NAME

shape − Set/update/query shaped window information

SYNOPSIS

package require shape 0.4

shape *option window* ?*arg ...*?

DESCRIPTION

The **shape** command is used to manipulate windows so as to make them have non-rectangular shapes. It also permits the querying of the current status of windows to see what shape they have. The **shape** can have any of several forms, depending on the *option* argument:

shape bounds *window* ?*kind*?

Returns the bounding box of the shape of *window*, or an empty string if the window does not currently have a shape set for it (i.e. it is rectangular.) The *kind* option allows the selection of which shape to retrieve the bounding box of. The **-bounding** option indicates that the bounding shape of the window is to be queried (the default.) The **-clip** option indicates that the clipping shape of the window is to be queried.

shape get *window* ?*kind*?

Returns a list of rectangles describing the shape of *window*. If *window* currently has no shape set for it, this command returns the Xserver's current idea of the rectangular shape of *window* which might or might not be ‘‘sensible’’. The *kind* option allows the selection of which shape to retrieve the shape of. The **-bounding** option indicates that the bounding shape of the window is to be queried (the default.) The **-clip** option indicates that the clipping shape of the window is to be queried.

shape offset *window* ?*kind*? *xOffset yOffset*

Moves the shape of *window* by *xOffset* pixels right and *yOffset* pixels down, relative to the overall position of *window*. The contents of *window* do not move.

The *kind* option allows the selection of which shape to move. The **-bounding** option indicates that the bounding shape of *window* is to be moved. The **-clip** option indicates that the clipping shape of *window* is to be moved. The **-both** option (the default) moves both the bounding and the clipping shapes of *window*.

shape set *window* ?*kind*? ?-*offset xOffset yOffset*? *sourcekind* ?*arg ...*?

Sets the shape of *window* to the shape specified by the source described by *sourcekind ...*, possibly pre-offset by (*xOffset*,*yOffset*), much as in the **shape offset** command. Shape sources are described in ‘‘SHAPE SOURCES’’ below.

The *kind* option allows the selection of which shape to set. The **-bounding** option indicates that the bounding shape of *window* is to be set. The **-clip** option indicates that the clipping shape of *window* is to be set. The **-both** option (the default) sets both the bounding and the clipping shapes of *window*.

shape update *window operation* ?*kind*? ?*-offset xOffset yOffset*? *sourcekind* ?*arg* ...?

Updates the shape of *window*, applying a shape using set operation *operation*. The shape applied is specified by the source described by *sourcekind ...* and is possibly pre-offset by (*xOffset*,*yOffset*), much as in the **shape offset** command. Shape sources are described in "SHAPE SOURCES" below.

The *kind* option allows the selection of which shape to set. The **-bounding** option indicates that the bounding shape of *window* is to be set. The **-clip** option indicates that the clipping shape of *window* is to be set. The **-both** option (the default) sets both the bounding and the clipping shapes of *window*.

The following set operations (and unique abbreviations of them) are supported:

set (aliases: **:=**, **=**)

The shape specified by the source becomes the shape of *window*.

union (aliases: **+=**, **||**)

The new shape of *window* consists of all areas specified in either the old shape of *window*, or by the source.

intersect (aliases: ***=**, **&&**)

The new shape of *window* consists of all areas specified in both the old shape of *window*, and by the source.

subtract (aliases: **-=**)

The new shape of *window* consists of all areas specified by the old shape of *window* that are not specified by the source.

invert (no aliases)

The new shape of *window* consists of all areas specified by the source that are not specified by the old shape of *window*.

shape version

Return the version number of the non-rectangular window extension installed on the X server that serves the screen that **.** is on. Note that this is *not* the version number of the Tcl/Tk extension, which is reported in the (global) variables **shape_version** and **shape_patchLevel**.

This returns a dummy value on Windows.

SHAPE SOURCES

There are six types of shape sources, though they are not universally supported:

bitmap *bitmap*

This uses the outline of *bitmap* (which may be any string acceptable to **Tk_GetBitmap**) as a shape source. The same information is supplied to operations involving bounding and clipping shapes.

This operation is not supported on Windows.

photo *photoImageName*

This uses (with a suitably patched Tk core) the transparency information from the given photo image (specifi ed using any string acceptable to **Tk_GetPhoto**). The same information is supplied to operations involving bounding and clipping shapes.

rectangles *rectangleList*

This uses the union of the rectangles in *rectangleList* as a source of shape information (the same information is supplied to operations involving bounding and clipping shapes). Each rectangle is described a list of four numbers *x1*, *y1*, *x2* and *y2* where (*x1*,*y1*) is the top-left corner of the rectangle, and (*x2*,*y2*) is the bottom-right corner of the rectangle.

reset This resets the shape of the window the operation is being applied to to a rectangle with the dimensions of the window (or at least, the dimensions that the Xserver thinks that the window has.) This operation is only valid as part of the **shape set** command. It ought to mark the window as not being shaped at all, but there is no protocol for this in version 1.0 of the X non-rectangular window extension, making it an impossible operation.

text *string font*

This uses an image of *string* drawn in *font* as a source of shape information (the same information is supplied to operations involving bounding and clipping shapes) where *font* is any string acceptable to **Tk_GetFont**. Note that only spaces are supported as whitespace characters in *string*, and that the effects of using newline and tab characters is undefi ned.

This operation is not supported on Windows.

window *window*

This uses *window* as a source of shape information, with the bounding shape of *window* being used to supply bounding information, and the clipping shape of *window* being used to supply clipping information.

KINDS OF SHAPE

Every window has two kinds of shape; their **bounding** and their **clipping** shapes. The **bounding** shape of a window describes the area covered by that window, and the **clipping** shape of a window describes the part of the window occupied by the contents of that window. The area of the window contained in the bounding area of the window but not in the clipping area of the window is (X) border of the window. This border is not useful under Tk (which manages all widget borders internally,) so the default kinds that the various operations work on are such that read-only operations work by default on the bounding shape (via the **-bounding** option) and read-write operations work by default on both the clip and the bounding shapes (via the **-both** option.) The **-clip** option is unlikely to be useful in normal operation.

Windows only supports one kind of shape, so the difference between clipping and bounding regions is ignored.

CAVEATS

When setting the shape of a toplevel window, make sure that you perform an **update idletasks** between the creation of the toplevel and applying a **shape set** or **shape update** operation to it. This is because Tk works by reparenting toplevel windows into a (user-invisible) container, so allowing the introduction of a menubar to windows without having to alter the internal dimensions of the toplevel. Because this is done on an idle handler, you must wait until after that idle handler has executed before applying a reshape operation to a toplevel. Failure to do this will result in the reshaped toplevel being apparently encased in a toplevel window that is unresponsive to all events (like Expose events). While no harm is done to the application, this an undesirable effect that you probably do not want in your application. Sometime, I will have a go at fixing this.

This extension can only handle applications that use a single screen, or at least, it can only handle those that only attempt to use shaped windows on the same screen as **.** Since this is the vast majority of all applications, this should not prove too onerous a restriction for the present. *This might have been fixed. Need to test...*

Updating window shapes can be a very expensive operation, so you are advised to avoid doing it as much as possible (do not use it as a way of doing animation − it does not work well.) Of course, if you need to change the shape, then do so, but you should try to find ways that do it as little as possible due to it being an expensive operation.

Some other X clients can run far slower if they have to interact with shaped windows. The troublemakers are not always the clients that you would expect to cause problems.

DEMO PROGRAMS

All these programs are located in `shape0.4/demos`.

dragger.tcl
This creates a window that when Mouse Button 1 is depressed in, brings up a coloured cursor that (partially) changes colour depending on whether the cursor is over the source window or not.

fancytext.tcl
A fairly simple, but effective, demonstration of how to make a simple piece of text look far more effective using moving lines clipped to the text to suggest the letters without showing them all at once.

fingerprint.tcl
This puts a grubby-looking fingerprint on the screen. The fingerprint can be dragged about the screen (the cursor changes when it is possible to drag the fingerprint). Thanks to John LoVerso `<loverso@opengroup.org>` for this little demo!

TO-DO

I need fix the caveat relating to toplevels and check for and resolve any conflicts with some of the other capabilities of toplevels (especially the **-menu** and **-use** options).

I want to write some code to install the extension.

I *need* to test the code for Windows, but I have no compiler that can target the platform. Also, anyone who can supply code to convert bitmaps or text to regions is welcome to contribute!

I would like input about using non-rectangular windows on other Tk platforms (notably the Mac.)

It would be rather nice to be able to anchor cutouts to corners/edges of a window and have them handle window resizes correctly. Possibly also a scaling operation could be added.

At some point, I ought to look into more complex alpha blending, which some people report that is supported in a few places. It does have the disadvantage of only being able to work on systems with at least 15 or 16 bits per pixel; 8-bpp displays (like the ones I use) simply do not have the hardware to cope with this...

There are probably loads of other things that need doing, but I can’t think of them right now.

AUTHOR

Donal K. Fellows `<fellowsd@cs.man.ac.uk>`

KEYWORDS

shape, window