");
$lst -> insert('end', "Item 5");

my $txt = $mw -> Scrolled("Text",-scrollbars=>'e');
$txt -> insert('end',"To Hack With It

     To Compute
     Or Not To Compute
     That Is The Question
     Whether 'Tis Nobler In The Memory Bank
     To Suffer The Slings And Circuits Of Outrageous Functions
     Or To Take Up Arms Against A Sea OfTransistors,
     Or Rather Transponders
     Transcondu--
     Trans
     Er           Oh, To Hack With It");


my $side = "left";
$lst -> pack(-side => $side, -fill => 'both', -expand => 1);
$adj -> packAfter($lst, -side => $side);
$txt -> pack(-side => $side, -fill => 'both', -expand => 1);

MainLoop;
```

Scrolled

Scrolled is a derived widget that creates a widget with attached scrollbar(s) If you have tried to create a widget and attach scrollbars to it, you will doubtlessly understand the usefulness of this feature It greatly reduces the number of code for these situations

```
my $widget = $parent -> Scrolled('WIDGET' ?,-scrollbars=>WHERE? ?,?);
```

WIDGET can be any widget that supports scrolling For example, Text, Listbox, etc

Some Options

-scrollbars => SIDE	Expects as argument the position where the scrollbars should be created: **w, e** or **n, s** or a combination of them If the one or both positions are prefixed with **o** the scrollbar will only show up if there is a 'real' need to scroll

Example:

```
use Tk;
my $mw = new MainWindow;

my $txt = $mw -> Scrolled('Text',-scrollbars=>"oe") -> pack;
$txt -> insert('end',
"Arthur: \"It's at times like this I wish I'd listened to my mother\"
Ford : \"Why, what did she say?\"
Arthur: \"I don't know, I never listened\"
                Douglas Adams");

MainLoop;
```

The same example using hand coded scrollbars

```
use Tk;
my $mw = new MainWindow;
```

```
my $textarea = $mw -> Frame();
my $txt = $textarea -> Text(-width=>40, -height=>10);
my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                  -xscrollcommand=>['set',$srl_x]);
$txt -> insert('end',
"Arthur: \"It's at times like this I wish I'd listened to my mother\"
Ford : \"Why, what did she say?\"
Arthur: \"I don't know, I never listened\"
                  Douglas Adams");

#Geometry
$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>1,-column=>1,-columnspan=>2);

MainLoop;
```

As you can see, the Scrolled widget saves quite a bit of typing

Geometry Management : Grid, Pack

grid

The grid command is used to communicate with the grid geometry manager that arranges widgets in rows and columns inside of another window, called the geometry master (or master window) The grid command can have any of several forms, depending on the option argument
In short, grid is the name given to the thingy that will place your widget where you want it to be placed

Some Options

-sticky => STYLE	This option may be used to position (or stretch) the widget within its cell STYLE is a string that contains zero or more of the characters n, s, e or w Each letter refers to a side (north, south, east, or west) that the slave will "stick" to If both n and s (or e and w) are specified, the slave will be stretched to fill the entire height (or width) of its cavity
-ipadx => AMOUNT	The AMOUNT specifies how much horizontal internal padding to leave on each side of the slave(s) This is space is added inside the slave(s) border
-ipady => AMOUNT	The AMOUNT specifies how much vertical internal padding to leave on each side of the slave(s) Options same as -ipadx
-padx => AMOUNT	The amount specifies how much horizontal external padding to leave on each side of the slave(s), in screen units AMOUNT may be a list of two values to specify padding for left and right separately
-pady => AMOUNT	The amount specifies how much vertical external padding to leave on the top and bottom of the slave(s), in screen units Options same as -padx
-row => N	Insert the slave so that it occupies the Nth row in the grid Row numbers start with 0 If this option is not supplied, then the slave is arranged on the same row as the previous slave specified on this call to grid, or the first unoccupied row if this is the first slave
-column => N	Insert the slave so that it occupies the N'th column in the grid Options same as -row
-rowspan => N	Insert the slave so that it occupies N rows in the grid The default is one row
-columnspan => N	Insert the slave so that it occupies N columns in the grid

Example

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

```
# Main Window
my $mw = new MainWindow;
#Text Area
my $txt = $mw -> Text(-width=>40, -height=>10);
my $srl_y = $mw -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $mw -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
                  -xscrollcommand=>['set',$srl_x]);

#Geometry Management
$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");

MainLoop;
```

Lets take a closer look at the commands -
$txt -> grid(-row=>1,-column=>1);
This line will tell the interpreter to put the widget called '$txt' in the first row of the first column of its parent widget
The below digram will help you understand

	Column 1	Column 2
Row 1	'$xt' widget will be here	'$srl_y' widget's place
Row 2	'$srl_x' widget's position	

Using grid requires a bit of experience - but if you know HTML it would help a lot The rows and columns are just like those in HTML tables The concept is the same but the codes are very different

---

pack

Pack is also a geometry manager like Grid - but much simpler You don't have to specify the rows and columns as you did for grid This is for you lazybones out there Just put `$widget -> pack;` and the widget will be packed But for more complex arrangements with pack one must use frames

Some Options

-expand=>BOOLEAN	Specifies whether the slaves should be expanded to consume extra space in their master Sounds a lot like slaves eating up their masters, don't it? It just means that when the application is resized the widgets will automatically fill the space
-fill=>STYLE	This will stretch the widget in **x** direction, **y** direction or **both** directions
-side=>SIDE	Specifies which side of the master the slave(s) will be packed against Must be **left, right, top**, or **bottom** Defaults to **top**

Now for the example

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

# Main Window
my $mw = new MainWindow;
my $lab = $mw -> Label(-text=>"Do You Remember When");
my $txt = $mw -> Text();
$txt->insert('end',"A Computer Was Something On TV From A Science Fiction Show
A Window Was Something You Hated To Clean
And Ram Was The Cousin Of A Goat
```

```
Meg Was The Name Of My Girlfriend
And Gig Was Your Thumb Upright
Now They All Mean Different Things
And That Mega Bytes

An Application Was For Employment
A Program Was A TV Show
A Cursor Used Profanity
A Keyboard Was A Piano

Compress Was Something You Did To The Garbage
Not Something You Did To A File
And If You Unzipped Anything In Public
You'd Be In Jail For A While

Log On Was Adding Wood To The Fire
Hard Drive Was A Long Trip On The Road
A Mouse Pad Was Where A Mouse Lived
And A Backup Happened To Your Commode

Cut You Did With A Pocket Knife
Paste You Did With Glue
A Web Was A Spider's Home
And A Virus Was The Flu

I Guess I'll Stick To My Pad And Paper
And The Memory In My Head
I Hear Nobody's Been Killed In A Computer Crash
But, When It Happens They Wish They Were Dead");

my $srl = $mw -> Scrollbar(-command=>[$txt,'yview']);
$txt -> configure(-yscrollcommand=>[$srl,'set']);

#The packing commands
$lab -> pack;
$txt -> pack(-expand => 1, -fill => "both", -side => "left");
$srl -> pack(-expand => 1, -fill => "y");

MainLoop;
```

Some Common Widget Options

The capitalized word must be replaced with any of the bold words Check the manual for all the options Also make sure that the widget you are using support the option you are using

-anchor=>POSITION	Position widget relative to: **n ne nw s se sw e w** or **center**
-background=>COLOR	Background color
-borderwidth=>WIDTH	Width of border Choosing '0' will hide the border
-command=>SCRIPT	Executes SCRIPT when invoked
-cursor=>CURSOR	Mouse cursor to display when mouse in this widget
-disabledforeground=>COLOR	Foreground color when the widget is disabled
-font=>FONTNAME	Use FONTNAME for text style It is given in this format - FONTNAME SIZE STYLE For example, `-font => "fixed 12 bold"`
-foreground=>COLOR	Foreground when the state of the widget is normal
-	Can sets horizontal (and optionally vertical) widget size

geometry=>WIDTHxHEIGHT	
-height=>VALUE	Set vertical height of the widget
-image=>IMAGE	The image to display in widget
-justify=>JUSTIFICATION	Multi-line justification: **left, right** or **center**
-orient=>ORIENTATION	For objects like scale and scrollbar Must be **horizontal** or **vertical**(or **h** or **v**)
-padx=>NUMBER	Horizontal padding The space on top and bottom of the given widget For example, -padx=>5
-pady=>NUMBER	Vertical padding This is the space on either side of the given widget For example, -pady=>2
-relief=>RELIEF	3D bevel border - RELIEF must be **flat, groove, raised, ridge, solid** or **sunken** Very useful when using frame
-state=>STATE	Set widget to: **normal, disabled** or **active**
-text=>STRING	Display text string
-variable=>VARNAME	Display text variable If the widget is manipulated, the value of the variable will change automatically and if this variable's value is changed, the widget will change to reflect the changes in the variable
-underline=>CHAR	Which character position in string to underline This must be a number - 0 for the first character, 1 for the second and so on
-width=>WIDTH	Sets horizontal widget size
-wraplength=>LENGTH	Word wrapping maximum string length

Some Tk Commands

Bind

The bind command associates Perl code with events If you want to do something when the user double-clicks a item in a listbox or when he/she press any button(say F1), **bind** is what you need Lets bind something in the next example

Syntax:
`$widget -> bind(<sequence>, Callback);`
<sequence> stands for the sequence of button/mouse presses It should be given in the following pattern <modifier-modifier-type-detail> For example
<Control-Alt-Key-t> - Control+Alt+T
Control and *Alt* are the modifiers and *Key* is the type with *t* as the detail See the next example and you will understand

```
#!/usr/local/bin/perl
use Tk;
# Main Window
my $mw = new MainWindow;

my $lab = $mw -> Label(-text=> " The Bind command ",
        -font=>"ansi 12 bold") -> pack;
my $lst = $mw -> Listbox() -> pack;
$lst -> insert('end',"Don't double-click this");
$lst -> insert('end',"Don't double-click this either");
```

```
$lst -> insert('end',"Don't even think of double-clicking this");
$lst -> insert('end',"You may double-click this");
$lst -> insert('end',"No Negative Nay Nope Get the message?");
#Bind the double click event to the list box
$lst -> bind ('<Double-ButtonPress-1>', sub { double() });

my $all_keys = $mw -> Label(-justify=>"left",
        -text=>"Press any of the following
Control+A
Control+Shift+A
Control+Alt+T
Right click
Control+Escape") -> pack;

#Exit when the escape key is pressed
$mw -> bind('<Key-Escape>', sub { exit });
#Shows a helping dialog box when F1 is pressed
$mw -> bind('<Key-F1>',sub { help(); });
#Binds misc keys
$mw -> bind('<Control-Key-a>', sub {
        $mw -> messageBox(-message=>"You pressed Control+A, did'nt you?")
});#Control+A
$mw -> bind('<Control-Key-A>', sub {
        $mw -> messageBox(-message => "Control+Shift+A, right?");
        });#Control+Shift+A
$mw -> bind('<Control-Alt-Key-t>', sub {
        $mw -> messageBox(-message => "Control, Alt and T");
});#Control+Alt+T
$mw -> bind('<ButtonPress-3>', sub {
        $mw -> messageBox(-message => "The right way to click");
});#Right click
$mw -> bind('<Control-Key-Escape>', sub {
        $mw -> messageBox(-message => "You must be a married man
What you pressed remindes married men of what they never will have
 - Control or Escape");
});#Control+Escape

MainLoop;

#The helping function
sub help {
        $mw -> messageBox(-message=>"Did you ask of help?
You ain't getting any
Ha Ha Ha!");
}

#This happens at double-click
sub double {
        $mw -> messageBox(-message=>"You double clicked something
This script is simple - so it won't display what you clicked on
But if you want a sript that is able to do that,
write to me at binnyva\(at\)hotmail\(dot\)com
and I will send you a better script");
}
```

Now What?

Perilous seas crossed, thought scorching desert, we have come here after overcoming countless dangers to life and limb We have reached the end of our quest, the end of our journey Now What?

Now you are on your own You must strive to achieve excellence in the field of Perl/Tk programming Buy books, visit websites, steal codes Do anything to increase your knowledge on this subject There are an amazing array of websites out there that would supply you with tutorials, programs, games etc on Perl/Tk Go get them Read code of other people and study from them Remember, creativity is great, but plagiarism is faster

But the most important thing - program Get those fingers typing, make programs in Perl/Tk Experiment Make

mistakes Correct them There is no better way of learning a subject than by doing it

See references for more links on Perl/Tk

Reference

Books

'**Mastering Perl/Tk**' by Steve Lidie and Nancy Walsh is a good book that teaches how to create Graphical User Interfaces in Perl

Learning Perl/Tk

Programming Perl' by Larry Wall is the standard book on Perl If you consider serious Perl programming, this is a must This is about Perl rather than about Perl/Tk

Get More Books on Perl/Tk

Manual

Make yourself familiar with the manual that comes with Perl To know more about any commands, say print, just type `perldoc -f print` in the command prompt This works on both Unix and windows machines Other that that, Perl ships with tons of HTML tutorials that teaches everything in perl

External Sites

Perl's Usenet group at complangperltk is of great help to any Perl Tk programmer

http://wwwperlorg/ has been for ages the primary resource site for many perl developers

http://learnperlorg/ is a site that has resources for both beginners and advanced perl programmers

Appendix

Appendix A : About the Author

My name is Binny V A I have been programming in Perl and Tcl/Tk for a while now - but only recently have I started programming with Perl/Tk So I am not an expert in Perl/Tk But I have already written a tutorial for Tcl/Tk So I thought that I will translate the Tk parts to perl and give the world a new Perl/Tk program So don't be angry if you see a lot of similarities between Tcl/Tk Tutorial and this Perl/Tk tutorial I can plagiarise from myself, can't I? Anyway, all questions, suggestions, criticisms etc can be directed to binnyva@gmailcom You can know more about me at my website For all the programs that I made in Perl, go to the perl page in my site

If you have liked this tutorial, I have wrote another tutorial on Tcl/Tk language Check it out
I also wrote tutorials on CGI-Perl and JavaScript

Appendix B : Commonly Made mistakes in Perl/Tk

There are a few 'irregularities' at the places where Perl and Tk meet These places are often a source of problems for new Perl/Tk programmers Here is a list of most common mistakes made by new Perl/Tk programmers I made quite a few of them myself

Callbacks

Calling a function from a button is almost always trouble for the inexperienced Assume that you created a function called `showGreeting` `$button = $mw -> Button(-text=>"Hello",-command=>showGreeting);` #- Won't work
`$button = $mw -> Button(-text=>"Hello",-command=>showGreeting());` #- Again, won't work
`$button = $mw -> Button(-text=>"Hello",-command=>&showGreeting);` #- Will not work
This is the porper way of doing it
`$button = $mw -> Button(-text=>"Hello",-command=>\&showGreeting);` #- Finally Something that works
`$button = $mw -> Button(-text=>"Hello",-command=>sub { showGreeting(); } );` #- Another method of doing it

Variables
This has a similar effect as the above problem
`$entry = $mw -> Entry(-textvariable=>$var);` #- This is not the right way
`$entry = $mw -> Entry(-textvariable=>\$var);` #- Correct way

Using qw//
Don't use a white space inside a value if you are using the qw// method to configure options See below for more details

---

Appendix C : Tcl/Tk And Perl/Tk

If you are more squinted with the Tcl/Tk's way of doing it, you will be happy to know that there is a way of giving the options in the Tcl/Tk style

```
$label = $mw -> Label(qw/-text Hello -font courierfont -relief raised/) ->
pack();
```

qw function will split the given string at white space It can be understood as being roughly equivalent to:

```
split(' ', q/STRING/);
```

So the no space is allowed inside values if you are using this For example, the following line will create an error

```
$label = $mw -> Label(qw/-text "Hello World" -font courierfont -relief raised/)
-> pack();
```

Even using quotes(") will give unexpected results

```
$label = $mw -> Label(qw/-text "Hello" -font courierfont -relief "raised"/) ->
pack();
```

Appendix D : Codes

Almost all the programs in the tutorial are available in a zipped format for download

---

Appendix E : FeedBacks

How do you like this tutorial? Pen some comments below

If you have any questions send me an e-mail at binnyva@gmailcom If you leave a question in the above feedback form, be sure to give you e-mail address if you want me to respond

Appendix F : Comments

Index

Anonymous at 11 Mar, 2008 04:26

Hi
every time i try to access perl-tk module it gives segmentation fault i used apt-get to install perl and perl-tk modules

Anonymous at 01 Jun, 2008 10:30
try installing them with **CMA**

Introduction

Anshuman Atre at 26 Feb, 2007 08:41
Hi,

I have just finished a great tool with many command line options It also has a basic CLI based user interface to make it interactive

I happen to a own the O'Rielly "Perl/Tk" that I bought for a steal a few years ago, and am considering adding a GUI to my tool So I downloaded the Perl/Tk source code from CPAN and failed to compile it using gcc on Solaris However, it compiled fine on my Linux box

I tried out a few code snippets and these are my first impressions about Perl/Tk:

1 Yes, it's powerful enough to provide a cool front end to my tool
2 The coding seems pretty straight forward, though repeatative and monotonous (with lots of trial and error iterations)
3 The end user, however would need Perl/Tk installed on his/her box to able to run my Perl/Tk, which is NOT shipped by default How do I convince him/her to spend a good part of an hour to get Perl/Tk installed, so that he can use my tool!? It'd be easier for him/her to use the (ample) command line options I give, or use the rudimentary text based interface

So my question: If I spend a good part of a month writing a Perl/Tk interface, what are the chances that an average (not a programmer/admin) user/operator gets to use it?

Thanks!

Binny V A at 27 Feb, 2007 05:04
Tk interface, unfortunately, looks really outdated in linux systems However, in windows systems they look great If to want cool looking interfaces in Linux, try something like Gtk for perl

My advice is to provide the GUI along with the application - but after separating the application from the GUI The app should work without the GUI if that is how the user wants it The GUI will be a value add

Mariano at 27 Mar, 2007 05:22
If the user is currently using a Windows terminal you could generate an exe with your app in wich you can include all the used libraries (ie Tk)

If not, it means it is very easy to install the module

Anonymous at 14 Mar, 2008 08:37
Use Perl2Exe to compile it into a exe, it automatically includes the Perl/Tk binaries (If your on windows,otherwise, lul linux fails) Use serialsws to find a serialfor perl2exe

purnachander at 05 Jul, 2007 08:08
Hi,

I'm trying to add GUI to my tool So i started learning Perl/Tk module Can you tell me where I can get documentation for different values of an option

For eg, The option "-background" is used to set the background color I know how to set my background to common colors like, green, black,etc However, if I'd like to choose my own color from RGB How can I do it?

Similarly, if i need to know various values available for an option, where should i search for documentation

Binny V A at 05 Jul, 2007 11:07
Just search the net - you will get Perl/Tk Docs without much difficulty For instance
Perl/Tk Documentation

Anonymous at 30 Aug, 2007 11:58
I do not see the answer in that documentation I"m looking for the proper syntax to specify the color in RGB instead of a color name purnachander, did you ever find it?

balaji at 30 Jul, 2007 09:46
hi ,
i scripting a tool for vlsi design using perl But i dont know how to make the gui for it Can u elaborately tell me wat ar all steps to be carried out

Anonymous at 09 Aug, 2007 03:21
if i use this method for installing Tk

perl -MCPAN -e shell
cpan> install Bundle::CPAN
cpan> reload cpan
cpan> install Tk

i am finding this error
Fetching with LWP:
wwwperlorg/CPAN/authors/id/N/NI/NI-S/Tk-804027targz
LWP failed with code[500] message[read timeout]

can somebody help me pls!!!!!

Binny V A at 09 Aug, 2007 07:04
Your best bet is just to try again I think it is a temporary server issue

senthil at 04 Sep, 2007 06:10
hi,
i am senthil i need a code of perl tk i need to get a input from user using the one input like visual basic so get value to assign the one variable
like "$input= ccnb"

ccnb is get from user
senthil_v@newgenimagingcom

Binny V A at 04 Sep, 2007 09:11
The details are available in the Entry page

aflexo at 11 Dec, 2007 04:07
Great samples in tutorial! These just helped me to understand the whole thing about Perl/TK programming))

dsw at 18 Dec, 2007 08:50
This interface does not look outdated If you are looking for something more polished, then I suggest you write a tutorial about using QT Oldschool is the only school :-)

Binny V A at 18 Dec, 2007 11:08
Unfortunately, you are right - in the case of Linux Tk interface in Linux is just horrible But it looks rather good in Windows Plus - I have not looked at QT yet - I have learned a bit of Gtk - but nowhere enough to write a tutorial

PhillC at 10 Jan, 2008 07:21
First, thanks for a great tutorial using PerlTK Your style and examples work for me very well

I'm embarking on my first Perl GUI programming experience After investigating a number of options such at GTK2 and wxWindows, I think Tk appeals to me most While the PerlTk GUI look and feel may be a little outdated, I like the syntax used to create it I especially like the grid packaging system

I've been reading about the Tcl::Tk module currently available in CPAN It seems that it has some advantages over PerlTK, specifically speed and the ability to use the latest Tcl/Tk 85 widgets such as Tile It also appears to support PerlTK syntax, but of course there are some differences

I was wondering if you'd consider a tutorial highlighting the Tcl::Tk module vs PerlTk differences, and especially which widgets are included in Tcl:Tk by default and how to call external Tcl widgets In fact, maybe just a single page explaining these differences might be really useful

Regards,
Phill

Binny V A at 13 Jan, 2008 07:21
I'll look into it - but chances are that I will not be able to do this study Currently I am in Web Development - I have not been active at the Tk front So, my knowledge about those systems are a bit outdated Anyway, thanks for the suggestion - I will look into it

Anonymous at 27 Feb, 2008 04:58
assigning a variable to backgrounds and things like that
textvariable is for text, but I cannot find a similar entry for frames or anything else outside of text in a label
I get a message that says unknown color name "SCALAR(0x18a3f6c)" if I use code similar to $frame=$mw->Frame(-background=>\$variable)->pack();
but if I use $frame=$mw->Frame(-background=>$variable)->pack(); without the slash the widget is assigned to the one color permanently
this is a significant problem to me and I would appreciate someone else trying to solve this, because I have been unable to find Docs myself to assist me

Chanio at 03 Mar, 2008 05:20
I appreciate very much your great and honest work!
This complete tutorial is a very helpful tool for both: the users and the perl/Tk community

Wouldn't it suit better if your site had a wiki style?
I imagine perldocs published on-line in a wiki fashion, all full of hyperlinks

THXS! Have a nice year!
Alberto (Argentina)

Anonymous at 14 Mar, 2008 06:23
Finally figured out how to specify rgb colors for widgets Posting it here in case anybody else is looking

Put a # in front of the 6 digit rgb code and enclose it with single quotes

Ex

-background=>'#ff0000'

Kumar P at 03 Apr, 2008 06:45
I have an assignment to create a GUI, which should be able to parse the MPEG Video Bit Stream and show us various properties/semantics of the underlying video Is perl/TK the right choice for developing such application?

Jefffrey at 04 Apr, 2008 10:35
I've started writing a database and point of sale front end for my dive shop I'm using perl because it's the only language I'm somewhat fluent in I have used Tk before but not Gtk2 Can anyone tell me if there is enough info out there to get the hang of Gtk2 fairly quickly or is there a learning curve

Chanio at 10 Apr, 2008 08:04
If someone wants to get a more VB-like result with Tk you might want to use the free: Zooz GUI builder

It is not better than doing it all by coding It is just another way of complementing your knowledge of Perl-Tk And, perhaps, learning something new!

Rathnakar at 21 Apr, 2008 09:41
To get Gui you can use perl/Tk modules which would give more flexibility
I found it very useful Refer to the examples in the site to get a better picture

Anonymous at 07 May, 2008 07:47
A wonderfull tutorial that helped me a lot and I still keep for references !

My Thanks

Anonymous at 20 May, 2008 12:51
Thanks Binny,
You have done a great job on the Tk tutorial I am actually looking for help on Windows(active_state) based Tks In particular utf8 support for Tks using foreign language fonts such as Russian, Mongolian, Chinese Do you know whether Tks will support utf8?
Vanilla

Hello World

Anonymous at 30 Jan, 2007 05:34

couldn't connect to display ":0" at /opt/perl/lib/site_perl/582/PA-RISC11-thr
ead-multi/Tk/MainWindowpm line 55
MainWindow->new() at welcomepl line 5

Binny V A at 31 Jan, 2007 04:31
I have never seen this error - but do you have X server running? Tk needs a GUI display

Anonymous at 17 Apr, 2007 12:16
you need to set your display environment variable, and then also if you are telneting and have further problems 'xhost +' from the originating machine

fred at 28 Apr, 2007 04:54
Hello,

When I run the hello example on my winxp box
it opens a command prompt It stays open until

I close the example

Is this the way all PTK programs work?

Thanks

Binny V A at 28 Apr, 2007 05:38
Yes - thats how they work Don't worry about it

Jayson at 19 Jun, 2007 03:52
ldso1: /usr/local/bin/perl: fatal:libctso: open failed: No such file or directory
Please help

Binny V A at 19 Jun, 2007 09:31
There is some issue with your perl installation - some libraries are missing Try downloading perl and installing it

AMP at 03 Aug, 2007 08:56
Probably not a problem w/ installation, Perl is most likely in /usr/bin/perl instead All the Linux distributions I have worked on have it in this location

kittu at 21 Jun, 2007 08:57
HI
Friends
i have some issues pls help me on this actually cgi/peral with NT 40 environment is moving by Linux any disadvantages and suggest if that is re-engineering with java , linux and give me some points and presently existing system having 5003 want to change to 583 so any deprcciation i did not find any problems pls help on that and with linux how to deploy cgi/pearl programs

Jonathan at 03 Aug, 2007 03:07
Kittu, Congratulations on your post I have never read anything more completely incomprehensible pls kepp up the good werk u ntrtane me

Anonymous at 21 Sep, 2007 10:59
Thanks for the tutorial Binny It's very good!

Anonymous at 24 Nov, 2007 08:49
Very clear and understandable Nice tutorial Thank You to the author

OddChild at 09 May, 2008 08:29
thank you for putting this together, it was most helpful

Anonymous at 21 May, 2008 07:00
this line is very slow:

$mw->messageBox(-message=>"Goodbye");

Is that supposed to be? Or just my environment issue?

Thanks

Anonymous at 21 May, 2008 07:02
Message box line is very slow Is that supposed to be? Or it is my environment issue?
Thanks

Sid at 21 May, 2008 09:01
New to pTk, pulling perl out of cold storage Your tutorial is helpful Thank you

Widget 1

Anonymous at 31 Jul, 2007 02:49
The example above with the function push_button is not showing the text "whatever" Also if i use the command use strict,how do i write the bare text whatever

Binny V A at 01 Aug, 2007 06:19
No - you misunderstood me That 'whatever' is not the output - I meant you can put the code you want in its place On hindsight, I think I should have commented it

Anonymous at 31 Jul, 2007 04:31
I think the function body(sub) should be written inside MainLoop itself I had posted the same question and I think the answer is this description

Jack at 01 Aug, 2007 03:17
Is there a command to toggle the visibility of a widget? preferably label and entry widgets

Binny V A at 01 Aug, 2007 06:28
There is the destroy() function that will delete a widget

```
my $lab = $mw -> Label(-text=>"Enter name:") -> pack();
$lab->destroy();
```

ANGIE at 08 Sep, 2007 04:52
YOUR CODE IS SO GREAT YOU SHOULD START A PAID CERTIFICATION COURSE
YOU ROCK , MAN!

Anonymous at 21 Sep, 2007 11:38
As they say in Shakespeare's time, "You da Man Home-Slice"! Thanks for this!

Anonymous at 21 Sep, 2007 11:43
Do you have an example of a button that the "-state" changes from "disabled" to "normal" after some event?
The following didn't work for me

```
my $Enable_button="disable";


$mw->-> Button(-text => "Push Me",
-command =>\&push_button);
-state => $Enable_button,
or
-state => \$Enable_button,
$but -> pack();

then, later
$Enable_button = "normal";
```

Binny V A at 22 Sep, 2007 01:15
The right way to do this is

```
my $button = $wm-> Button(-text => "Push Me", -command =>\&push_button, -state=>"normal");
```

Later

$button->configure(-state=>'disabled');

Paul R at 24 Sep, 2007 07:11
Binny VA
That's it Thanks for your help

SibuNL at 11 Mar, 2008 12:59
Binny,

When I run a perl/TK simple hello world application by double clicking on it
under windows xp,
a command window also opened along with the hello world application window
How to avoid or hide the command window from appearing on screen

Thanks in advance,

SibuNL

Binny V A at 11 Mar, 2008 03:53
I don't think you can do anything about that

Kerbiquet at 18 Mar, 2008 04:07
Hi,

Check this example, taken from the Perl Cookbook edited by O'Reilly: wwworeillycom/catalog/perlckbk2/tochtml

The windows console pops up by diseappear immediatly

Kerbiquet at 18 Mar, 2008 06:33

You have written a Perl program for the Windows port of Perl and Tk, but you get a DOS shell window every time you start your program
Add this to the start of your program:

```
BEGIN {
if ($^O eq 'MSWin32') {
require Win32::Console;
Win32::Console::Free( );
}
}
```

The Win32::Console module lets you control the terminal window that launched your program All you need to do is close that window (or Free it in, in the peculiar parlance of the Windows API) and voila no pesky DOS shell window
The documentation for the Win32::Console module, which is included with distributions of Perl destined for Microsoft systems

Anonymous at 20 Mar, 2008 09:21
Entry() takes a 1-line inputwat to do if i want t to capture a text area as input & manipulaate it?

Binny V A at 21 Mar, 2008 03:59
Its in the next page - Text widget

Anonymous at 06 May, 2008 01:29
can you please tell me how can i delete or make the scrolledlistbox invisible in the main window?
I am developing 1 application in TK(VTcl)

Anonymous at 09 May, 2008 02:37
Hi Binny,
I have a sub mainp(), which runs in loop and may take less than a second or few minutes sometimes, I want to exit from the program before mainp() finishes I have the following code could you please suggest me how to achieve this?

```
my $print = $right1->Button(-text => 'Build Index',
-command => \&mainp)->
grid(qw/-row 7 -column 1 -sticky se/);
my $exit = $right1->Button(-text => ' Exit ',
-command => [$mw => 'destroy'])->
grid(qw/-row 7 -column 2 -sticky sw/);
```

Anonymous at 19 May, 2008 01:02
Hi,

I am trying to create and delete widgets label & entry dynamically I am able to create widgets but unable to delete the specified range of widgets I am using Grid manager Could you please help me out for the same?

Anonymous at 12 Jun, 2008 09:11
If you execute the perl script with wperlexe instead of perl, no command console will be displayed If you edit the actions for the pl type you can make a Run with wperl

Widget 2

Anonymous at 01 Apr, 2007 03:25
In the text area in the example above the scroll bar on the x axis doesn't ever get used because the text always starts a new line instead of going off the side What can i do about this?

Binny V A at 01 Apr, 2007 09:27
This is because the Text widget defaults to wrapping To disable this behaviour, just use this code

```
my $txt = $textarea -> Text(-width=>40, -height=>10,-wrap=>'none');
```

See "`-wrap=>'none'`" - that will disable the wrap The other options for wrap are '`word`' and '`char`'

Anonymous at 01 Apr, 2007 11:36
thanks!

Anonymous at 01 Apr, 2007 11:36
thanks!

Anonymous at 01 Apr, 2007 01:45
When i tried this it didn't work and said:

```
Tk::Error: Odd number of args to Tk::Text->new()
Tk callback for
Tk callback for frame
Tk callback for frame1
Tk::Widget::new at /usr/lib/perl5/vendor_perl/588/i586-linux-thread-multi/Tk/Widgetpm line 164
Tk::Widget::__ANON__ at /usr/lib/perl5/vendor_perl/588/i586-linux-thread-multi/Tk/Widgetpm line 256
Odd number of args to Tk::Text->new()
at (pathname)/scrollbar_and_grid_tut line 14
```

what does it mean and how can i fix it?

Binny V A at 02 Apr, 2007 06:02
Please provide the exact code you used Also, could you email it to me? That would be better than using this comment system My email is binnyva, gmail

Anonymous at 02 Apr, 2007 07:49
Hi,
Could you please tell how to position the "text messge" in the center of the window as well as position the "Quit" button Thanks

Binny V A at 02 Apr, 2007 11:44
I am not sure I understand your question Could you just create a small image(using MS Paint or something) on how you want the layout and send it to my? That way I can understand what you need My email is binnyva, gmail

Anonymous at 04 Apr, 2007 05:42
Hi,
Is there any way of centrally aligning text inside a text wigit?

Binny V A at 04 Apr, 2007 06:28
Try this code
```
my $lab = $mw -> Label(-text=>"Hello",-anchor=>"center") -> pack();
```

Anonymous at 06 Apr, 2007 09:47
Sorry, i meant in a text wigit, not a label wigit
I wanted centrally aligned text spanning more that one line with a different relief to a label
please help!

Anonymous at 04 Apr, 2007 06:31
Hi,
Is there any way of centrally aligning text inside a text wigit?

Anonymous at 04 Apr, 2007 06:32
soz

Anonymous at 04 Apr, 2007 06:43
how do you make a text wigit readonly?

Amit at 05 Apr, 2007 09:28
Is there any way to lock the Text Field So tat only contents are displayed none can edit ???

Binny V A at 05 Apr, 2007 10:17
Try using `-state=>'disabled'` If you have other doubts like this, take a look at the Tk reference manual Even if the manual is for Tcl/Tk, it can be used for Perl/Tk as well

Tk Manual

Anonymous at 06 Apr, 2007 07:58
If the state is disabled is it still possible to edit the contents from within the program?

Binny V A at 08 Apr, 2007 05:52
No - it is not possible

Venugopal at 04 May, 2007 04:46
Hi,

I have a requirement to list the contents of a file in a GUI, with the options to make a selection(one or more) from the list and store the selected items to another variable or another file I have used list,scroll bar and could list the contents

of the file But how could I add the selection feature? Here is my code Could you please help me how could I achieve this

```
$list = `type $file`;
my $mw = new MainWindow; # Main Window
$mw->title("Output");
$mw->resizable(0,0);
my $textarea = $mw -> Frame(); #Creating Another Frame
my $txt = $textarea -> Text();

my $srl_y = $textarea -> Scrollbar(-orient=>'v',-command=>[yview => $txt]);
my $srl_x = $textarea -> Scrollbar(-orient=>'h',-command=>[xview => $txt]);
$txt -> configure(-yscrollcommand=>['set', $srl_y],
-xscrollcommand=>['set',$srl_x]);

$txt -> grid(-row=>1,-column=>1);
$srl_y -> grid(-row=>1,-column=>2,-sticky=>"ns");
$srl_x -> grid(-row=>2,-column=>1,-sticky=>"ew");
$textarea -> grid(-row=>5,-column=>1,-columnspan=>2);
$txt -> insert('end',"$list");

MainLoop;
```

Thanks in Advance

Anonymous at 10 Jul, 2007 09:11
Hi,

I'm trying to set scrollbars for Mainwindow However, I'm failing to do it Can you please hint me how to go through this?

Tcl/Tk programmer at 06 Oct, 2007 05:20
How do you automatically send text data to a text widget? For example, tail -f /var/log/messages I'm working on a real-time status script for the job Many thanks in advance, Binny?

Anonymous at 14 Apr, 2008 10:08
This is the most excellent tutorial Is there any way I can get a copy of this to use locally, off line? Thank you

Anonymous at 11 Jun, 2008 02:43
Fantastic tutorial, I used your Tcl/Tk one as well and found it very helpful I do think though that scrollbars are the most difficult section to understand for a beginner, perhaps you could spend a bit more time on them? Just a thought:)

Widget 5

Anonymous at 23 Apr, 2007 09:54
what is a tear off entry?

Anonymous at 17 Jul, 2007 05:36
It's a part of a menu where you can click a little button, and then, you can click on anything in the little menu while clicking something else in the big menu

Greg at 23 May, 2007 12:28
Do you know how to change the background color for the separator?

I've tried using the following code but it has no affect:
```
$file->separator( -background=> 'white' );
```

I would be very thankful for any help you could give!

ELJONTO at 19 Jun, 2007 12:10
Can you use the menu that you have used in the above example and change its background and foreground colours? Or do i have to make my own menu?

jack at 31 Jul, 2007 11:37
how do you create an option that has more than one word?

Anonymous at 30 Apr, 2008 08:11
my $curlist = $mw -> Optionmenu(-options => ["first option", "second option"], -variable=> \$var) -> pack();

Dont use the qw function

Widget 6

Anonymous at 14 Feb, 2008 11:10
When I first read this page, I thought you were dismissing Canvas widgets as impotent

But further reading suggests you meant "important", as they certainly seem to have some power Which is it?

Stuart Doty

Binny V A at 16 Feb, 2008 12:48
My bad - its fixed now

Anonymous at 27 May, 2008 11:39
hi,
i want to make a real time gui which updates itself but adding elements to the canvas or deleting them
Can you be more illustrative using Canvas and how can i integrate it with an environment which updates itself periodically

Thanks

Geometry Management

Is there any way to 'ungrid' something?
When I click a radio button I want one frame to be replaced with another preprepared frame
Thanks

dec at 10 Sep, 2007 08:04
$widget->gridForget;

Anonymous at 06 Jul, 2007 11:24
When I run this Code on my Solaris Box which has 588 perl Insatlled I get a CoreDUMP
Any Reason Why?

#!/usr/local/bin/perl
use Tk;

```
use strict;
# Main Window
my $mw = new MainWindow;
my $response = $mw -> messageBox(-message=>"Really quit?",
-type=>'yesno',-icon=>'question');

if( $response eq "yes" ) {
exit
}
else {
$mw -> messageBox(-type=>"ok",
-message=>"I know you like this application!");
}
MainLoop;
```

The Stack trace is given below:

```
$pstack core
core 'core' of 28384: /usr/local/bin/perl /temp
ff1428c0 t_delete (43261c, ff1bc008, ff1c27cc, ff1c284c, ff1c2848, 0) + 68
ff141f50 _malloc_unlocked (68, 43261c, ff1bc008, 68, 43261c, 0) + 18c
ff141da8 malloc (68, 0, 20, 8, 215e4, fee9b938) + 20
feea624c _XEnq (0, ffbeebc4, 14e2c8, ffbeeb5c, 14e2c8, 20) + 28
fee9ccdc _XReply (14e2c8, 39, 0, 1, 39, 14e2c8) + 498
feea7088 XInternAtom (0, ff04b2fc, 0, 14e2c8, a5c46890, 40) + bc
fefd7474 Tk_InternAtom (38a4f0, ff04b2fc, fefcac28, ff031240, 1a8e0, 1) + 4c
ff0312b8 UpdateWmProtocols (3f67b0, 2c00, 153a, 1400, ff05c30c, 38a5e8) + e8
ff02addc TkWmMapWindow (38a4f0, ff067160, 0, 38a5e8, ff05c30c, 0) + 194
ff03703c Tk_MapWindow (38a4f0, 0, 12634, ff1fc910, ff20eda0, 38002) + 3c
fefed014 MapFrame (38a6d8, 32c8, 6f354, ff1f44f4, ff05c30c, 3000) + 68
ff1fde8c TclServiceIdle (0, ff1fed8c, 0, 1, 377d30, 388828) + 98
ff1fc910 Tcl_DoOneEvent (22, 0, 12634, ff3cdf18, ff20eda0, 0) + 1b0
fefecfe4 MapFrame (370770, 32c8, 6f354, ff1fd5b0, ff05c30c, 3000) + 38
ff1fde8c TclServiceIdle (0, ff1fed8c, 0, 0, 377d30, 36ab80) + 98
ff1fc910 Tcl_DoOneEvent (22, 0, 12634, fefb451c, ff20eda0, 0) + 1b0
fefe261c Tk_UpdateObjCmd (2e9408, 22a8f4, 2, ff07532c, 20, 32c8) + 90
fefba8b0 Call_Tk (13984c, 2, 226514, fefe258c, 1397a8, 1396f0) + 314
fefbd294 XSTkCommand (2585a8, 1, fefe258c, 2, 226514, 0) + 158
fefbd334 XStoTclCmd (3400, 32fc, 4, 226510, ff05c30c, 1398a0) + 90
00096a18 Perl_pp_entersub (1, 80, 2585a8, 139400, 226518, 247918) + 664
0008cfac Perl_runops_standard (24e230, 136c00, 132c00, 0, 0, 8d044) + 4c
00028dd8 S_run_body (0, ffbeef40, 8, 139800, 139800, 28ed58) + 164
00028acc perl_run (139a60, 139400, 139400, 139e90, 139e90, 137000) + 240
00024f6c main (2, ffbef4fc, 132c00, 139800, 136c00, 139800) + a4
```

The Tk Version installed is 804027

Niraj at 01 Sep, 2007 02:47

how to run a perl script in GUI

I have a code for transcribing a DNA into a RNA written in perl script but how should i get its output in GUI layout? please reply soon

Anonymous at 09 Dec, 2007 09:03

My doubt is about how to implement reset in Perl tkI have to refresh the values stored from the entry by using get() and again take the new values so can i use the same window for that and use refresh button??Reply me

Now What?

Anonymous at 15 Jan, 2007 01:08
Hi,

I am just starting to learn Perl/Perl-TK so I googled this topic
and found your tutorial web site on this subject I am just finished
the tutorial and found to be very helpful for a newbie like me
Thank your for providing this tutorial and for sharing your expertise
Also I will follow your suggestion "There is no better way of learning a subject than by doing it"

Thanks,

Ethan

Anonymous at 08 Feb, 2007 04:15
And the humour keeps one going through it!
Thanks for taking the time to put it together!

gobble at 03 Apr, 2007 03:17
Nice tutorial Very helpful

Giulio at 23 Apr, 2007 03:06
Very good job!

Thank a lot

Giulio

If it's possible fit with more examples ! ;-D

Aryan at 29 May, 2007 12:20
Great Work:)

Regards
Aryan

Anonymous at 29 May, 2007 01:27
Good tutorial, i have learned lots about TK

Anonymous at 07 Jun, 2007 06:01
Thank you

I really needed this

Anonymous at 10 Jul, 2007 10:58
really a usefull e-learning for a newbie in perl/tk
hats off for keep up the good work going

thanks a lot for the tutorial!!!!!!!!

Anonymous at 13 Jul, 2007 02:59
Thank you very much for putting this tutorial together It was extermely helpful to me

Jonathan at 01 Aug, 2007 01:59
Thanks for publishing this tutorial It has been a great help; your karma explodes

Anonymous at 15 Aug, 2007 06:07
Thank you very much It was really helpful

RajasekarV at 28 Sep, 2007 01:39
I found this site very useful Its great a platfarm for beginners to startwith
more examples could make in more better Hats off for great work

Regards,
Rajasekar V

Raja at 28 Sep, 2007 01:41
Hats off for your great work

JZ at 30 Sep, 2007 05:47
bin-co you're the greatest with you expertise in both TCL/Tk and Perl/Tk And your tutorial is exemplary work! You really know how to convey complex concepts clearly to the reader Many, many kudos to you!

Binny V A at 30 Sep, 2007 07:06
Thanks everyone

Daniel Wilson at 06 Oct, 2007 10:59
Great tutorial - helped me out loads :)

LeRoi at 10 Jan, 2008 04:53
Very nice tutorial Thanks!!!!

Anonymous at 03 Feb, 2008 09:15
Hi Binny,

I am a perl/Tk newbie Your site was a great help Thanks Can you point out to me as to how my application can accept multiline formatted textual input from the user? (to save it to an external file) The application has to to wait on the user input and then proceed I just cant seem to get my head around it

Thanks a lot again for the tips here :-)

Binny V A at 04 Feb, 2008 08:24
Try using the Text widget

Anonymous at 14 Feb, 2008 11:45
Very useful tutorial fo newbie
Helped me get into a fast track programming!

Anonymous at 04 Mar, 2008 06:13
Very helpful ! thanks

Anonymous at 12 Apr, 2008 04:01
This is a great tutorial I'm really happy that I found this

Thanks!

Anonymous at 22 May, 2008 09:50
Excellent perl/tk tutorial I loved it5 start award to you

Bob at 11 Jun, 2008 03:14
Hi !

This is a superb tutorial on perl/tk
Many thanks for taking the time to put it together

Best regards
Bob

Appendix

Anonymous at 19 Apr, 2007 05:48
This is an excellent tutorial It gives you all the information you need to start writing GUI programs, and gives you information and were to go and to get more

Laurence at 12 Jul, 2007 12:32
Thank you so much for this tutorial, it has helped me loads
Keep up the good work

Pankaj at 09 Aug, 2007 12:12
Quite informative Got all the information i needed as a beginner
Thanks

Maria J at 16 Aug, 2007 06:24
I find this tutorial very, very helpful Thanks a lot for your clear, well structured explanation and great examples

Mark at 24 Aug, 2007 11:54
I agree excellent starting point, short and to the point Much better than the bask in my own glory types that frequent a lot of the linux post sites

Doug at 25 Sep, 2007 12:28
I agree with the others, this is an Excellent tutorial! Thanks for setting it up It helped me to quickly get back into shape working with tk after a break of 10 years I just wish the authors of the published books followed your writing style of clear examples!

Keep up the fantastic work!

Anonymous at 29 Nov, 2007 10:06
This tutorial was really of great help to me as i was a begginner If we have two labels how can the second label be printed in the next line by using pack geometry manager(i used framalso)I could not get that option By using grid i was able to do thisHope you will reply me as soon as possibleThank you

Binny V A at 30 Nov, 2007 02:00

```
pack first_label

pack second_label
```

That should do it

Avidan Zhang at 22 Jan, 2008 12:28
Very helpful tutorial! Easy for new comers to make first step in TK under PERL

Thank you very much!