

for Perl/Tk 402.003 - Perl 5.004 / Tk 402.003

Perl program designed and created by
Larry Wall <larry@wall.org>

Tk designed and created by
John Ousterhout <John.Ousterhout@Eng.Sun.COM>

Perl/Tk designed and created by
Nick Ing-Simmons <nick@ni-s.u-net.com>

Reference guide format designed and created by
Johan Vromans <jvromans@squirrel.nl>

Reference guide contents written by
Paul Raines <raines@slac.stanford.edu>
Jeff Tranter <Jeff_Tranter@Mitel.COM>
Steve Lidie <Stephen.O.Lidie@Lehigh.EDU>

Contents

1. General Perl/Tk Widget Information	2
2. Perl/Tk Special Variables	5
3. Widget Scroll Commands	5
4. The Canvas Widget	6
5. The Entry Widget	11
6. The Listbox Widget	12
7. The Menu Widget	13
8. The Text Widget	15
9. Other Standard Widgets	18
10. Perl/Tk Widgets	23
11. Images	32
12. Window Information	34
13. The Window Manager	36
14. Geometry Management	38
15. Bindings	39
16. Other Perl/Tk Commands	40

Conventions

fixed denotes literal text.
this means variable text, i.e. things you must fill in.
word is a keyword, i.e. a word with a special meaning.
 ?...? denotes an optional part.

1. General Perl/Tk Widget Information

All Perl/Tk programs must have a **use Tk** statement. To use special Perl/Tk widgets like **Dialog** a **use Tk::Dialog** statement is required.

All widgets are created with

```
$widget = $parent->widgetClass(?-option1 => value1, ...?);
```

where *widgetClass* is the name of the class of widget desired (eg. **Button**) and *parent* is a Perl/Tk widget reference of the new widget's parent. The Perl object reference is stored in `$widget`, which becomes a child of `$parent`, creating the widget hierarchy.

All widget creation commands can have the optional *Name => resourceName* parameter to associate a resource database name with the widget.

Every Perl/Tk program requires a *main window*, the topmost widget in the hierarchy, created with

```
$mw = MainWindow->new;
```

The following command creates a new button widget `$b` and uses the **grid** geometry manager to map it:

```
$b = $mw->Button(-text => "Hello World")->grid;
```

Widget configuration options may be passed in the creation method. Options begin with a "-" and are always followed by a value, usually an integer or string, sometimes a Perl scalar, array, hash or code reference. After creation, options may be changed using the **configure** widget command

```
$widget->configure(-option1 => value1, ...);
```

and queried using the **cget** command

```
$widget->cget(-option);
```

The last statement in a Perl/Tk program calls **MainLoop** to initiate event processing.

Perl/Tk Callbacks

A *callback* is a scalar, either a code reference or a method name as a string. Either of these styles can take parameters by passing an array reference, with the first element the code reference or method name, and subsequent elements subroutine parameters.

```
\&subroutine        [\&subroutine ?, args?]
```

```
sub {...}            [sub {...} ?, args?]
```

```
'methodName'      ['methodName' ?, args?]
```

Note that **bind** *callbacks* are implicitly passed the bound widget reference as the first argument of the parameter list. Refer to the section *Bindings* for related information.

Notes

`$widget->selectionGet(?-selection => selection? ?, -type => type?);`
Retrieves *selection* from `$widget`'s display using representation *type*.

`$widget->selectionHandle(?-selection => sel? ?, -type => type? ?, -format => fmt? ,win => callback);`
Arranges for *callback* to be run whenever *sel* of *type* is owned by *win*.

`$widget->selectionOwner(?-selection => selection?);`
Returns path name of `$widget` which owns *selection* on `$widget`'s display.

`$widget->selectionOwn(?-selection => selection? ?, -command => callback?);`
Causes `$widget` to become new owner of *selection* and arranges for *command* to be run when `$widget` later loses the *selection*.

`$widget->send(?-async? interp => callback);`
Execute *callback* in the Tk application *interp* on `$widget`'s display. If `-async` is specified, the `$widget->send` command will return immediately.

To receive commands from a foreign application define a subroutine `Tk::Receive($widget, commandString)`. Run with taint checks on and untaint the received data.

`$widget->setPalette(color);`
Set the default background color and compute other default colors.

`$widget->setPalette(name => color ?, name => color ...?);`
Set the default color for the named color options explicitly.

`$widget->waitVariable(varRef);`
Pause program until global variable *varRef* is modified.

`$widget->waitVisibility;`
Pause program until `$widget`'s visibility has changed.

`$widget->waitWindow;`
Pause program until `$widget` is destroyed.

Common Widget Options

Some of the widget options common to several widgets are described here for brevity.

`-activebackground => color`
Background color of widget when it is active.

`-activeborderwidth => width`
Width in screen units of widget border when it is active.

`-activeforeground => color`
Foreground color of widget when it is active.

`-anchor => anchorPos`
How information is positioned inside widget. Valid *anchorPos* values are **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, and **center**.

`-background => color`
Background color of widget in normal state (Abbrev: **-bg**).

`-bitmap => bitmap`
Bitmap to display in the widget (**error**, **gray12**, **gray50**, **gray25**, **hourglass**, **info**, **questhead**, **question**, **warning**, **@pathName**).

`-borderwidth => width`
Width in screen units of widget border in normal state (Abbrev: **-bd**).

`-class => className`
Assign a new *className* to a widget.

`-command => callback`
A Perl/Tk callback describing the Perl code to run when widget is invoked.

`-cursor => [bitmap, mask, foreground, background]`
An array reference describing the cursor to display when mouse pointer is in widget.

`-disabledforeground => color`
Foreground color of widget when it is disabled.

`-exportselection => boolean`
Whether or not a selection in the widget should also be the X selection.

`-font => font`
Font to use when drawing text inside the widget.

`-foreground => color`
Foreground color of widget in normal state (Abbrev: **-fg**).

`-geometry => widthxheight`
Geometry for the widget's window. The units for *width* and *height* depend on the particular widget (usually in characters when widget has text).

`-height => height | textChars`
Height of widget. Units depend on widget.

`-highlightbackground => color`
Color of the rectangle drawn around the widget when it does not have the input focus.

`-highlightcolor => color`
Color of the rectangle drawn around the widget when it has the input focus.

`-highlightthickness => width`
Width in screen units of highlight rectangle drawn around widget when it has the input focus.

-image => image
Image to display in the widget (see Images).

-insertbackground => color
Color to use as background in the area covered by the insertion cursor.

-insertborderwidth => width
Width in screen units of border to draw around the insertion cursor.

-insertofftime => milliseconds
Time the insertion cursor should remain “off” in each blink cycle.

-insertontime => milliseconds
Time the insertion cursor should remain “on” in each blink cycle.

-insertwidth => width
Width in screen units of the insertion cursor.

-jump => boolean
Whether to notify scrollbars and scales connected to the widget to delay updates until mouse button is released.

-justify => left | center | right
How multiple lines line up with each other.

-menuitems => [[type, label ?, -option => value?]]
A list of list of menuitem *types*, like **Button**, with the text *label* and optional parameters. *menuitems* can be nested.

-orient => horizontal | vertical
Which orientation widget should use in layout.

-padx => width
Extra external space in screen units to request for the widget in X-direction. Use *-ipadx* the request additional internal horizontal space.

-pady => height
Extra external space in screen units to request for the widget in Y-direction. Use *-ipady* to request additional internal vertical space.

-relief => flat | groove | raised | ridge | sunken
3-D effect desired for the widget’s border.

-repeatdelay => milliseconds
Time a button or key must be held down before it begins to auto-repeat.

-repeatinterval => milliseconds
Time between auto-repeats once action has begun.

-selectbackground => color
Background color to use when displaying selected items.

-selectborderwidth => width
Width in screen units of border to draw around selected items.

-selectforeground => color
Foreground color to use when displaying selected items.

-setgrid => boolean
Whether this widget controls the resizing grid for its toplevel window.

-size => size
A size in screen units. Values are in pixels unless an optional one letter suffix modifier is present — **c** (cm), **i** (inch), **m** (mm), or **p** (points).

-state => normal | disabled (| active for button-type widgets)
Current state of widget.

-takefocus => focusType
If 0 or 1, signals that the widget should never or always take the focus. If

\$widget->clipboardClear;
Claim ownership of clipboard on *\$widget*’s display, clearing its contents.

\$widget->destroy;
Destroy the given window and its descendents.

Exists(\$widget);
Returns 1 if there exists a window for *\$widget*, ” (false) if no such window exists.

\$widget->focus;
Set focus window to *\$widget*.

\$widget->focusCurrent;
Returns focus window on *\$widget*’s display.

\$widget->focusFollowsMouse;
Change focus model of application so focus follows the mouse pointer.

\$widget->focusForce;
Sets the input focus for *\$widget*’s display to *\$widget* even if another application has it.

\$widget->focusLast;
Returns the window which most recently had focus and is a descendent of *\$widget*’s toplevel.

\$widget->focusNext;
Returns the next window after *\$widget* in focus order.

\$widget->focusPrev;
Returns the previous window before *\$widget* in focus order.

\$widget->grabCurrent;
Returns name of current grab window on *\$widget*’s display. If *\$widget* is omitted, returns list of all windows grabbed by the application.

\$widget->grabRelease;
Releases grab on *\$widget*.

\$widget->grab;
Sets a local grab on *\$widget*.

\$widget->grabGlobal;
Sets a global grab.

\$widget->grabStatus;
Returns **none**, **local**, or **global** to describe grab state of *\$widget*.

\$widget->lower(?belowThis?);
Places *\$widget* below window *belowThis* in stacking order.

\$widget->optionAdd(pattern => value ?, priority?);
Adds option with *pattern value* at *priority* (0-100) to database.

\$widget->optionClear
Clears option database and reloads from user’s Xdefaults.

\$widget->optionGet(name, class);
Obtains option value for *\$widget* under *name* and *class* if present.

\$widget->optionReadfile(pathName ?, priority?);
Reads options from Xdefaults-style file into option database at *priority*.

\$widget->Popup(menu, x, y ?, entry?);
Post popup *menu* so that *entry* is positioned at root coords *x y*.

\$widget->raise(?aboveThis?);
Places *\$widget* above window *aboveThis* in stacking order.

\$widget->selectionClear(?-selection => selection?);
Clears *selection* (default **PRIMARY**) on *\$widget*’s display.

Configure	KeyPress, Key	Unmap
Destroy	KeyRelease	Visibility
Enter	Motion	

Details: for buttons, a number 1-5
for keys, a keysym (/usr/include/X11/keysymdef.h)

Tags: widget instance (applies to just that window)
toplevel window (applies to all its internal windows)
class name (applies to all widgets in class)
'all' (applies to all windows)

Codes: pass to XEvent structure (see below)

%	single % sign	w	width field
#	last event's serial field	x	x field
a	above field	y	y field
b	button number	A	ASCII character
c	count field	B	border_width field
d	detail field	E	send_event field
f	focus field	K	keysym as text
h	height field	N	keysym as decimal
k	keycode field	R	root window
m	mode field	S	sub_window identifier
o	override_redirect field	T	type field
p	place field	W	window pathname
s	state field	X	x_root field
t	time field	Y	y_root field

\$widget->break;

Exit *callback* and shortcircuit **bindtags** search.

\$eventStructure = \$widget->XEvent;

Fetch the X11 event structure, then invoke the `$eventStructure` method and pass one of the preceding *Codes* to fetch the corresponding X-Event field information for `$widget`.

\$eventStructure = \$Tk::event;

A binding can localize the X11 event structure with `$Tk::event` rather than calling **XEvent**.

\$eventField = Ev(Code);

Binding callbacks can be nested using the `Ev(...)` "constructor". `Ev(...)` inserts callback objects into the argument list. When Perl/Tk prepares the argument list for the callback it is about to call it spots these special objects and recursively applies the callback process to them.

16. Other Perl/Tk Commands

\$widget->appname(?newName?);

Set the interpreter name of the application to *newName*.

\$widget->bell;

Ring the X bell on `$widget`'s display.

\$widget->bisque;

Set default color palette to old bisque scheme.

\$widget->clipboardAppend(?-format =>fmt? ?, -type =>type?, data);

Append *data* to clipboard on `widget`'s display.

empty, Tk decides. Otherwise, evaluates argument as script with widget name appended as argument. Returned value must be 0, 1 or empty.

-text => string

Text to be displayed inside the widget.

-textvariable => varRef

A reference to a Perl scalar variable which contains a text string to be displayed inside the widget, or which is modified by the widget.

-troughcolor => color

Color to use for the rectangular trough areas in widget.

-underline => index

Integer index of a character to underline in the widget.

-width => width | textChars

Width of widget. Units depend on widget.

-wraplength => length

Maximum line length in screen units for word-wrapping.

-xscrollcommand => cmdPrefix

Prefix for a command used to communicate with horizontal scrollbars.

-yscrollcommand => cmdPrefix

Prefix for a command used to communicate with vertical scrollbars.

2. Perl/Tk Special Variables

\$Tk::library

Directory containing library of Tk modules, widgets and scripts.

\$Tk::patchLevel

Integer specifying current patch level for Tcl/Tk.

\$Tk::strictMotif

When non-zero, Tk tries to adhere to Motif look-and-feel as closely as possible.

\$Tk::version

Current version of Tcl/Tk that Perl/Tk is based on, in *major.minor* form.

\$Tk::VERSION

Current version of Perl/Tk.

3. Widget Scroll Commands

The Canvas, Listbox and Text widgets support the following scrolling commands. The Entry widget supports the **xview** command and the **scan** command with the y coordinate dropped.

Refer to the section *Perl/Tk Widgets* and learn how Perl/Tk greatly simplifies managing scrollbars.

\$widget->scanMark(x, y);

Records *x* and *y* as widget's current view anchor.

\$widget->scanDragto(x, y);

Shift the view by 10 times the difference between the coordinates *x* and *y* and the current view anchor coordinates.

\$widget->xview;

Return a two element list specifying the fraction of the horizontal span of the widget at the left and right edges of the window.

`$widget->xviewMoveto(fraction);`

Adjust the view in the window so that *fraction* of the total width of the widget is off-screen to the left.

`$widget->xviewScroll(number => units | pages);`

Shift the view by *number* one-tenth's (**unit**) or nine-tenth's (**pages**) the window's width in the horizontal direction.

`$widget->yview;`

Return a two element list specifying the fraction of the vertical span of the widget at the top and bottom edges of the window.

`$widget->yviewMoveto(fraction);`

Adjust the view in the window so that *fraction* of the total height of the widget is off-screen to the top.

`$widget->yviewScroll(number => units | pages);`

Shift the view by *number* one-tenth's (**unit**) or nine-tenth's (**pages**) the window's height in the vertical direction.

The Text Widget also supports the following:

`$text->yview(?-pickplace,? index);`

Changes view of widget's window to make character at *index* visible. If **-pickplace** is specified, *index* will appear at the top of the window.

The Entry (**xview** only) and Listbox widget also supports the following:

`$listbox->xview(index);`

Adjusts view so that character position *index* is at left edge.

`$listbox->yview(index);`

Adjusts view so that element at *index* is at top of window.

4. The Canvas Widget

Canvas Options

-background	-insertbackground	-selectborderwidth
-borderwidth	-insertborderwidth	-selectforeground
-cursor	-insertofftime	-takefocus
-height	-insertontime	-width
-highlightbackground	-insertwidth	-xscrollcommand
-highlightcolor	-relief	-yscrollcommand
-highlightthickness	-selectbackground	

-closeenough => float

How close the mouse cursor must be to an item before it is considered to be "inside" the item.

-confine => boolean

Whether it is allowable to set the canvas's view outside the scroll region.

-scrollregion => [corners]

List reference of four coordinates describing the left, top, right, and bottom of a rectangular scrolling region.

-xscrollincrement => distance

Specifies the increment for horizontal scrolling in screen units.

-yscrollincrement => distance

Specifies the increment for vertical scrolling in screen units.

\$slave->gridForget;

Removes (and unmaps) *\$slave* from grid of its master.

\$slave->gridInfo;

Returns list describing configuration state of *\$slave*.

\$master->gridLocation(x, y);

Returns column and row containing screen units *x y* in *\$master*. If *x, y* is outside grid, -1 is returned.

\$master->gridPropagate(?boolean?);

Set/get whether *\$master* tries to resize its ancestor windows to fit grid.

\$master->gridRowconfigure(row ?, -minsize => size? ?, -weight => int?);

Set/get minimum row size and relative row weight.

\$master->gridSize;

Returns size of grid (in columns then rows) for *\$master*.

\$master->gridSlaves(?-row => row? ?, -column => column?);

With no options, a list of all slaves in *\$master* is returned. Otherwise, returns a list of slaves in specified row and/or column.

15. Bindings

Note that **bind callbacks** are implicitly passed the bound widget reference as the first argument of the parameter list.

\$widget->bind;

Returns list of all event descriptors for which a binding exists for *\$widget*.

\$widget->bind(tag);

Returns list of all event descriptors for which a binding exists for *tag*.

\$widget->bind(<modifier-modifier-type-detail>);

Returns the callback bound to the given event descriptor for *\$widget*.

\$widget->bind(tag, <modifier-modifier-type-detail>);

Returns the callback bound to the given event descriptor for *tag*.

\$widget->bind(<modifier-modifier-type-detail> => callback);

Binds *callback* to the given sequence for *\$widget*

\$widget->bind(tag, <modifier-modifier-type-detail> => callback);

Binds *callback* to the given sequence for *tag*

\$widget=>bindtags(?tagList?);

Sets the current precedence order of tags for *\$widget* to *tagList*. By default a Perl/Tk widget's *taglist* is (*class, instance, toplevel, 'all'*). Note that this ordering differs from that of Tcl/Tk's (*instance, class, toplevel, 'all'*).

Modifiers:

Any	Triple	Button5, B5	Mod3, M3
Control	Button1, B1	Meta, M	Mod4, M4
Shift	Button2, B2	Mod1, M1	Mod5, M5
Lock	Button3, B3	Mod2, M2	Alt
Double	Button4, B4		

Types:

ButtonPress, Button	Expose	Leave
ButtonRelease	FocusIn	Map
Circulate	FocusOut	Property
Colormap	Gravity	Reparent

14. Geometry Management

The pack Command

`$widget->pack(?-options?);`

Details how slave window `$widget` should be managed.

-after => *sibling* -in => *master* -pady => *pixels*
 -anchor => *anchor* -ipadx => *pixels* -fill => **none** | **x** | **y** | **both**
 -before => *sibling* -ipady => *pixels* -side => **top** | **bottom** | **left** | **right**
 -expand => *boolean* -padx => *pixels*

`$widget->packForget;`

Unmanages the given slave.

`$widget->packInfo;`

Returns list containing current pack configuration.

`$master->packPropagate(?boolean?);`

Enables or disables propagation for the window `$master`.

`$master->packSlaves;`

Returns lists of slaves in the window `$master`.

The place Command

`$widget->place(-option => value ?, -option => value ...?);`

Details how `$widget` should be managed.

-anchor => *anchor* -relheight => *size* -x => *location*
 -height => *size* -relwidth => *size* -y => *location*
 -in => *master* -relx => *location* -bordermode => **inside** | **outside** | **ignore**
 -width => *size* -rely => *location*

`$widget->placeForget;`

Unmanages `$widget`.

`$widget->placeInfo;`

Returns list containing current place configuration of `$widget`.

`$master->placeSlaves;`

Returns lists of slaves in the window `$master`.

The grid Command

`$widget->grid(?-option => value ...?);`

-column => *n* -ipady => *amount* -row => *n*
 -columnspan => *n* -padx => *amount* -rowspan => *n*
 -in => *other* -pady => *amount* -sticky => ?**n** | **s** | **e** | **w** | **ns** | **ew**?
 -ipadx => *amount*

`$master->gridBbox(column, row);`

Returns an *approximate* bounding box in pixels of space occupied by `column, row`.

`$master->gridColumnconfigure(column ?, -minsize => size? ?, -weight => int?);`

Set/get minimum column size and relative column weight.

Coordinate examples: 5 (pixel), 2.2i (inch), 4.1c (cm), 3m (mm), 21p (pts)

Larger y-coordinates refer to points lower on the screen.

Larger x-coordinates refer to points farther to the right.

Character positions: '*charIndex*', '**end**', '**insert**', '**sel.first**', '**sel.last**', '@x,y'

Canvas Commands

`$canvas->addtag(tag, searchSpec ?, arg, arg ...?);`

Add `tag` to the list of tags associated with each item that satisfy `searchSpec`. See Canvas Search Specs below.

`$canvas->bbox(tagOrId ?, tagOrId ...?);`

Returns a list (x1, y1, x2, y2) giving an *approximate* bounding box for all the items named by the `tagOrId` arguments.

`$canvas->bind(tagOrId ?, sequence => callback?);`

Associates `callback` to be invoked on events specified with `sequence` with the items given by `tagOrId`.

`$canvas->canvax(screenx ?, gridspacing?);`

Returns the canvas x-coordinate that is displayed at screen x-coordinate `screenx` possibly rounding to nearest multiple of `gridspacing` units.

`$canvas->canvay(screeny ?, gridspacing?);`

Returns the canvas y-coordinate that is displayed at screen y-coordinate `screeny` possibly rounding to nearest multiple of `gridspacing` units.

`$canvas->coords(tagOrId ?, x0, y0 ...?);`

Query or modify the coordinates that define an item.

`$canvas->createType(x, y ?x, y ...? ?, -option=>value ...?);`

Create a new item of type `Type` at specified coordinates and with list options. Currently `Type` may be: **Arc** **Bitmap** **Image** **Line** **Oval** **Polygon** **Rectangle** **Text** **Window**.

`$canvas->dchars(tagOrId, first ?, last?);`

For items given by `tagOrId`, delete the characters in the range given by `first` and `last` (defaults to `first`), inclusive.

`$canvas->delete(?tagOrId ...?);`

Delete each of the items given by each `tagOrId`.

`$canvas->dtag(tagOrId ?, tagToDelete?);`

Remove tag `tagToDelete` from the taglist of items given by `tagOrId`.

`$canvas->find(searchSpec ?, arg, arg ...?);`

Returns a list of the items that satisfy the specification `searchSpec`. See Canvas Search Specs below.

`$canvas->focus(tagOrId);`

Set the focus to the first textual item given by `tagOrId`.

`$canvas->gettags(tagOrId);`

Return a list of the tags associated with the first item given by `tagOrId`.

`$canvas->icursor(tagOrId, index);`

Set the insertion cursor for the item(s) given by `tagOrId` to just before the character position `index`.

`$canvas->index(tagOrId, index);`

Returns a decimal string giving the numerical index within `tagOrId` corresponding to character position `index`.

\$canvas->insert(*tagOrId*, *beforeThis*, *string*);
Insert *string* just before character position *beforeThis* in items given by *tagOrId* that support textual insertion.

\$canvas->itemcget(*tagOrId*, *-option*);
Returns the value *-option* for the item given by *tagOrId*.

\$canvas->itemconfigure(*tagOrId* ?, *-option => value ...?*)
Modifies item-specific options for the items given by *tagOrId*.

\$canvas->lower(*tagOrId* ?, *belowThis*?);
Move the items given by *tagOrId* to a new position in the display list just before the first item given by *belowThis*.

\$canvas->move(*tagOrId*, *xAmount*, *yAmount*);
Move the items given by *tagOrId* in the canvas coordinate space by adding *xAmount* and *yAmount* to each items x and y coordinates, respectively.

\$canvas->postscript(*-option => value ...?*);
Generate an Encapsulated Postscript representation for part or all of the canvas. See Canvas Postscript Options below.

\$canvas->raise(*tagOrId* ?, *aboveThis*?);
Move the items given by *tagOrId* to a new position in the display list just after the first item given by *aboveThis*.

\$canvas->scale(*tagOrId*, *xOrigin*, *yOrigin*, *xScale*, *yScale*);
Rescale items given by *tagOrId* in canvas coordinate space to change the distance from *xOrigin*, *yOrigin* by a factor of *xScale*, *yScale* respectively.

\$canvas->scan(*args*);
See Widget Scroll Commands above.

\$canvas->selectAdjust(*tagOrId*, *index*);
Adjust nearest end of current selection in *tagOrId* to be at *index* and set the other end to be the new selection anchor.

\$canvas->selectClear;
Clear the selection if it is in the widget.

\$canvas->selectFrom(*tagOrId*, *index*);
Set the selection anchor in *tagOrId* to just before the character at *index*.

\$canvas->selectItem;
Return id of the selected item. Returns a empty string if there is none.

\$canvas->selectTo(*tagOrId*, *index*);
Set the selection to extend between *index* and anchor point in *tagOrId*.

\$canvas->type(*tagOrId*);
Returns the type of the first item given by *tagOrId*.

\$canvas->xview | yview(*args*);
See Widget Scroll Commands above.

Canvas Search Specifications

above => tagOrId
Selects the item just after the one given by *tagOrId* in the display list.

all
Selects all the items in the canvas.

below => tagOrId
Selects the item just before the one given by *tagOrId* in the display list.

closest => x, y ?, halo? ?, start?
Select the topmost, closest item to @x,y that is below *start* in the display list. Any item closer than *halo* to the point is considered to overlap it.

\$mw->frame;
Returns the X window identifier for the outermost decorative frame containing *\$mw*. If *\$mw* has none, returns X id of *\$mw* itself.

\$mw->geometry(*?newGeometry?*);
Changes geometry of *\$mw* to *newGeometry*.

\$mw->grid(*?baseWidth, baseHeight, widthInc, heightInc?*);
Indicates that *\$mw* is to be managed as a gridded window with the specified relation between grid and pixel units.

\$mw->group(*?pathName?*);
Gives path name for leader of group to which *\$mw* belongs.

\$mw->iconbitmap(*?bitmap?*);
Specifies a bitmap to use as icon image when *\$mw* is iconified.

\$mw->iconify;
Arrange for *\$mw* to be iconified.

\$mw->iconmask(*?bitmap?*);
Specifies a bitmap to use to mask icon image when *\$mw* is iconified.

\$mw->iconname(*?newName?*);
Specifies name to use as a label for *\$mw*'s icon.

\$mw->iconposition(*?x, y?*);
Specifies position on root window to place *\$mw*'s icon.

\$mw->iconwindow(*?pathName?*);
Sets path name of window to use as the icon when *\$mw* is iconified.

\$mw->maxsize(*?width, height?*);
Specifies maximum size *\$mw* may be resized to in each direction.

\$mw->minsize(*?width, height?*);
Specifies minimum size *\$mw* may be resized to in each direction.

\$mw->overrideredirect(*?boolean?*);
Set or unset the override-redirect flag of *\$mw* commonly used by window manager to determine whether window should decorative frame.

\$mw->positionfrom(*? 'program' | 'user' ?*);
Indicate from whom the *\$mw*'s current position was requested.

\$mw->protocol(*?name? ?, callback?*);
Specify a Perl callback to be invoked for messages of protocol *name*.

\$mw->resizable(*?widthBoolean, heightBoolean?*);
Specifies whether *\$mw*'s width and/or height is resizable.

\$mw->sizefrom(*?program | user?*);
Indicate from whom the *\$mw*'s current size was requested.

\$mw->state;
Returns current state of *\$mw*: **normal**, **iconic**, or **withdrawn**.

\$mw->title(*?string?*);
Set title for *\$mw*'s decorative frame to *string*.

\$mw->transient(*?master?*);
Informs window manager that *\$mw* is a transient of the window *master*.

\$mw->withdraw;
Arranges for *\$mw* to be withdrawn from the screen.

\$widget->screenvisual;
Returns the visual class of *\$widget*'s screen. Maybe one of: 'directcolor', 'grayscale', 'pseudocolor', 'staticcolor', 'staticgray', or 'truecolor'.

\$widget->screenwidth;
Returns the width in pixels of *\$widget*'s screen.

\$widget->toplevel;
Returns the pathname of the toplevel window containing *\$widget*.

\$widget->visual;
Returns the visual class of *\$widget* (see *\$widget->screenvisual*).

\$widget->visualsavailable;
Returns a list whose elements describe the visuals available for *\$widget*'s screen including class and depth..

\$widget->vrootheight;
Returns the height of the virtual root window associated with *\$widget*.

\$widget->vrootwidth;
Returns the width of the virtual root window associated with *\$widget*.

\$widget->vrootx;
Returns the x-offset of the virtual root window associated with *\$widget*.

\$widget->vrooty;
Returns the y-offset of the virtual root window associated with *\$widget*.

\$widget->width;
Returns *\$widget*'s width in pixels.

\$widget->x;
Returns x-coordinate, in *\$widget*'s parent, of the upper-left corner of *\$widget*.

\$widget->y;
Returns y-coordinate, in *\$widget*'s parent, of the upper-left corner of *\$widget*.

13. The Window Manager

Many of the following methods have one or more optional arguments which, when specified, set something, but when missing, get something. The widget is typically a `MainWindow` or a `Toplevel`.

\$mw->aspect(?minNumer, minDenom, maxNumer, maxDenom?);
Inform window manager of desired aspect ratio range for *\$mw*.

\$mw->client(?name?);
Store *name* in *\$mw*'s `WM_CLIENT_MACHINE` property. Informs window manager of client machine on which the application is running.

\$mw->colormapwindows(?windowList?);
Store *windowList* in *\$mw*'s `WM_COLORMAP_WINDOWS` property which identifies the internal windows within *\$mw* with private colormaps.

\$mw->command(?value?);
Store *value* in *\$mw*'s `WM_COMMAND` property. Informs window manager of command used to invoke the application.

\$mw->deiconify;
Arrange for *\$mw* to be mapped on the screen.

\$mw->focusmodel(?'active' | 'passive'?);
Specifies the focus model for *\$mw*.

enclosed => x1, y1, x2, y2
Selects all the items completely enclosed within *x1, y1, x2, y2*.

overlapping => x1, y1, x2, y2
Selects all the items that overlap or are enclosed within *x1, y1, x2, y2*.

withtag => tagOrId
Selects all the items given by *tagOrId*.

Canvas Item Types

\$canvas->createArc(x1, y1, x2, y2 ?, -option => value ...?);
-fill => color -stipple => bitmap -width => outlineWidth
-outline => color -tags => tagList

-extent => degrees
Size of the angular range occupied by arc.

-outlinestipple => bitmap
Bitmap stipple to use to draw arc's outline.

-start => degrees
Starting angle measured from 3-o'clock position.

-style => pieslice | chord | arc
How to "complete" the region of the arc.

\$canvas->createBitmap(x, y ?, -option => value ...?);
-anchor => anchorPos -bitmap => bitmap -tags => tagList
-background color -foreground color

\$canvas->createImage(x, y ?, -option => value ...?);
-anchor => anchorPos -image => image -tags => tagList?

\$canvas->createLine(x1, y1, ... xN, yN ?, -option => value ...?);
-fill => color -stipple => bitmap -width => outlineWidth
-smooth => boolean -tags => tagList

-arrow => none | first | last | both
Specify on which ends of the line to draw arrows.

-arrowshape => shape
Three element list which describes shape of arrow.

-capstyle => butt | projecting | round
How to draw caps at endpoints of the line. Default is `butt`.

-jointstyle => bevel | miter | round
How joints are to be drawn at vertices. Default is `miter`.

-splinesteps => number
Degree of smoothness desired for curves.

\$canvas->createOval(x1, y1, x2, y2 ?, -option => value ...?);
-fill => color -stipple => bitmap -width => outlineWidth
-outline => color -tags => tagList

\$canvas->createPolygon(x1, y1, ... xN, yN ?, -option => value ...?);
-fill => color -smooth => boolean -tags => tagList
-outline => color -stipple => bitmap -width => outlineWidth

-splinesteps => number
Degree of smoothness desired for curved perimeter.

\$canvas->createRectangle(x1, y1, x2, y2 ?, -option => value ...?);
-fill => color **-stipple => bitmap** **-width => outlineWidth**
-outline => color **-tags => tagList**

\$canvas->createText(x, y ?, -option => value ...?);
-anchor => anchorPos **-font => font** **-tags => tagList**
-fill => color **-stipple => bitmap** **-text => string**

-justify => left | right | center
How to justify text within its bounding region.

-width => lineLength
Maximum line length for the text. If zero, break only on \n.

\$canvas->createWindow(x, y ?, -option => value ...?);
-anchor => anchorPos **-tags => tagList**

-height => height Height in screen units to assign item's window.
-width => width Width in screen units to assign item's window.
-window => pathName
Window to associate with item.

Canvas Postscript Options

\$canvas->postscript(?-option => value ...?);

-colormap => varRef
Specifies a color mapping to use where *varRef* is an array variable whose elements specify Postscript code to set a particular color value.

-colormode => color | grey | mono
Specifies how to output color information.

-file => pathName
Specifies the name of the file in which to write the Postscript. If not specified, the Postscript is returned as the result of the command.

-fontmap => varRef
Specifies a font mapping to use where *varRef* is an array variable whose elements specify the Postscript font and size to use as a two element list.

-height => size
Specifies the height of the area of the canvas to print. Defaults to the height of the canvas window

-pageanchor => anchor
Specifies which point of the printed area should be appear over the positioning point on the page. Defaults to **center**.

-pageheight => size
Specifies that the Postscript should be scaled in both x and y so that the printed area is *size* high on the Postscript page.

-pagewidth => size
Specifies that the Postscript should be scaled in both x and y so that the printed area is *size* wide on the Postscript page.

-pagex => position
Set the x-coordinate of the positioning point on the page to *position*.

\$widget->MainWindow;
Returns a reference to the display's **MainWindow**.

\$widget->manager;
Returns the name of the geometry manager currently responsible for *\$widget*.

\$widget->name;
Returns *\$widget*'s name within its parent, as opposed to its full path name.

\$widget->parent;
Returns the path name of *\$widget*'s parent.

\$widget->pathname(id);
Returns the path name of the window whose X identifier is *id* on *\$widget*'s display.

\$widget->pixels(number);
Returns the number of pixels in *\$widget* corresponding to the distance given by *number*, rounded to the nearest integer.

\$widget->pointerx;
Returns mouse pointer's x coordinate on *\$widget*'s screen.

\$widget->pointerxy;
Returns mouse pointer's x and y coordinates on *\$widget*'s screen.

\$widget->pointery;
Returns mouse pointer's y coordinate on *\$widget*'s screen.

\$widget->reqheight;
Returns a decimal string giving *\$widget*'s requested height, in pixels.

\$widget->reqwidth;
Returns a decimal string giving *\$widget*'s requested width, in pixels.

\$widget->rgb(color);
Returns a list of the three RGB values that correspond to *color* in *\$widget*.

\$widget->rootx;
Returns the x-coordinate, in the root window of the screen, of the upper-left corner of *\$widget* (including its border).

\$widget->rooty;
Returns the y-coordinate, in the root window of the screen, of the upper-left corner of *\$widget* (including its border).

\$widget->server;
Returns server information on *\$widget*'s display.

\$widget->screen;
Returns the name of the screen associated with *\$widget*, in the form *displayName.screenIndex*.

\$widget->screncells;
Returns the number of cells in the default color map for *\$widget*'s screen.

\$widget->screndepth;
Returns the depth (bits per pixel) of *\$widget*'s screen.

\$widget->screenheight;
Returns the height in pixels of *\$widget*'s screen.

\$widget->screenmmheight;
Returns the height in millimeters of *\$widget*'s screen.

\$widget->screenmmwidth;
Returns the width in millimeters of *\$widget*'s screen.

- from** => *x1, y1, x2, y2*
Specifies a rectangular region of the image file to copy from.
- shrink**
Will clip image so copied region is in bottom-right corner.
- to** => *x, y*
Specifies coords of the top-left corner in image to copy into.
- \$image->redither;**
Redither the image.
- \$image->write**(*pathName* ?, *-option => value ...?*);
Writes image data from image into file *pathName*.
- format** => *format-name*
Specifies image format for the file.
- from** => *x1, y1, x2, y2*
Specifies a rectangular region of the image to copy from.

12. Window Information

- \$widget->allmapped;**
Returns 1 if *\$widget* and all its ancestors are mapped, 0 otherwise.
- \$widget->atom**(*name*);
Returns integer identifier for atom given by *name* on *\$widget*'s display.
- \$widget->atomname**(*id*);
Returns textual name of atom given by integer *id* on *\$widget*'s display.
- \$widget->cells;**
Returns number of cells in the colormap for *\$widget*.
- \$widget->children;**
Returns list containing path names of all the children of *\$widget*.
- \$widget->class;**
Returns the class name of *\$widget*.
- \$widget->colormapfull;**
Return 1 if the colormap for *\$widget* is full, 0 otherwise.
- \$widget->containing**(*rootX, rootY*);
Returns path name of window containing the point *rootX rootY* on *\$widget*'s display.
- \$widget->depth;**
Returns the depth (bits per pixel) of *\$widget*.
- \$widget->fpixels**(*number*);
Returns floating-point value giving the number of pixels in *\$widget* corresponding to the distance given by *number*.
- \$widget->geometry;**
Returns the pixel geometry for *\$widget*, in the form *widthxheight+x+y*.
- \$widget->height;**
Returns height of *\$widget* in pixels.
- \$widget->id;**
Returns a hexadecimal string indicating the X identifier for *\$widget*.
- \$widget->interps;**
Returns a list of all Perl interpreters registered on *\$widget*'s display.
- \$widget->ismapped;**
Returns 1 if *\$widget* is currently mapped, 0 otherwise.

- pagey** => *position*
Set the y-coordinate of the positioning point on the page to *position*.
- rotate** => *boolean*
Whether the printed area is to be rotated 90 degrees. ("landscape").
- width** => *size*
Specifies the width of the area of the canvas to print. Defaults to the width of the canvas window.
- x** => *position*
Set the x-coordinate of the left edge of canvas area to print.
- y** => *position*
Set the y-coordinate of the top edge of canvas area to print.

5. The Entry Widget

Entry Widget Options

- | | | |
|----------------------|---------------------|--------------------|
| -background | -highlightcolor | -relief |
| -borderwidth | -highlightthickness | -selectbackground |
| -cursor | -insertbackground | -selectborderwidth |
| -exportselection | -insertborderwidth | -selectforeground |
| -font | -insertofftime | -state |
| -foreground | -insertontime | -takefocus |
| -highlightbackground | -insertwidth | -textvariable |
| | -justify | -width |

-show *boolean*

Whether to show actual character or "*" in entry.

Entry Indices: *number* (starts at 0), 'anchor', 'end', 'insert', 'sel.first', 'sel.last', '@x'

Entry Widget Commands

- \$entry->bbox**(*index*);
Returns a list (x, y, width, height) giving an *approximate* bounding box of character given by *index*.
- \$entry->delete**(*first* ?, *last*?);
Delete characters from *first* through character just before *last*.
- \$entry->get;**
Returns the *\$entry*'s string.
- \$entry->icursor**(*index*);
Display insertion cursor just before character at *index*.
- \$entry->index**(*index*);
Returns the numerical index corresponding to *index*.
- \$entry->insert**(*index, string*);
Insert *string* just before character at *index*.
- \$entry->scan**(*-option, args*);
See Widget Scroll Commands above.
- \$entry->selectionAdjust**(*index*);
Adjust nearest end of current selection to be at *index* and set the other end to the anchor point.

\$entry->selectionClear;
Clear the selection if currently in the widget.

\$entry->selectionFrom(*index*);
Set the anchor point to be at *index*.

\$entry->selectionPresent;
Returns 1 if any characters are selected, 0 otherwise.

\$entry->selectionRange(*start*, *end*);
Select the characters from *start* through character just before *end*.

\$entry->selectionTo(*index*);
Set the selection to extend between *index* and anchor point.

6. The Listbox Widget

Listbox Widget Options

-background	-height	-selectborderwidth
-borderwidth	-highlightbackground	-selectforeground
-cursor	-highlightcolor	-setGrid
-exportselection	-highlightthickness	-takefocus
-font	-relief	-width
-foreground	-selectbackground	-xscrollcommand
		-yscrollcommand

-selectMode `single|browse|multiple|extended`

Listbox Indices: *number* (starts at 0), `'active'`, `'anchor'`, `'end'`, `'@x,y'`

Listbox Widget Commands

\$listbox->activate(*index*);
Sets the active element to *index*.

\$listbox->bbox(*index*);
Returns a list (*x*, *y*, *width*, *height*) giving an *approximate* bounding box of character given by *index*.

\$listbox->curselection;
Returns list of indices of all elements currently selected.

\$listbox->delete(*index1* ?, *index2*?);
Delete range of elements from *index1* to *index2* (defaults to *index1*).

\$listbox->get(*index1* ?, *index2*?);
Return as a list contents of elements from *index1* to *index2*.

\$listbox->index(*index*);
Returns position *index* in *number* notation.

\$listbox->insert(*index* ?, *element* ...?);
Insert specified elements just before element at *index*.

\$listbox->nearest(*y*);
Return index of element nearest to *y*-coordinate.

\$listbox->scan(*args*);
See Widget Scroll Commands above.

The Bitmap image method

-background => *color*
Set background color for bitmap.

-data => *string*
Specify contents of bitmap in X11 bitmap format.

-file => *pathName*
Gives name of file whose contents define the bitmap in X11 bitmap format.

-foreground => *color*
Set foreground color for bitmap.

-maskdata => *string*
Specify contents of mask in X11 bitmap format.

-maskfile => *pathName*
Gives name of file whose contents define the mask in X11 bitmap format.

The Photo image method

-data => *string*
Specify contents of image in a supported format.

-format => *formatName*
Specify format for data specified with the **-data** or **-file** options.

-file => *pathName*
Gives name of file whose contents define the image in supported format.

-height => *number*
Specifies pixel height of the image.

-palette => *paletteSpec*
Set the resolution of the color cube to be allocated for image.

-width => *number*
Specifies pixel width of the image.

\$image->blank;
Blanks the image so it has no data and is completely transparent.

\$image->copy(*sourceImage* ?, *-option* => *value* ...?);
Copy a region from *sourceImage* to *\$image* using given options.

-from => *x1*, *y1*, *x2*, *y2*
Specifies rectangular region of source image to be copied.

-to => *x1*, *y1*, *x2*, *y2*
Specifies rectangular region of target image to be affected.

-shrink
Will clip target image so copied region is in bottom-right corner.

-zoom => *x*, *y*
Magnifies source region by *x* *y* in respective direction.

-subsample => *x*, *y*
Reduces source image by using only every *x* *y*th pixel.

\$image->get(*x*, *y*);
Returns RGB value of pixel at coords *x* *y* as list of three integers.

\$image->put(*data* ?-*to* => *x1*, *y1*, *x2*, *y2*?);
Sets pixels values for the region *x1* *y1* *x2* *y2* for 2-D array *data*.

\$image->read(*pathName* ?,*-option* => *value* ...?);
Reads image data from file *pathName* into *\$image* using given options.

-format => *format-name*
Specifies image format of file.

- file => pathName**
Path name of file with embedded POD directives.
- scrollbars => scrollSpec**
Scrollbar specifications.

Scrolled

Perl/Tk includes the special constructor **Scrolled** which creates a widget with attached scrollbars, as long as the widget class supports the x/y scrollcommand(s), as **Canvas**, **Entry**, **Listbox** and **Text** do. Scrollbars can be required, or might spring into existence only when needed.

```
$scrolled = $parent->Scrolled(widgetClass ...);
```

You may specify option/value pairs which are passed to the *widgetClass* constructor.

- scrollbars => scrollbarSpecs**
The strings *n s e w* specify top, bottom, left or right scrollbars, respectively. The string *sw* creates two scrollbars, on the left and bottom of the widget. The string *o* means optional and *r* required, so 'rwos' means required on 'west' (vertical), optional on 'south' (horizontal).

11. Images

Images are created using the **DefineBitmap**, **Bitmap** and **Photo** methods, described below. Bitmaps have a pixel depth of 2 bits, whereas Photos can be full color 24 bit GIF89, XPM, XBM or BMP objects.

```
$widget->DefineBitmap(bitmapName, bitColumns, bitRows, bitVector);  
Define a bitmap named bitmapName directly in Perl code. The first bitColumns bits packed in bitVector are row one.
```

Here are image manipulation methods common to the remaining two image types:

- \$image->delete;**
Deletes the image.
- \$image->height;**
Returns pixel height of image.
- \$image->imageNames;**
Returns a list of the names of all existing images.
- \$image->type;**
Returns the type of image.
- \$image->imageTypes;**
Returns a list of valid image types.
- \$image->width;**
Returns pixel width of image.

When an image is created via the **DefineBitmap**, **Bitmap** or **Photo** method, Tk creates an image object reference to the the image. For all image types, this object supports the **cget** and **configure** methods in the same manner as widgets for changing and querying configuration options. Of course, image configuration commands can still be passed to the creation method:

```
$image = $mw->Photo(-file => '/home/bug/photo.gif');
```

- \$listbox->selectionAnchor(index);**
Set the selection anchor to element at *index*.
- \$listbox->selectionClear(first ?, last?);**
Deselect elements between *first* and *last* inclusive.
- \$listbox->selectionIncludes(index);**
Returns 1 if element at *index* is selected, 0 otherwise.
- \$listbox->selectionSet(first ?,last?);**
Add all elements between *first* and *last* inclusive to selection.
- \$listbox->see(index);**
Adjust the view in window so element at *index* is completely visible.
- \$listbox->size**
Returns number of elements in listbox.
- \$listbox->xview | yview(args);**
See Widget Scroll Commands above.

7. The Menu Widget

Menu Widget Options

- activebackground** **-borderwidth** **-font**
- activeborderwidth** **-cursor** **-foreground**
- activeforeground** **-disabledforeground** **-relief**
- background**

- postcommand => callback**
Specify Perl command to invoke immediately before the menu is posted.
 - selectcolor => color**
Specifies indicator color for checkbutton and radiobutton entries.
 - tearoff => boolean**
Whether to include a tear-off entry at top of menu.
 - tearoffcommand => callback**
Specifies command to be run when menu is torn off. The name of the menu and the new torn-off window will be appended on invocation.
 - transient => boolean**
Whether menu should be displayed as transient or not.
- Entry Types: *cascade*, *checkbutton*, *command*, *radiobutton*, *separator*
- Menu Indices: *number* (starts at 0, normally the *tearoff* item), **'active'**, **'last'**, **'none'**, **'@y'**, **'matchPattern'**

Menu Widget Commands

- \$menu->activate(index);**
Change state of entry at *index* to be sole active entry in menu.
- \$menu->add(type ?, -option => value ...?);**
Add new entry of type *type* to bottom of menu. See below for options.
- \$menu->delete(index1 ?, index2?);**
Delete all entries between *index1* and *index2* inclusive.
- \$menu->entrycget(index, -option);**
Return current value of *-option* for entry at *index*.

\$menu->entryconfigure(*index* ?, *-option =>value ...?*);
Set option values for entry at *index*.

\$menu->index(*index*);
Returns the numerical index corresponding to *index*.

\$menu->insert(*index*, *type* ?, *-option => value ...?*);
Same as **add** but inserts new entry just before entry at *index*.

\$menu->invoke(*index*);
Invoke the action of the menu entry at *index*.

\$menu->post(*x*, *y*);
Display menu on screen at root-window coordinates given by *x*, *y*.

\$menu->postcascade(*index*);
Post submenu associated with cascade entry at *index*.

\$menu->type(*index*);
Returns type of menu entry at *index*.

\$menu->unpost;
Unmap window so it is no longer displayed.

\$menu->yposition(*index*);
Returns the y-coordinate within the menu window of the topmost pixel in the entry specified by *index*.

Menu Entry Options

The following options work for all cascade, checkbutton, command, and radiobutton entries unless otherwise specified.

-activebackground	-bitmap	-image
-activeforeground	-font	-state
-background	-foreground	-underline

-accelerator => *string*
Specifies string to display at right side of menu entry.

-command => *callback*
Perl command to execute entry is invoked.

-indicatoron => *boolean*
Whether indicator for checkbutton or radiobutton entry should be displayed.

-label => *string*
Textual string to display on left side of menu entry.

-menu => *pathName*
Pathname to a menu to post when cascade entry is active.

-offvalue => *value*
Value to store in checkbutton entry's associated variable when deselected.

-onvalue => *value*
Value to store in checkbutton entry's associated variable when selected.

-selectcolor => *color*
Color for indicator in checkbutton and radiobutton entries.

-selectimage => *image*
Image to draw in indicator for checkbutton and radiobutton entries.

-value => *value*
Value to store in radiobutton entry's associated variable when selected.

-variable => *varRef*
Name of global variable to set when checkbutton or radiobutton is selected.

-justify => **let** | **right** | **center**
When there are multiple lines of text displayed in a tab, this option determines the justification of the lines.

-createcmd => *callback*
The *callback* invoked the first time the page is shown on the screen.

-raisecmd => *callback*
The *callback* invoked whenever this page is raised by the user.

-state => **normal** | **disabled**
Specifies whether this page can be raised by the user.

-underline => *integer*
Specifies the integer index of a character to underline in the tab.

-wraplength => *integer*
This option specifies the maximum line length of the label string on this tab.

\$note->delete(*page*);
Deletes the page identified by *page*.

\$note->pagecget(*page*, *-option*);
Returns the current value of the configuration option given by *-option* in the page given by *page*. Options may have any of the values accepted in the **add** method.

\$note->pageconfigure(*page*, *-option => value*);
Like configure for the page indicated by *page*. Options may be any of the options accepted by the **add** method.

\$note->raise(*page*);
Raise the page identified by *page*.

\$note->raised;
Returns the name of the currently raised page.

Optionmenu

-activebackground	-foreground	-relief
-activeforeground	-height	-state
-anchor	-highlightbackground	-takefocus
-background	-highlightcolor	-text
-bitmap	-highlightthickness	-textvariable
-borderwidth	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength
-font	-pady	

-options => [*optionList*]
The menuitems, specified as a reference to a list of strings.

-command => *callback*
The callback to invoke after *varRef* of *-textvariable* is set.

Pod

-borderwidth	-highlightbackground	-relief
-cursor	-highlightcolor	-takefocus
-height	-highlightthickness	-width

`$hlist->showEntry(?entryPath?);`
Shows the list entry *entryPath*.

`$hlist->xview;`
Returns a list of two real fractions between 0 and 1 describing the horizontal span that is visible in the window.

`$hlist->xview(entryPath);`
Adjusts the view in the window so that the list entry identified by *entryPath* is aligned to the left edge of the window.

`$hlist->xviewMoveto(fraction);`
Adjusts the view in the window so that *fraction* of the total width of the HList is off-screen to the left.

`$hlist->xviewScroll(int, units | pages);`
Shifts the view in the window left or right according to *int* and *what*.

`$hlist->yview;`
Returns a list of real fractions between 0 and 1 describing the vertical span that is visible in the window.

`$hlist->yview(entryPath);`
Adjusts the view in the window so that *entryPath* is displayed at the top of the window.

`$hlist->yviewMoveto(fraction);`
Adjusts the view in the window so that *fraction* of the total height of the HList is off-screen to the top.

`$hlist->yviewScroll(int, 'units' | 'pages');`
Shifts the view in the window up or down according to *int* and *what*.

NoteBook

`-borderwidth` `-highlightbackground` `-relief`
`-cursor` `-highlightcolor` `-takefocus`
`-height` `-highlightthickness` `-width`

`-dynamicgeometry => boolean`
False ensures all notebook pages are the same size. True resizes the notebook as different pages are selected.

`-ipadx => pixels`
The amount of internal horizontal padding around the pages.

`-ipady => pixels`
The amount of internal vertical padding around the pages.

`$note->add(page, -option => value);`
Adds a page with name *page* to the notebook.

`-anchor => n|ne|e|se|s|sw|w|nw|center`
Specifies how the information in a tab is to be displayed.

`-bitmap => bitmap`
Specifies a bitmap to display on the tab of this page. The bitmap is displayed only if none of the `-label` or `-image` options are specified.

`-image => image`
Specifies an image to display on the tab of this page. The image is displayed only if the `-label` option is not specified.

`-label => string`
Specifies the text string to display on the tab of this page.

8. The Text Widget

Text Widget Options

<code>-background</code>	<code>-highlightthickness</code>	<code>-selectbackground</code>
<code>-borderwidth</code>	<code>-insertbackground</code>	<code>-selectborderwidth</code>
<code>-cursor</code>	<code>-insertborderwidth</code>	<code>-selectforeground</code>
<code>-exportselection</code>	<code>-insertofftime</code>	<code>-setgrid</code>
<code>-font</code>	<code>-insertontime</code>	<code>-state</code>
<code>-foreground</code>	<code>-insertwidth</code>	<code>-takefocus</code>
<code>-height</code>	<code>-padx</code>	<code>-width</code>
<code>-highlightbackground</code>	<code>-pady</code>	<code>-xscrollcommand</code>
<code>-highlightcolor</code>	<code>-relief</code>	<code>-yscrollcommand</code>

`-spacing1 => size` Space in screen units above paragraphs.
`-spacing2 => size` Space in screen units between paragraph lines.
`-spacing3 => size` Space in screen units below paragraphs.
`-tabs => tabList`
Set of tab stops as a list of screen distances giving their positions. Each stop may be followed by one of `left`, `right`, `center`, or `numeric`.
`-wrap => 'none' | 'char' | 'word'`
How to wrap lines.

Text Indices

Syntax: *base* ?*modifier* ... ?

Base: *'line.char'* (*line* starts at 1, *char* starts at 0), *'@x,y'*, *'end'*, *'mark'*, *'tag.first'*, *'tag.last'*, *'pathName'*

Modifier: *'± count chars'*, *'± count lines'*, *'linestart'*, *'lineend'*, *'wordstart'*, *'wordend'*

Ranges: Ranges include all characters from the start index up to but not including the character at the stop index.

Text Tag Options

<code>-background</code>	<code>-justify</code>	<code>-spacing2</code>
<code>-borderwidth</code>	<code>-relief</code>	<code>-spacing3</code>
<code>-font</code>	<code>-spacing1</code>	<code>-wrap</code>
<code>-foreground</code>		

`-bgstipple => bitmap` Stipple pattern for background.

`-fgstipple => bitmap` Stipple pattern for foreground.

`-lmargin1 => size` Left margin of first line of a paragraph.

`-lmargin2 => size` Left margin of wrapped lines of a paragraph.

`-offset => size` Offset of baseline from normal baseline.

`-overstrike => boolean`
Whether to overstrike text.

`-rmargin => size` Right margin of all lines.

`-tabs => tabList` Set of tab stops (see `-tabs` above).

`-underline => boolean`
Whether to underline text.

Text Embedded Window Options

Use `-window` to pass any Perl/Tk widget reference to `windowCreate`. Manage embedded windows with `windowConfigure` and `windowCget`.

`-align => top | center | bottom | baseline`
Where window is displayed on the line.

`-create => script`
Script to create and return window pathname if no `-window` option is given.

`-stretch => boolean`
Whether window should be stretched vertically to fill line.

`-window => widget`
Name of window to display

Text Widget Commands

`$text->bbox(index);`
Returns a list (*x*, *y*, *width*, *height*) giving an *approximate* bounding box of character given by *index*.

`$text->compare(index1, op, index2);`
Compares indices *index1* and *index2* according to relational operator *op*.

`$text->delete(index1 ?, index2?);`
Delete range of characters (*index2* defaults to *index1* + 1 *char*).

`$text->dlineinfo(index);`
Returns a list (*x*, *y*, *width*, *height*, *baseline*) describing the screen area taken by display line at *index*.

`$text->dump(?switches, ? index1 ?, index2?);`
Returns detailed info on text widget contents in range *index1* to *index2*. Switches include `-all`, `-mark`, `-tag`, `-text`, `-window` for specifying type of info returned. The switch `-command => callback` exists to invoke a procedure on each element type in the range.

`$text->get(index1 ?, index2?);`
Returns string of characters in range (*index2* defaults to *index1* + 1 *char*).

`$text->index(index);`
Returns position *index* in *line.char* notation.

`$text->insert(index ?, string ?, tagList, string, tagList ...?);`
Insert *string* into text at *index* applying tags from *tagList*.

`$text->markGravity(markName => ?left | right?);`
Returns (or sets) which adjacent character a mark is attached to.

`$text->markNames`
Returns a list of the names of all marks currently set.

`$text->markNext | markPrevious(index);`
Return name of next/previous mark at or after/before *index*.

`$text->markSet(markName => index);`
Set mark *markName* to position just before character at *index*.

`$text->markUnset(markName => ?, markName ...?);`
Remove each mark specified so they are no longer usable as indices.

`$text->scan(args);`
See Widget Scroll Commands above.

`$hlist->infoAnchor;`
Returns the *entryPath* of the current anchor.

`$hlist->infoBbox(entryPath);`
Returns a list of four numbers describing the visible bounding box of *entryPath*.

`$hlist->infoChildren(?entryPath?);`
If *entryPath* is given, returns a list of its children entries, otherwise returns a list of the toplevel.

`$hlist->infoData(?entryPath?);`
Returns the data associated with *entryPath*.

`$hlist->infoDragsite;`
Returns the *entryPath* of the current dragsite.

`$hlist->infoDropstie;`
Returns the *entryPath* of the current dropstie.

`$hlist->infoExists(entryPath);`
Returns a boolean value indicating whether *entryPath* exists.

`$hlist->infoHidden(entryPath);`
Returns a boolean value indicating whether *entryPath* is hidden.

`$hlist->infoNext(entryPath);`
Returns the *entryPath* of the list entry immediately below this list entry.

`$hlist->infoParent(entryPath);`
Returns the name of the parent of *entryPath*.

`$hlist->infoPrev(entryPath);`
Returns the *entryPath* of the list entry immediately above this list entry.

`$hlist->infoSelection;`
Returns a list of selected entries.

`$hlist->itemCget(entryPath, col, option);`
Returns the current value of *option* for *entryPath* at column *col*.

`$hlist->itemConfigure(entryPath, col, ?, option?, => ?value);`
Query or modify the configuration options of *entryPath* at column *col*.

`$hlist->itemCreate(entryPath, col, ?, -itemtype => type? ?, option => value);`
Creates a new display item at column *col* of *entryPath*.

`$hlist->itemDelete(entryPath, col);`
Deletes the display item at column *col* of *entryPath*.

`$hlist->itemExists(entryPath, col);`
Returns true if there is a display item at column *col* of *entryPath*.

`$hlist->nearest(y);`
Returns the *entryPath* of the visible element nearest to Y-coordinate *y*.

`$hlist->see(entryPath);`
Adjust the view so that *entryPath* is visible.

`$hlist->selectionClear(?from? ?, to?);`
Deselect the list entries *from* through *to*, inclusive.

`$hlist->selectionGet;`
This is an alias for the `infoSelection` widget command.

`$hlist->selectionIncludes(entryPath);`
Returns true if *entryPath* is currently selected.

`$hlist->selectionSet(from ?, to);`
Selects all of the list entrie(s) between between *from* and *to*, inclusive.

\$hlist->dragsiteSet(entryPath);
Sets the dragsite to the list entry identified by *entryPath*.

\$hlist->dragsiteClear;
Remove the dragsite, if any.

\$hlist->dropsiteSet(entryPath);
Sets the dropsite to the list entry identified by *entryPath*.

\$hlist->dropsiteClear;
Remove the dropsite, if any.

\$hlist->entrycget(entryPath, -option);
Returns the current value of the configuration option *-option* for *entryPath*.

\$hlist->entryconfigure(entryPath ?, -option => value?);
Query or modify the configuration options of the list entry *entryPath*.

\$hlist->headerCget(col, option);
If column *col* has a header display item, returns the value of the specified option of the header item.

\$hlist->headerConfigure(col ?, option? ?=> value?);
Query or modify the configuration options of the header display item of column *col*.

\$hlist->headerCreate(col ?, -itemtype => type? ?, -option => value ...?);
Creates a new display item as the header for column *col*.

-borderwidth => width
Specifies the border width of this header item.

-headerbackground => color
Specifies the background color of this header item.

-relief => relief
Specifies the relief type of the border of this header item.

\$hlist->headerDelete(col);
Deletes the header display item for column *col*.

\$hlist->headerExists(col);
Returns true if a header display item exists for column *col*.

\$hlist->headerSize(col);
Returns a two element list of the form [*width, height*] of the header display item for column *col*;

\$hlist->hideEntry(entryPath);
Hides the list entry identified by *entryPath*.

\$hlist->indicatorCget(entryPath, option);
Returns the value of the specified option of the indicator for *entryPath*.

\$hlist->indicatorConfigure(entryPath ?, option? ?=> value?);
Query or modify the configuration options of the indicator display item of *entryPath*.

\$hlist->indicatorCreate(entryPath ?, -itemtype => type? ?, option => value);
Creates a new display item as the indicator for *entryPath*.

\$hlist->indicatorDelete(entryPath);
Deletes the indicator display item for *entryPath*.

\$hlist->indicatorExists(entryPath);
Return true if an indicator display item exists for *entryPath*.

\$hlist->indicatorSize(entryPath);
Returns a two element list of the form [*width, height*] of the indicator display item for *entryPath*.

\$text->search(?switches, ? pattern, index ?, stopIndex?);
Returns index of first character matching *pattern* in text range *index* to *stopIndex*. Switches: *-forwards*, *-backwards*, *-exact*, *-regexp*, *-count var*, *-nocase*

\$text->see(index);
Adjust the view in window so character at *index* is completely visible.

\$text->tagAdd(tagName, index1 ?, index2?);
Apply tag *tagName* to range (*index2* defaults to *index1 + 1 char*).

\$text->tagBind(tagName ?, sequence ?, script??);
Arrange for *script* to be run whenever event *sequence* occurs for a character with tag *tagName*.

\$text->tagCget(tagName => -option);
Return current value of *-option* for tag *tagName*.

\$text->tagConfigure(tagName ?, -option ?, value ?, -option => value ...?);
Modifies tag-specific options for the tag *tagName*.

\$text->tagDelete(tagName ?, tagName ...?);
Delete all tag information for given tags.

\$text->tagLower(tagName ?, belowThis?);
Change priority of tag *tagName* so it is just below tag *belowThis*.

\$text->tagNames(?index?);
Returns a list of the names of all tags associated with character at *index*. If *index* is not given, returns list of all tags defined in widget.

\$text->tagNextrange(tagName, index1 ?, index2?);
Searches character range *index1* to *index2* (default **end**) for the first region tagged with *tagName*. Returns character range of region found.

\$text->tagPrevrange(tagName, index1 ?, index2?);
Like **tagNextrange** but searches backwards from *index1* to *index2* (default 1.0).

\$text->tagRaise(tagName ?, aboveThis?);
Change priority of tag *tagName* so it is just above tag *aboveThis*.

\$text->tagRanges(tagName);
Returns a list describing all character ranges tagged with *tagName*.

\$text->tagRemove(tagName, index1 ?, index2?);
Remove tag *tagName* for all characters in range *index1* to *index2*.

\$text->windowCget(index, -option);
Return current value of *-option* for embedded window at *index*.

\$text->windowConfigure(index ?, -option ?, value ?, -option => value ...?);
Modifies embedded window-specific options for the window at *index*.

\$text->windowCreate(index ?, -option => value ...?);
Create a new embedded window at position *index* with specified options.

\$text->windowNames;
Returns list of names of all windows embedded in widget.

\$text->xview | yview(args);
See Widget Scroll Commands above.

9. Other Standard Widgets

Button

<code>-activebackground</code>	<code>-font</code>	<code>-pady</code>
<code>-activeforeground</code>	<code>-foreground</code>	<code>-relief</code>
<code>-anchor</code>	<code>-height</code>	<code>-state</code>
<code>-background</code>	<code>-highlightbackground</code>	<code>-takefocus</code>
<code>-bitmap</code>	<code>-highlightcolor</code>	<code>-text</code>
<code>-borderwidth</code>	<code>-highlightthickness</code>	<code>-textvariable</code>
<code>-command</code>	<code>-image</code>	<code>-underline</code>
<code>-cursor</code>	<code>-justify</code>	<code>-width</code>
<code>-disabledforeground-padx</code>		<code>-wraplength</code>

`$button->flash;`

Alternate checkbox between active and normal colors.

`$button->invoke;`

Toggle the selection state of the checkbox and invoke the Perl command specified with `-command`, if any.

Checkbox

<code>-activebackground</code>	<code>-font</code>	<code>-pady</code>
<code>-activeforeground</code>	<code>-foreground</code>	<code>-relief</code>
<code>-anchor</code>	<code>-height</code>	<code>-state</code>
<code>-background</code>	<code>-highlightbackground</code>	<code>-takefocus</code>
<code>-bitmap</code>	<code>-highlightcolor</code>	<code>-text</code>
<code>-borderwidth</code>	<code>-highlightthickness</code>	<code>-textvariable</code>
<code>-command</code>	<code>-image</code>	<code>-underline</code>
<code>-cursor</code>	<code>-justify</code>	<code>-width</code>
<code>-disabledforeground-padx</code>		<code>-wraplength</code>

`-indicatoron => boolean`

Whether or not the indicator should be drawn.

`-offvalue => value`

Value given to variable specified with `-variable` option when the checkbox is deselected.

`-onvalue => value`

Value given to variable specified with `-variable` option when the checkbox is selected.

`-selectcolor => color`

Color used to fill in indicator when selected.

`-selectimage => image`

Image displayed in indicator when selected.

`-variable => varRef`

Variable to associate with checkbox.

`$checkbox->deselect;`

Deselect the checkbox.

`$checkbox->flash;`

Alternate checkbox between active and normal colors.

`-separator => string`

Specifies the character to be used as the separator character when interpreting the path-names of list entries. By default the character "." is used.

`-width => width`

Specifies the desired width for the window in characters.

`$hlist->add(entryPath ?, -option => value?);`

Creates a new list entry with the pathname `entryPath`.

`-at => index`

Insert the new list at the position given by position `index`.

`-after => index`

Insert the new list entry after the entry identified by `index`.

`-before => index`

Insert the new list entry before the entry identified by `index`.

`-data => string`

Specifies a string to associate with this list entry.

`-itemtype => imagetext | text | window`

Specifies the default type of display item.

`-state => normal | disabled`

Specifies whether this entry can be selected or invoked by the user.

`$hlist->addchild(parent ?, -option => value?);`

Adds a new child entry to the children list of the list entry `parentPath` and returns the new `entryPath`.

`$hlist->anchorSet(entryPath);`

Sets the anchor to the list entry identified by `entryPath`.

`$hlist->anchorClear;`

Removes the anchor, if any.

`$hlist->cget(-option);`

Returns the current value of `-option`.

`$hlist->ColumnWidth(col ?, -char? ?, width?);`

Queries or sets the width of the column `col`.

`$hlist->ColumnWidth(col => ' ');`

An empty string indicates that the width of the column should be just wide enough to display the widest element in this column.

`$hlist->columnWidth(col, width);`

Set column `col` to pixel width `width`.

`$hlist->columnWidth(col, -char => nchars);`

The width is set to be the average width occupied by `nchars` number of characters of the font specified by the `-font` option.

`$hlist->configure(?-option => value?);`

Query or modify the configuration options of the widget.

`$hlist->deleteAll;`

Delete all list entries.

`$hlist->deleteEntry(entryPath);`

Delete list entry `entryPath`.

`$hlist->deleteOffsprings(entryPath);`

Delete all offsprings of `entryPath`.

`$hlist->deleteSiblings(entryPath);`

Delete all the list entries that share the parent `entryPath`.

`$fselect->Show;`

Map the FileSelect toplevel window; return selection or undef.

HList

<code>-background</code>	<code>-height</code>	<code>-selectborderwidth</code>
<code>-borderwidth</code>	<code>-highlightbackground</code>	<code>-selectforeground</code>
<code>-cursor</code>	<code>-highlightcolor</code>	<code>-setGrid</code>
<code>-exportselection</code>	<code>-highlightthickness</code>	<code>-takefocus</code>
<code>-font</code>	<code>-relief</code>	<code>-width</code>
<code>-foreground</code>	<code>-selectbackground</code>	<code>-xscrollcommand</code>
		<code>-yscrollcommand</code>

`-browsecmd => callback`

Specifies the callback invoked when the user browses through the entries in the HList widget.

`-columns => integer`

Specifies the number of columns in this HList widget.

`-command => callback`

Specifies the callback invoked when the user selects a list entry.

`-drawBranch => boolean`

True if a branch line should be drawn to connect list entries to their parents.

`-header => boolean`

Specifies whether headers should be displayed (see the `header` method below).

`-height => integer`

Specifies the desired height for the window in number of characters.

`-indent => pixels | textchars`

Specifies the amount of horizontal indentation between a list entry and its children.

`-indicator => boolean`

Specifies whether the indicators should be displayed. See the `indicator` methods below.

`-indicatorCmd => callback`

Specifies the callback executed when the user manipulates the indicator of an HList entry. The default callback is invoked with one implicit argument, the `entryPath` of the entry whose indicator has been triggered.

`-itemType => imagetext | text | window`

Specifies the default type of display item.

`-selectBackground => color`

Specifies the background color for the selected list entries.

`-selectBorderWidth => width`

Specifies the width of the 3-D border to draw around selected items.

`-selectForeground => color`

Specifies the foreground color for the selected list entries.

`-selectMode => single browse multiple extended`

Specifies one of several styles for manipulating the selection.

`-sizeCmd => callback`

Specifies the callback executed whenever the HList widget changes size.

`$checkbox->invoke;`

Toggle the selection state of the checkbox and invoke the Perl command specified with `-command`, if any.

`$checkbox->select;`

Select the checkbox.

`$checkbox->toggle;`

Toggle the selection state of the checkbox.

Frame

<code>-borderwidth</code>	<code>-highlightbackground</code>	<code>-relief</code>
<code>-cursor</code>	<code>-highlightcolor</code>	<code>-takefocus</code>
<code>-height</code>	<code>-highlightthickness</code>	<code>-width</code>

`-background => color`

Same as standard expect it may be the empty string to preserve colormap.

`-class => name`

Class name to use in querying the option database and for bindings.

`-colormap => colormap`

Colormap to use for the window if different from parent.

`-visual => visual`

Visual info to use for the window if different from parent.

Label

<code>-anchor</code>	<code>-height</code>	<code>-pady</code>
<code>-background</code>	<code>-highlightbackground</code>	<code>-relief</code>
<code>-bitmap</code>	<code>-highlightcolor</code>	<code>-takefocus</code>
<code>-borderwidth</code>	<code>-highlightthickness</code>	<code>-text</code>
<code>-cursor</code>	<code>-image</code>	<code>-textvariable</code>
<code>-font</code>	<code>-justify</code>	<code>-underline</code>
<code>-foreground</code>	<code>-padx</code>	<code>-width</code>
		<code>-wraplength</code>

Menubutton

<code>-activebackground</code>	<code>-foreground</code>	<code>-relief</code>
<code>-activeforeground</code>	<code>-height</code>	<code>-state</code>
<code>-anchor</code>	<code>-highlightbackground</code>	<code>-takefocus</code>
<code>-background</code>	<code>-highlightcolor</code>	<code>-text</code>
<code>-bitmap</code>	<code>-highlightthickness</code>	<code>-textvariable</code>
<code>-borderwidth</code>	<code>-image</code>	<code>-underline</code>
<code>-cursor</code>	<code>-justify</code>	<code>-width</code>
<code>-disabledforeground</code>	<code>-padx</code>	<code>-wraplength</code>
<code>-font</code>	<code>-pady</code>	

`-indicatoron => boolean`

If true then a small indicator will be displayed on the buttons's right side and the default menu bindings will treat this as an option menubutton.

`-menu => pathName`

Pathname of menu widget to post when button is invoked.

Message

-anchor	-highlightbackground	-relief
-background	-highlightcolor	-takefocus
-borderwidth	-highlightthickness	-text
-cursor	-justify	-textvariable
-font	-padx	-width
-foreground	-pady	

-aspect *integer*

Ratio of text width to text height times 100 to use to display text.

Radiobutton

-activebackground	-font	-pady
-activeforeground	-foreground	-relief
-anchor	-height	-state
-background	-highlightbackground	-takefocus
-bitmap	-highlightcolor	-text
-borderwidth	-highlightthickness	-textvariable
-command	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength

-indicatoron => *boolean*

Whether or not the indicator should be drawn.

-selectcolor => *color*

Color used to fill in indicator when selected.

-selectimage => *image*

Image displayed in indicator when selected.

-value => *value*

Value given to variable specified with **-variable** option when the radiobutton is selected.

-variable => *varRef*

Variable to associate with radiobutton.

\$radiobutton->deselect;

Deselect the radiobutton.

\$radiobutton->flash;

Alternate radiobutton between active and normal colors.

\$radiobutton->invoke;

Toggle the selection state of the radiobutton and invoke the Perl command specified with **-command**, if any.

\$radiobutton->select;

Select the radiobutton.

Scale

-activebackground	-highlightbackground	-repeatdelay
-background	-highlightcolor	-repeatinterval
-borderwidth	-highlightthickness	-state
-cursor	-orient	-takefocus
-foreground	-relief	-troughcolor

Dialog

-borderwidth	-highlightbackground	-relief
-cursor	-highlightcolor	-takefocus
-height	-highlightthickness	-width

-title => *string*

Title to display in the dialog's decorative frame.

-text => *string*

Message to display in the dialog widget.

-bitmap => *bitmap*

Bitmap to display in the dialog.

-default_button => *string*

Text label of the button that is to display the default ring.

-buttons => [*@button_labels*]

A reference to a list of button label strings.

\$dialog->Show(?-global?);

Show dialog and return the selection as a string. The grab is local unless *-global* is specified.

FileSelect

-borderwidth	-highlightbackground	-relief
-cursor	-highlightcolor	-takefocus
-height	-highlightthickness	-width

-width => *width*

Width of file and directory list boxes.

-height => *height*

Height of file and directory list boxes.

-directory => *pathName*

Starting directory path name.

-filelabel => *string*

Label for file entry widget.

-filelistlabel => *string*

Label for file listbox widget.

-filter => *string*

Limit search to the wildcard *string*.

-dirlabel => *string*

Label for directory entry widget.

-dirlistlabel => *string*

Label for directory listbox widget.

-accept => *callback*

Override FileSelect Accept subroutine with your own.

-create => *boolean*

True if it's okay to create directories.

-verify => [*-verifyOptions*]

A list of Perl file test operators and/or *callbacks* to your own verify subroutine. The subroutine is implicitly called with a directory path name and a file (or directory) name, and possibly optional arguments.

-cursor **-highlightcolor** **-takefocus**
-height **-highlightthickness** **-width**

-listwidth => width
 Specifies the character width of the popup listbox.

-variable => varRef
 Where entered value is stored.

-browsecmd => callback
 Specifies a function to call when a selection is made. It is passed the widget and the text of the entry selected. This function is called after *varRef* has been assigned the value.

-listcmd => callback
 Specifies the function to call when the button next to the entry is pressed to popup the choices in the listbox. This is called before popping up the listbox, so it can be used to populate the entries in the listbox.

\$browse->insert(index, string);
 Inserts the text of string at the specified index. This string then becomes available as one of the choices.

\$browse->delete(index1 ?, index2?);
 Deletes items from *index1* to *index2*.

ColorEditor

-borderwidth **-highlightbackground** **-relief**
-cursor **-highlightcolor** **-takefocus**
-height **-highlightthickness** **-width**

-title => string
 Toplevel title, default = ' '.

-cursor=> cursor
 A valid Tk cursor specification (default is *top_left_arrow*). This cursor is used over all ColorEditor hot spots.

-command => callback
 Optional replacement for **set_colors** color configurator.

-widgets => [widgetList]
 A reference to a list of widgets for the color configurator to color.

-display_status => boolean
 True to display the ColorEditor status window when applying colors.

-add_menu_item => itemString
 'SEP' (a separator), or a color attribute menu item.

-delete_menu_item => itemString
 'SEP', a color attribute menu item, or color attribute menu ordinal.

\$cedit->Show;
 Map the Coloreditor toplevel window.

\$cedit->delete_widgets([widgetList]);
 A reference to a list of widgets to remove from ColorEditor's consideration.

-font

-bigincrement => number
 A real value to use for large increments of the scale.

-command => callback
 Specified a Perl command to invoke when scale's value is changed. The scale's value will be appended as an additional argument.

-digits => integer
 An integer specifying how many significant digits should be retained.

-from => number
 A real value corresponding to left or top end of the scale.

-label => string
 A string to display as label for the scale.

-length => size
 Specifies the height (width) for vertical (horizontal) scales.

-resolution => number
 Real value to which scale's value will be rounded to an even multiple of.

-showvalue => boolean
 Whether or not scale's current value should be displayed in side label.

-sliderlength => size
 Size of the slider, measured along the slider's long dimension.

-sliderrelief => relief
 Specify the relief used to display the slider.

-tickinterval => number
 A real value to specify the spacing between numerical tick marks displayed.

-to => number
 A real value corresponding to the right or bottom end of the scale.

-variable => varRef
 Name of a global variable to link to the scale.

-width => width
 Narrow dimension of scale (not including border).

\$scale->coords(?value?);
 Returns x and y coordinates of point corresponding to *value*.

\$scale->get(?x, y?);
 If x, y is given, returns scale value at that coordiante postion. Otherwise, scale's current value is returned.

\$scale->identify(x, y);
 Returns string indicating part of scale at postion x, y. Maybe one of 'slider', 'trough1', 'trough2' or empty.

\$scale->set(value);
 Changes the current value of scale to *value*.

Scrollbar

-activebackground **-highlightcolor** **-repeatdelay**
-background **-highlightthickness** **-repeatinterval**
-borderwidth **-jump** **-takefocus**
-cursor **-orient** **-troughcolor**
-highlightbackgroundrelief

-activerelief => *relief*
Relief to use when displaying the element that is active.

-command => *callbackPrefix*
Prefix of a Perl command to invoke to change the view in the widget associated with the scrollbar.

-elementborderwidth => *width*
Width of borders around internal elements (arrows and slider).

-width => *width*
Narrow dimension of scrollbar (not including border).

Elements: arrow1, trough1, slider, trough2, arrow2

\$scrollbar->activate(?element?);
Display *element* with active attributes.

\$scrollbar->delta(deltaX, deltaY);
Returns fractional position change for slider movement of *deltaX* *deltaY*.

\$scrollbar->fraction(x, y);
Returns a real number between 0 and 1 indicating where the point given by pixel coords *x y* lies in the trough area of the scrollbar.

\$scrollbar->get;
Returns current scrollbar settings as the list *{first last}*.

\$scrollbar->identify(x, y);
Returns name of element under pixel coords *x y*.

\$scrollbar->set(first, last);
Describes current view of associated widget where *first* and *last* are the percentage distance from widget's beginning of the start and end of the view.

Toplevel

-borderwidth **-highlightbackground** **-relief**
-cursor **-highlightcolor** **-takefocus**
-height **-highlightthickness** **-width**

-background => *color*
Same as standard but may be empty to preserve colormap space.

-class => *string*
Class name for the window to be used by option database.

-colormap => *colormap*
Color map to use for window. May be the word **new**, pathname of other toplevel, or empty for the default colormap of screen.

-screen => *screen*
Screen on which to place the window.

-visual => *visual*
Specifies visual to use for window.

10. Perl/Tk Widgets

Here are Tix and user contributed widgets particular to Perl/Tk.

Adjuster

-borderwidth **-highlightbackground** **-relief**
-cursor **-highlightcolor** **-takefocus**
-height **-highlightthickness** **-width**

\$widget->packAdjust(?packOptions?);

If *\$widget* is packed with **-side => left | right** then width is adjusted. If packed **-side => top | bottom** then height is adjusted.

Ballon

-borderwidth **-highlightbackground** **-relief**
-cursor **-highlightcolor** **-takefocus**
-height **-highlightthickness** **-width**

-initwait => *delay*

Milliseconds to wait without activity before popping up a help balloon (default 350 milliseconds). This applies only to the popped up balloon; the status bar message is shown instantly.

-state => **balloon | status | both | none**

Indicates that the help balloon, status bar help, both or none, respectively, should be activated when the mouse pauses over the client widget.

-statusbar => *\$widget*

Specifies the widget used to display the status message. This widget should accept the **-text** option and is typically a Label.

\$balloon->attach(-options);

Attaches the widget indicated by *widget* to the help system.

-statusmsg => *statusMessage*

The message shown on the status bar when the mouse passes over this client. If not specified but **-msg** is specified then the message displayed on the status bar is the same as the argument for **-msg**.

-balloonmsg => *balloonMessage*

The message displayed in the balloon when the mouse pauses over the client. As with **-statusmsg** if this is not specified, then it takes its value from the **-msg** specification. If neither **-balloonmsg** nor **-msg** are specified, then an empty balloon will be popped up.

-msg => *defaultMessage*

The catch-all for **-statusmsg** and **-balloonmsg**. This is a convenient way of specifying the same message to be displayed in both the balloon and the status bar for the client.

\$balloon->detach(\$widget);

Detaches the specified widget from the help system.

BrowseEntry

-borderwidth **-highlightbackground** **-relief**