ITGraph Custom Control Version 1.2

Properties Events Methods Dialogs Usage New Features

ITGraph is a Visual Basic Custom Control (VBX) that can be used to create, display and manipulate graphical structures from a Visual Basic application. The ITGraph.VBX control provides properties and methods to support interrogation and manipulation of the graph by the application, and events for management of user interaction. In addition to the ITGraph.VBX custom control itself, a Visual Basic module, ITGraph.BAS, is supplied. This file contains Const definitions for the various settings of the ITGraph properties, and should be loaded into any project that uses the ITGraph control.

This help file describes the itgraph properties, events, methods and dialog boxes. The usage section covers basic topics pertaining to ITGraph usage, and provides sample code for common operations. You should scan through these topics first to get an overview of ITGraph capabilities and operation.

The New Features section delineates changes in the control between Versions 1.0, 1.1, and 1.2, and should be scanned if you are currently using an earlier version of ITGraph. Throughout the help file, items that have been added will be marked with (1.1) or (1.2) to indicate the version of ITGraph where the changes were made.

ITGraph Methods

AddItem
Clear
*Move
*Refresh
RemoveItem
*SetFocus
*ZOrder

* Standard Visual Basic Methods

ITGraph Properties

About	DrawScale	ItemGraphicTop (1.1)	PrintToWnd
*Align	*Enabled	ItemGraphicWidth (1.1)	QueryCount
ArrangeMode	FillColor	ItemHeight (1.2)	QueryData (1.1)
AutoArrange	*FontItalic	ItemId	QueryItem
AutoMouseEvent (1.2)	*FontBold	ItemIndex (1.2)	QueryItemHandle
AutoMouseEvents (1.2)	*FontName	ItemLabelAlign (1.1)	QueryState
*BackColor	*FontSize	ItemLabelHeight (1.1)	Redraw (1.1)
*BorderStyle	*FontStrike	ItemLabelLeft (1.1)	RemoveFrom
ConnectFromHandle	*FontUnder	ItemLabelTop (1.1)	RubberBand
ConnectFromIndex	*ForeColor	ItemLabelWidth (1.1)	SaveAs
ConnectionAlign (1.1)	Gap	ItemShape	SelectedIndex (1.2)
ConnectionArrow	GraphicAllowImport (1.1)	ItemTextColor	SelectRectEnabled
ConnectionColor	GraphicCount (1.1)	ItemWidth (1.2)	SelectRectHeight
ConnectionData	GraphicName (1.1)	ItemXpos	SelectRectLeft
ConnectionId (1.1)	GraphicPath (1.1)	ItemYpos	SelectRectTop
ConnectionLabel	Graphics (1.1)	*Left	SelectRectWidth
ConnectionLineWidth (1.1)	GraphicSelect (1.1)	LineWidth	ShapeCount (1.1)
ConnectTo	*Height	List	ShapeName (1.1)
ConnectToHandle	*HelpContextID	ListCount	ShapeSelect (1.1)
ConnectToIndex	*hWnd	LoadFrom	StoreGraphics (1.1)
*CtlName	*Index	*MousePointer	*TabIndex
*DragIcon	IsDirty (1.2)	*Name	*TabStop
DragItems	ItemBorderColor	NewIndex	*Tag
*DragMode	ItemData	*Parent	*Top
DrawArrows	ItemDrawLabel (1.1)	PrintGraph	*Visible
DrawBackLinks	ItemFillColor	PrintHeader (1.1)	*Width
DrawColored	ItemGraphic (1.1)	PrintRectHeight (1.1)	XSpan
DrawConnLabels	ItemGraphicAlign (1.1)	PrintRectLeft (1.1)	XSpace
DrawDir	ItemGraphicHeight (1.1)	PrintRectTop (1.1)	YSpace
DrawHandles	ItemGraphicLeft (1.1)	PrintRectWidth (1.1)	YSpan
DrawItemLabels	ItemGraphicStyle (1.1)	PrintToDC	ZoomSelectRect

* Standard Visual Basic Properties

ITGraph Events

Click	ItemResize (1.2)
DblClick	*KeyDown
DragDrop	*KeyPress
DragOver	*KeyUp
*GotFocus	LineClick
ItemClick	LineDblClick
ItemConnect	*LostFocus
ItemDblClick	MouseDown (1.2)
ItemDrag (1.2)	MouseUp (1.2)
ItemMouseMove	SelectRect

* Standard Visual Basic Events

ITGraph Dialogs

ITGraph Usage

New Features

Several new features have been added to ITGraph Versions 1.1 and 1.2. The following is a summary of the additional capabilities.

Changes in Version 1.2

1. Event Handling is now fully customizable. The AutoMouseEvents and AutoMouseEvent properties can be used to configure how ITGraph responds to mouse events. All three mouse buttons are supported in conjunction with combinations of the Shift, Ctrl and Alt keys.

2. Nearly all the mouse events have been extended to include *Button* (the mouse button pressed to initiate the event), *Shift* (which of the Shift, Ctrl and Alt keys were pressed), and *X* and *Y* (the mouse coordinates at the time of the event, where applicable).

   Note! Users of previous versions of ITGraph will have to update the arguments to these events in their applications.

3. An ItemDrag event has been added to report the dragging of a node.

4. A SelectedIndex property has been added to designate a node as being selected. The selected node is shown with eight sizing handles around it.

5. Nodes can now be individually resized. Two new properties, ItemWidth and ItemHeight, have been added to support this capability. Consequently, changing XSpan and YSpan no longer affects the sizes of existing nodes. A user can resize a node by selecting it and then dragging the sizing handles. An ItemResize event has been added to report the resizing of a node by a user.

6. An ItemIndex property has been added to allow reordering of nodes. The ItemIndex mainly affects the order in which nodes are drawn in the tree layout.

7. The PrintGraph property has changed. Instead of taking a scale factor as an argument, it is now passed a device context in which to print. The graph will be printed according to the current setting of the DrawScale property.

8. MouseDown and MouseUp events have been added. TheMouseDown event is useful for implementing popup menus in your application.

9. ITGraph is now compatible with Microsoft Visual C++ and Borland Delphi. Example programs have been provided for both environments.

Changes in Version 1.1

1. Now supports the import of bitmaps and metafiles as ITGraph "graphics". The Graphics property supports design-time import of graphics. The GraphicCount, GraphicName, GraphicPath, GraphicAllowImport and GraphicSelect properties support run-time import and user selection of graphics. The ItemGraphic, ItemGraphicStyle, ItemGraphicAlign, ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight properties control graphic appearance on a node. The StoreGraphics property determines whether graphics will be saved and loaded with graphs by the LoadFrom and SaveAs properties.

2. Now supports user selection of node shapes via the ShapeSelect property. The ShapeCount and ShapeName properties provide information about the shapes.

3. Now supports text placement and alignment for nodes via the ItemLabelAlign, ItemLabelLeft, ItemLabelTop, ItemLabelWidth and ItemLabelHeight properties. The ItemDrawLabel property controls whether or not a particular node's label will be shown. The ConnectionAlign property controls the alignment of text for connections. Text can be multi-line in nodes and connections, simply by adding carriage returns in the text. In Visual Basic, a carriage return can be added programmatically by appending **chr$(13)** to the node's text.

4. Now includes a ConnectionId property, which is automatically assigned to new connections analagously to the ItemId property.

5. Now includes a QueryData property and four new query types (see QueryState): ITG_QueryMatchConnectionData, ITG_QueryMatchConnectionId, ITG_QueryMatchItemData and ITG_QueryMatchItemId, to support easier access to nodes and connections.

6. The PrintGraph property has been enhanced so that graphs can be printed across multiple pages. The PrintHeader property is used to set the headers and footers for the pages.

7. The capabilities of the PrintToDC and PrintToWnd properties have been enhanced. The PrintRectLeft, PrintRectTop, PrintRectWidth and PrintRectHeight properties define a destination rectangle and the SelectRectLeft, SelectRectTop, SelectRectWidth and SelectRectHeight properties now specify an area in the graph to be printed.

8. The Redraw property has been added. This controls graph updating and is useful for speeding up application manipulation of the graph. Other enhancements reduce the amount of screen updating necessary when changes are made, and drawing and mouse processing has been optimized.

9. A ConnectionLineWidth property has been added, which is used to set line widths for individual connections.

10. The AddItem method now takes an additional optional parameter which specifies the list index of a new node. This is useful if the node order is important. The ITG_ModeTree setting of the ArrangeMode property utilizes the node list order to determine the order of placement in the graph.

KAbout Property

Applies To

ITGraph

Description

Pops up a dialog box with information about the ITGraph Custom Control.

Usage

At design time, click on "..." in the properties window to see the ITGraph about box.

ArrangeMode Property

Applies To

ITGraph

Description

Specifies the mode for laying out the graph's nodes and connections.

Usage

[*form*.]*control*.**ArrangeMode**[= *mode*]

Read/Write at run time and design time.

Setting

The ArrangeMode settings are:

Setting	Description
ITG_ModeHierarchy	Layout the graph in a hierarchical fashion.
ITG_ModeCompact	Layout the graph as compactly as possible while trying to minimize crossings of connections.
ITG_ModeFlowChart	Layout the graph as a flowchart.
ITG_ModeTree	Layout the graph as a tree.

Remarks

The ArrangeMode setting determines two behaviors of the graph. First, when AutoArrange is ITG_AutoArrange, or when it is set to ITG_ArrangeNow, the ArrangeMode determines how the nodes will be laid out. Secondly, ArrangeMode determines how connections will be drawn between nodes. Usually, you will not want to change ArrangeMode once a graph has been created. One exception may be to select the ITG_ModeCompact ArrangeMode to provide connections directly between nodes once the initial layout has be produced.

Data Type

Integer (Enumerated)

AutoArrange Property

Applies To

ITGraph

Description

Specifies whether the graph's nodes will be automatically arranged using one of ITGraph's layout procedures.

Usage

[*form*.]*control*.**AutoArrange**[= *mode*]

Read/Write at run time and design time.

Setting

The AutoArrange settings are:

Setting	Description
ITG_ManualArrange	Do not rearrange nodes in the graph.
ITG_AutoArrange	Graphically rearrange the graph whenever a change is made.
ITG_ArrangeNow	Graphically rearrange the nodes exactly one time, then revert to ITG_ManualArrange for the AutoArrange property (valid only at run-time).

Remarks

The ITG_ArrangeNow option is valid only at run time. It causes the graph to be rearranged according to the current settings of ArrangeMode and other properties, then sets AutoArrange to ITG_ManualArrange. If an application generates a graph, it would be best to set AutoArrange to ITG_ManualArrange, create the graph, then set AutoArrange to ITG_ArrangeNow to layout the nodes. For an interactive application where a user is adding nodes, and automatic arrangement is desired, the ITG_AutoArrange option should be used.

Data Type

Integer (Enumerated)

AutoMouseEvent Property

Applies To

ITGraph

Description

Used to configure ITGraphs response to a mouse event.

Usage

[*form*.]*control*.**AutoMouseEvent** (*event*) [= *action*]

Hidden at design time. Read/Write at run time.

Remarks

The AutoMouseEvent property is used to set or retrieve ITGraphs response to a particular mouse event. The *event* is composed of a single mouse button identifier (ITG_LeftButton, ITG_MiddleButton, or ITG_RightButton) added to any combination of modifier keys (ITG_ShiftDown, ITG_CtrlDown and ITG_AltDown). The *action* cab be ITG_meNone (do nothing), or a composition from the following constants: ITG_meMouseEvent (generate mouse events), ITG_meConnect (connect nodes), ITG_meConstrained (constrain mouse movement), ITG_meDrag (drag nodes), ITG_meSelect (select an area on the graph) and ITG_meSize (resize nodes). See Working With Events for a more detailed description of ITGraphs event handling scheme.

Data Type

Integer

AutoMouseEvents Property

Applies To

ITGraph

Description

Brings up the AutoMouseEvents Setup dialog box.

Usage

[*form*.]*control*.**AutoMouseEvents**[= *dummyString*]

Write only at run time. Select only at design time (double-click on name or click on "..." in the property window).

Remarks

The AutoMouseEvents property is used to bring up the AutoMouseEvents Setup dialog box, where ITGraphs response to mouse events can be configured. The dialog can also be brought up at run time by setting AutoMouseEvents to a dummy string, the value being irrelevant.

Data Type

String

ConnectFromHandle

Applies To

ITGraph

Description

Set or retrieve the handle from which the current connection originates from its source node.

Usage

[*form*.]*control*.**ConnectFromHandle**[= *handle*]

Read/Write at run time. Hidden at design time.

Setting

The ConnectFromHandle settings depend on the current value of DrawDir. For a DrawDir of ITG_XxxToYyy, ConnectFromHandle would be set to **1** for side Xxx and **3** for side Yyy. The values start at **1** (= side Xxx) and procede clockwise to **4**. The table below shows the handle sides corresponding to the various DrawDir and ConnectFromHandles settings.

Setting	ITG_RightToLeft	ITG_LeftToRight	ITG_BottomToTop	ITG_TopToBottom
1	Right	Left	Bottom	Top
2	Bottom	Top	Left	Right
3	Left	Right	Top	Bottom
4	Top	Bottom	Right	Left

Remarks

The setting of this property is only relevant when ArrangeMode is set to ITG_ModeFlowChart. When the ConnectFromHandle property is set, the graph will be redrawn. If the AutoArrange property is set to ITG_AutoArrange, a new graph layout will be generated to reflect the change. If no connection is current, ConnectFromHandle will return **0**.

Data Type

Integer (Enumerated)

ConnectFromIndex Property

Applies To

ITGraph

Description

Used to retrieve the target index of the current connection.

Usage

index = [*form*.]*control*.**ConnectToIndex**

Read only at run time. Hidden at design time.

Remarks

If there is no current connection, ConnectFromIndex returns **-1**.

Data Type

Integer

ConnectionAlign Property (1.1)

Applies To

ITGraph

Description

Set or retrieve the text alignment used for the current connection.

Usage

[*form*.]*control*.**ConnectionAlign**[= *alignment*]

Read/Write at run time. Hidden at design time.

Setting

The ConnectionAlign settings are:

Setting	Description
ITG_AlignLeft	Left justify text.
ITG_AlignCenter	(Default)Center text.
ITG_AlignRight	Right justify text.

Remarks

When the ConnectionAlign property is set, the graph will be redrawn. If no connection is current, ConnectionAlign will return **-1**. Note that the ConnectionAlign property only affects the text if the connection label has more than one line.

Data Type

Integer (Enumerated)

ConnectionArrow

Applies To

ITGraph

Description

Set or retrieve the type of arrowhead used for the current connection.

Usage

[*form*.]*control*.**ConnectionArrow**[= *arrowhead*]

Read/Write at run time. Hidden at design time.

Setting

The ConnectionArrow settings are:

Setting	Description
ITG_AHNone	No arrowhead.
ITG_AHSolid	(Default)Solid, filled, triangular arrowhead.
ITG_AHHollow	Hollow triangular arrowhead.
ITG_AHOutline	Triangular, open arrowhead.
ITG_AHSolidCircle	Solid, filled, circular arrowhead.
ITG_AHHollowCircle	Hollow circular arrowhead.

Remarks

When the ConnectionArrow property is set, the graph will be redrawn. If no connection is current, ConnectionArrow will return **-1**.

Data Type

Integer (Enumerated)

ConnectionColor Property

Applies To

ITGraph

Description

Set or retrieve the color used for the current connection.

Usage

[*form*.]*control*.**ConnectionColor**[= *color*]

Read/Write at run time. Hidden at design time.

Remarks

In addition to a valid color value, ConnectionColor can be set to **-1**, in which case the connection will be drawn in ForeColor. This is the default setting for all connections. When the ConnectionColor property is changed, the graph will be redrawn. If no connection is current, ConnectionColor will return **-1**.

Data Type

Long (Color or -1)

ConnectionData Property

Applies To

ITGraph

Description

Set or retrieve a user-defined data value associated with the current connection.

Usage

[*form*.]*control*.**ConnectionData**[= *data*]

Read/Write at run time. Hidden at design time.

Remarks

The ConnectionData property can be used to associate a user-defined Long value with each connection. By default, this value is set to **0**. If no connection is current, ConnectionData will return **0**.

Data Type

Long

ConnectionId Property

Applies To

ITGraph

Description

Unique long index assigned by ITGraph to the current connection.

Usage

[*form*.]*control*.**ConnectionData**[= *data*]

Read only at run time. Hidden at design time.

Remarks

When a connection is added with the ConnectTo property, it is assigned a unique ConnectionId. This value never changes for the connection, even if others are deleted or inserted, or the graph is saved or retrieved from disk. ConnectionId values are numbered beginning with **1**. When a graph is cleared by the Clear method, the next connection to be added will be assigned a ConnectionId of **1**.

Data Type

Long

ConnectionLabel Property

Applies To

ITGraph

Description

Used to set or retrieve the label of the current connection.

Usage

[*form*.]*control*.**ConnectionLabel**[= *label*]

Read/Write at run time. Hidden at design time.

Remarks

When the ConnectionLabel property is set, the graph will be repainted. If no connection is current, ConnectionLabel will return an empty string.

Data Type

String

ConnectionLineWidth Property

Applies To

ITGraph

Description

Set or retrieve the line width used for the current connection.

Usage

[*form*.]*control*.**ConnectionLineWidth**[= *line_width*]

Read/Write at run time. Hidden at design time.

Remarks

Specifies the width of the line used to draw a connection. A value of **0** draws the thinnest line possible. Higher values result in increasingly thicker lines. A value of **-1** causes the connection to be drawn using the line thickness specified by the control's LineWidth property.

Data Type

Integer (Line Width or -1)

ConnectTo Property

Applies To

ITGraph

Description

Add a new connection or select and existing connection.

Usage

[*form*.]*control*.**ConnectTo** (*from_index*) = *to_index*

Write only at run time. Hidden at design time.

Remarks

The ConnectTo property establishes a connection between two nodes in the graph.. If *from_index* and *to_index* are valid node indices, a connection from node *from_index* to node *to_index* will be added to the graph. If the AutoArrange property is set to ITG_AutoArrange, the graph will be recomputed and redrawn.

The new connection will become the current connection. If there is already a connection between the specified nodes, that connection will be made current, but the graph will not be redrawn.

Data Type

Integer

ConnectToHandle Property

Applies To

ITGraph

Description

Set or retrieve the handle to which the current connection connects to its target node.

Usage

[*form*.]*control*.**ConnectToHandle**[= *handle*]

Read/Write at run time. Hidden at design time.

Setting

The ConnectToHandle settings depend on the current value of DrawDir. For a DrawDir of ITG_XxxToYyy, ConnectToHandle would be set to **1** for side Xxx and **3** for side Yyy. The values start at **1** (= side Xxx) and procede clockwise to **4**. The table below shows the handle sides corresponding to the various DrawDir and ConnectToHandles settings.

Setting	ITG_RightToLeft	ITG_LeftToRight	ITG_BottomToTop	ITG_TopToBottom
1	Right	Left	Bottom	Top
2	Bottom	Top	Left	Right
3	Left	Right	Top	Bottom
4	Top	Bottom	Right	Left

Remarks

The setting of this property is only relevant when ArrangeMode is set to ITG_ModeFlowChart. When the ConnectToHandle property is set, the graph will be redrawn. If the AutoArrange property is set to ITG_AutoArrange, a new graph layout will be generated to reflect the change. If no connection is current, ConnectToHandle will return **0**.

Data Type

Integer (Enumerated)

ConnectToIndex Property

Applies To

ITGraph

Description

Used to retrieve the source index of the current connection.

Usage

index = [*form*.]*control*.**ConnectToIndex**

Read only at run time. Hidden at design time.

Remarks

If there is no current connection, ConnectToIndex returns **-1**.

Data Type

Integer

DragItems Property

Applies To

ITGraph

Description

The DragItems property is no longer used by ITGraph. It has been subsumed in the new event handling mechanism. See Working with Events for more details.

DrawArrows Property

Applies To

ITGraph

Description

Specifies whether arrowheads will be drawn on connections.

Usage

[*form*.]*control*.**DrawArrows**[= *boolean*]

Read/Write at run time and design time.

Setting

The ConnectionArrow settings are:

Setting	Description
True	(Default) Draw arrowheads on connections.
False	Don't draw arrowheads.

Remarks

Whether an arrowhead is actually drawn and how it will appear for a particular connection is determined by the the ConnectionArrow property for that connection. When the DrawArrows property is changed, the graph will be repainted.

Data Type

Integer (Boolean)

DrawBackLinks Property

Applies To

ITGraph

Description

When ArrangeMode is ITG_ModeHierarchy or ITG_ModeTree, specifies if and how back-links will be drawn.

Usage

[*form*.]*control*.**DrawBackLinks**[= *mode*]

Read/Write at run time and design time.

Setting

The DrawBackLinks settings are:

Setting	Description
ITG_NoBackLinks	Don't show back-links on the graph.
ITG_SolidBackLinks	Draw back-links using solid lines.
ITG_DashedBackLinks	(Default) Draw back-links using dashed lines.

Remarks

When the DrawBackLinks property is changed, the graph will be repainted.

Data Type

Integer (Enumerated)

DrawColored Property

Applies To

ITGraph

Description

Specifies whether the graph will be drawn in color or black and white.

Usage

[*form*.]*control*.**DrawColored**[= *boolean*]

Read/Write at run time and design time.

Setting

The DrawColored settings are:

Setting	Description
True	(Default) Draw the graph using the specified colors.
False	Draw the graph in black and white. The background and interiors of nodes and hollow arrowheads will be white. Node outlines, connections, arrowheads and text will be black.

Remarks

The DrawColored property is useful for printing a graph when the colored version is not legible. When the DrawColored property is changed, the graph will be repainted.

Data Type

Integer (Boolean)

DrawConnLabels

Applies To

ITGraph

Description

Specifies whether labels will be drawn on connections.

Usage

[*form*.]*control*.**DrawConnLabels**[= *boolean*]

Read/Write at run time and design time.

Setting

The DrawConnLabels settings are:

Setting	Description
True	(Default) Draw labels on connections.
False	Don't draw labels on connections.

Remarks

The label for a particular connection must be set with the ConnectionLabel property. When the DrawConnLabels property is changed, the graph will be repainted.

Data Type

Integer (Boolean)

DrawDir Property

Applies To

ITGraph

Description

Specifies the preferred orientation for laying out the graph's nodes.

Usage

[*form*.]*control*.**DrawDir**[= *mode*]

Read/Write at run time and design time.

Setting

The DrawDir settings are:

Setting	Description
ITG_RightToLeft	The preferred flow of the graph is from right to left.
ITG_LeftToRight	The preferred flow of the graph is from left to right.
ITG_BottomToTop	The preferred flow of the graph is from bottom to top.
ITG_TopToBottom	(Default) The preferred flow of the graph is from top to bottom.

Remarks

The setting of the DrawDir property guides the automatic layout procedures in producing a graph layout. The setting should be according to normal flow for the desired graph type. Flow Charts, for example, usually flow from top to bottom, so an ITG_TopToBottom value would be used for this type of graph. If the AutoArrange property is set to ITG_AutoArrange, the graph will be recomputed and redrawn when DrawDir is set.

Data Type

Integer (Enumerated)

DrawHandles Property

Applies To

ITGraph

Description

Specifies whether node handles will be shown in the ITG_ModeFlowChart ArrangeMode.

Usage

[*form*.]*control*.**DrawHandles**[= *boolean*]

Read/Write at run time and design time.

Setting

The DrawHandles settings are:

Setting	Description
True	(Default) Draw handles on nodes.
False	Don't draw handles on nodes.

Remarks

This property is only relevant if ArrangeMode is set to ITG_ModeFlowChart. When DrawHandles is set to **True**, handles will be drawn to the left, right, above and below the nodes. Clicking in one handle and dragging to another will cause an ItemConnect event to be sent to the control, specifying the source and destination handles as well as the node indices. When the DrawHandles property is changed, the graph will be repainted.

Data Type

Integer (Boolean)

DrawItemLabels Property

Applies To

ITGraph

Description

Specifies whether labels will be drawn on graph nodes.

Usage

[*form*.]*control*.**DrawItemLabels**[= *boolean*]

Read/Write at run time and design time.

Setting

The DrawItemLabels settings are:

Setting	Description
True	(Default) Draw labels on graph nodes.
False	Don't draw labels on graph nodes.

Remarks

The label for a particular node must be set with the List property. When the DrawItemLabels property is changed, the graph will be repainted. Note that if a particular node's ItemDrawLabel property is set to **False**, then its label will not be drawn no matter what the setting of DrawItemLabels.

Data Type

Integer (Boolean)

DrawScale Property

Applies To

ITGraph

Description

Sets the magnification level for the graph.

Usage

[*form*.]*control*.**DrawScale**[= *scale*]

Read/Write at run time and design time.

Setting

The DrawScale settings are:

Setting	Description
0	Draw the graph so that it fits in the control window, but no larger than normal size (i.e. when DrawScale = 100).
< 100	Shrink the graph to the specified percentage of normal size.
100	(Default) Draw the graph at normal size.
> 100	Expand the graph to the specified percentage of normal size.

Remarks

When DrawScale is set, the graph will be redrawn at the specified scale. A node that was previously centered in the control window will be kept in the center at the new scale if possible.

Data Type

Integer (% scale)

FillColor Property

Applies To

ITGraph

Description

Set or retrieve the color used for the interiors of graph nodes.

Usage

[*form*.]*control*.**FillColor**[= *color*]

Read/Write at run time and design time.

Remarks

FillColor is the default color to be used for the interiors of all graph nodes. If a node's ItemFillColor is set to **-1**, it will be filled with FillColor. If the DrawColored flag is **False**, all nodes will be filled white, irrespective of the settings of FillColor and ItemFillColor.

Data Type

Long (Color)

Gap Property

Applies To

ITGraph

Description

Specifies the distance (in pixels) between the start or end of a connection and its point of branching.

Usage

[*form*.]*control*.**Gap**[= *gap_pixels*]

Read/Write at run time and design time.

Remarks

The exact interpretation of the Gap property depends on the setting of ArrangeMode and DrawDir.

If ArrangeMode is ITG_ModeHierarchy, Gap applies only in the direction of DrawDir. It is the spacing between the start point of the connection and the first branch point and between the last branch point and the end point of the connection.

If ArrangeMode is ITG_ModeCompact, Gap is ignored. Spacing is determined entirely by the XSpace and YSpace properties.

If ArrangeMode is ITG_ModeFlowChart, Gap specifies the distance in any direction from the start or end point of a connection and the first or last branch of the connection.

If ArrangeMode is ITG_ModeTree, Gap applies only in the direction of DrawDir. It is the spacing between the start point of the connection and the first branch point and between the last branch point and the end point of the connection. Depending on the settings of Gap, XSpace and YSpace, you can use the ITG_ModeTree mode to produce a tree (Gap = **0**, and whichever of XSpace or YSpace is the primary graph axis set to the spacing), an org chart (Gap = spacing, XSpace or YSpace of graph axis set to **0**), or a block diagram (Gap, XSpace, YSpace all set to **0**).

Data Type

Integer

GraphicAllowImport Property (1.1)

Applies To

ITGraph

Description

Specifies whether the user can import graphics at runtime via the "Select a Graphic" dialog box.

Usage

[*form*.]*control*.**GraphicAllowImport**[= *mode*]

Read/Write at run time and design time.

Setting

The GraphicAllowImport settings are:

Setting	Description
ITG_NoImportGraphics	(Default) The user can only select existing graphics. Importing new graphics is not allowed.
ITG_ImportGraphics	The user is allowed to import new graphics, and modify or delete graphics imported at runtime. Design time graphics cannot be modified or deleted.
ITG_ModifyDTGraphics	The user is allowed to import new graphics at runtime. The user may modify (change the name/shape) of any graphic. Only graphics imported at runtime may be deleted.
ITG_DeleteDTGraphics	The user is allowed to import new graphics at runtime. Any graphic (runtime or design time) may be modified or deleted.

Data Type

Integer (Enumerated)

GraphicCount Property (1.1)

Applies To

ITGraph

Description

Specifies the number of currently loaded graphics.

Usage

count = [*form*.]*control*.**GraphicCount**

Read only at run time. Hidden at design time.

Data Type

Integer

GraphicName Property (1.1)

Applies To

ITGraph

Description

Specifies the name assigned to a graphic.

Usage

[*form*.]*control*.**GraphicName** (*index*)[= *name*]

Read/Write at run time. Hidden at design time.

Remarks

The GraphicName property sets or retrieves the name associated with a graphic. The GraphicCount property can be used to determine how many graphics are currently loaded. The *index* values range from 0 to GraphicCount-1. The value of GraphicName for an invalid *index* value is an empty string.

Setting GraphicName for a valid *index* changes the name of the indexed graphic. If the assigned name is the same as the name of another graphic, its name will be modified so as to preserve unique names for all the graphics.

Setting GraphicName with *index* equal to the value of GraphicCount creates a new (initially empty) graphic. This must be done before GraphicPath can be set.

Setting GraphicName to an empty string causes the indexed graphic to be deleted. Any nodes whose ItemGraphic property were set to the deleted graphic will have their ItemGraphic property set to 0 (no graphic). All other nodes will retain their graphics, though the ItemGraphic values themselves may change. Graphic 0 cannot be deleted.

Data Type

String

GraphicPath Property (1.1)

Applies To

ITGraph

Description

Used to specify a graphic to be imported.

Usage

[*form*.]*control*.**GraphicPath** (*index*) = *path*

Write only at run time. Hidden at design time.

Remarks

The GraphicPath property is used to specify the path of a graphic to be imported. The string must be a valid path to a bitmap or windows metafile. The GraphicCount property can be used to determine how many graphics are currently loaded. The *index* values range from 0 to GraphicCount-1. Graphic path can only be set for a valid *index* value. To add a new graphic, first set the GraphicName property for an index of GraphicCount, then set the GraphicPath property of that graphic. If the GraphicPath property of a node is set to an empty string, nothing will be drawn for nodes referencing that graphic.

Data Type

String

Graphics Property (1.1)

Applies To

ITGraph

Description

Brings up the ITGraph Graphic Table dialog box.

Usage

Hidden at run time. Select only at design time (double-click on name or click on "..." in the property window).

Remarks

The Graphics property is used to bring up the ITGraph Graphic Table dialog box, where graphics can be imported at design time. Graphics that are imported at design time will be saved with the form on which they appear.

GraphicSelect Property (1.1)

Applies To

ITGraph

Description

Brings up the Select a Graphic dialog box, where the user can choose from available graphics, import new graphics, and modify or delete existing ones.

Usage

[*form*.]*control*.**GraphicSelect**[= *graphic_index*]

Read/Write at run time. Hidden at design time.

Remarks

Setting GraphicSelect to a valid *graphic_index* brings up the Select a Graphic dialog box, with the specified graphic selected. When done, GraphicSelect will contain the index of the graphic selected by the user, or **-1** if no graphic was selected. The setting of the GraphicAllowImport property determines whether or not the user will be able to import new graphics, or modify or delete design time graphics, imported via the Graphics property.

Data Type

Integer

IsDirty Property

Applies To

ITGraph

Description

Specifies whether the contents of the graph have changed.

Usage

[*form*.]*control*.**IsDirty**[= *boolean*]

Read/Write at run time. Hidden at design time.

Setting

The IsDirty settings are:

Setting	Description
True	The graphs contents have changed.
False	The graphs contents have not changed since the last Clear, LoadFrom or SaveAs.

Remarks

The IsDirty property can be used to determine whether or not the graph has changed since it was last cleared, saved or loaded. This information can be used to alert the user prior to clearing the graph, loading a new graph, or exiting the application. Typically, a YES/NO/CANCEL box is brought up with the question Save changes to graph?. If the user chooses YES, the graph should be saved. If NO, the operation should proceed without saving the graph. If CANCEL is chosen, the operation should be cancelled, and the graph should remain as it is.

The IsDirty property is set to **True** whenever a stored property of the graph is changed. Event mappings and other properties that dont affect the graphs appearance do not set the IsDirty flag. The Clear method and LoadFrom and SaveAs properties are the only ways IsDirty is set to **False** by ITGraph. Your application can set or clear the flag as desired.

Delphi Users: Setting ITGraphs Font in Delphi does not cause the IsDirty flag to be set. Therefore, if the font is changed, the application should explicitly set IsDirty to **True**.

Data Type

Integer (Boolean)

ItemBorderColor Property

Applies To

ITGraph

Description

Specifies the color with which the specified node will be outlined.

Usage

[*form*.]*control*.**ItemBorderColor** (*index*)[= *color*]

Read/Write at run time. Hidden at design time.

Remarks

ItemBorderColor sets the border color for a particular node. If it is set to **-1**, then the node's border will be drawn using ForeColor. If DrawColored is **False**, the node's border will be black.

Data Type

Long (Color or -1)

ItemData Property

Applies To

ITGraph

Description

Set or retrieve a user-defined data value associated with the specified node.

Usage

[*form*.]*control*.**ItemData** (*index*)[= *data*]

Read/Write at run time. Hidden at design time.

Remarks

The ItemData property can be used to associate a user-defined Long value with each node in the graph. By default, this value is set to **0** for each node.

Data Type

Long

ItemDrawLabel Property (1.1)

Applies To

ITGraph

Description

Specifies whether or not a node's label will be drawn.

Usage

[*form*.]*control*.**ItemDrawLabel** (*index*)[= *boolean*]

Read/Write at run time. Hidden at design time.

Setting

The ItemDrawText settings are:

Setting	Description
True	(Default) Draw the node's label.
False	Don't draw the node's label.

Remarks

Specifies for a particular node whether or not its label will be drawn. Note that if the DrawItemLabels property is set to **False**, no node labels will be drawn, irrespective of the settings of ItemDrawLabel.

Data Type

Integer (Boolean)

ItemFillColor Property

Applies To

ITGraph

Description

Specifies the color with which the specified node will be filled.

Usage

[*form*.]*control*.**ItemFillColor** (*index*)[= *color*]

Read/Write at run time. Hidden at design time.

Remarks

ItemFillColor sets the interior color for a particular node. If it is set to **-1**, then the node will be filled using FillColor. If DrawColored is **False**, the node's interior will be white.

Data Type

Long (Color or -1)

ItemGraphic Property (1.1)

Applies To

ITGraph

Description

Specifies the graphic associated with a node.

Usage

[*form*.]*control*.**ItemGraphic** (*index*)[= *graphic_index*]

Read/Write at run time. Hidden at design time.

Remarks

graphic_index is a positive index that identifies a graphic. It ranges from 0 to GraphicCount - 1. Graphic 0 is the null graphic - nothing is drawn. Other graphics can be imported at design time or runtime, and associated with nodes by setting ItemGraphic. If graphics are deleted, the value of ItemGraphic for a node may change if the index of its associated graphic changes. Also, if the graphic associated with a node is deleted, the node's ItemGraphic property will be set to 0.

Data Type

Integer

ItemGraphicAlign Property (1.1)

Applies To

ITGraph

Description

Set or retrieve the alignment for the graphic of a node within the node's graphic rectangle.

Usage

[*form*.]*control*.**ItemGraphicAlign** (*index*)[= *alignment*]

Read/Write at run time. Hidden at design time.

Setting

The ItemGraphicAlign settings are:

Setting	Description
ITG_AlignTopLeft	Top left corner.
ITG_AlignTopCenter	Top center.
ITG_AlignTopRight	Top right corner.
ITG_AlignMiddleLeft	Middle left.
ITG_AlignMiddleCenter	(Default) Center.
ITG_AlignMiddelRight	Middle right.
ITG_AlignBottomLeft	Bottom left corner.
ITG_AlignBottomCenter	Bottom center.
ITG_AlignBottomRight	Bottom right corner.

Remarks

The ItemGraphicAlign property specifies how a node's graphic (if any) will be placed within the node's graphic rectangle.

Data Type

Integer (Enumerated)

ItemGraphicHeight Property (1.1)

Applies To

ITGraph

Description

Specifies the height of a node's graphic rectangle.

Usage

[*form*.]*control*.**ItemGraphicHeight** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

When ItemGraphicStyle is ITG_GraphicIconFill or ITG_GraphicIconFillIso, then the node's graphic is drawn so as to fit in the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight. These parameters are in graph coordinates, and are relative to the top-left corner of the node. If ItemGraphicHeight is **0**, then the graphic will extend to the bottom of the node.

Data Type

Integer

ItemGraphicLeft Property (1.1)

Applies To

ITGraph

Description

Specifies the left edge of a node's graphic rectangle.

Usage

[*form*.]*control*.**ItemGraphicLeft** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

When ItemGraphicStyle is ITG_GraphicIconFill or ITG_GraphicIconFillIso, then the node's graphic is drawn so as to fit in the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight. These parameters are in graph coordinates, and are relative to the top-left corner of the node.

Data Type

Integer

ItemGraphicStyle Property (1.1)

Applies To

ITGraph

Description

Set or retrieve the drawing style for a node's graphic.

Usage

[*form*.]*control*.**ItemGraphicStyle** (*index*)[= *draw_style*]

Read/Write at run time. Hidden at design time.

Setting

The ItemGraphicStyle settings are:

Setting	Description
ITG_GraphicNone	Don't draw the node's graphic.
ITG_GraphicIconFill	Draw the node's graphic so that it fills the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight.
ITG_GraphicIconFillIso	Draw the node's graphic so that it fills the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight, while maintaining the original aspect ratio of the graphic.
ITG_GraphicShapeFill	(Default) Draw the node's graphic so that it fills the node's bounding rectangle.
ITG_GraphicShapeFillIso	Draw the node's graphic so that it fills the node's bounding rectangle, while maintaining the original aspect ratio of the graphic.

Remarks

The ItemGraphicStyle setting is only relevant if a graphic has been set for the node via the ItemGraphic property. The ITG_GraphicXXXFillIso settings maintain the aspect ratio of the graphic. This may mean that the graphic will actually be smaller in one dimension than the target rectangle. In that case, the graphic is aligned within its rectangle according to the setting of ItemGraphicAlign. The non-Iso settings will distort the graphic unless the target rectangle has the same aspect ratio as the graphic itself.

Data Type

Integer (Enumerated)

ItemGraphicTop Property (1.1)

Applies To

ITGraph

Description

Specifies the top of a node's graphic rectangle.

Usage

[*form*.]*control*.**ItemGraphicTop** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

When ItemGraphicStyle is ITG_GraphicIconFill or ITG_GraphicIconFillIso, then the node's graphic is drawn so as to fit in the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight. These parameters are in graph coordinates, and are relative to the top-left corner of the node.

Data Type

Integer

ItemGraphicWidth Property (1.1)

Applies To

ITGraph

Description

Specifies the width of a node's graphic rectangle.

Usage

[*form*.]*control*.**ItemGraphicWidth** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

When ItemGraphicStyle is ITG_GraphicIconFill or ITG_GraphicIconFillIso, then the node's graphic is drawn so as to fit in the rectangle defined by ItemGraphicLeft, ItemGraphicTop, ItemGraphicWidth and ItemGraphicHeight. These parameters are in graph coordinates, and are relative to the top-left corner of the node. If ItemGraphicWidth is **0**, then the graphic will extend to the right edge of the node.

Data Type

Integer

ItemHeight Property (1.2)

Applies To

ITGraph

Description

Specifies the height of the node's bounding rectangle.

Usage

[*form*.]*control*.**ItemHeight** (*index*)[= *height*]

Read/Write at run time. Hidden at design time.

Remarks

When a node is added with the AddItem method, ItemHeight is set to the value of the YSpan property.

Data Type

Integer

ItemId Property

Applies To

ITGraph

Description

Unique long index assigned by ITGraph to the node.

Usage

id = [*form*.]*control*.**ItemId** (*index*)

Read only at run time. Hidden at design time.

Remarks

When a node is added with the AddItem method, it is assigned a unique ItemId. This value never changes for the node, even if others are deleted or inserted, or the graph is saved or retrieved from disk. ItemId values are numbered beginning with **1**. When a graph is cleared by the Clear method, the next node to be added will be assigned an ItemId of **1**.

Data Type

Long

ItemIndex Property (1.2)

Applies To

ITGraph

Description

Used to change the index of a node in the graph.

Usage

[*form*.]*control*.**ItemIndex** (*index*) = *new_index*

Write only at run time. Hidden at design time.

Remarks

An items index is relevant to ITGraph on two occasions. Firstly, the order in which nodes are drawn is determined by the index. The highest index is always on top if nodes overlap. Secondly, if ArrangeMode is set to ITG_ModeTree, nodes on the same level are drawn in order of increasing ItemIndex. Change ItemIndex allows your application to reorder the nodes. Setting ItemIndex to **0** is analogous to the Send To Back command in Visual Basics Edit menu.

Data Type

Integer

ItemLabelAlign Property (1.1)

Applies To

ITGraph

Description

Set or retrieve the alignment for the label of a node within the node's label rectangle.

Usage

[*form*.]*control*.**ItemLabelAlign** (*index*)[= *alignment*]

Read/Write at run time. Hidden at design time.

Setting

The ItemLabelAlign settings are:

Setting	Description
ITG_AlignTopLeft	Top left corner.
ITG_AlignTopCenter	Top center.
ITG_AlignTopRight	Top right corner.
ITG_AlignMiddleLeft	Middle left.
ITG_AlignMiddleCenter	(Default) Center.
ITG_AlignMiddelRight	Middle right.
ITG_AlignBottomLeft	Bottom left corner.
ITG_AlignBottomCenter	Bottom center.
ITG_AlignBottomRight	Bottom right corner.

Remarks

The ItemLabelAlign property specifies how a node's label (if any) will be aligned within the node's label rectangle.

Data Type

Integer (Enumerated)

ItemLabelHeight Property (1.1)

Applies To

ITGraph

Description

Specifies the height of a node's label rectangle.

Usage

[*form*.]*control*.**ItemLabelHeight** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

Specifies the height of the rectangle used to draw the node's label. The label rectangle is defined relative to the top-left corner of the node's bounding rectangle. If ItemLabelHeight is set to **0**, then the label rectangle extends to the bottom of the node.

Data Type

Integer

ItemLabelLeft Property (1.1)

Applies To

ITGraph

Description

Specifies the left edge of a node's label rectangle.

Usage

[*form*.]*control*.**ItemLabelLeft** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

Specifies the left edge of the rectangle used to draw the node's label. The label rectangle is defined relative to the top-left corner of the node's bounding rectangle.

Data Type

Integer

ItemLabelTop Property (1.1)

Applies To

ITGraph

Description

Specifies the top of a node's label rectangle.

Usage

[*form*.]*control*.**ItemLabelTop** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

Specifies the top of the rectangle used to draw the node's label. The label rectangle is defined relative to the top-left corner of the node's bounding rectangle.

Data Type

Integer

ItemLabelWidth Property (1.1)

Applies To

ITGraph

Description

Specifies the width of a node's label rectangle.

Usage

[*form*.]*control*.**ItemLabelWidth** (*index*)[= *pixels*]

Read/Write at run time. Hidden at design time.

Remarks

Specifies the width of the rectangle used to draw the node's label. The label rectangle is defined relative to the top-left corner of the node's bounding rectangle. If ItemLabelWidth is set to **0**, then the label rectangle extends to the right edge of the node.

Data Type

Integer

ItemShape Property

Applies To

ITGraph

Description

Specifies the shape for drawing the node.

Usage

[*form*.]*control*.**ItemShape** (*index*)[= *shape_index*]

Read/Write at run time. Hidden at design time.

Remarks

The ItemShape settings are:

Setting	Description
ITG_ShapeNone	No shape. Useful if a graphic represents the node.
ITG_ShapeDefault	(Default) Same as ITG_ShapeRectangle.
ITG_ShapeRectangle	A rectangle.
ITG_ShapeEllipse	An ellipse.
ITG_ShapeRoundRect	A rounded rectangle.
ITG_ShapeParallelogram1	A parallelogram with the bottom shifted right.
ITG_ShapeParallelogram2	A parallelogram with the top shifted right.
ITG_ShapeHexagon	A hexagon.
ITG_ShapePage	A page of paper, with the top right corner folded over.
ITG_ShapeDiamond	A diamond.
ITG_ShapeOctagon	An octagon.
ITG_ShapePapers1	A stack of paper, with the top sheet in the top left.
ITG_ShapePapers2	A stack of paper, with the top sheet in the bottom left.

Data Type

Integer (Enumerated)

ItemTextColor Property

Applies To

ITGraph

Description

Specifies the color with which the specified node's text will be drawn.

Usage

[*form*.]*control*.**ItemTextColor** (*index*)[= *color*]

Read/Write at run time. Hidden at design time.

Remarks

ItemTextColor sets the text color for a particular node. If it is set to **-1**, then the node's caption will be drawn in ForeColor. If DrawColored is **False**, the node's text will be black.

Data Type

Long (Color or -1)

ItemWidth Property (1.2)

Applies To

ITGraph

Description

Specifies the width of the node's bounding rectangle.

Usage

[*form*.]*control*.**ItemWidth** (*index*)[= *position*]

Read/Write at run time. Hidden at design time.

Remarks

When a node is added with the AddItem method, ItemWidth is set to the value of the XSpan property.

Data Type

Integer

ItemXpos Property

Applies To

ITGraph

Description

Specifies the left edge of the node's bounding rectangle.

Usage

[*form*.]*control*.**ItemXpos** (*index*)[= *position*]

Read/Write at run time. Hidden at design time.

Remarks

If your application sets ItemXpos, you will probably want to set AutoArrange to ITG_ManualArrange so that the graph is not recomputed later. The horizontal center of a node can be found by adding ItemWidth/2 to the ItemXpos value.

Data Type

Integer

ItemYpos Property

Applies To

ITGraph

Description

Specifies the top edge of the node's bounding rectangle.

Usage

[*form*.]*control*.**ItemYpos** (*index*)[= *position*]

Read/Write at run time. Hidden at design time.

Remarks

If your application sets ItemYpos, you will probably want to set AutoArrange to ITG_ManualArrange so that the graph is not recomputed later. The vertical center of a node can be found by adding ItemHeight/2 to the ItemYpos value.

Data Type

Integer

LineWidth Property

Applies To

ITGraph

Description

Specifies the default line width to be used for drawing connections between nodes.

Usage

[*form*.]*control*.**LineWidth**[= *width*]

Read/Write at run time and design time.

Remarks

A LineWidth of **0** causes lines to always be thin, irrespective of the setting of DrawScale. If LineWidth is greater than 0, it will be scaled according to the setting of DrawScale. The LineWidth property specifies the default line width for connections. All connections with a ConnectionLineWidth of **-1** will be drawn with the width specified by LineWidth, otherwise the connection's own line width setting will be used.

Data Type

Integer

List Property

Applies To

ITGraph

Description

Specifies the label associated with a node.

Usage

[*form*.]*control*.**List** (*index*)[= *string*]

Read/Write at run time. Hidden at design time.

Remarks

The list property sets or retrieves the label associated with a node. Note that the initial value for this property is set by AddItem method.

Data Type

String

ListCount Property

Applies To

ITGraph

Description

Specifies the number of nodes in the graph.

Usage

count = [*form*.]*control*.**ListCount**

Read only at run time. Hidden at design time.

Data Type

Integer

LoadFrom Property

Applies To

ITGraph

Description

Load a saved ITGraph graph from disk.

Usage

[*form*.]*control*.**LoadFrom** = *path*

Write only at run time. Hidden at design time.

Remarks

Setting the LoadFrom property to the path of an ITGraph file will clear the current contents of the control and replace it with the graph found in the specified file. Properties that affect the way in which the graph is drawn are saved in the graph file. Those that affect event handling are not.

Data Type

String (file path)

NewIndex Property

Applies To

ITGraph

Description

Index of the the last node added to the graph.

Usage

index = [*form*.]*control*.**NewIndex**

Read only at run time. Hidden at design time.

Remarks

The value of the NewIndex property will either be a valid node index (greater than or equal to zero), or **-1**. NewIndex should only be used immediately after a node has been added with the AddItem.

Data Type

Integer

PrintGraph Property

Applies To

ITGraph

Description

Print the current graph on the printer.

Usage

[*form*.]*control*.**PrintGraph** = *hDC*

Write only at run time. Hidden at design time.

Remarks

Prints the graph on the specified device context (*hDC*). The graph will be drawn at the same scale as the screen image. If the graph won't fit on the page at the specified scale, it will be split across multiple pages. The PrintHeader property can be used to specify custom headers and footers for the pages of the graph.

If you are creating a custom report and need to include the graph in the printout, you may want to use the PrintToDC property to better control the way the graph is printed.

C++ Users: PrintGraph does not work with the MFC print preview feature. In order to support print preview, PrintGraphs pagination and printing must be implemented in your application. The C++ demonstration program shows how this can be done.

Data Type

Integer

PrintHeader Property (1.1)

Applies To

ITGraph

Description

Specify custom headers and footers to be used when printing the graph with PrintGraph.

Usage

[*form*.]*control*.**PrintHeader**[= *print_header*]

Read/Write at run time and design time.

Remarks

If *print_header* is a simple text string, it will be printed in the top-center of each page printed by PrintGraph. Several formatting characters can be inserted in the string to customize the behavior as follows:

Format	Description
&1	Place the following text (up to the next placement marker) in the top-left corner of each page.
&2	Place the following text (up to the next placement marker) in the top-center of each page.
&3	Place the following text (up to the next placement marker) in the top-right corner of each page.
&4	Place the following text (up to the next placement marker) in the bottom-left corner of each page.
&5	Place the following text (up to the next placement marker) in the bottom-center of each page.
&6	Place the following text (up to the next placement marker) in the bottom-right corner of each page.
&p	Insert the current page number in this position.
&n	Insert the total number of pages in this position.
&&	Insert the '&' character in this position.

As an example, the following lines will print the current graph, with "Test Graph" at the top-center of each page, the current date in the top right, and "Page # of #" at the bottom center.

```
ITGraph1.PrintHeader = "&2Test Graph&3" & Date & "&5Page&p of &n"
ITGraph1.PrintGraph = 100
```

Data Type

String

PrintRectHeight Property

Applies To

ITGraph

Description

Set or retrieve the height of the print rectangle.

Usage

[*form*.]*control*.**PrintRectHeight**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

PrintRectLeft Property

Applies To

ITGraph

Description

Set or retrieve the left of the print rectangle.

Usage

[*form*.]*control*.**PrintRectLeft**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

PrintRectTop Property

Applies To

ITGraph

Description

Set or retrieve the top of the print rectangle.

Usage

[*form*.]*control*.**PrintRectTop**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

PrintRectWidth Property

Applies To

ITGraph

Description

Set or retrieve the width of the print rectangle.

Usage

[*form*.]*control*.**PrintRectWidth**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

PrintToDC Property

Applies To

ITGraph

Description

Draws a portion of the graph in the specified device context.

Usage

[*form*.]*control*.**PrintToDC**[= *hDC*]

Write only at run time. Hidden at design time.

Remarks

When the PrintToDC property is set to a Windows device context handle, the portion of the graph specified by the selection rectangle will be drawn to that device context. The graph will be scaled to fit the print rectangle, which should be set to the desired viewport in the device context prior to setting PrintToDC. The PrintToDC property is useful for including an ITGraph graph in a printout or custom layout.

Note: If you set PrintToDC to **0**, the selection rectangle will be set to the bounding rectangle of the entire graph. This can be used if you want PrintToDC to print the entire graph. Alternatively, if SelectRectWidth and SelectRectHeight are both set to **0**, the entire graph will be printed.

Data Type

Integer (Device Context Handle)

PrintToWnd Property

Applies To

ITGraph

Description

Draws a portion of the graph in the specified window.

Usage

[*form*.]*control*.**PrintToWnd**[= *hWnd*]

Write only at run time. Hidden at design time.

Remarks

When the PrintToWnd property is set to a Windows window handle, the portion of the graph specified by the selection rectangle will be drawn in the specified window. The graph will be scaled to fit the print rectangle, which should be set to the desired viewport in the target window prior to setting PrintToWnd. The PrintToWnd can be used to display the graph in another control or window.

If you set PrintToWnd to **0**, the selection rectangle will be set to the bounding rectangle of the entire graph. This can be used if you want PrintToWnd to print the entire graph. Alternatively, if SelectRectWidth and SelectRectHeight are both set to **0**, the entire graph will be printed.

If you set PrintRectWidth and PrintRectHeight both to **0**, then the printed portion of the graph will fill the client area of *hWnd*.

Data Type

Integer (Window Handle)

QueryCount Property

Applies To

ITGraph

Description

The number of nodes or connections which satisfy the current query.

Usage

count = [*form*.]*control*.**QueryCount**

Read only at run time. Hidden at design time.

Remarks

Returns the number of nodes or connections that match the current query as initiated by setting the QueryState property.

Data Type

Integer

QueryData Property (1.1)

Applies To

ITGraph

Description

The reference data for the current query in ITG_QueryMatchItemId, ITG_QueryMatchItemData, ITG_QueryMatchConnectionId and ITG_QueryMatchConnectionData queries.

Usage

[*form*.]*control*.**QueryItem**[= *data*]

Read/Write at run time. Hidden at design time.

Remarks

When set, establishes the reference data for an upcoming query of type ITG_QueryMatchItemId, ITG_QueryMatchItemData, ITG_QueryMatchConnectionId or ITG_QueryMatchConnectionData. Any query currently in progress is invalidated.

Data Type

Long

QueryItem Property

Applies To

ITGraph

Description

The reference node for the current query or the current node in an ITG_QuerySelectRectItems query.

Usage

[*form*.]*control*.**QueryItem**[= *item*]

Read/Write at run time. Hidden at design time.

Remarks

When set, establishes the reference node for an upcoming query and sets QueryItemHandle to **0**. Any query currently in progress is invalidated.

During an ITG_QuerySelectRectItems, the QueryItem property will contain the index of the current node retrieved by the query.

Data Type

Integer

QueryItemHandle Property

Applies To

ITGraph

Description

The reference handle for a query.

Usage

[*form*.]*control*.**QueryItemHandle**[= *handle*]

Write only at run time. Hidden at design time.

Remarks

When set, establishes the reference handle for an upcoming query. Any query currently in progress is invalidated. A setting of **0** indicates that the query will apply to all handles of the node specified by QueryItem.

Data Type

Integer (Enumerated)

QueryState Property

Applies To

ITGraph

Description

Control and monitor an ITGraph query.

Usage

[*form*.]*control*.**QueryState** = *query_cmd*

query_valid = [*form*.]*control*.**QueryState**

Read/Write only at run time. Hidden at design time.

Remarks

ITGraph queries can be initiated and controlled by setting QueryState to one of the following values:

Setting	Description
ITG_QueryGetSources	Initiates a query to get the sources of a node. The QueryItem property must previously have been set. If only sources that connect to a specific handle are desired, the QueryItemHandle property should also be set. The first connection matching the query will be made the current connection, and the QueryCount property will be set to the total number of connections matching the query.
ITG_QueryGetTargets	Initiates a query to get the targets of a node. The QueryItem property must previously have been set. If only targets that emmanate from a specific handle are desired, the QueryItemHandle property should also be set. The first connection matching the query will be made the current connection, and the QueryCount property will be set to the total number of connections matching the query.
ITG_QueryGetConnections	Initiates a query to get all connections of a node. The QueryItem property must previously have been set. If only connections that connect to a specific handle are desired, the QueryItemHandle property should also be set. The first connection matching the query will be made the current connection, and the QueryCount property will be set to the total number of connections matching the query.
ITG_QueryGetSelectRectItems	Initiates a query to get all nodes intersected by the current selection rectangle. The selection rectangle is set prior to sending a SelectRect event, or can be directly set *via* the SelectRectLeft, SelectRectTop, SelectRectWidth and SelectRectHeight properties. The index of the first node matching the query can be retrieved from the QueryItem property, and the QueryCount property will contain the number of nodes matching the query.
ITG_QueryIterate	Iterates to the next node or connection in the query. If the query returns nodes, the QueryItem property is set to its index, otherwise, current connection is set to the next connection matching the query.
ITG_QueryMatchItemId	Initiates a query to get the index of the node with a particular ItemId. The QueryData property must previously have been set to the ItemId of the desired node. The index of the node matching the query can be retrieved from the QueryItem property, and the QueryCount property will contain the number of nodes matching the query (either 0 or 1).
ITG_QueryMatchItemData	Initiates a query to get the index of the nodes with a particular value for ItemData. The QueryData property must previously have been set to the desired ItemData value. The index of the first node matching the query can be retrieved from the QueryItem property, and the QueryCount property will contain the number of nodes matching the query.

ITG_QueryMatchConnectionId	Initiates a query to get the connection with a particular ConnectionId. The QueryData property must previously have been set to the ConnectionId of the desired connection. If a connection matches the query, it will be made the current connection, and the QueryCount property will be set to the total number of connections matching the query (either 0 or 1).
ITG_QueryMatchConnectionData	Initiates a query to get the connections with a particular value for ConnectionData. The QueryData property must previously have been set to the desired ConnectionData value. The first connection matching the query will be made the current connection, and the QueryCount property will be set to the total number of connections matching the query.

If QueryState is read, it will be either **True** or **False**, depending on whether the node or connection retrieved by the current query is valid.

Data Type

Integer (Enumerated)

Redraw Property (1.1)

Applies To

ITGraph

Description

Specifies whether the graph will be updated to reflect changes to its structure.

Usage

[*form*.]*control*.**Redraw**[= *boolean*]

Read/Write at run time and design time.

Setting

The Redraw settings are:

Setting	Description
True	(Default) Update the graph as changes are made.
False	Don't update the graph when changes are made.

Remarks

The Redraw flag is useful to delay updating of the graph when several changes are being done at one time. When Redraw is set to **True**, any changes previously done will take effect on the graph, and it will be redrawn. When a lengthy operation is being performed (e.g. creating a graph by program control), first set Redraw to **False**, then perform the graph modifications, then set Redraw back to **True**.

Data Type

Integer (Boolean)

RemoveFrom Property

Applies To

ITGraph

Description

Remove a connection between two graph nodes.

Usage

[*form*.]*control*.**RemoveFrom** (*from_index*) = *to_index*

Write only at run time. Hidden at design time.

Remarks

If *from_index* and *to_index* are valid indices, the connection from node *from_index* to node *to_index* will be removed from the graph. If the AutoArrange property is set to ITG_AutoArrange, the graph will be recomputed and redrawn.

Data Type

Integer

RubberBand Property

Applies To

ITGraph

Description

Specifies whether rubber-band lines will be shown when dragging connections between nodes or handles.

Usage

[*form*.]*control*.**RubberBand**[= *mode*]

Read/Write at run time and design time.

Remarks

The settings for RubberBand are:

Setting	Description
ITG_RBNone	(Default) No rubber band lines.
ITG_RBItems	Rubber band lines when dragging between nodes.
ITG_RBHandles	Rubber band lines when dragging between handles.
ITG_RBBoth	Rubber band lines when dragging between nodes or handles.

The setting of the RubberBand property merely determines whether or not rubber-band lines will be shown during connection dragging. Setting RubberBand to ITG_RBNone does not prevent ItemConnect messages from being sent, it merely disables display of a rubber-band line.

Data Type

Integer (Enumerated)

SaveAs Property

Applies To

ITGraph

Description

Load a saved ITGraph graph from disk.

Usage

[*form*.]*control*.**SaveAs** = *path*

Write only at run time. Hidden at design time.

Remarks

Setting the SaveAs property to a file path will save the current graph in the specified file, replacing it if it already exists. Properties that affect the way in which the graph is drawn are saved in the graph file. Those that affect event handling are not.

Data Type

String (file path)

SelectedIndex Property (1.2)

Applies To

ITGraph

Description

Index of the currently selected node in the graph.

Usage

index = [*form*.]*control*.**SelectedIndex**

Read/Write at run time. Hidden at design time.

Remarks

The value of the SelectedIndex property will either be a valid node index (greater than or equal to zero), or **-1**. Setting SelectedIndex to the index of a node will cause that node to be drawn with sizing handles. If permitted, the user can the resize that node. A good time to set the SelectedIndex property is in response to an ItemClick event. SelectedIndex should be set to **-1** in response to a Click event, so as to allow deselection.

Data Type

Integer

SelectRectEnabled Property

Applies To

ITGraph

Description

The SelectRectEnabled property is no longer used by ITGraph. It has been subsumed in the new event handling mechanism. See Working with Events for more details.

SelectRectHeight Property

Applies To

ITGraph

Description

Set or retrieve the height of the selection rectangle.

Usage

[*form*.]*control*.**SelectRectHeight**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

SelectRectLeft Property

Applies To

ITGraph

Description

Set or retrieve the left of the selection rectangle.

Usage

[*form*.]*control*.**SelectRectLeft**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

SelectRectTop Property

Applies To

ITGraph

Description

Set or retrieve the top of the selection rectangle.

Usage

[*form*.]*control*.**SelectRectTop**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

SelectRectWidth Property

Applies To

ITGraph

Description

Set or retrieve the width of the selection rectangle.

Usage

[*form*.]*control*.**SelectRectWidth**[= *pixels*]

Read/Write at run time. Hidden at design time.

Data Type

Integer

ShapeCount Property

Applies To

ITGraph

Description

Specifies the number of node shapes.

Usage

count = [*form*.]*control*.**ShapeCount**

Read only at run time. Hidden at design time.

Data Type

Integer

ShapeName Property

Applies To

ITGraph

Description

Specifies the name assigned to a shape.

Usage

string = [*form*.]*control*.**ShapeName** (*index*)

Read only at run time. Hidden at design time.

Remarks

The ShapeName property retrieves the name associated with a shape. The ShapeCount property can be used to determine how many shapes there are. The *index* values range from 0 to ShapeCount-1. The value of ShapeName for an invalid *index* value is an empty string.

Data Type

String

ShapeSelect Property (1.1)

Applies To

ITGraph

Description

Brings up the Select a Shape dialog box, where the user can choose a node shape.

Usage

[*form*.]*control*.**ShapeSelect**[= *shape_index*]

Read/Write at run time. Hidden at design time.

Remarks

Setting ShapeSelect to a valid shape index brings up the Select a Shape dialog box, with the specified shape selected. When done, ShapeSelect will contain the index of the shape selected by the user, or **-1** if no shape was selected.

Data Type

Integer

StoreGraphics Property

Applies To

ITGraph

Description

Specifies whether graphics will be loaded/stored together with a graph.

Usage

[*form*.]*control*.**StoreGraphics**[= *mode*]

Read/Write at run time and design time.

Setting

The StoreGraphics settings are:

Setting	Description
ITG_NoLoadGraphics	Graphics are neither loaded nor saved with a graph.
ITG_NoSaveGraphics	Graphics are loaded with a graph, but not saved.
ITG_SaveUsedGraphics	(Default) Graphics are loaded with a graph. All referenced graphics are saved when a graph is saved.
ITG_SaveRTGraphics	Graphics are loaded with a graph. All referenced graphics and all graphics imported at run time are saved when a graph is saved.
ITG_SaveDTGraphics	Graphics are loaded with a graph. All graphics (run time and design time) are saved when a graph is saved.

Remarks

StoreGraphics controls how graphics are treated by the LoadFrom and SaveAs properties. If set to ITG_SaveUsedGraphics, ITG_SaveRTGraphics or ITG_SaveDTGraphics, then the specified graphics are saved by SaveGraph in the graph file, together with the graph itself. LoadGraph uses the StoreGraphics property to determine whether or not to load any graphics that are saved with the graph. If StoreGraphics is anything other than ITG_NoLoadGraphics, then the graphics are loaded. No graphic will be loaded if there is already a graphic with the same name. Nodes will always try to link to a graphic with the same name as was saved with the graph; if the referenced graphic doesn't exist, the node's ItemGraphic property is set to **0**.

Data Type

Integer (Enumerated)

XSpace Property

Applies To

ITGraph

Description

Specifies the horizontal spacing between graph nodes.

Usage

[*form*.]*control*.**XSpace**[= *integer*]

Read/Write at run time and design time.

Remarks

The exact horizontal spacing between nodes may actually be greater than XSpace, depending on the settings of the ArrangeMode, DrawDir and Gap properties. If the AutoArrange property is set to ITG_AutoArrange, setting XSpace at run time will cause the graph to be recomputed and redrawn to reflect the change in node spacing.

Data Type

Integer

XSpan Property

Applies To

ITGraph

Description

Specifies the width, in pixels, for new nodes in the graph.

Usage

[*form*.]*control*.**XSpan**[= *integer*]

Read/Write at run time and design time.

Remarks

Note that XSpan is used as the default width for new nodes. Changing XSpan does not affect any of the existing nodes. This behavior is different than that of versions prior to 1.2.

Data Type

Integer

YSpace Property

Applies To

ITGraph

Description

Specifies the vertical spacing between graph nodes.

Usage

[*form*.]*control*.**YSpace**[= *integer*]

Read/Write at run time and design time.

Remarks

The exact vertical spacing between nodes may actually be greater than YSpace, depending on the settings of the ArrangeMode, DrawDir and Gap properties. If the AutoArrange property is set to ITG_AutoArrange, setting YSpace at run time will cause the graph to be recomputed and redrawn to reflect the change in node spacing.

Data Type

Integer

YSpan Property

Applies To

ITGraph

Description

Specifies the height, in pixels, for new nodes in the graph.

Usage

[*form*.]*control*.**YSpan**[= *integer*]

Read/Write at run time and design time.

Remarks

Note that YSpan is used as the default height for new nodes. Changing YSpan does not affect any of the existing nodes. This behavior is different than that of versions prior to 1.2.

Data Type

Integer

ZoomSelectRect Property

Applies To

ITGraph

Description

Brings the portion of the graph bounded by the selection rectangle to the center of the ITGraph window. Zoom to fill the window with the selected portion if desired.

Usage

[*form*.]*control*.**ZoomSelectRect** = *boolean*

Write only at run time. Hidden at design time.

Setting

The ZoomSelectRect settings are:

Setting	Description
True	Scale the graph to fit the selection rectangle.
False	Maintain the current scale, center the selection rectangle in the window.

Remarks

The ZoomSelectRect property can be set to **True** in response to a SelectRect event to implement a zoom window for your application. If you wish to center the graph on a particular node, set the selection rectangle to that node's rectangle (ItemXPos, ItemYPos, XSpan and YSpan), then set ZoomSelectRect to **False**. This will bring the node to the center without changing the scale of the graph.

Data Type

Integer (Boolean)

AddItem Method

Adds a new node to the ITGraph graph at run time.

Syntax

*control.***AddItem** *item*[,*index*]

Remarks

The **AddItem** method has these parts:

Part	Description
control	ITGraph control.
item	String expression to be used as the label for the new node in the graph.
index	Optional ListIndex for the new node. An *index* of 0 causes the new node to be added at the beginning of the list.

AddItem adds a new node to an ITGraph graph. The node's List property is set to *item* and the the ItemId property is set to a unique value. The graph's NewIndex property will contain the index of the newly added node or **-1** if there is an error. If the graph's AutoArrange property is set to ITG_AutoArrange, the graph will be recomputed and redrawn to include the new node.

Delphi Users: Delphi's AddItem method only takes a single argument, the name of the new node. To insert a node at a specific index, use the following code:

```
control.AddItem(item);
control.ItemIndex(.control.NewIndex.) := index;
```

Clear Method

Clears an ITGraph graph at run time.

Syntax

*control.***Clear**

Remarks

control is an ITGraph control.

Clear removes all nodes and connections from the graph. No ITGraph properties are affected by this method.

C++ Users: The CVBControl does not support a Clear method. See the demo program for an example of how to call ITGraphs Clear method from C++.

RemoveItem Method

Removes a node from an ITGraph graph at run time.

Syntax

*control.***RemoveItem** *index*

Remarks

The **RemoveItem** method has these parts:

Part	Description
control	ITGraph control.
index	Integer index of the graph node to be removed. The *index* must be greater than or equal to **0** and less than ListCount.

RemoveItem removes a node from an ITGraph graph. Along with the node, all connections to the node will be removed. If the AutoArrange property is set to ITG_AutoArrange, the graph will be recomputed and redrawn to reflect the changes.

Click Event

Example

Applies To

ITGraph

Description

Occurs when the user presses and then releases a mouse button over a blank (i.e. not over a node, handle, or connection) area in an ITGraph window.

Syntax

*ctlname*_**Click** (*Index* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

Button indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse click, in graph coordinates. A useful application of this event is to add a node to the graph at the location that the user clicks. Example shows how a node could be added to the graph, centered at the location clicked by the user.

DblClick Event

Example

Applies To

ITGraph

Description

Occurs when the user double-clicks the left mouse button over a blank (i.e. not over a node, handle, or connection) area in an ITGraph window.

Syntax

*ctlname*_**DblClick** (*Index* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

Button indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse click, in graph coordinates. A useful application of this event is to return from a zoom to a lower drawing scale as in Example.

Example: Adding A Node By Clicking

The following code shows how you might respond to a Click event by adding a node to the graph at the location that was clicked. If you allow users to add nodes at any location in the graph, the AutoArrange property should be set to ITG_ManualArrange to prevent the graph from being recomputed when new nodes are added.

```
Sub ITGraph1_Click (Button As Integer, Shift As Integer, X As Integer,
Y As Integer)
  ITGraph1.AddItem "<New Node>"
  ITGraph1.ItemXpos(ITGraph1.NewIndex) = X - (ITGraph1.XSpan/2)
  ITGraph1.ItemYpos(ITGraph1.NewIndex) = Y - (ITGraph1.YSpan/2)
End Sub
```

DragDrop Event

Applies To

ITGraph

Description

Occurs when a drag-and-drop operation is completed as a result of dragging a control over an ITGraph control and releasing the mouse button; or using the Drag method with its *action* argument = 2 (Drop).

Syntax

*ctlname*_**DragDrop** ([*Index* **As Integer**,] *Source* **As Control**, *X* **As Single**, *Y* **As Single**)

Remarks

Refer to the standard Visual Basic description of the DragDrop event for details. The ITGraph DragDrop event is the same except that the *X* and *Y* values are graph coordinates scaled to twips. You can determine the drop point in graph coordinates by dividing *X* and *Y* by Screen.TwipsPerPixelX and Screen.TwipsPerPixelY respectively.

DragOver Event

Applies To

ITGraph

Description

Occurs when a drag-and-drop operation is in progress. You can use this event to monitor when the mouse pointer enters, leaves, or is directly over a valid target. The mouse pointer position determines which target object receives this event.

Syntax

*ctlname*_**DragOver** ([*Index* **As Integer**,] *Source* **As Control**, *X* **As Single**, *Y* **As Single**, *State* **As Integer**)

Remarks

Refer to the standard Visual Basic description of the DragOver event for details. The ITGraph DragOver event is the same except that the *X* and *Y* values are graph coordinates scaled to twips. You can determine the drop point in graph coordinates by dividing *X* and *Y* by Screen.TwipsPerPixelX and Screen.TwipsPerPixelY respectively.

ItemClick Event

Applies To

ITGraph

Description

Occurs when the user presses and then releases the left mouse button over a node or handle in the graph.

Syntax

*ctlname*_**ItemClick** (*Index* **As Integer**, *ItemIx* **As Integer**, *ItemHandle* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

ItemIx identifies the node that was clicked, or **-1** if the user didn't click on a node or handle. If a handle was clicked, *ItemHandle* specifies which handle was clicked, otherwise it is **0**. Note that if the DrawHandles property is **False** or ArrangeMode property is not ITG_ModeFlowChart, there will be no handles shown on the graph and therefore *ItemHandle* will always be **0**. *Button* indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

ItemConnect Event

Applies To

ITGraph

Description

Occurs when the user presses the left mouse button while the mouse is over a node or handle, drags the mouse to another node or handle, and releases the mouse button.

Syntax

*ctlname*_**ItemConnect** (*Index* **As Integer**, *FromIx* **As Integer**, *FromHandle* **As Integer**, *ToIx* **As Integer**, *ToHandle* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**)

Remarks

Occurs when the user presses the left mouse button while the mouse is over a node or handle, drags the mouse to another node or handle, and releases the mouse button. *FromIx* identifies the starting node and *ToIx* the ending node. If the mouse was dragged between two handles, *FromHandle* and *ToHandle* identify the selected handles corresponding to nodes *FromIx* and *ToIx*, respectively. If the mouse was dragged between two nodes, *FromHandle* and *ToHandle* will both be **0**. Note that if the DrawHandles property is **False** or ArrangeMode property is not ITG_ModeFlowChart, there will be no handles shown on the graph and therefore *FromHandle* and *ToHandle* will always be **0**. The setting if the RubberBand property determines whether rubber band lines will be shown while a connection is being drawn between two nodes or handles.

ItemDblClick Event

Applies To

ITGraph

Description

Occurs when the user presses and then releases the left mouse button twice over a node or handle in the graph.

Syntax

*ctlname*_**ItemDblClick** (*Index* **As Integer**, *ItemIx* **As Integer**, *ItemHandle* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

ItemIx identifies the node that was clicked, or **-1** if the user double-clicked on a blank area in the graph. If a handle was clicked, *ItemHandle* specifies which handle was clicked, otherwise it is **0**. Note that if the DrawHandles property is **False** or ArrangeMode property is not ITG_ModeFlowChart, there will be no handles shown on the graph and therefore *ItemHandle* will always be **0**. *Button* indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

ItemDrag Event (1.2)

Applies To

ITGraph

Description

Occurs after an item is dragged to a new location in the graph.

Syntax

*ctlname*_**ItemDrag** ([*Index* **As Integer**,] *X0* **As Integer**, *Y0* **As Integer**, *X1* **As Integer**, *X1* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**)

Remarks

Specifies the initial (*X0*, *Y0*) and final (*X1*, *Y1*) values for a nodes ItemXpos and ItemYpos properties when a node is dragged. The drag can effectively be cancelled by setting the nodes ItemXpos and ItemYpos properties to the *X0* and *Y0* parameters, respectively. *Button* indicates the button which initiated the dragging and *Shift* specifies the state of the Shift, Ctrl and Alt keys when the drag was initiated.

ItemMouseMove Event

Applies To

ITGraph

Description

Occurs when the mouse passes over a node or handle in the graph.

Syntax

*ctlname*_**ItemMouseMove** (*Index* **As Integer**, *ItemIx* **As Integer**, *ItemHandle* **As Integer**, *IsDragging* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

ItemIx identifies the node over which the mouse moved, or **-1** if the cursor is not over a node or handle. *ItemHandle* identifies the handle under the cursor, or **0** if the mouse is not over a handle. If a drag operation is in progress (i.e. one of the mouse buttons is down), *IsDragging* will be **True**. If the DrawHandles property is **False** or ArrangeMode property is not ITG_ModeFlowChart, there will be no handles shown on the graph and therefore *ItemHandle* will always be **0**. The ItemMouseMove event can be used to set the cursor, change a node's properties, or retrieve a node's properties when the mouse passes over it. If *IsDragging* is **True**, then *Button* indicates which mouse button initiated the drag and *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

ItemResize Event (1.2)

Applies To

ITGraph

Description

Occurs after an item is resized by the user.

Syntax

*ctlname*_**ItemResize** ([*Index* **As Integer**,] *L0* **As Integer**, *T0* **As Integer**, *W0* **As Integer**, *H0* **As Integer**, *L1* **As Integer**, *T1* **As Integer**, *W1* **As Integer**, *H1* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**)

Remarks

Specifies the initial (*L0*, *T0*, *W0*, *H0*) and final (*L1*, *T1*, *W1*, *H1*) values for a nodes ItemXpos, ItemYpos, ItemWidth and Height properties when a node is resized by a user. The node can be returned to its original size by setting ItemXpos, ItemYpos, ItemWidth and ItemHeight to the *L0*,*T0*, *W0*, and *H0* parameters, respectively. *Button* indicates the button which initiated the resize and *Shift* specifies the state of the Shift, Ctrl and Alt keys when the resize was initiated.

LineClick Event

Applies To

ITGraph

Description

Occurs when the user presses and releases the left mouse button on a connection in the graph.

Syntax

*ctlname*_**LineClick** (*Index* **As Integer**, *FromIx* **As Integer**, *ToIx* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

FromIx identifies the source of the connection and *ToIx* identifies the target of the connection. If the user clicks in a location which could be more than one connection, *FromIx*, *ToIx*, or both will be **-1**. If only *ToIx* is **-1**, then the user clicked on a connection/connections the go to multiple targets. The application may want to provide a selection box for the user to choose a target. A query on *FromIndex* with QueryState set to ITG_QueryTargets can be used to determine the possible targets. *Button* indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

LineDblClick Event

Applies To

ITGraph

Description

Occurs when the user double-clicks the left mouse button on a connection in the graph.

Syntax

*ctlname*_**LineDblClick** (*Index* **As Integer**, *FromIx* **As Integer**, *ToIx* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

FromIx identifies the source of the connection and *ToIx* identifies the target of the connection. If the user double-clicks in a location which could be more than one connection, *FromIx*, *ToIx*, or both will be **-1**. If only *ToIx* is **-1**, then the user clicked on a connection/connections the go to multiple targets. The application may want to provide a selection box for the user to choose a target. A query on *FromIndex* with QueryState set to ITG_QueryTargets can be used to determine the possible targets. *Button* indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

MouseDown Event (1.2)

Applies To

ITGraph

Description

Occurs when the user presses a mouse button in an ITGraph window.

Syntax

*ctlname*_**MouseDown** (*Index* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

Button indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates. A useful application of this event is to initiate a popup menu.

MouseUp Event (1.2)

Applies To

ITGraph

Description

Occurs when the user releases a mouse button in an ITGraph window.

Syntax

*ctlname*_**MouseUp** (*Index* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**, *X* **As Integer**, *Y* **As Integer**)

Remarks

Button indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys. *X* and *Y* give the position of the mouse, in graph coordinates.

SelectRect Event

Applies To

ITGraph

Description

Occurs when the user drags a selection rectangle in the graph.

Syntax

*ctlname*_**SelectRect** (*Index* **As Integer**, *L* **As Integer**, *T* **As Integer**, *W* **As Integer**, *H* **As Integer**, *Button* **As Integer**, *Shift* **As Integer**)

Remarks

L, *T*, *W*, and *H* identify the left, top, width and height of the selection rectangle, respectively - in graph coordinates. This event is sent when the user presses a mouse button, drags the mouse and then releases it. A dashed selection rectangle is drawn as the user drags the mouse. See the Working With Events section for a description of how to enable dragging of a selection rectangle. When this event is generated, the SelectRectLeft, SelectRectTop, SelectRectWidth and SelectRectHeight properties will be set to the same values as the *L*, *T*, *W* and *H* event parameters. In response to the SelectRect event, the ZoomSelectRect property can be used to zoom in on the selected area. The ITG_QuerySelectRectItems setting of the QueryState property can be used to determine which nodes fall inside the selection rectangle. *Button* indicates which mouse button generated the event. *Shift* specifies the state of the Alt, Ctrl and Shift keys.

Setting up the ITGraph Control

There are several important properties which should be set at design time for an ITGraph control. These properties control display characteristics of the entire graph:

At design time, the ITGraph window shows a default node in the center of the screen. You should adjust the properties described below so that the colors and styles are as you desire.

The AutoArrange property should be set to either ITG_ManualArrange or ITG_AutoArrange. Set it to ITG_AutoArrange if you want the graph to be updated when new nodes or connections are added. If you only want the graph to be updated when explicitly required by your application, then set AutoArrange to ITG_ManualArrange at design time. You can set it to ITG_ArrangeNow at runtime to arrange the graph when needed.

Set the ArrangeMode, DrawDir and DrawBackLinks properties according to the layout needs of your application. Even if you don't use the automatic layout features, these properties will affect the way in which connections are drawn between nodes. The XSpan, YSpan, and Gap properties affect the spacing of nodes and connections. The exact interpretation of these properties depends on the settings of ArrangeMode and DrawDir. The DrawScale property should probably be set either to **100** for WYSIWYG drawing or **0** to keep everything in the control window.

Set the XSpace, YSpace, BackColor, FillColor, ForeColor, and LineWidth properties to determine the default sizes and colors for nodes and connections. The DrawArrows, DrawColored and DrawItemLabels properties further affect the components that will be shown in the graph.

Use the AutoMouseEvents and RubberBand properties to configure how mouse events will be processed by your application. Using these options, you can implement node dragging, zooming, multiple-node selections, and rubber band lines for connections, among other possibilities.

Adding Nodes to the ITGraph Control

New nodes are added to an ITGraph at run time only via the AddItem method. This method takes as an argument the initial value for the node's List property. An optional second argument specifies the position to insert the new node in the node list. This is particularly useful for the tree layout, where the appearance of the graph depends on the insertion order. The AddItem method sets the NewIndex property of the control to the index of the new node in the graph. Note that this index value is valid until another node is added or deleted from the graph, at which time it may be shifted; if you need a permanent reference to a node, the ItemId property gives you an identifier which will never be changed. Alternatively, you can use the ItemData property to associate your own data with a node.

When a new node is added, several other properties should be set before the graph is redrawn. The ItemShape, ItemFillColor, ItemBorderColor and ItemTextColor properties control the display characteristics of the node.

If your application uses manual placement of nodes, then you should also set the ItemXpos and ItemYpos properties at this time. These properties are particularly useful if you implement a drag-drop capability where the user can drop nodes directly onto the graph as in Example.

If you need to remove a node from the graph, use the RemoveItem method, supplying the index of the node you wish to remove.

Example: Adding Nodes By Drag-Drop

The following code shows how you might allow a user to drop a node onto the ITGraph control. To prevent the new node from being moved, the AutoArrange property must be set to ITG_ManualArrange. Notice the conversion at the beginning from Twips to Pixels for the drop coordinates. The call to the Refresh method at the end causes the graph to be redrawn (although none of the nodes are moved).

```
Sub ITGraph1_DragDrop (Source As Control, x As Single, y As Single)
  x = x / Screen.TwipsPerPixelX
  y = y / Screen.TwipsPerPixelY

  ITGraph1.AddItem "New Node"
  ITGraph1.ItemXpos(ITGraph1.NewIndex) = x - ITGraph1.XSpan / 2
  ITGraph1.ItemYpos(ITGraph1.NewIndex) = y - ITGraph1.YSpan / 2
  ITGraph1.ItemShape(ITGraph1.NewIndex) = ITG_ShapeRectangle
  ITGraph1.ItemBorderColor(ITGraph1.NewIndex) = &H000000
  ITGraph1.ItemFillColor(ITGraph1.NewIndex) = &H0000FF
  ITGraph1.ItemTextColor(ITGraph1.NewIndex) = &H00FF00
  ITGraph1.Refresh
End Sub
```

Working with Connections

Example

A connection can be added to the graph by setting the ConnectTo property of the source to the index of the new target. The new connection will be added to the graph and made the current connection and the graph will be redrawn to include the additional connection. This is now the time to change the connection's properties from the default values. If ArrangeMode is ITG_ModeFlowChart, then you may want to set the ConnectFromHandle and ConnectToHandle to specify the way the connection should be drawn; the DrawHandles property determines whether handles will be shown on the graph. The ConnectionColor property can be set to give the connection its own color, or can be set to **-1** to use the setting of ForeColor. The ConnectionArrow property determines the type of arrowhead (if any) to be drawn at the end of the connection; the DrawArrows property determines whether any arrows will be drawn at all. A label can be assigned to the connection by setting the ConnectionLabel property and an application-defined long integer can be associated with the connection *via* the ConnectionData property. The DrawConnLabels property determines whether or not to draw labels on connections that have them.

When ArrangeMode is ITG_ModeHierarchy or ITG_ModeTree, connections are classified into two categories. Regular connections proceed according to the specified value of DrawDir. Backlinks, which only occur in graphs with circularities, proceed in the opposite direction. Also, a connection from a node to itself is considered a backlink. The distinction between these two types of connections may affect how the graph is drawn. The setting of the DrawBackLinks property determines whether or not backlinks will be drawn, and in what style (solid or dashed lines).

To access or retrieve the properties of a connection, it is necessary to make it the current connection. There are several ways this can be done. If you already know the source and target indices (e.g. from an event), then you can make the connection current by setting the ConnectTo property of the source to the index of the target. Since this connection is already in the graph, ConnectTo will not add a new connection or redraw the graph. Alternatively, you can query connections belonging to a node in the graph. First, set the QueryItem property to the index of the reference node. If you are interested in connections related to a particular handle of the node, set the QueryItemHandle property to the index of the desired handle. Start your query by setting QueryState to the code for the type of query you want. This will start a query to get all connections or all incoming or outgoing connections from the node or handle. The QueryCount property tells you how many connections match the query criteria. QueryState tells you if the current connection is valid. If it is, you can access the properties of that connection as described above. QueryState is also used to iterate through the matching connections. This is done by setting QueryState to ITG_Iterate, then checking its value to see if the current connection is valid (i.e. another connection matched the criteria) or invalid (i.e. you have seen all the matching connections).

A connection can be removed by setting the RemoveFrom of the source to the index of the target. Additionally, if a node is removed by the RemoveItem method, any connections into or out of that node will also be removed.

Three ITGraph events are relevant for working with connections. The LineClick event is fired when the user clicks on a connection in the graph with the left mouse button. The LineDblClick event is fired when the user double-clicks on a line with the left mouse button. You may want to respond to one or both of these events by selecting the connection (perhaps by changing its color), returning information on the selected connection, or some other appropriate action. Note that the LineClick and LineDblClick events return the source and target indices of the connection. These can be used to select the connection with the ConnectTo property. If more than one connection is clicked at the same time, the LineClick and LineDblClick events will indicate this by providing **-1** in place of node indices. Your application can then use the provided index to iterate through the possible matching connections.

The ItemConnect event is fired when a user clicks on a node with the left mouse button, drags the mouse to a different node, and releases the mouse button. An appropriate response to this event may be to add a connection between the two nodes. The ItemConnect event provides the indices of the selected nodes as well as handle indices. If the user drags between two nodes, the handle indices will both be **0**. If the user drags between two handles, the indices will indicate which handles were chosen. The RubberBand property determines whether connection dragging will be accompanied by a rubber

band line. The RubberBand property does not affect whether or not ItemConnect events will be sent; it is the application's responsibility to respond (or not respond) appropriately to the ItemConnect event.

Example: Finding the Targets of a Node

The following code shows how you might respond to an ItemDblClick event by placing the names of all targets of the selected node in a ListBox control named TargList. If a node is double-clicked, all targets of the node are listed. If a handle is double-clicked, only those connections emmanating from the handle are shown.

```
Sub ITGraph1_ItemDblClick (ItemIx As Integer, ItemHandle As Integer,
Button As Integer, Shift As Integer, X As Integer, Y As Integer)
  Dim targIndex As Integer

  TargList.Clear
  ITGraph1.QueryItem = ItemIx
  ITGraph1.QueryItemHandle = ItemHandle
  ITGraph1.QueryState = ITG_QueryGetTargets
  While ITGraph1.QueryState
    targIndex = ITGraph1.ConnectToIndex
    TargList.AddItem ITGraph1.List(targIndex)
    ITGraph1.QueryState = ITG_QueryIterate
  Wend
End Sub
```

The first step in the code is to specify the query parameters. The QueryItem property is set to the index of the node that was double-clicked and the QueryItemHandle property is set to the handle that was clicked (or 0 if a node was clicked). The query for targets of the selected node is then initiated by setting QueryState to ITG_QueryGetTargets. In a while loop, the QueryState property is checked. If true, then the query has returned a connection whose properties (in this case the target index) can be read. At the end of the while loop, QueryState is set to ITG_QueryIterate to get the next connection in the query.

Similarly, the sources of a node can be found simply by initializing QueryState to ITG_QueryGetSources instead of ITG_QueryGetTargets as above.

Current Connection

There can be at most one current connection. Several operations that affect connection properties apply to the current connection. When a new connection is added or selected (by setting the ConnectTo property) or retrieved by a query (using the QueryXXX properties) it becomes the current connection.

Source

If there is a connection from one node to another, the first node is a source of the second node. The ConnectTo property is used to make a node a source of another node.

Target

If there is a connection from one node to another, the second node is a target of the first node. The ConnectTo property is used to add a target to a node.

File Operations

The ITGraph control includes two special properties used for saving and loading graphs from the disk. The SaveAs property allows you to specify a path to save the graph. You may want to use the .ITG extension for ITGraph files to distinguish them from other types of files. Previously saved files can be loaded by using the LoadFrom method. The CMDialog control can be used to query the user for a path to save a file or for a graph to be loaded.

All ITGraph properties that affect the way a graph is drawn are saved in the graph file. Those properties that affect the control's behavior are not saved. Properties NOT saved are: Align, AutoArrange, AutoMouseEvent, AutoMouseEvents, BorderStyle, CtlName, DragIcon, DragItems, DragMode, Enabled, GraphicAllowImport, Height, HelpContextID, hWnd, Index, IsDirty, Left, MousePointer, Name, Parent, PrintHeader, PrintRectHeight, PrintRectLeft, PrintRectTop, PrintRectWidth, Redraw, RubberBand, SelectedIndex, SelectRectEnabled, SelectRectHeight, SelectRectLeft, SelectRectTop, SelectRectWidth, StoreGraphics, TabIndex, TabStop, Tag, Top, Visible, Width and ZoomSelectRect.

Note that the AutoArrange property is not saved in the graph file. If a graph is loaded when the ITGraph control's AutoArrange property is set to ITG_AutoArrange, the graph will be rearranged.

The StoreGraphics property determines whether bitmaps and metafiles will be saved in a graph file, or loaded from the file. This property's setting affects both loading and saving.

Printing

Printing an ITGraph graph can be accomplished by setting the PrintGraph property to the destination device contest. Since color graphs dont look good, setting the DrawColored flag to **False** before printing will optimize the graph for black and white printing. The PrintHeader property allows you to customize the headers and footers placed on pages printed by PrintGraph.

The PrintToDC and PrintToWnd properties can be used to "print" an image of the graph or portion of the graph into another Visual Basic control or an application window or report. The first function is given a handle to a device context while the second is given a handle to a window. Many Visual Basic controls provide an hWnd property that can be used to retrieve the control's window handle. Prior to setting the PrintToDC or PrintToWnd properties, the selection rectangle should be set to a rectangle in the graph. This is the portion of the graph that will be printed. If SelectRectWidth and SelectRectHeight are both set to **0**, then the entire graph will be printed. The print rectangle, defined by PrintRectLeft, PrintRectTop, PrintRectWidth and PrintRectHeight, specifies a viewport in the destination device context or window. The portion of the graph delineated by the selection rectangle will be scaled to fit the print rectangle.

The selection rectangle's bounds can be set using the SelectRectLeft, SelectRectTop, SelectRectWidth and SelectRectHeight properties.

Note: If you set PrintToWnd or PrintToDC to **0**, the selection rectangle will be set to the bounding rectangle of the graph.

Using the Selection Rectangle

Example1 Example2 Example3

Associated with the graph is a current selection rectangle. The selection rectangle can be set in response to a user SelectRect event or any other time by setting the SelectRectLeft, SelectRectTop, SelectRectWidth and SelectRectHeight properties.

One useful application of the selection rectangle is to implement a zoom window. When the user drags a rectangle with the left or right mouse button, the graph will be zoomed to show the area that was selected. To implement this capability, dragging of a selection rectangle must be enabled as described in Working With Events. Next, the application must respond to the SelectRect event by setting the ZoomSelectRect to **1**, causing the selected area to fill the ITGraph control window. This is illustrated by **Example1**.

Another application of the select rectangle is to allow the user to select a node from a list of node names, and then center the graph on that node. This is particularly useful for large graphs. Such a capability can be implemented by setting the selection rectangle bounds to the bounds of the desired node and then setting the ZoomSelectRect property to **0**, which centers the selection rectangle in the graph window, but doesn't change the scale. This is illustrated by **Example2**.

A query can be used to perform a particular action on all the nodes that the selection rectangle intersects. The ITG_QuerySelectRectItems query is provided for this purpose. **Example3** shows how this is done.

Additionally, the selection rectangle defines the portion of the graph to be printed by the PrintToDC and PrintToWnd properties.

Example: Implementing a Zoom Window

The following code shows how you might implement a zoom window in response to a SelectRect event. Note that you must set the SelectRectEnabled property to **True** in order for the user to be able to drag a selection rectangle and for the SelectRect event to be called.

```
Sub ITGraph1_SelectRect (L As Integer, T As Integer, W As Integer,
H As Integer, Button As Integer, Shift As Integer)
  ITGraph1.ZoomSelectRect = 1
End Sub
```

That's all there is to it! The user can now select an area of the graph to zoom in on. You may want to provide some way to zoom out again. A convenient place to do this is in response to the DblClick event.

```
Sub ITGraph1_DblClick (Button As Integer, Shift As Integer, X As Integer,
Y As Integer)
  ITGraph1.DrawScale = 0   'Fit to screen
End Sub
```

In our example, we respond to a double-click on the ITGraph screen by setting the DrawScale property to **0**, which causes the entire graph to fit on the screen.

Example: Centering On A Node

The following code shows a procedure that will center the graph on a node whose index is passed to the procedure.

```
Sub CenterOnNode (NodeIx As Integer)
  ITGraph1.SelectRectLeft = ITGraph1.ItemXpos(NodeIx)
  ITGraph1.SelectRectTop = ITGraph1.ItemYpos(NodeIx)
  ITGraph1.SelectRectWidth = ITGraph1.ItemWidth(NodeIx)
  ITGraph1.SelectRectHeight = ITGraph1.ItemHeight(NodeIx)
  ITGraph1.ZoomSelectRect = 0
End Sub
```

Setting ZoomSelectRect to **0** causes the selection rectangle to be centered in the ITGraph window, but without changing the scale of the graph.

Example: Marking Selected Nodes

The following code shows how you might respond to a SelectRect event by changing the interior color of all the nodes in the rectangle to red. Note that you must set the SelectRectEnabled property to **True** in order for the user to be able to drag a selection rectangle and for the SelectRect event to be called.

```
Sub ITGraph1_SelectRect (L As Integer, T As Integer, W As Integer,
H As Integer, Button As Integer, Shift As Integer)
  Const COLOR_RED = &HFF
  ITGraph1.QueryState = ITG_QuerySelectRectItems
  While ITGraph1.QueryState
    ITGraph1.ItemFillColor(ITGraph1.QueryItem) = COLOR_RED
    ITGraph1.QueryState = ITG_Iterate
  Wend
End Sub
```

Setting QueryState to ITG_QuerySelectRectItems initiates a query to get all the nodes intersected by the current selection rectangle. Since this is being done in response to a SelectRect event, the selection rectangle will be the area selected by the user. The selection rectangle could also be set by the application if desired. QueryState will be true as long as node are found. The QueryItem property can be used to get the index of the last node found by the query. The next node is found by setting QueryState to ITG_Iterate.

Import Graphic Dialog (1.1)

Applies To

ITGraph

Purpose

Used to select a graphic to be imported at design time or run time.

Description

The "Import Graphic" dialog box appears when the Select button is pressed on the New Graphic Information or Information for Graphic dialog boxes.

The Import Graphic dialog box is essentially a file selection dialog, for choosing a graphic file to be imported as an ITGraph graphic. The controls on the dialog are as follows:

Control	Description
File Name	The "File Name" list box contains a list of the files in the specified drive and directory that are of the desired file types. Double-clicking on a file in the list box will close the Import Graphic dialog box, and place the path of the selected file into the Path box of the Graphic Information dialog box. The File Name edit box contains the path of the selected file, relative to the current directory. A path can be typed directly in this dialog box if desired.
List Files of Type	The "List Files of Type" combo box can be used to limit the files displayed in the File Name list box to files of a certain type. The options are "Graphic Files," "Windows Bitmaps" and "Windows Metafiles."
Directories	The "Directories" list box contains the directory tree for the selected drive. Double clicking on a directory name expands the tree to show the subdirectories, and sets the current directory (shown above the list box) to the selected directory.
Drives	The "Drives" combo box contains a list of the drives on your system. The current drive is shown in the edit box.
OK	Close the Import Graphic dialog box and import the graphic file specified in the File Name edit box.
Cancel	Close the Import Graphic dialog box without importing a new graphic.

New Graphic Information Dialog (1.1)
Information for Graphic # of # Dialog (1.1)

Applies To

ITGraph

Purpose

Used to specify or modify the name and path for an ITGraph graphic. For a new graphic, the "New Graphic Information" dialog box is shown. The "Information for Graphic # of #" dialog box is shown when an existing graphic is being modified.

Description

The "New Graphic Information" dialog box appears when the New... button is pressed on the ITGraph Graphic Table or Select a Graphic dialog boxes. The "Information for Graphic # of #" dialog box appears when a graphic is double clicked in the "ITGraph Graphic Table" dialog box, or when the Modify button is pressed on the "Select a Graphic" dialog box.

The controls on these dialog boxes are as follows:

Control	Description
Name	Specifies the name of the graphic. This is the name referenced by the graphic's GraphicName property. Every graphic must have a name, and these names must be unique. FOr the "Information for Graphic # of #" dialog box, this name will be filled in with the name of the graphic being modified.
Path	Used to specify the path of a graphic image to be imported. If no image should be associated with the graphic, the Path box can be left empty. Since the original image path is not stored with a graphic, the Path box will always be initially empty when a graphic is modified.
Select	Brings up the Import Graphic dialog box, for selection of a graphic file. The selected path is placed in the Path box. If a file is selected and the Name box is empty, the name of the selected file will also be placed in the Name box.
Accept	Close the Import Graphic dialog box and import or modify the graphic. This button will be dimmed if there is no entry in the Name box. If there is already a graphic with the same name, a message box will appear, and you will be requried to change the name of the graphic before importing.
Cancel	Close the Graphic Information dialog box without importing or modifying a graphic.

ITGraph Graphic Table Dialog (1.1)

Applies To

ITGraph

Purpose

Used to view, add, modify and delete ITGraph graphics at design time.

Description

The ITGraph Graphic Table dialog box is accessed by double clicking on the Graphics property in the Visual Basic Properties window. The controls on the dialog are as follows:

Control	Description
Graphics	The Graphics list box lists the names of all the graphics currently loaded. They are listed in the order they were added. The first entry, "<No Graphic>" is always present and cannot be modified or deleted. Above the list box is a line listing the number of graphics in the list. When a graphic is selected in the list box, it is drawn to the right of the list box. Double clicking on a graphic (other than "<No Graphic>") brings up the Information for Graphic # of # dialog box, where the graphic's information can be modified.
Remove	Removes the selected graphic. This button will be dimmed if no graphic is selected, or if "<No Graphic>" is selected.
New...	Brings up the New Graphic Information dialog box, for the definition of a new graphic to be imported.
Done	Closes the ITGraph Graphic Table dialog box.

Select a Graphic Dialog (1.1)

Applies To

ITGraph

Purpose

Used to view, select, add, modify and delete ITGraph graphics at run time.

Description

The Select a Graphic dialog box is brought up at run time by setting the GraphicSelect propery. If set to a valid graphic index, the specified graphic will be initially selected in the dialog box. The controls on the dialog are as follows:

Control	Description
Graphics	The Graphics list box alphabetically lists the names of all the graphics currently loaded. The entry, "<No Graphic>" is always present and cannot be modified or deleted. Above the list box is a line listing the number of graphics in the list. When a graphic is selected in the list box, it is drawn to the right of the list box. The Remove and Modify buttons are lit or dimmed according to which graphic is selected and the current setting of the GraphicAllowImport property. Double clicking on a graphic selects that graphic and closes the dialog box. The selected graphic's index is stored as the value of the GraphicSelect property, for retrieval by the application.
Remove	Removes the selected graphic. This button will be dimmed if no graphic is selected, if "<No Graphic>" is selected, or if the setting of GraphicAllowImport precludes removal of the selected graphic. If GraphicAllowImport is set to ITG_NoImportGraphics, the Remove button will not be shown.
Modify	Brings up the Information for Graphic # of # dialog box to modify the selected graphic. This button will be dimmed if no graphic is selected, if "<No Graphic>" is selected, or if the setting of GraphicAllowImport precludes modification of the selected graphic. If GraphicAllowImport is set to ITG_NoImportGraphics, the Modify button will not be shown.
New...	Brings up the New Graphic Information dialog box, for the definition of a new graphic to be imported. If GraphicAllowImport is set to ITG_NoImportGraphics, the New... button will not be shown.
Accept	Closes the Select a Graphic dialog box, setting the GraphicSelect property to the index of the selected graphic.
Cancel	Closes the Select a Graphic dialog box without selecting any graphic. The GraphicSelect property will be set to **-1** to indicate that Cancel was pressed.

Select a Shape Dialog (1.1)

Applies To

ITGraph

Purpose

Used to view and select ITGraph shapes at run time.

Description

The Select a Shape dialog box is brought up at run time by setting the ShapeSelect propery. If set to a valid shape index, the specified shape will be initially selected in the dialog box. The controls on the dialog are as follows:

Control	Description
Graphics	The Shapes list box alphabetically lists the names of all the ITGraph node shapes. The entry, "<No Shape>" indicates that no shape will be drawn for the node. This is useful if the node is being drawn as a graphic instead. Above the list box is a line listing the number of shapes in the list. When a shape is selected in the list box, it is drawn to the right of the list box. Double clicking on a shape selects that shape and closes the dialog box. The selected shape's index is stored as the value of the ShapeSelect property, for retrieval by the application.
Accept	Closes the Select a Shape dialog box, setting the ShapeSelect property to the index of the selected shape.
Cancel	Closes the Select a Shape dialog box without selecting any graphic. The ShapeSelect property will be set to **-1** to indicate that Cancel was pressed.

AutoMouseEvents Setup Dialog (1.2)

Applies To

ITGraph

Purpose

Used to configure ITGraph event responses.

Description

The AutoMouseEvents Setup dialog box is brought up at run time by setting the AutoMouseEvents property, or at design time by double clicking on the property in the properties window. The controls on the dialog are as follows:

Control	Description
List Box	The list box at the top of the AutoMouseEvents Setup dialog box lists all the mouse events and their corresponding actions. Each row in the list corresponds to a combination of one of the mouse buttons (left, middle or right) together with any of the modifier keys (shift, control and alt). This information is represented by the first four columns. The first shows the button, while the next three have stars for the selected modifiers (alt in the A column, control in the C column and shift in the S column). The fifth column lists the types of actions that can be initiated by that combination of mouse button and modifier keys. For a more detailed description of the options see Working With Events. Selecting a row in the list will cause the Button, Modifiers and Events groups to be filled in with the information from the selected row.
Button Group	When a row in the list is selected, either Left, Middle or Right will be chosen, depending on the value in the Button column of the selected row. Choosing a different button will cause the appropriate row in the list box to be selected.
Modifiers Group	When a row in the list is selected, the Alt, Ctrl and Shift checkboxes will be checked according to the values in the A, C and S columns of the selected row. Checking and unchecking the boxes will cause the appropriate row in the list box to be selected.
Events Group	Shows which events will be performed for the button/modifiers combination defined for the selected row in the list box. Any combination of the boxes can be checked with the exception of Connect and Drag, which cannot be checked at the same time.
Done	Closes the AutoMouseEvents Setup dialog box.

Working With Events

ITGraph events are generated in response to user actions such as clicking with the mouse, typing on the keyboard, or dragging and dropping an item on the control. Your application determines exactly which events will be handled and what the response will be.

The events list shows all the events generated by ITGraph. Several are standard events, and reference should be made to the Visual Basic documentation for more information. Events which are unique to ITGraph are described in this help file. Four of the standard Visual Basic events have been modified slightly for ITGraph. The Click, DblClick, DragDrop and DragOver events have been extended to include the coordinates (within the graph) of the mouse at the time of the event, and in the case of Click and DblClick, the mouse button which generated the event and the state of the modifier keys (Shift, Ctrl and Alt).

The ItemMouseMove, ItemClick, ItemDblClick, ItemDrag and ItemResize events are generated when a node is passed over by the mouse, clicked, double clicked, dragged to a new location or resized, respectively. LineClick and LineDblClick events are generated when a connection is respectively clicked or double clicked. A SelectRect event is generated when the user drags out a selection rectangle. Again, the events specify the mouse button which caused the event, the state of the modifier keys and any relevant coorinates in the graph.

Most events have two special parameters: *Button* and *Shift*. The first, *Button*, can have three possible values: ITG_LeftButton, ITG_MiddleButton and ITG_RightButton. The *Button* parameter specifies which button caused the event to be generated. The *Shift* parameter contains three flags which indicate the state of the modifier keys. It can be a combination of zero to three of the flags: ITG_ShiftDown, ITG_CtrlDown and ITG_AltDown. Note that the *Button* and *Shift* parameters relate to the initiation of the event and may not correspond to the machine state when the event is actually generated. An ItemDrag event, for example, specifies the mouse button that was pressed down to start the drag and finally released to drop the node in a new position. All intermediate clicking of other mouse buttons is irrelevant. Likewise the *Shift* parameter will specify the state of the modifier keys when the mouse button was first pressed. Henceforth, they can be released without affecting the ItemDrag event.

There are two ways to configure ITGraphs events: at design time through the AutoMouseEvents property and associated dialog box, or at run time through the AutoMouseEvent property. In both cases, what is specified is the types of actions that can occur in response to the user clicking the mouse with a particular combination of modifier keys. Each mouse button and modifier combination can have a different response. Possible actions are summarized in the table below:

Action Flag	Description
None	Dont do anything in response to the button/modifier combination.
MouseEvent	Generate mouse events, depending on where the user clicked. This enables the Click, DblClick, ItemMouseMove, ItemClick, ItemDblClick, LineClick, LineDblClick, MouseDown and MouseUp events.
Connect	If the user clicks on a node or handle, initiate a connection. This enables the ItemConnect event. A rubber-band line will be shown while the mouse is being dragged only if the RubberBand property is appropriately set.
Constrained	In combination with the Drag flag, limits dragging to a single direction. This option will frequently be associated with a Shift key.

Drag	Enables dragging a node in the graph, resulting in an ItemDrag event.
Select	Enables dragging a selection rectangle in graph, resulting in a SelectRect event.
Size	Enables resizing of nodes, generating an ItemResize event. Note that a node must be selected (see SelectedIndex) in order to be resized by the user.

Subject to certain constraints, these options can be combined as desired. The Drag and Connect flags cannot be used together. Drag and Connect take precedence over Select, so if Select and either Drag or Connect are enabled, clicking on a node and dragging will result in an ItemDrag or ItemConnect event rather than a SelectRect event. When a mouse event occurs, the AutoMouseEvent entry is checked for the appropriate button and modifier key combination. If a reasonable action has been specified, processing for the event is initiated. Otherwise, given that the action includes the MouseEvent flag, the next most general handler is checked. The order of search is: (1) original modifier keys, (2) without alt key, (3) without control key, (4) without shift key. If no handlers can be found, the event is ignored.

An example scheme might be to set the Left button to MouseEvent+Connect+Select+Size, the Left+Shift to MouseEvent+Connect+Select+Size+Constrained and all the rest of the left button combinations to MouseEvent, which allows them to pass to the more general handlers. This has the effect of ignoring the Ctrl and Alt keys and treating the Shift button as a movement constrainer. All Middle and Right button actions should be set to None to disable events from those buttons.

Note that a particular ITGraph event may be generated by several different button/modifier combinations. The applications event handler may need to check the *Button* and *Shift* parameters to determine how the event was generated and respond appropriately.