Name

tkwintrack — track and outline windows in a Tk GUI, and display their pathname

Synopsis

```
package require tkwintrack
```

Description

This reference page corresponds to tkwintrack version 2.0

The package *tkwintrack* is a little inspection tool for developers of Tk GUI's. It extends the interpreter of a Tk application with the machinery to track the individual windows (widgets) in a GUI using the mouse pointer. Upon detection of a window, its outline is highlighted and its pathname is displayed in a small separate widget: the *pathname widget*. Window pathnames displayed in the pathname widget may be copied to the clipboard using a keyboard shortcut.

After loading the package into the client application, tkwintrack starts in configuration mode: a dialog is displayed which lets the user adjust appearance and behaviour of the package's functionality (see also the section Configuration). After clicking the *Continue* button in this dialog, tkwintrack enters detection mode.

Detection mode

In detection mode, the pathname widget appears on screen and the user may track windows using the mouse pointer. While moving the mouse around inside the GUI, individual windows are detected, highlighted and their pathname displayed in the pathname widget. When the mouse pointer is over the pathname widget, the following mouse button actions are available:

- right-clicking opens the configuration dialog. This temporarily turns off detection mode.
- pressing and holding the left mouse button allows dragging of the pathname widget.

The following functionalities are accessible through keyboard shortcuts, irrespective of the position of the mouse pointer or the pathname widget:

- Toggling detection mode off/on. Toggling detection mode off withdraws the pathname widget and highlighted window outlines from the screen, thus letting the client GUI resume its unaffected appearance. Default keyboard shortcut is **Alt+7**.
- Copying the pathname of the currently detected window to the clipboard. Default keyboard shortcut is **Alt+8**.
- Recalling the pathname widget.

  Recalling the pathname widget is useful in case the pathname widget was configured to not follow the mouse pointer, and the pathname widget was dragged away from the active area of the client application (or conversely). Once that happened, the pathname widget can't be reached anymore with the mouse pointer unless a window in the client application is moved towards the pathname widget. The recall keyboard shortcut obviates this awkward action: pressing it moves the pathname widget from anywhere on the screen to the location of the mouse pointer. Default keyboard shortcut is **Alt+9**.

The default key sequences for the keyboard shortcuts are rather uncommon. This minimizes the probability of interfering with the bindings of the client application. Please see the section “Configuration mode” for how to configure them differently.

Configuration mode

Clicking the right mouse button when the mouse pointer is over the pathname widget, displays the configuration dialog. It allows the user to adjust appearance and behaviour of various functionalities of the package. The following options are available:

Outline highlighting

- Outline color
  The color used for the outline drawn around a tracked window.
- Outline width
  The width of the outline drawn around a tracked window. Default is 2 pixels.

Pathname widget

- Background color
  The background color (fill) of the pathname widget.
- Foreground color
  The color of the text and border of the pathname widget.
- Follow mouse
  This setting determines whether the pathname widget follows the mouse pointer when a new window is detected. Default is on.

Keyboard shortcuts

Moving the mouse pointer over the value for a keyboard shortcut (the key sequence) activates key sequence selection mode, and the key sequence becomes highlighted. While the value is highlighted, the user can define a new key sequence for the shortcut by pressing keys. The shortcut may exist of a single regular key, optionally preceded by one or two modifier keys (Control, Shift, Alt, Meta, Mod1 through Mod5). As soon as a valid key sequence is detected, it is accepted and the key selection mode is deactivated.

Tkwintrack remembers the settings for up to 10 different client applications. Upon loading the package in a client application, tkwintrack automatically applies the settings as they were in the previous session for that specific client application. The setting for *follow mouse* is excluded from this memory function: the follow mouse mode is always enabled when tkwintrack starts. If the limit of 10 client applications is exceeded, tkwintrack discards the oldest settings.

Requirements

Tkwintrack has been tested on Linux and MS Windows operating systems. It is possible that tkwintrack runs without any problem on the macOS operating system, but this has not been tested, and therefore macOS is not officially supported. Running the test script on macOS is expected to reveal whether there are issues on that platform.

Tkwintrack requires Tk 8.5 or newer. Furthermore, it requires the windetect library 1.0.

Limitations and known issues

Tkwintrack relies on a client application with a regular Tcl/Tk interpreter, offering standard commands and Tk bindings as provided by the Tcl/Tk language. Tkwintrack doesn't cope with situations where standard Tcl/Tk commands were removed or renamed, and it relies on the binding tag *all* being associated with each widget in the client application.

Specific limitations exist for the following functional areas of tkwintrack:

Outline highlighting
The outline is a composed widget that consists of four lines which are drawn just inside a tracked window. Inside these lines the mouse pointer is insensitive to the client GUI, which causes the following oddities:

- Bindings to mouse button actions on the tracked window (defined by the client application) do not work;
- The mouse cursor may be different from the cursor which the client application configured for the tracked window;
- Very small windows that are completely covered by the outline, cannot be reached or detected if the mouse pointer comes from inside the currently tracked window.

For these reasons, it is advised to choose the outline width as small as acceptable.

Keyboard shortcuts
If the client application loses input focus, keyboard shortcuts for tkwintrack don't work. Therefore, choices with respect to the **focus** command, made when programming the client application, are relevant for this functionality of tkwintrack.

Programming constructs that reenter the event loop
Programming constructs in tkwintrack that reenter the event loop may interfere with the order of execution of actions scheduled by the client application. During the lifetime of the running program, there are a few occasions where tkwintrack reenters the event loop. All except one of these are incidental. Specifically:

- When the pathname widget is created **update idletasks** is called. This happens once during the run time of the program, at program initialization.
- Each time when the pathname is copied to the clipboard **update idletasks** is called.
- Each time when the configuration dialog is (re)displayed, one call to **update idletasks** and one call to **tkwait visibility** are made.
- Each time when an invalid keyboard shortcut is entered in the configuration dialog, **update idletasks** is called.
- If the pathname widget follows the mouse, **update idletasks** is called each time when a new window is detected. This is the only case that occurs while operating the mouse to track windows, and it can be prevented by configuring tkwintrack to not follow the mouse.

Test script

Tkwintrack comes with a test script `tkwintrack.test`, which is invoked as follows:

```
tclsh tkwintrack.test ?option value? ?option value? ...
```

or, on unix-derived operating systems, if the script file has been made executable:

```
tkwintrack.test ?option value? ?option value? ...
```

There are two types of option-value pairs that the command line accepts:

- configuration options for the Tcl test harness tcltest. These are passed on to tcltest. Please see the documentation for package tcltest for details.
- configuration options specific to testing of tkwintrack. Currently, there is just one such option: *`-wtune`*.

The function of this option is to adjust waiting times, used by various tests in the test script. Many tests have a stage where the redrawing of the screen needs to be completed before proceeding. Note that the test script runs without any user-interaction, and the commands **tkwait visibility** or **update idletasks** cannot always be used, or are to no avail. In such situations, tkwintrack relies on an estimate of the time needed for the screen update. This is awkward because the time required for a screen update to complete depends on system performance, something that can change even while tests are running. The estimated waiting times are very generic and course-grained. Although they tend to stay on the safe side, they may not meet the needs in specific circumstances, especially on less performant systems.

To address this situation, the waiting times for screen updates have been made adjustable. Adjusting is done by using the option *`-wtune numval`*, which affects the waiting times for all tests in the test script in the same way. The default value for *`numval`* is 1.0. The higher the value for *`numval`*, the longer the waiting time.

If tests are failing because of waiting times that are too short, it is advised to retry the script execution with a higher value for *`numval`*. Typical symptoms of tests failing for this reason are background errors and hangs (the event loop stalls).

Author

Erik Leunissen

See also

bind(n), bindtags(n), focus(n), tcltest(n), tkwait(n), update(n), windetect(n)

License