DS-Animated Cursor Controls Help

Thank you for using the Dave Software Animated Cursor Controls. These controls are the full versions. You are welcome to distribute them to other people, either as the controls or in your programs. When distributing the controls, you must make no charge (including disk charges) for them. The controls can be downloaded from http://www.bigfoot.com/~davesoft/dsaniset.zip

If you find any bugs in the controls, or wish to make suggestions about them, please feel free to email me at davesoft@bigfoot.com

Before reading this document, you may wish to check out the demo program included with the controls. This demonstrates the effects of all the properties. If you find any errors in this documentation, please let me know and I will correct them.

The controls are now at version 1.01

Overview

These controls do two things:

1) They enable you to use Animated Cursors as images on windows that will actually animate
2) They fix an annoying limitation of VB that prevents the use of Colour Static Cursors and, of course, Animated Cursors

These controls have already gone through a vast amount of beta testing (the Cursor component made it to Beta #3!). They should be working with a good degree of stability now. Please note that the techniques used for the Cursor Control are immensely complicated – there may still be mistakes in the code. If you do find a fault, see if it is consistent and then let me know about it and I will do my best to fix it. Hopefully, Microsoft will eventually enable the use of Animated Cursors in VB and so the control will become redundant.

Windows NT Note

The controls behave erratically on Windows NT and the property pages do not function correctly. The problems appear to be linked to the Windows NT use of Unicode strings. Windows 2000 promises to fix this problem by eradicating the cross-platform differences between Windows NT and Windows 95/98. For this reason, I have no plans to make these controls function under Windows NT. They work, but I cannot guarantee that they will function completely.

Properties

Property	Information		
	Type	Default	Reference
Image Control			
Animate	Boolean	True	When True, the image will Animate. When False, only Frame #1 is displayed
Animation	String	N/A	Set this to the filename of the cursor required. The return value of this property should be disregarded. If you wish to send the binary data (for example, if the cursor is in a resource file). Then prefix the data with a “>” and send it as the Filename field.
Artist	String	N/A	Returns the name of the Artist who wrote the cursor or {Undefined} if it was not in the file. Read-only
BackColor	RGB Colour	Light Grey	Sets the Mask colour for the control. All the parts of the cursor that are in the “mask” colour will be

			set to this colour
CursorName	String	N/A	Returns the name of the cursor or {Untitled} if it was not in the file. Read-only
GetData	String	N/A	Technically, a function! Returns a string containing the binary data of the file loaded
LastError	Enum	NoError	NoError – Nothing wrong FileSystem – Error reading the disk or bad filename FileCorrupt – File is not static/animated cursor Unexpected – Unexplained error This value must be read as soon as the call is made. Calls to certain properties will clear the value of this property
OLEDropMode	Enum	aciNone	Determines if the control accepts OLE Dropping
Time	Long	N/A	Returns the length of time, in milliseconds, that one complete cycle takes. Returns –1 when no graphic is loaded
Visible	Boolean	True	Determines if the control is displayed or not
Cursor Control			
Cursors	Class	N/A	See below for details of the CursorCol Class. I recommend that you initialise the control at design time using the Property and do not actually use this property – it's easier!

Events

Only the Image Control has any events. The events are all standard Visual Basic events – please see the Visual Basic documentation for details on these.

Usage

The Image Control should be pretty much self-explanatory from the properties. The Cursor Control requires a bit more understanding. Firstly, only controls that have a MousePointer property **and** an hWnd property can be used by this control. If you want a cursor on a Label control (which, in the case annoyingly, has no hWnd – at other times this fact is handy!!!) then draw a picture box that is the same size, place the label inside it and set the cursor for the Picture Box. This is a quick work around this problem. Also note that, for the cursor to apply, the MousePointer property of the object must be set to "0 – Default" or "1 – Arrow". For some annoying reason known only to Microsoft, the PropertyPage and UserDocument objects do not implement the UserControl.ParentControls property to correctly – I'm afraid that the control does not work on Property Pages and User Documents.

CursorCol

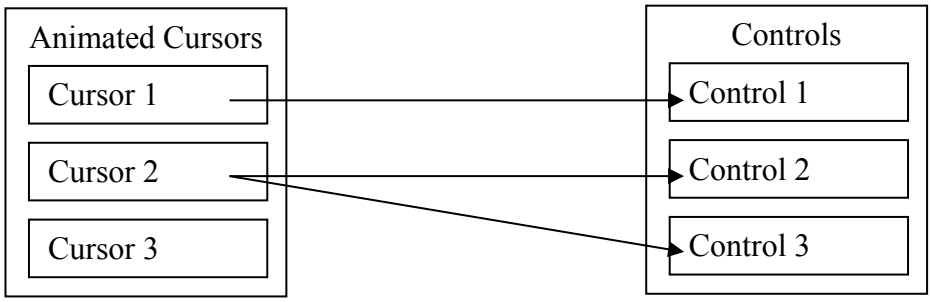
This class is used to manipulate how the control behaves. Think of the Cursors property as the book, and the DSAniCursor control as the reader. They are actually totally independent of each other!

NB Only place one Animated Cursor Control on each form. If a second is placed, the form may not function correctly. A warning message is displayed when you do this. If you persist and distribute an application that has two controls on one form, because of the risk of interference with other applications, this nice little message is displayed…

Is this an incentive to obey the last remark? Yes, I thought so!! This message is no joke – one control on the form will manage all cursors and controls.

Overview

The CursorCol is an object that manipulates two collections. One is a collection of all the objects on the form that the DSAniCursor is on, the second is a collection of all the Animated Cursors that you wish to use. You can only assign one Animated Cursor to a control, but one Animated Cursor may be assigned to more that one control thus…



In this example, Cursor 3 is unused, but still available. Please note that Controls collection is not pre-initialised. You simply add control names to it. Also the names are not checked – you can add any name you like, but the cursor will only be applied if it is the name of a control on the form. Control arrays are

referenced as, for example, BtnData(0). The CursorCol class exposes methods for adding and removing cursors to the control, and for setting the cursor for a control.

Method F = Function, S = Sub	Information	
	Parameters	Reference
S AddCursor	Name (Variant – Collection Key), Filename (String)	Name is the name by which you will refer to the cursor when using it with SetControl. Filename is the filename of the cursor to load. You can send data by prefixing it with the “>” character just as for the Animation property of the Image Control
F ControlCount	Returns Long	Returns the number of controls that have had cursors assigned to them
F ControlData	Control-Name (String). Returns String	Returns the name of the cursor that the control “Control-Name” has been assigned to
F CursorCount	Returns Long	Returns the number of cursors that have been loaded
F CursorData	Name (Variant – Collection Key). Returns String	Returns the binary data of the cursor specified by Name.
F GetLastError	Returns long	Returns the last error. Check as soon as the call is made or the error will be cleared. The error is set 0 after the function is called
S RemoveControl	Name (String)	Removes the cursor applied to a control
S RemoveCursor	Name (Variant – Collection Key)	Removes a cursor from the collection and all of the controls that have that cursor associated with them.
S SetControl	Name (String), Cursor (Variant – Collection Key)	This assigns the loaded cursor “Cursor” to the control “Name”. Name must be the exact name of the control (not case sensitive) or nothing will happen.

Error Codes
0: No Error
1: File not found
2: Disk error
3: Already in collection
4: Data not found
5: Invalid Key (Contained Chr(0) or "{None Set}" was supplied)

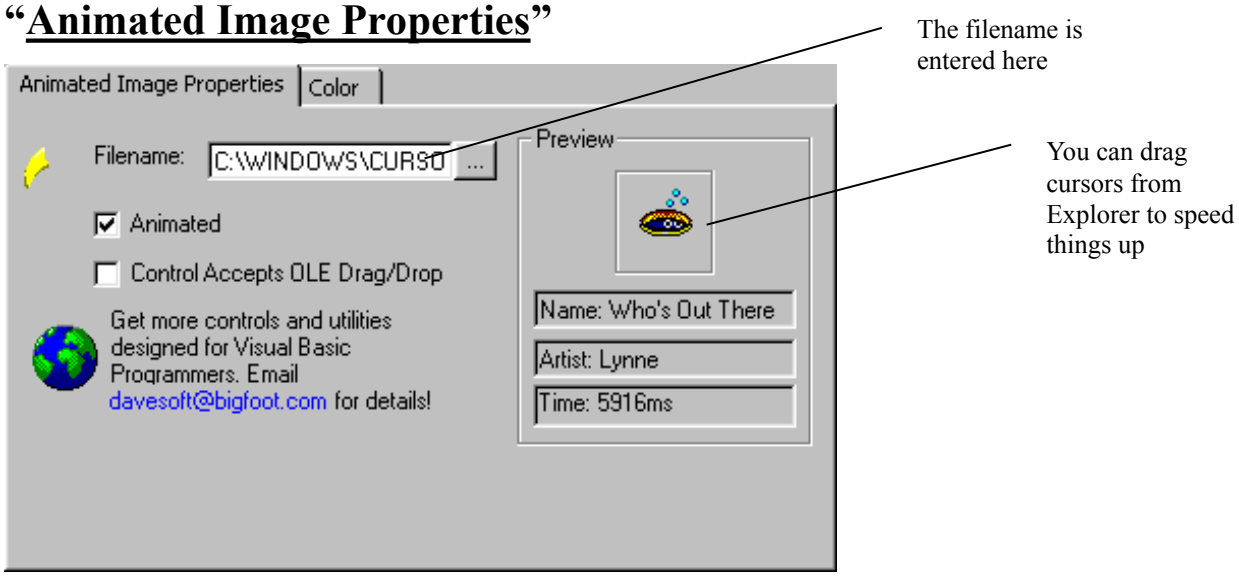
This lists the error codes used by each routine. Note that the CursorCount and ControlCount do not set the GetLastError property. This means that you can still retrieve the error value after calling them. The “exception” codes shown are the only ones that can ever be “thrown” by the routine.

Method	Throws	Clears Last Error
AddCursor	1,2,3,5	Yes
RemoveCursor	4	Yes
CursorCount		No
CursorData	4	Yes
SetControl	4,5	Yes
RemoveControl	4	Yes
ControlCount		No
ControlData	4	Yes

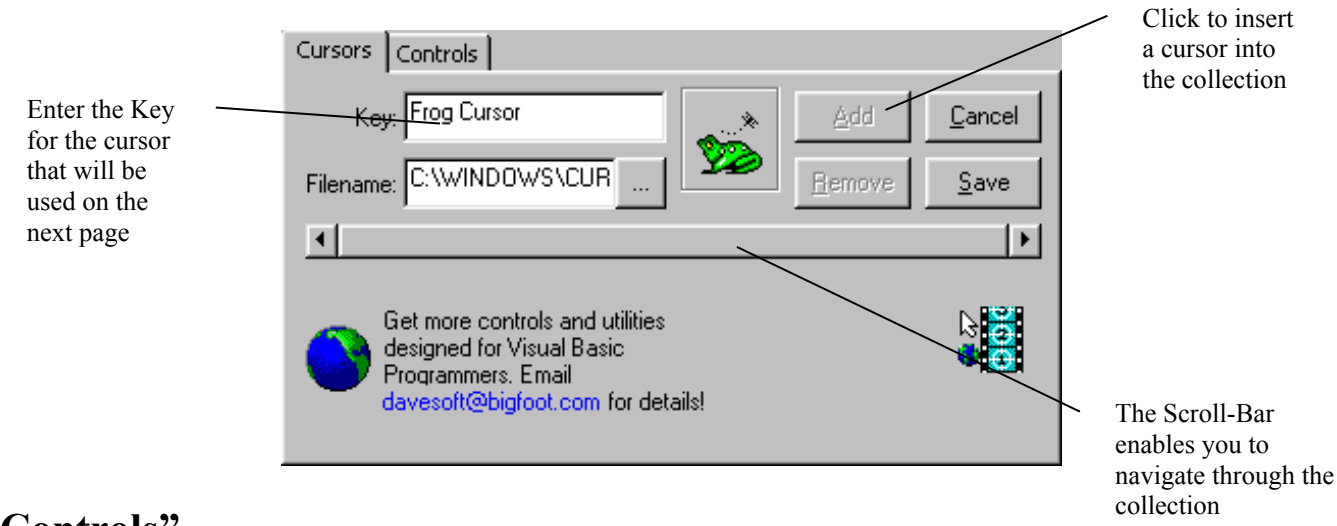
Property Pages

Both controls have two property pages. The Image control has a custom page, "Animated Image Properties", and a standard VB Colour page. The Cursor Control has two custom pages, "Cursors" and "Controls". The VB Colour page is documented in the VB help file. The other three are below…

"Animated Image Properties"



"Cursors"



"Controls"

This property page has the advantage that you can't try and specify a control that doesn't exist!