

Installation of NTGraph ActiveX on WIN7 (64bit)

(1) Copy file "NTGraph.ocx" to folder

C:\Windows\SysWOW64

(2) In cmd.exe (command prompt), change address to the above folder, then type to register

```
regsvr32 NTGraph.ocx
```

or type to unregister (if you want to remove it)

```
regsvr32 /u NTGraph.ocx
```

(3) In MFC "Insert ActiveX control", choose NTGraph, enjoy!

<http://www.codeproject.com/Articles/3214/2D-Graph-ActiveX-Control>

[Home](#) [Articles](#) [Quick Answers](#) [Discussions](#) [Learning Zones](#) [Features](#) [Help!](#) [The Lounge](#) [Search](#)[» Desktop Development »](#)
[Miscellaneous » Charting Controls](#)

2D Graph ActiveX Control

Licence
First Posted **19 Nov 2002**
Views **541,343**
Bookmarked **285 times**

See Also

- [More like this](#)
- [More by this author](#)


By **Nikolai Teofilov** | 5 Aug 2003 | [Article](#)

[VC6](#) [VC7](#) [Win2K](#) [WinXP](#) [MFC](#) [Dev](#) [Intermediate](#)

An ActiveX control for 2D data visualisation

[Article](#) [Browse Code](#) [Stats](#) [Revisions](#) [Alternatives](#)



4.90 (156 votes)  354



Is your email address OK? You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please click [here](#) to have a confirmation email sent so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](#).



Download source files - 381 Kb

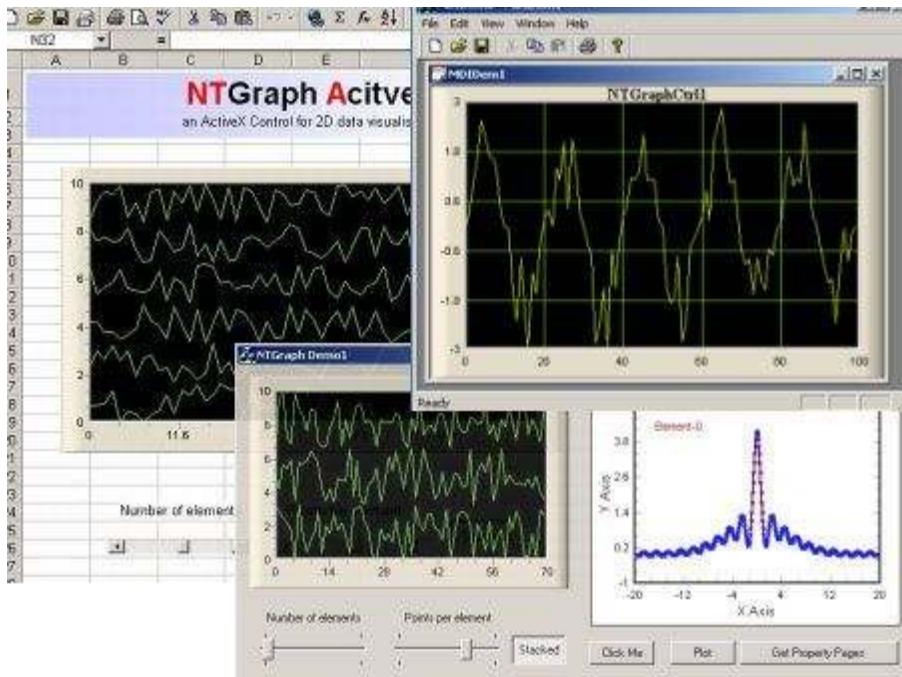


Download demo projects - 200 Kb



Download control binaries - 72 Kb

[Add your own alternative version](#)



Introduction

This is a simple OCX control, which allows you to plot two-dimensional data. Despite the large set of controls that comes with VC++, there is no out-of-the-box control that provides a simple and straightforward 2D data visualization. The [ActiveX control tutorial](#) by

Hot News: [HTML5: A blessing or a curse?](#)
The Code Project Insider. [Free each morning.](#)

Kapil Chaturvedi inspired me to write my own control, mostly because I wanted to customize the source code when needed. Over time, the functionality of the ActiveX control became more elaborate, and finally I made decision to publish what I have in hand.

What Can It Do?

The control is able to plot a large number of points and updating one or more plots on the graph with new data, replacing the old plot with the new plot. Multiple plots with individual properties such as name, line and point style, width, could be customized at runtime. At runtime, the control is capable of displaying its own property pages (double click on the control area or by invoking the ShowProperties method) and showing short help as a result of the user pressing **F1** while the control has the focus. By setting the TrackMode property you should be able to switch between a different modes such as tracking cursor coordinates while moving (left mouse button pressed), zooming, XY-, X-, and Y-panning. Finally the control snapshot could be copied to the clipboard, printed, or saved as a bitmap file.

What doesn't it do?

You cannot plot 3D data, but you can use the [NTGraph3D ATL/STL/OpenGL activeX control](#) to do that :-)

What's New?

- The Log Axes Mode works now, showing the log10 grid, and appropriated labels, it also converts the graph element's data
- The control's snapshot could be now saved as a bitmap file, many thanks to **Robert Harber** for providing the code!
- Added abilities to dynamically creating annotation labels, that can be on different colors, orientations, and also could be hidden or visible.
- Added "Annotations" property page that provides fully access to the annotation list in the real/design mode.
- Added abilities to dynamically drawing of multiply cursors, with a different colors, crosshair styles, floating/fixed, or snapped to the currently selected element!
- Added "Cursors" property page that provides fully access to the cursor list in the real/design mode.
- Added axis formatting, that allows a customization of the bottom and left axis labels.
- Added "Format" property page that provides access to the axis format properties, and a templates for of commonly used data formats such as: Numbers, Exponential, Symbolic, Date, and Time!
- Added Time format for the graph axes. To use it, you should set the `XTime/YTime` property to `True`.

You also have to convert the date/time data to `double` format. The Date/Time format is implemented as a floating-point value, measuring days from midnight, 30 December 1899. So, midnight, 31 December 1899 is represented by 1.0. Similarly, 6 AM, 1 January 1900 is represented by 2.25, and midnight, 29 December 1899 is 1.0. However, 6 AM, 29 December 1899 is 1.25.

For more info refer to MSDN for the class: `COleDateTime`!

How to test the control

You can use the ActiveX Control Test Container, and load the

Related Articles

[2D Graph ActiveX control in C++ with ATL \(no MFC dependency\)](#)

[3D Graph ActiveX Control](#)

[A 2D Graph Component With Zoom Capability](#)

[WPF: A graph control](#)

[Real Time 2D Graph for CE](#)

[A Simple Graph Control](#)

[Autoscaling Graph Control](#)

[Bar Graph Control](#)

[2D Drawing with an OpenGL Control](#)

[GDI+ Plot ActiveX Control](#)

[C2DPushGraph: A Push Graph Control](#)

[C2DPushGraph: A Push Graph Control](#)

[C# 2.0 Graphing Control](#)

[A Complete ActiveX Web Control Tutorial](#)

[Bar Graph Control](#)

[A Graph Tree Drawing Control for WPF](#)

[A WPF Graph Control Library](#)

[2D Multi-Parameter Pie Control](#)

[FontComboBox ActiveX control](#)

[Using 2D controls in a 3D environment](#)

Test.dsm macro from the menu Tools\Macros... You can write your own routines to test the control behavior (look at Test.dsm macro)

How to use the control

To use this OCX control, embed it in an application that supports the use of OCX controls. Microsoft Visual Basic applications, Office applications and applications created with the Microsoft Developer Studio's AppWizard can support the use of OCX controls. There are two files required to use this control. They are:

- NTGraph.hlp -The help file for this control.
- NTGraph.ocx -The NTGraph controls code and data.

Before the ActiveX control can be used in your application, it must be registered as a COM Component in the system registry. This is a self registering control. This means that to register the control in the system registry you only need to have an application load the control and call the control's exported function DllRegisterServer. You can use the REGSVR32 utility or have your setup program do this.

How to use the REGSVR32 utility?

Copy *NTGraph.ocx* to your directory and type:

```
regsvr32 NTGraph.ocx
```

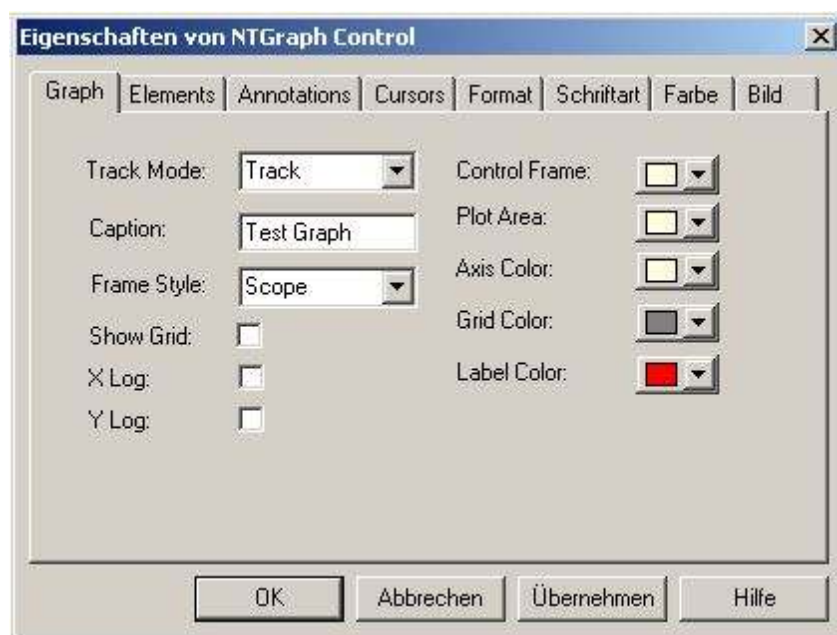
```
regsvr32 /u NTGraph.ocx (Unregister server)
```

Customizing The Control

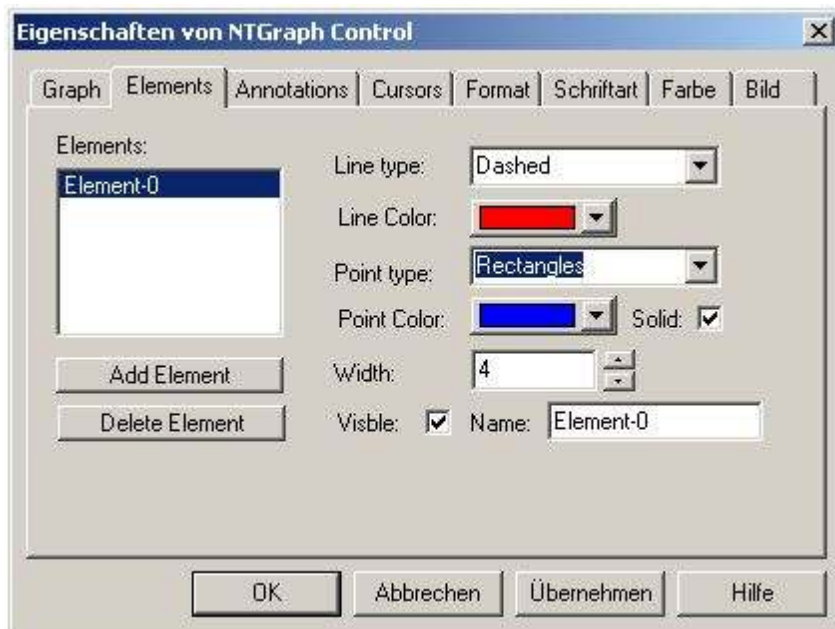
You can change the properties of this control during design time, or in run time to affect how the control will plot the data.

Use the new control property pages:

Graph Property Page



Elements Property Page



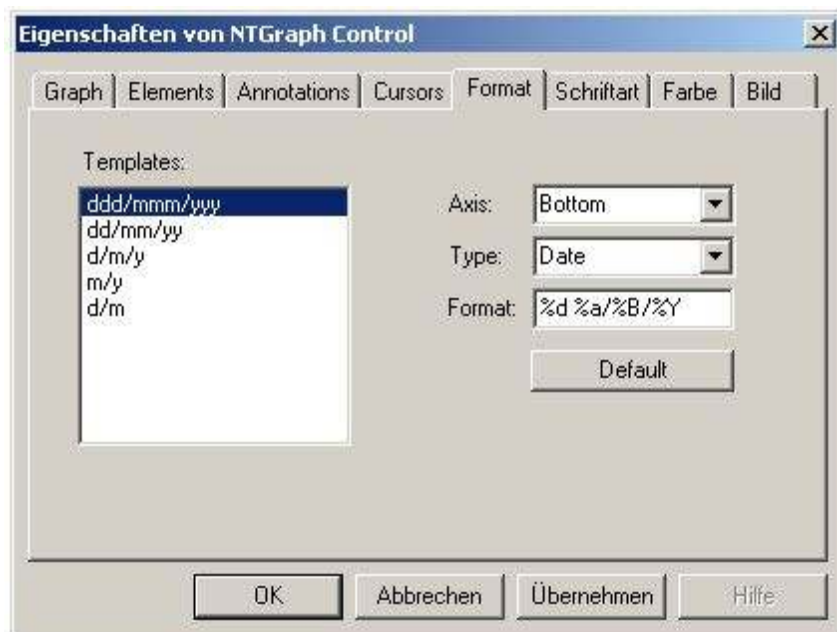
Annotations Property Page



Cursors Property Page



Format Property Page



You can include the control in your project by following the standard steps for ActiveX controls:

1. Create MFC Dialog project or MDI/SDI project with View class derived from `CFormView`
2. Choose menu Project|Add To Project|Components and Controls...
3. Open the Registered ActiveX Control gallery
4. Choose the `NTGraph` Control and click Insert
5. Visual C++ will generate the class `CNTGraph`
6. Then you can define variable of the type as `CNTGraph`.

The control's customization options are straightforward:

[Collapse](#) | [Copy Code](#)

```
// Customize Graph Properties
m_Graph.SetBackColor (RGB(0,0,0));
m_Graph.SetAxisColor (RGB(0,255,0));
m_Graph.SetLabelColor (RGB(128,255,255));

// Control's Frame and Plot area options
```



```

m_Graph.SetFrameColor((RGB(0,0,0));
m_Graph.SetPlotAreaColor(RGB(212,222,200));

m_Graph.SetFrameStyle(2) // (1) - Flat
                        // (2) - Scope (raised frame and
                        // (3) - 3DFrame (a bitmap frame
sunken plot area borders) picture)

m_Graph.SetGridColor(RGB(192,192,192));
m_Graph.SetShowGrid (TRUE);

m_Graph.SetCursorColor (RGB(255,0,0));
m_Graph.SetTrackMode (1);

m_Graph.SetGraphTitle("XY Plot");
m_Graph.SetXLabel ("X Axis");
m_Graph.SetYLabel ("Y Axis");
m_Graph.SetRange(0.,10,-1,1.);

```

You don't need to call the control `Invalidate()` function every time you change a Graph property. The changes are automatically reflected on the control appearance.

To load the data into the control...

[Collapse](#) | [Copy Code](#)

```

//
//
// Customize Graph Elements
//
//
// The Graph elements are dynamically allocated!

// Element 0 is allocated by default
// Even after a call to the ClearGraph method,
// the Element-0 is automatically allocated.

m_Graph.SetElementLineColor(RGB(255,0,0));
m_Graph.SetElementLinetype(0);
m_Graph.SetElementWidth(1);

m_Graph.SetElementPointColor(RGB(0,0,255));
m_Graph.SetElementPointSymbol(3);
m_Graph.SetElementSolidPoint(TRUE);

// Allocate a new element: Element-1
m_Graph.AddElement();

m_Graph.SetElementColor (RGB(0,255,0));
m_Graph.SetElementLinewidth(1);
m_Graph.SetElementLinetype(2);

// Allocate a new element: Element-2
m_Graph.AddElement();

m_Graph.SetElementColor (RGB(0,0,255));
m_Graph.SetElementLinetype(3);

// Now change again the properties of Element-1
m_Graph.SetElement(1);

m_Graph.SetElementColor (RGB(0,0,255));
...

//
// Load Data int the Graph Elements
//

double y;
for (int i = 0; i < NumberOfElements; i++)
{
    for (int x = 0; x < NumberOfPoints; x++)

```

```

{
    y = (double)rand() / RAND_MAX * 10.0;
    y = y / 3 + 10.0 / 2 * i + 1;
    m_Graph.PlotXY(x, y, i);
    // or PlotY(double data, long ElementID)
}

```

The same story for Visual Basic Users:

[Collapse](#) | [Copy Code](#)

```

With NTGraph1
    .PlotAreaColor = vbBlack
    .FrameStyle = Frame
    .Caption = ""
    .XLabel = ""
    .YLabel = ""

    .ClearGraph 'delete all elements and create a new one
    .ElementLineColor = RGB(255, 255, 0)
    .AddElement ' Add second elements
    .ElementLineColor = vbGreen

    For X = 0 To 100
        Y = Sin(X / 3.15) * Rnd - 1
        .PlotY Y, 0
        Y = Cos(X / 3.15) * Rnd + 1
        .PlotXY X, Y, 1
        .SetRange 0, 100, -3, 3
    Next X
End With

```

NTGraph Properties:

Graph

- short Appearance
- BSTR Caption
- short Appearance
- BSTR Caption
- BSTR XLabel
- BSTR YLabel
- OLE_COLOR ControlFrameColor
- OLE_COLOR PlotAreaColor
- OLE_COLOR AxisColor
- OLE_COLOR GridColor
- OLE_COLOR LabelColor
- OLE_COLOR CursorColor
- IPictureDisp* ControlFramePicture
- IPictureDisp* PlotAreaPicture
- IFontDisp* LabelFont
- IFontDisp* TickFont
- IFontDisp* TitleFont
- IFontDisp* IdentFont
- FrameType FrameStyle
- short XGridNumber
- short YGridNumber
- boolean ShowGrid
- boolean XLog
- boolean YLog
- double XCursor
- double YCursor

Elements

- short Element
- short ElementCount
- OLE_COLOR ElementLineColor
- OLE_COLOR ElementPointColor
- LineType ElementLinetype
- short ElementWidth
- SymbolType ElementPointSymbol
- boolean ElementSolidPoint
- boolean ElementShow
- TrackModeState TrackMode
- BSTR ElementName
- boolean ElementIdent

Annotations

- short Annotation
- short AnnoCount
- BSTR AnnoLabelCaption
- double AnnoLabelX
- double AnnoLabelY
- OLE_COLOR AnnoLabelColor
- boolean AnnoLabelHorizontal
- boolean AnnoVisible

Cursors

- short Cursor
- short CursorCount
- short CursorMode (0 - Fixed; 1 - Floating; 2 - Snapped to currently selected element)
- double CursorX
- double CursorY
- OLE_COLOR CursorColor
- short CursorStyle (0 - Crosshair; 1 - X hairline only; 2 - Y hairline only;)
- boolean CursorVisible

Format

- boolean XTime
- boolean YTime
- BSTR FormatAxisBottom
- BSTR FormatAxisLeft

Methods

Graph

- void SetRange(double xmin, double xmax, double ymin, double ymax)
- void AutoRange()
- void CopyToClipboard()
- void PrintGraph()
- void ShowProperties()

Elements

- void AddElement()
- void DeleteElement(short ElementID)

- `void ClearGraph()`
- `double GetElementXValue(short index, short ElementID)`
- `void SetElementXValue(short index, short ElementID, double newValue)`
- `double GetElementYValue(short index, short ElementID)`
- `void SetElementYValue(short index, short ElementID, double newValue)`
- `void PlotXY(double X, double Y, short ElementID)`
- `void PlotY(double Y, short ElementID)`

Annotations

- `void AddAnnotation()`
- `void DeleteAnnotation(short AnnotationID)`

Cursors

- `void AddCursor()`
- `void DeleteCursor(short CursorID)`

Tracking Mode constants

- None = 0
- Track = 1 Track cursor position (hold mouse button pressed)
- Cursor = 2 Cursor position by single click
- Zoom = 3 Unzoom (right mouse button click)
- PanXY = 4
- PanX = 5
- PanY = 6

Frame Style Constants

- Flat = 0
- Scope = 1 (raised frame and sunken plot area borders)
- 3DFrame = 2 (a bitmap frame picture)

Line style constants

- Solid = 0
- Dash = 1
- Dot = 2
- DashDot = 3
- DashDotDot = 4
- Null = 5
- XYStep = 6
- YXStep = 7
- Bars = 8
- Stick = 9

Symbol style constants

- Nosym = 0
- Dots = 1
- Rectangles = 2
- Diamonds = 3
- Asterisk = 4
- DownTriangles = 5
- UpTriangles = 6
- LeftTriangles = 7
- RightTriangles = 8

Yep, that's it!

Enjoy!

Send mail to nteofilov@yahoo.de with questions or comments about this article.

History

22 Nov 2002 - v1.0 Initial release

01 Dec 2002 - v1.1

- Added new method copy2clipboard.
- Added the ability to draw elements with a different number of points. (by A.Hoffman)
- New method added by A.Hoffman to Show/Hide the Graph Element
- Bug fix. Thanks to A.Hofmann for help.
- Fixed some drawing problems. Thanks to Judd.
- Added custom font support.
- Zoom Mode: Not implemented yet, but reserved.

26 Jan 2003 - v2.0 (Flicker Free versiton of the control)

- Thanks to Keith Rule for the `class CMemDC`
- ZoomMode Implemented.
- Added tooltip, showing current cursor position.
- Added new method `Autorange`.
- Added new property `ElementLinewidth`.
- Added new property `ElementLinetype`.
- Fixed some drawing problems.
- Added (*Test.htm*) a brief info on how to add the control to your web page.

09 Mar 2003 - v2.1

- PanMode Implemented.
- Modified `SetElementColor`, `SetElementLinewidth`, `SetElementLinetype`, so that they accept as a first parameter the ElementID.
- Fixed some drawing problems. Should be clean now.
- GDI leak Fixed.

01 Jun 2003 - v3.0 New release!

- Thanks to **Chris Maunder** for `Colour Picker control`

02 Aug 2003 - v4.0 Final release!

- Thanks to **Robert Harber** for the useful discussions, ideas and code.
- Thanks to **tagi1** for fixing the printing font problem.
- Thanks to **Judd** for testing the control.

Note that since there are significant changes in the last release you should remove (first unregister and than delete) all old versions of the control from your projects! The VBA users who use the control inside of Office applications should also remove the following file NTGRAPHLib.exd in the Temp directory of your computer. This file is automatically created by the Office application and saves properties of the former created control instances. You have to delete this file before you insert the new version of the control, in order to get correctly names in the property browser.