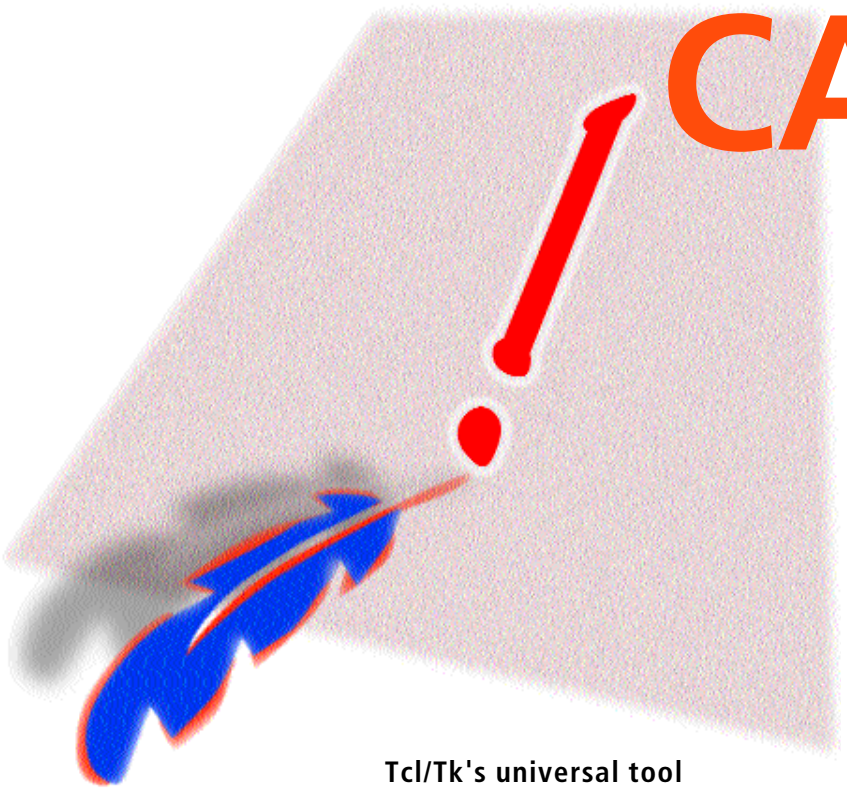
Universal tool
A WIDE CANVAS

CARSTEN ZERBST



Tcl/Tk's universal tool for 2D graphics is the Canvas widget. It can represent, manipulate and animate simple and complex graphic objects and enable mouse access.

Most user interfaces can be designed using the usual GUI elements like buttons, entry fields and labels. But some tasks go beyond the limits of these widgets. Whether you want to illustrate a factory's material flow, conjure up the romance of steam trains or just want to give your program an interface that is a bit out of the ordinary, look no further than the Canvas widget.

Canvas offers you a virtual canvas that allows you to display objects such as lines, surfaces, bitmaps and fonts. However, possibilities are not limited to display. Objects can also be manipulated, either in a pre-programmed process or interactively by the user. Canvas can also export the finished masterpiece as postscript. Due to its functional range, it forms the basis of many drawing programs (Impress for example).

The source text in Listing 1 demonstrates some of the Canvas widget's features. As in the last instalment of Tcl, we're dealing with the representation and manipulation of type 1 fonts (postscript fonts). The screenshot in Figure 1 is taken from the detailed version of the program, which draws an exclamation mark consisting of an upright bar (actually a polygon) and a spherical base.

Objects

Firstly, two arrows are going to illustrate the 1000x1000 point design space of type 1 characters. The coordinate axes are pointing upwards and to the right. However, the Canvas widget uses its own coordinate system that cannot be changed. In this, the axes point downwards and to the right, as in X11. It is the programmer's job to create objects in Canvas with coordinates that have been converted – or can convert them using *canvasName scale Element*.

For the first two lines of Listing 1 the y coordinate's sign is simply reversed. A negative scaling factor along the y axis has the same effect for the remaining elements.

New Canvas objects are always created in the same way. The syntax is *canvasName create Type Coordinates Attributes*. Table 1 contains a listing of important attributes.

If the commands from the listing are entered one at a time (such as in Tkcon), not much of the two lines is visible at first. The Canvas widget must first be told which region to display. This is done with the *-scrollregion* option. Considering the size and resolution of today's monitors, the display could be smaller. This is achieved by simply scaling down the arrows (in our example by a factor of *0.4)*.

Table 1: Attributes of Canvas Objects

Tags	List of tags
-fill Colour	colour of lines and surfaces
-outline Colour	colour of outlines
-width Width	width of lines and outlines
-dash Pattern	line pattern, e.g. "-.."
-stipple Pattern	bitmap for shading, e.g. gray25
-arrow Where	arrow point (none, first, last, both)
-smooth Boolean	splines instead of polygon

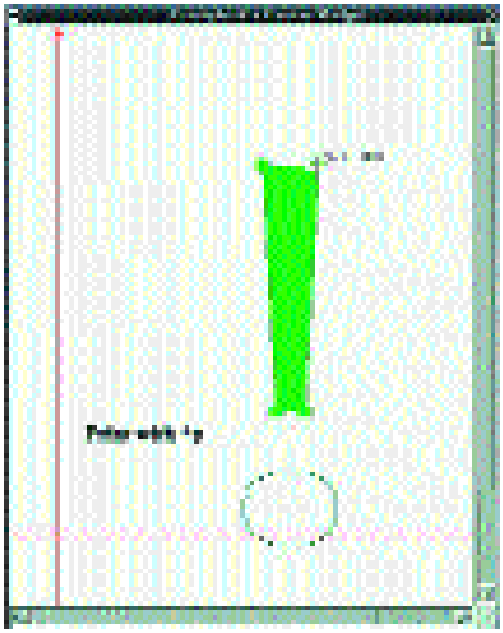


Figure 1: The complete example draws an exclamation mark with several check points and displays the current mouse position

Manipulated objects

Each manipulation must specify which object it is referring to. This can be done in two ways: using IDs or tags (markings). Each object is assigned a unique ID when it is created. This could be stored for later use, but it's easier to use tags. Each object can contain one or more tags, by which it can be addressed instead of by its ID. Two tags always exist – *all* for all objects and *current* for the most recent object.

After scaling down, the arrows should be completely visible. However, the scroll region still has to be adapted to the new dimensions. Instead of specifying the area directly as before, we will use the command *bbox*. It determines the region within Canvas that is occupied by objects.

We want to create an exclamation mark as a simple outline. The widget itself can deal with scaling and reflection in the y direction. Like the arrows before, the bar is scaled down as soon as it is created. A negative scaling factor along the y axis takes care of reflection. Scaling only ever applies to coordinates – line width or text size are not affected. In our example, one corner of the bar still contains a little square.

Spherical objects

After the bar, it's the turn of the base. It does not simply consist of a circle, but has a more complex shape formed of several curves. In order to be able to represent curves instead of straight lines, the option *-smooth true* exists for lines and polygons. The display uses splines, which smooth the transition between two consecutive line segments. To introduce a bend into a spline curve, the bending point must be contained twice in the list of coordinates. Type 1 fonts use bezier curves, defined

Listing 1: Canvas widget with some objects

```
canvas .c -width 400 -height 500 -bg white \
 -xscrollcommand [list .hscroll set] \
 -yscrollcommand [list .vscroll set]
scrollbar .hscroll -orient horizontal -command [list .c xview]
scrollbar .vscroll -orient vertical -command [list .c yview]

grid .c .vscroll -sticky news
grid .hscroll -sticky ew
grid columnconfigure . 0 -weight 1
grid rowconfigure . 0 -weight 1

# Two lines with arrow points
.c create line -100 0 1100 0 -fill red -arrow last -tags coord
.c create line 0 400 0 -1100 -fill red -arrow last -tags coord
.c configure -scrollregion {-100 -1100 400 1100}

# Scaling
set scale 0.4
.c scale coord 0 0 $scale $scale
.c configure -scrollregion [.c bbox all]

# Bar
set item [.c create polygon 440 800 560 800 530 270 470 270 \
 -fill seagreen2 -outline seagreen4 -tags outl1 ]
.c scale $item 0 0 $scale -$scale

# A node
.c create rectangle 430 790 450 810 -tags {node outl1} \
 -fill seagreen2 -outline seagreen4
.c scale node 0 0 $scale -$scale

# A curve
set item [.c create line 400 60 400 100 450 140 500 140 \
 -smooth true -fill seagreen4 ]
.c scale $item  0 0 $scale -$scale

# Some text
.c create text 100 -100 -text "Print with ^p"

# Output of coordinates
proc coords {x y} {
 set x [expr {[.c canvasx $x]/$::scale}]
 set y [expr {-[.c canvasy $y]/$::scale}]
 puts stdout "x: $x\ty: $y"
}
bind .c <Motion> {coords %x %y}

# Selection
proc deselect {} {
 .c itemconfigure outl1 -outline green4 -fill green2
 .c bind outl1 <Button-1> select
}
proc select {} {
 .c itemconfigure outl1 -fill firebrick1 -outline firebrick4
 .c bind outl1 <Button-1> deselect
}
.c bind outl1 <Button-1> select

# Printing the visible region
proc printing {} {
 puts stderr "Print postscript canvas.ps"
 set fd [open canvas.ps w]
 puts $fd [.c postscript ]
 close $fd
}
bind . <Control-p> printing
```

News from the Tcl world

Jeffrey Hobbes has published a new version of Tkcon, a tool for every Tcl developer. It would be unfair on Tkcon to describe it simply as a substitute for the normal Tcl command line interface. Tkcon offers the same usability when working with Tcl that you will be accustomed to from Tcsh or Bash. Tkcon automatically completes file names as well as Tcl commands and variables. In addition, it has other features that you will be familiar with from fully-fledged editors, such as syntax highlighting and display of bracket levels. You can browse through name spaces with ease, extensions installed in the system can be loaded at a mouse-click, and much more besides. Tkcon makes working with Tcl even more fun.

Tkcon as universal tool

Tkcon not only offers valuable support when trying out new things but also helps with debugging. It is even ideally suited to writing applications, as it can display individual variables or load improved source text at runtime.

The Tcl extension Snack is undergoing quite a bit of development. This is nothing to do with fatty foods rich in carbohydrates, but rather with sound. Language researchers at the Royal College of Stockholm have created a tool that can deal with many sound processing tasks. Snack can record and play sounds, edit and distort them and carry out further processing. If you're planning on dissecting your MP3 files, you may as well do it with Snack. Wavesurfer (see Figure 2), a handy program for editing audio files, uses Snack. The way to your own MP3 player has never been as easy as with this extension.

Reading matter

If you'd like to see what others are getting up to with Tcl/Tk, we'd recommend a look at the pages of the 2nd European Tcl/Tk User Meeting. There is a wide range of papers, the focus this year was on the use of Tcl on the Web. Even though not nearly as much fuss is made about Tcl as about some of its alternatives, Tcl is working behind the scenes of AOLserver and Vignette's Story server, both of which are hardly the smallest in their field. Fringe areas such as the coupling of COBOL with Tcl are examined, along with the application of Tcl for game control or as a testing tool.

Figure 2: Wavesurfer not only plays and processes audio files, but also displays wave forms graphically. This picture shows a WAV file

Interactive objects

To be able to select or move objects with the mouse, they must first react to it. The command *canvasName bind TagOrID Event Command* is used to instruct one or more Canvas objects to react to a specified event. In our example, when you click on the bar, it changes colour.

Lastly, we want to output the whole thing as postscript. Using *canvasName postscript,* this is no problem either. If no printing area is specified, the output only contains the visible Canvas region. Otherwise, the required section has to be specified. When using texts in Canvas, it is advisable to remember that there are often more fonts installed than the printer will recognise. Either limit yourself to the 35 standard fonts or embed the additional fonts in the postscript file at a later date.

How many dimensions?

The Canvas widget offers a lot of functionality for 2D graphics, either purely for display purposes, for user interfaces or for creating graphics with a lot of interactions. Help is also available. For instance, Pstoedit can prepare many postscript files so that they can be represented with Canvas. Gnuplot can output its graphics directly in Canvas widget format.

If, however, you're thinking more in terms of 3D for graphics, Canvas won't make you happy. Depending on your exact requirements, the OpenGL widget Togl may be better suited, or VTK for processing and representing scientific data, or the game engine Nebula Device.

After this rather picture-heavy instalment, the next issue of Tcl will describe how to design a really user-friendly Tcl/Tk application. ■

Info

Tkcon: http://tkcon.sourceforge.net
Snack: http://www.speech.kth.se/snack/
2nd European Tcl/Tk User Meeting: http://www.tu-harburg.de/skf/tcltk
AOLserver: http://aolserver.com
Wiggles: http://www.wiggles.com
Animated steam loco: http://mini.net/cgi-bin/wikit/1329.html
Xtcc: http://www.tu-harburg.de/skf/tcltk/papers2000/xtcc.pdf
Impress: http://www.ntlug.org/~ccox/impress/index.html
Pstoedit: http://www.geocities.com/SiliconValley/Network/1958/pstoedit/
Togl: http://sourceforge.net/projects/togl/
VTK: http://www.kitware.com
Nebula Device: http://www.radonlabs.de

■

The author

Carsten Zerbst is a member of staff at Hamburg-Harburg Technical University. Apart from researching service integration on board ships, he investigates Tcl in all its forms.

by two nodes and two check points. The example for the first segment only uses four points.

Work with Canvas often requires the mouse position. The location within the system of screen coordinates is of less interest than the Canvas position. The commands *canvasx* and *canvasy* convert the position, taking the current scroll position into account, but not scaling. The *coords* function divides the coordinates by the scaling factor and then outputs the converted coordinates on the command line.