Help for MouseWheel

Properties Events Methods Frequently Asked Questions

How To Buy This Software

Order Form

Getting Custom Controls Written

Licensing Information

Description

Add IntelliMouse support to your 32-bit applications with the Mabry MouseWheel control. The wheel is supported in Microsoft Office 97 and now it can be supported in your applications.

MouseWheel is a VB5 ActiveX control that allows your programs to take advantage of the powerful scrolling capabilities of the new IntelliMouse by Microsoft. Your users can easily scroll and select data without removing their hand from the mouse.

Uses:

- Select radio buttons and toggle checkboxes.
- Quickly scroll through large amounts of text.
- Set the value of a spin control.
- Provide for both horizontal and vertical scrolling.
- Perfect for any object that utilizes scrolling, such as list boxes, speadsheets, File and Directory lists, etc.
- Scrolling is handled automatically.

Features:

- Since event notifications occur both before and after the user moves the wheel, your application can intercept the wheel movement to enhance or override scrolling.
- Events indicate which mouse button is pressed so your application actions can be tailored for different combinations of wheel movement and button selection.
- A property allows you to specify whether the mouse coordinates are relative to the object that the mouse is over or relative to the entire screen.
- Another property determines whether events are fired for the control under the mousepointer or the control with focus.

MouseWheel can be used in any 32-bit development environment that supports ActiveX controls.

File Name

MWHEEL.OCX

ActiveX Compatibility

VB 4.0 (32-bit), 5.0 and 6.0

ActiveX Built With

Microsoft Visual Basic v5

ActiveX - Required DLLs

VB 5.0 Run-Time DLLs
MSVBVM50.DLL

Distribution Note When you develop and distribute an application that uses this control, you should install the control file into the user's Windows SYSTEM directory. The control file has version information built into it. So, during installation, you should ensure that you are not overwriting a newer

version.

Close

MouseWheel Properties

Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__CoordType__ Property

*__hWndNotify__ Property

*__Present__ Property

*__ScrollLines__ Property

*__ScrollWhich__ Property

Close

MouseWheel Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__AfterMouseWheel__ Event

*__BeforeMouseWheel__ Event

Close

MouseWheel Methods

Methods that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__HorzScroll__ Method

*__Refresh__ Method

Close

Frequently Asked Questions

General Questions

I've installed Mousewheel, but when I attempt to bring up the Custom Control dialog box in VB4 I get a "Object server not correctly registered" message. What's wrong and will this cause problems?

I've installed Mousewheel, but when I attempt to bring up the Custom Control dialog box in VB4 I get a "Object server not correctly registered" message. What's wrong and will this cause problems?

Frequently Asked Questions

This is a known Microsoft problem. ActiveX controls created in VB5 will install a registry key that VB4 doesn't recognize. Each time VB4 (or Access95) searches the registry to display the valid controls it finds this key that it doesn't recognize and shows the message box. You can either ignore this message or go to Microsoft's KnowledgeBase and look up Article ID: Q161827 for instructions on editing the registry to correct this.

How To Buy This Software

CREDITS

MouseWheel was written by Karl Peterson.

CONTACT INFORMATION

Orders, inquiries, technical support, questions, comments, etc. can be sent to mabry@mabry.com on the Internet. Our mailing address/contact information is:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Sales: 1-800-99-MABRY (U.S. Only)
Voice: 360-629-9278
Fax: 360-629-9278
Web: http://www.mabry.com

COST

The price of MouseWheel (control only) is US$25 (US$30 for International orders). The cost of MouseWheel and the Visual Basic source code (of the control itself) is US$75 (US$80 for International orders).

Prices are subject to change without notice.

Printed manuals are available at US$12.50 per copy.

DELIVERY METHODS

We can ship this software to you via air mail and/or e-mail.

Air Mail - you will receive diskettes, a printed manual (if purchased), and printed receipt if you choose this delivery method. The costs are:

US$10.00	US Priority Mail
US$15.00	Airborne Express 2nd Day (US deliveries only)
US$20.00	Airborne Express Overnight (US deliveries only)
US$20.00	Global Priority Mail (Int'l deliveries only; Western Europe, Pacific Rim and Canada only)
US$45.00	International Airborne Express (Int'l deliveries only)

E-Mail - We can ship this package to you via e-mail. You need to have an e-mail account that can accept large file attachments (which includes CompuServe, AOL, and most Internet providers). We will e-mail a receipt to you.

Be sure to include your full mailing address with your order. Sometimes (on the Internet) the package cannot be e-mailed, so we are forced to send it through the normal mails.

CompuServe E-Mail - CompuServe members can use the software registration forum (GO SWREG) to register this package. MouseWheel's SWREG ID number is 15753. The source code version's ID number is 15754. PLEASE NOTE: When you order through SWREG, we send the registered package to your CompuServe account (not your Internet or AOL account) within a few hours.

ORDER / PAYMENT METHODS

You can order this software by phone, fax, e-mail, mail. For your convenience, an order form has been provided that you can print out directly from this help file.

Please note that orders must include all information that is requested on our order form. Your shipment WILL BE DELAYED if we have to contact you for additional information (such as phone number, street address, etc.).

You can pay by credit card (VISA, MasterCard, American Express, Discover, NOVUS), check (U.S. dollars drawn on a U.S. bank), cash, International Money Order, International Postal Order, Purchase Order (established business entities only - terms net 30), or wire transfer.

WIRE TRANSFER INFORMATION

Here is the information you need regarding our account for a wire funds transfer:

Bank Name:	SeaFirst - Stone Way Branch
Bank Address:	3601 Stone Way North Seattle, WA 98103
Bank Phone:	206-585-4951
Account Name:	Mabry Software, Inc.
Routing Number:	12000024
Account Number:	16311706

If you are paying with a wire transfer of funds, please add US$25.00 to your order. This is the fee that SeaFirst Bank charges Mabry Software. Also, please ADD ANY ADDITIONAL FEES THAT YOUR BANK MAY CHARGE for wire transfer service. If you are paying with a wire transfer, we must have full payment deposited to our account before we can ship your order.

MouseWheel Order Form

Use the Print Topic... command from the File menu to print this order form.

Mail this form to:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Phone: 360-629-9278
Fax: 360-629-9278
Internet: mabry@mabry.com
Web: www.mabry.com

Where did you get this copy of MouseWheel?

_____

Name: _____

Ship to: _____

_____

_____

_____

Phone: _____

Fax: _____

E-Mail: _____

Credit Card #: _____ exp. _____

P.O. # (if any): _____ Signature _____

qty ordered ____ REGISTRATION
$25.00 ($30.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $10.00 per order for shipping and handling. Add $12.50 per printed manual.

qty ordered ____ SOURCE CODE AND REGISTRATION
$75.00 ($80.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $10.00 per order for shipping and handling. Add $12.50 per printed manual.

See Also

BeforeMouseWheel Event

See Also

AfterMouseWheel Event
HorzScroll Method

See Also

AfterMouseWheel Event

BeforeMouseWheel Event

See Also

BeforeMouseWheel Event

See Also

AfterMouseWheel Event

BeforeMouseWheel Event

HorzScroll Method

See Also

Refresh Method

ScrollLines Property

See Also

Present Property

ScrollLines Property

See Also

AfterMouseWheel Event
BeforeMouseWheel Event
Present Property
Refresh Method

See Also

AfterMouseWheel Event

BeforeMouseWheel Event

AfterMouseWheel Event

See Also Frequently Asked Questions

Description

Event fires immediately after the wheel has been turned. Scrolling has already occurred, if the control supports scrolling and if scrolling has not been cancelled in the BeforeMouseWheel event.

Syntax

Sub *object*_**AfterMouseWheel(**[*index* **As Integer**,] *hWnd* **As Long**, *delta* **As Long**, *shift* **As Long**, *button* **As Long**, *X* **As Long**, *Y* **As Long)**

The syntax of the **AfterMouseWheel** event has these parts:

Part	Description
object	A MouseWheel control.
index	An integer that identifies a control if it's in a control array.
hWnd	A long integer representing the hWnd property of the form or control receiving notification.
delta	A long integer indicating direction and magnitude of mouse wheel event. This value is either 120 or -120 (Microsoft's documentation notes that they may change this in future releases of the Intellimouse). Positive indicates the wheel was turned forward; negative means the wheel was turned backward.
shift	A long integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.
button	A long integer that identifies the state of the mouse buttons when the wheel was turned. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle or wheel button (bit 2). These bits correspond to the values 1, 2, and 4, respectively.
X	A long integer that specifies the current horizontal location of the mouse pointer. The value is always expressed in pixels, and is either relative to the screen or the notified control, based on the setting of the CoordType property.
Y	A long integer that specifies the current vertical location of the mouse pointer. The value is always expressed in pixels, and is either relative to the screen or the notified control, based on the setting of the CoordType property.

Remarks

The AfterMouseWheel event is most useful when you need to add "scrolling" capabilities to controls that aren't generally used in this manner. For example, to allow users to toggle a checkbox or flip between option buttons with their mouse wheel, use:

```
Private Sub MouseWheel1_AfterMouseWheel(ByVal hWnd As Long, ByValDelta As
Long, ByVal Shift As Long, ByVal Button As Long, ByVal X As Long,ByVal Y As
Long)
    Dim i As Long
    Select Case hWnd
        Case Check1.hWnd
            Check1.Value = Abs(Not CBool(Check1.Value))
        Case Option1(0).hWnd, Option1(1).hWnd, Option1(2).hWnd
            For i = 0 To 2
                Option(i).Value = (Option(i).hWnd = hWnd)
            Next i
    End Select
```

```
End Sub
```

BeforeMouseWheel Event

See Also Frequently Asked Questions

Description

Event fires immediately after the wheel has turned, but before scrolling occurs.

Syntax

Sub *object*_**BeforeMouseWheel(**[*index* **As Integer**,] *hWnd* **As Long**, *delta* **As Long**, *shift* **As Long**, *button* **As Long**, *X* **As Long**, *Y* **As Long**, *cancel* **As Boolean)**

The syntax of the **BeforeMouseWheel** event has these parts:

Part	Description
object	A MouseWheel control.
index	An integer that identifies a control if it's in a control array.
hWnd	A long integer representing the hWnd property of the form or control receiving notification.
delta	A long integer indicating direction and magnitude of mouse wheel event. This value is either 120 or -120 (Microsoft's documentation notes that they may change this in future releases of the Intellimouse). Positive indicates the wheel was turned forward; negative means the wheel was turned backward.
shift	A long integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.
button	A long integer that identifies the state of the mouse buttons when the wheel was turned. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle or wheel button (bit 2). These bits correspond to the values 1, 2, and 4, respectively.
X	A long integer that specifies the current horizontal location of the mouse pointer. The value is always expressed in pixels, and is either relative to the screen or the notified control, based on the setting of the CoordType property.
Y	A long integer that specifies the current vertical location of the mouse pointer. The value is always expressed in pixels, and is either relative to the screen or the notified control, based on the setting of the CoordType property.
cancel	Setting this value to True will prevent any automated scrolling from occurring in the control referenced by the hWnd parameter.

Remarks

The BeforeMouseWheel event is most useful when you need to override or enhance the auto-scrolling capabilities that MouseWheel adds to your forms. For example, you may want to scroll horizontally if the user simultaneously presses and turns the mouse wheel:

```
Private Sub MouseWheel1_BeforeMouseWheel(ByVal hWnd As Long, ByValDelta As
Long, ByVal Shift As Long, ByVal Button As Long, ByVal X As Long,ByVal Y As
Long, Cancel As Boolean)
    Select Case hWnd
        Case Text1.hWnd, ListView1.hWnd
        If Button = vbMiddleButton Then
            Call MouseWheel1.HorzScroll(hWnd, Delta)
            Cancel = True
        End If
    End Select
End Sub
```

CoordType Property

See Also Frequently Asked Questions

Description

Sets or returns whether coordinates are relative to the control or to the screen.

Syntax

object.**CoordType** [= *integer*]

The syntax of the **CoordType** property has these parts:

Part	Description
object	A MouseWheel control.
integer	An integer which determines the origin type of the coordinates.

Remarks

Returns or sets a value that determines whether mouse coordinates passed to the BeforeMouseWheel and AfterMouseWheel events are relative to the screen or relative to the control being notified.

This property can have one of two values:

Constant	Value	Description
ScreenRelative	0	Sets coordinate references relative to screen.
ControlRelative	1	Sets coordinate references relative to notified control.

Coordinates are always returned in pixels.

Data Type

Integer

HorzScroll Method

See Also Frequently Asked Questions

Description

Method used to initiate horizontal scrolling rather than, or in addition to, the default vertical scrolling.

Syntax

object.**HorzScroll***hWnd*,*delta*

The syntax of the **HorzScroll** method has these parts:

Part	Description
object	Required. A MouseWheel control.
hWnd	Required. A long integer that specifies the window handle of the control with which to perform automatic horizontal scrolling. The control must support horizontal scrolling.
delta	Required. A long integer indicating direction and magnitude of mouse wheel event. This value is either 120 or -120 (Microsoft's documentation notes that they may change this in future releases of the Intellimouse). Positive indicates the wheel was turned forward; negative means the wheel was turned backward.

Remarks

Uses the same strategy as default vertical scrolling to scroll a window horizontally. May be used in addition to, or in place of, default scrolling. For example, you may want to scroll horizontally if the user simultaneously presses and turns the mouse wheel:

```
Private Sub MouseWheel1_BeforeMouseWheel(ByVal hWnd As Long, ByValDelta As
Long, ByVal Shift As Long, ByVal Button As Long, ByVal X As Long,ByVal Y As
Long, Cancel As Boolean)
    Select Case hWnd
        Case Text1.hWnd, ListView1.hWnd
            If Button = vbMiddleButton Then
                Call MouseWheel1.HorzScroll(hWnd, Delta)
                Cancel = True
            End If
    End Select
End Sub
```

hWndNotify Property

See Also Frequently Asked Questions

Description

Returns or sets window handle values which are to be monitored in order to insure notification of mouse wheel events. Setting this property for a control prevents an override of the event by the operating system.

Syntax

object.**hWndNotify(** *hWnd* **)** [= *boolean*]

The syntax of the **hWndNotify** property has these parts:

Part	Description
object	A MouseWheel control.
hWnd	A long integer which contains the the .hWnd property of a control.
boolean	A boolean expression which indicates whether or not to monitor this control for mouse wheel events.

Remarks

In general, the MouseWheel control will automatically notify you, and attempt to scroll the appropriate window, whenever the user turns the mouse wheel. This ideal situation is complicated by the fact that Windows NT 4.0 now directly supports roller mice. Under Windows NT 4.0, this property must be set for controls that contain a scroll bar. By setting this property, you will guarantee that mouse wheel messages, for controls that contain scroll bars, will get through to your application. Again, this is only a factor for controls with scrollbars, such as multiline textboxes and listboxes.

The default handling by the operating system may be all you need or want. However, there may be times when notification is required. For example, if you want the ability to Cancel mouse wheel events in BeforeMouseWheel or issue a horizontal scroll in addition to or instead of the default vertical scroll, then you must set this property.

```
Private Sub Form_Load()
    '
    ' Turn on notification for these windows.
    ' Only required in WinNT.
    '
    MouseWheel1.hWndNotify(Text1.hWnd) = CBool(Check1.Value)
    MouseWheel1.hWndNotify(List1.hWnd) = CBool(Check1.Value)
    MouseWheel1.hWndNotify(ListView1.hWnd) = CBool(Check1.Value)
End Sub
```

Data Type

Boolean

Present Property

See Also Frequently Asked Questions

Description

Returns a value indicating whether or not the mouse wheel is enabled.

Syntax

object.**Present**

The syntax of the **Present** property has these parts:

Part	Description
object	A MouseWheel control.

Remarks

This property will report True if the mouse wheel has been enabled in the user's Control Panel. This property will report False if the mouse wheel is not present or if the mouse wheel has not been enabled.

This property is read-only and only available at run-time

Data Type

Boolean

Refresh Method

See Also Frequently Asked Questions

Description

Method used to refresh values returned by the Present and ScrollLines properties.

Syntax

object.**Refresh**

The syntax of the **Refresh** method has these parts:

Part	Description
object	Required. A MouseWheel control.

Remarks

Users are free to change the number of lines to scroll and the enabled state of the mouse wheel at any time. These values are either stored in the registry or may be obtained directly from the driver, depending on the operating system. The Refresh method queries for the latest settings.

ScrollLines Property

See Also Frequently Asked Questions

Description

Returns a value indicating how many "lines" to scroll with each turn of the wheel.

Syntax

object.**ScrollLines**

The syntax of the **ScrollLines** property has these parts:

Part	Description
object	A MouseWheel control.

Remarks

This value reflects the user's Control Panel setting for the number of "lines" (or records) to scroll with each turn of the mouse wheel.

If this value is -1, you should scroll an entire page of data (up or down).

This property is read-only and only available at run-time.

Data Type

Long

ScrollWhich Property

See Also Frequently Asked Questions

Description

Returns or sets a value which determines whether the control with focus or the control under the mouse is scrolled as the user turns the mouse wheel.

Syntax

object.**ScrollWhich** [= *integer*]

The syntax of the **ScrollWhich** property has these parts:

Part	Description
object	A MouseWheel control.
integer	An integer which determines which control receives notifications.

Remarks

Ordinarily, the control with focus receives notifications of scroll events. However, there are times when you may wish scrolling to follow the mouse. There is no set standard behavior, and use of this property is highly application specific.

If this property is set to ControlUnderMouse and the mouse pointer is not over any control, the form itself will recieve notification of mouse wheel events.

This property can have one of two values:

Constant	Value	Description
ControlWithFocus	0	Notifications sent to the ActiveControl.
ControlUnderMouse	1	Notifications sent to the control under the mouse pointer.

Data Type

Integer

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact us at the following:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Phone: 360-629-9278
Fax: 360-629-9278

Internet: mabry@mabry.com

You can also contact Zane Thomas. He can be reached at:

Zane Thomas
Post Office Box 121
Indianola, WA 98342

Internet: zane@mabry.com

Licensing Information

Legalese Version

Mabry Software grants a license to use the enclosed software to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited, and adherence to this requirement is the sole responsibility of the purchaser.

Customer written executable applications containing embedded Mabry products may be freely distributed, without royalty payments to Mabry Software, provided that such distributed Mabry product is bound into these applications in such a way so as to prohibit separate use in design mode, and that such Mabry product is distributed only in conjunction with the customers own software product. The Mabry Software product may not be distributed by itself in any form.

Neither source code for Mabry Software products nor modified source code for Mabry Software products may be distributed under any circumstances, nor may you distribute .OBJ, .LIB, etc. files that contain our routines. This control may be used as a constituent control only if the compound control thus created is distributed with and as an integral part of an application. Permission to use this control as a constituent control does not grant a right to distribute the license (LIC) file or any other file other than the control executable itself. This license may be transferred to a third party only if all existing copies of the software and its documentation are also transferred.

This product is licensed for use by only one developer at a time. Mabry Software expressly prohibits installing this product on more than one computer if there is any chance that both copies will be used simultaneously. This restriction also extends to installation on a network server, if more than one workstation will be accessing the product. All developers working on a project which includes a Mabry Software product, even though not working directly with the Mabry product, are required to purchase a license for that Mabry product.

This software is provided as is. Mabry Software makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

MABRY SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Mabry Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.

To be eligible for free technical support by telephone, the Internet, CompuServe, etc. and to ensure that you are notified of any future updates, please complete the enclosed registration card and return it to Mabry Software.

English Version

We require that you purchase one copy of a control per developer on a project. If this is met, you may distribute the control with your application royalty free. You may never distribute the LIC file. You may not change the product in any way that removes or changes the requirement of a license file.

We encourage the use of our controls as constituent controls when the compound controls you create are an integral part of your application. But we don't allow distribution of our controls as constituents of other controls when the compound control is not part of an application. The reason we need to have this restriction is that without it someone might decide to use our control as a constituent, add some trivial (or even non-trivial) enhancements and then sell the compound control. Obviously there would be little difference between that and just plain reselling our control.

If you have purchased the source code, you may not re-distribute the source code either (nor may you copy it into your own project). Mabry Software retains the copyright to the source code.

Your license is transferable. The original purchaser of the product must make the transfer request. Contact us for further information.

The sample versions of our products are intended for evaluation purposes only. You may not use the sample version to develop completed applications.