VisualNEO for Windows Help

Tabla de contenido

Welcome to VisualNEO

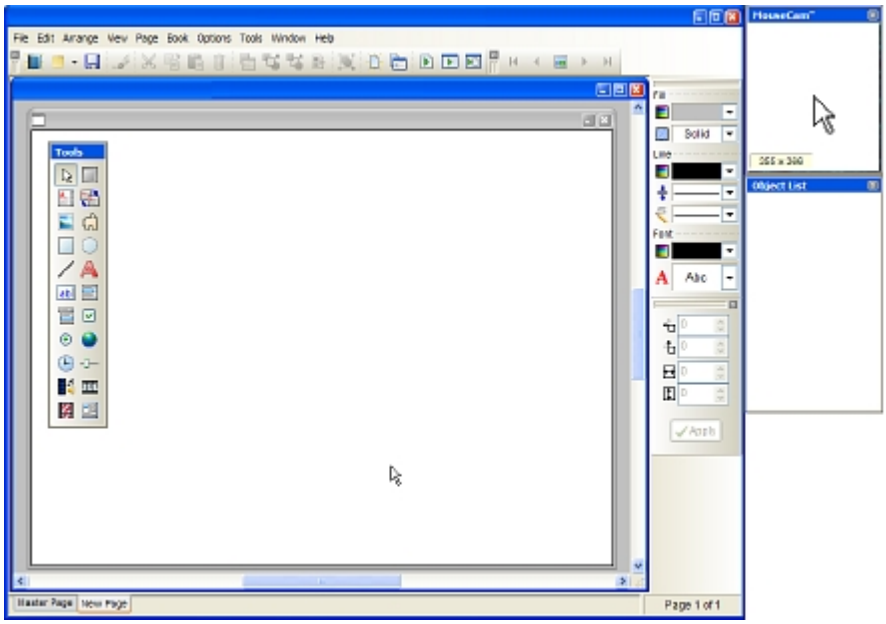
VisualNEO for Windows is a Rapid Application Development Software. It is designed to be extremely easy to learn and powerful. Even inexperienced users can quickly combine text, graphics, sound, animation and other elements to create interactive, software such as: electronic books, presentations, brochures, greeting cards, educational materials, computer-based training applications, catalogs, electronic magazines, games and many types of other applications.

VisualNEO Basics

This section will introduce you to VisualNEO for Windows's interface and provide you with a basic overview of the program's operation.

The VisualNEO Screen

When you start VisualNEO for Windows for the first time, a blank publication will be opened and you will be placed into authoring mode. As you may have already discovered, the VisualNEO for Windows screen consists of a large workspace surrounded by several groups of tools and dockable palettes. The parts of the screen are described below:



Menu Bar

The Menu Bar is a standard Windows component which appears at the top of most applications. VisualNEO for Windows's Menu Bar includes commands for opening, saving and modifying your publications. The commands found in the menu are described in detail in here.

Speed Bar

The Speed Bar contains buttons that provide single click access to several often used VisualNEO for Windows commands.

Navigation Buttons

To the right of the Speed Bar are five page Navigation Buttons. The first button sends you to the first page of the publication. The second button takes you to the previous page. The middle button sends you to the Master Page. The fourth button takes you to the next page. The fifth button takes you to the last page of the publication.

The Master Page typically contains elements that are common to most (or all) pages in a publication. Common elements can include navigational buttons, titles, page numbers, etc. You can add, modify and delete objects on the Master Page just as you would on any other page.

Work Space

The Work Space occupies the largest portion of VisualNEO for Windows's screen. This is the area where you will create and edit your publications. You may open several work windows at a time, each containing a different publication. If your VisualNEO for Windows screen does not contain a work window, you can open one by selecting New or Open from the File menu.

Page Tabs

When a work window is open, Page Tabs containing the titles of the publication's pages will

appear at bottom of the VisualNEO for Windows screen. Each page in a publication is assigned a unique name which is reflected on the tab. You may jump between pages by clicking on the tab of the page you want to go to. You can change the order of the pages by dragging the Page Tabs along the bottom of the VisualNEO for Windows screen.

Tool Palette

VisualNEO for Windows's Tool Palette contains a selection of tools that you will use to design your publications. The Tool Palette is described in detail in here.

Style Palette

The controls on the Style Palette are used to modify the appearance of the objects that make up your publication. Controls are available for changing an object's fill, outline and font attributes.

Object List

This palette contains a list of all the objects that have been placed on the current page. Like pages, each object is assigned a unique name. You may use the Object List to select specific objects for modification, or right click on an object to change its attributes. The Object List can be hidden by deselecting the Show Object List item in VisualNEO for Windows's View menu.

Object Information

The Object Info palette displays the selected object's size and position on the page. You can edit these settings manually if you need precise control over an object's coordinates.

MouseCam™ / Cursor Position

The MouseCam palette shows a magnified view of the area of the screen beneath the mouse pointer. This can be handy when positioning very small objects. You can adjust the magnification settings by right clicking on the MouseCam window.

Scroll Bar

If your publication's dimensions are greater than the size of your screen, you can use the scroll bars to bring other parts of your project into view.

Current Page Indicator

This box, in the lower/right corner, indicates the number of the current page displayed in the work space.

Customizing Your Workspace

You can customize your VisualNEO for Windows environment by positioning the palette windows anywhere on the screen, or by docking them to the left, right, top or bottom edges of the main program window. Docked palettes can be repositioned or undocked by dragging the small bar at the top or left of each palette window. Unneeded palettes can be hidden by clicking the small close box at the top of the window or by deselecting the appropriate item from VisualNEO for Windows's View menu. (Note: The Tool and Style palettes cannot be closed.)

Creating and Opening Applications

After you start VisualNEO for Windows, you can either create a new publication from scratch or open an existing publication.

To create a new publication, select New from VisualNEO for Windows's File menu and specify the desired screen size, color resolution and application type. An empty publication containing two blank pages will be created. One of the blank pages is the Master Page. The other is the first page of your publication. Additional pages can be added at any time by selecting Add Page from the Page menu. Other publication-wide settings can be defined by selecting App Properties from the App menu.

To open an existing VisualNEO for Windows publication, select the Open command from the File menu. A standard Windows file selector will appear, allowing you to select a publication from your hard drive. If you're new to VisualNEO for Windows, the only existing publications available will be the samples installed with VisualNEO for Windows.

Working with Pages

VisualNEO for Windows applications are composed of pages very much like a printed book. A publication may consist of a single page or up to several hundred pages. VisualNEO for Windows imposes no specific limit on the maximum number of pages a publication may contain, but most authors find it difficult to keep track of more than 200 or 300 pages. The actual limit depends on available memory, system resources, etc.

Each page in a publication is assigned a unique title. You can change the title assigned to a page if you like, but no two pages in a publication can have the same exact title. Page titles will appear in Page Tabs at the bottom of the VisualNEO for Windows screen. You can move between pages by clicking the tab of the desired page. You can also move between pages using the Page Navigation buttons, or by pressing the Page Up and Page Down keys on your keyboard.

Each publication also contains a special page called the **Master Page**. The Master Page typically contains elements that are common to most (or all) pages in a publication. Common elements may include navigational buttons, titles, page numbers, etc. You can add, modify and delete objects on the Master Page just as you would on any other page. The difference is, objects placed on the Master Page will also appear in the background of other pages.

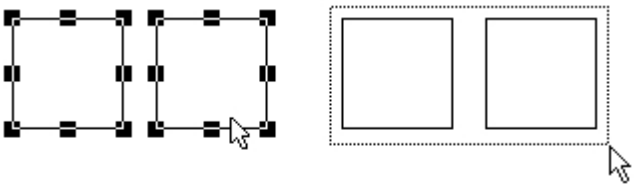
Note: *You cannot delete or rename the Master Page.*

The Page menu contains commands for adding, moving, copying and renaming pages. You can also move a page by dragging its Page Tab to a new location along the bottom of the VisualNEO for Windows screen.

Working with Objects

In VisualNEO for Windows, every element you add to your publication (text, pictures, buttons, etc.) is considered an object. An object is created using the tools found on VisualNEO for Windows's Tool Palette. Once created, objects can be selected, moved, resized, edited and deleted.

 Use the Tool Palette's Selection Tool to select, move and resize an object. Click on an object using the left mouse button to select it. Select multiple objects by holding down the Shift key while clicking on additional objects, or by clicking and dragging the mouse to surround the desired objects in a rectangle. Selected objects will be surrounded with small, box-like handles.



Once selected, an object or group may be moved by clicking on the object with the left mouse button, then dragging the object or group to a new location before releasing the mouse button. To resize an object, simply click and drag one of the object's selection handles. Holding down the Shift key while resizing an object retains the object's relative shape. Selected objects also can be cut or copied to the Windows Clipboard, or they can be removed from the publication by pressing the Delete key.

Depending on the object selected, clicking it with the right mouse button will allow you to modify the object's appearance and define how the object behaves when viewed by the reader of your publication. Most types of objects can also be assigned Actions to execute when clicked on by the reader.

Specific information about the different kinds of objects available in VisualNEO for Windows can be found here.

Moving Objects Between Pages and Other Publications

Selected objects can be copied or moved between pages within the same publication or even between two different publications.

One method for moving objects from one page to another is to drag the objects to one of the page tabs at the bottom of VisualNEO for Windows's screen. Release the mouse button and the objects will appear on the page corresponding to the selected tab. Holding down the Ctrl key on the keyboard during the move will duplicate the objects, leaving a copy on the original page.

Another method for copying and moving objects is to use the Windows Clipboard. Select the objects you wish to copy or move and choose Copy or Cut from VisualNEO for Windows's Edit menu. Then switch to the page you want the objects to be placed and select Paste from the Edit menu.

Objects can be moved to another publication using the same techniques. Use VisualNEO for Windows's Open command to load both the source and target publications. If necessary, use the Windows menu's Tile command to position the open publications so both are visible. Then, cut and paste or drag objects from one publication window to the other.

Because object names must be unique, VisualNEO for Windows may rename objects during paste or copy operations if there is a conflict with an existing object.

Defining App Properties

Publication-wide settings can be changed by selecting App Properties from the App menu. Properties that can be set include the publication's title, icon, mouse pointers, the shape and behavior of the publication's window, startup Actions, custom menu bars and more.

Testing an App

During the design phase of your project, you will want to test the publication periodically to get an idea of how it will look to your readers. You can do this at any time by selecting one of the **Run** commands from the **App** menu. When running in test mode, you will be able to preview page transition effects, animations, sounds, videos and see how buttons and other objects behave – just as a reader would see them. This provides you with the opportunity to find mistakes and make any necessary adjustments before your publication gets to your readers.

Saving an App

Before you get too far into designing your publication, it's a good idea to save what you've done to disk. To save a publication, select Save from the File menu. You will be prompted to give your publication a file name the first time you save it.

Compiling a App for Distribution

A typical VisualNEO for Windows publication contains text, images, animation, sound, video and other elements. These elements may be separate files stored in various locations on your computer. Before you can begin distributing your finished publication, you will need to package all of these elements into a stand-alone executable application. This final step in the process is called compiling.

VisualNEO for Windows's Compiler is capable of producing finished applications in four different formats. These include:

- Windows Standard Application (EXE)
- Windows Screen Saver (SCR)
- Windows System Tray Application (EXE)
- Web Browser Plug-In/ActiveX Control (OCX)

Each of these formats has advantages and disadvantages. Most VisualNEO for Windows publications are compiled as Windows applications (exe). A publication compiled to this format will generally look and behave like a traditional Windows-based program. In most cases, it will appear inside a window with a border and a title bar.

Most publications also can be compiled as screen savers, system tray applications or web browser plug-ins. However, these types of publications tend to be slightly more complex and require an understanding of Windows in order to achieve the best results.

Learning VisualNEO

The best way to learn VisualNEO for Windows is to try it yourself. The tutorials in this section will guide you through the creation of several types of VisualNEO for Windows publications. These tutorials just scratch the surface of what can be done with VisualNEO for Windows, but they should provide you with the basic knowledge needed to begin creating publications of your own.

If you haven't already, please read the topic VisualNEO for Windows Basics. These sections will provide you with some helpful background information about electronic publishing and VisualNEO for Windows's interface.

Tutorial 1: Creating a Basic Publication
Tutorial 2: Creating Hypertext Links

Note: *Names of people and companies used in these examples, with the exception of Historical figures, are fictitious. Any resemblance to existing companies or persons (living or dead) is purely coincidental.*

Tutorial 1: Creating a Basic App

The best way to learn VisualNEO for Windows is to try it yourself. The tutorials in this section will guide you through the creation of several types of VisualNEO for Windows publications. These tutorials just scratch the surface of what can be done with VisualNEO for Windows, but they should provide you with the basic knowledge needed to begin creating publications of your own.

This tutorial will demonstrate how to create a basic publication.

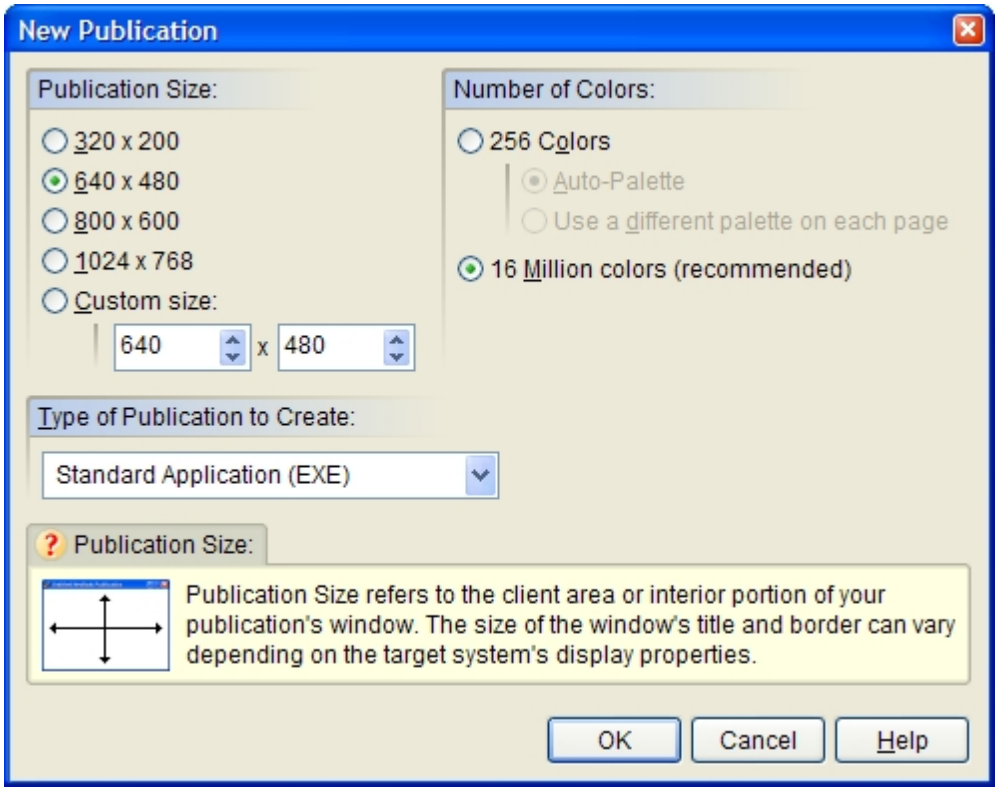
Creating a New Publication

When starting from scratch, you must define the publication's dimensions and color resolution.

1. Launch VisualNEO for Windows. If any publications are already open, use the **File** menu's **Close All** command to clear VisualNEO for Windows's work space.

2. Select **New** from VisualNEO for Windows's **File** menu.

The **New Publication** screen will appear.



3. Select a publication size of **640 x 480** pixels.

4. Select **16 Million Colors**.

5. Select **Standard Application (EXE)**.

6. Click **OK**.

An untitled publication will be created.

Defining Publication Wide Settings

Now we're ready to start creating our publication. We'll begin by configuring some settings that affect the entire publication.

1. Select App Properties > General from the **App** menu.

Here we can define how our publication will appear and how it will be used by our readers. The App Properties screen is divided into 11 sections indicated by the icon images on the left: General, Size/Colors, Window, Main Menu, Actions, Access, Security, Language, Interface, Screen Saver and Tray Menu. Each section contains a different category of settings used to configure the publication. The different sections can be displayed by clicking their corresponding icons.

2. In the first section (General) type `My First Publication` in the **Title** field.

3. Type your name in the **Author** field.

4. Click the **Access** icon to display that section of the App Properties screen. You may need to scroll down in order to bring the Access icon into view.

5. Remove the check mark from the **Allow Page Up, Page Down, Home and End Keys to change pages** option. We're going to create our own navigation controls, and we don't want readers to use the keyboard to change pages. Turning this option off will prevent that from happening.

6. Click **OK** to apply these changes to the publication.

Adding Pages to Our Publication

Our new publication already contains two blank pages, but we need to add one more.

1. Select **Add Page** from the **Page** menu.

The **Add Pages** screen will appear.

2. Enter "1" into the **Number of Pages to Add** field.

3. Click **OK**.

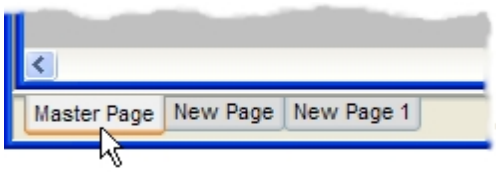
Creating a Button

Now we'll add a couple of simple page turning buttons to allow our readers to easily navigate among our publication's pages. So that these buttons appear on every page, we're going to place them on the **Master Page**.

The Master Page typically contains elements that are common to most (or all) pages in a publication. The difference between the Master Page and regular pages is that objects placed on the Master Page will also appear in the background of other pages. This is a real

time saver, since these objects only need to be created once and not for every page in the publication.

1. Display the **Master Page** by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "Master Page".



2. Select the **Push Button Tool** from the **Tool Palette**. 

3. Move the mouse pointer to the lower right portion of the Master Page. Press and hold down the left mouse button and draw a rectangle about one inch wide and one half an inch tall. Once the rectangle is the correct size, release the mouse button. Don't worry if your rectangle isn't perfect - we can fix that later.

After you release the mouse button, the **Push Button Properties** screen will appear allowing us to define what our button will do. The Push Button Properties screen is divided into three sections indicated by the icon images on the left: General, Appearance and Actions. To view the settings for a section, click the corresponding icon.

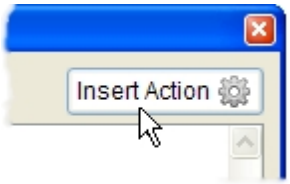
4. In the first section (General) type `Next` in the **Caption** field.

Text typed in the Caption field will appear on the face of the button.

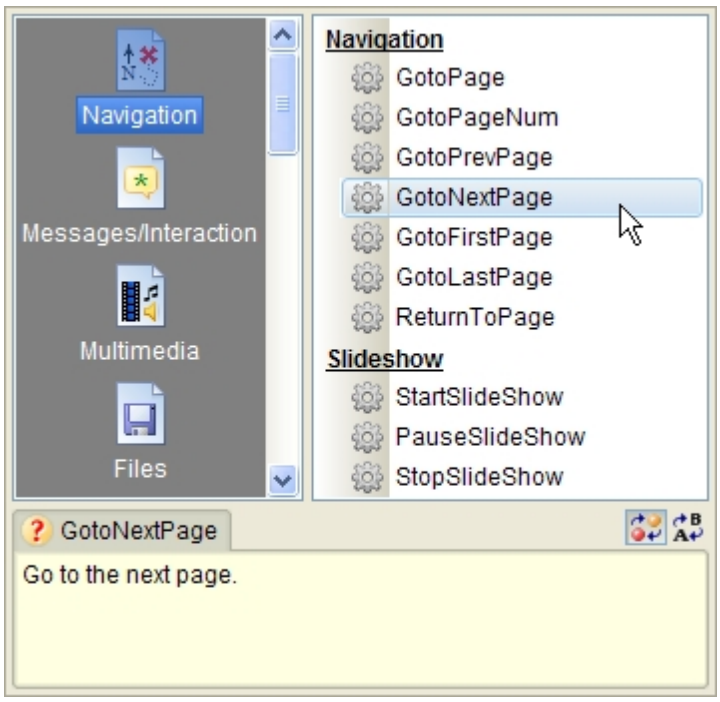
5. Click the **Actions** icon.

Now we'll tell VisualNEO for Windows what we want to happen when the reader clicks this button. (In the case of this particular button: go to the next page.) Our button will perform its task based on special instructions we provide. In VisualNEO for Windows, these instructions are called Actions. Most Actions can be inserted by simply selecting the desired command from a list.

6. Click the **Insert Action** button.



The **Insert Action** screen is divided into categories, indicated by the icon images on the left. Clicking an icon will display a list of Actions belonging to that category.

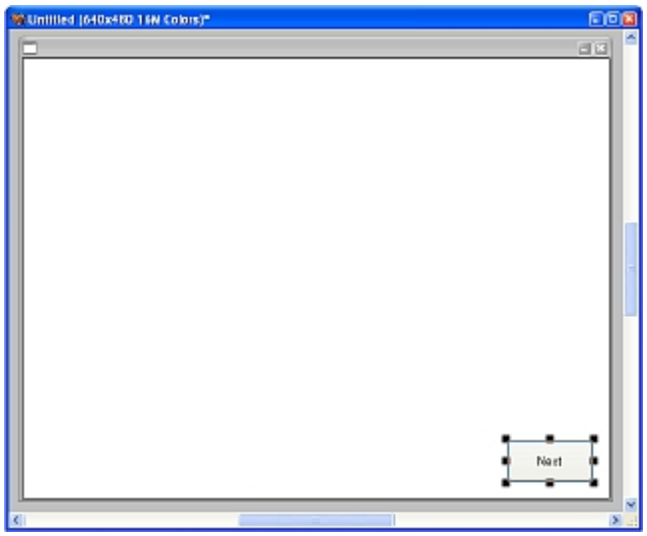


7. Click the **Navigation** category.

8. Select the **GotoNextPage** Action.

VisualNEO for Windows will automatically add the command to the button's Action Editor. This Action tells VisualNEO for Windows to advance to the next page whenever the button is pressed.

9. Click **OK**.



Notice that the button is surrounded by eight small black boxes or handles. This indicates that the object (in this case our button) is selected. Once selected, you can use the mouse to drag the object to a new location or change its shape by grabbing one of the handles. Selected objects can also be affected by changes made to the Style Palette's controls.

 Using the Tool Palette's Selection Tool to click on other objects or empty areas of the page will cause the handles to disappear and the button to become unselected. Clicking the button again will reselect the object and display the selection handles. (More information on selecting objects is available here.)

Creating a Second Button

Since our second page turning button is almost identical to the first, we can create it more quickly by using VisualNEO for Windows's Duplicate command.

1. Make sure the button you just created is selected. (If it's not, click it with the mouse.)

2. Choose **Duplicate** from the **Edit** menu.

A second, identical button will appear on your screen.

3. Use your mouse to drag the duplicate button to a position alongside the original button.

4. With the duplicate button still selected, choose **Object Properties** from the **Edit** menu.

5. Replace the button's current **Caption** with the word `Previous`.

6. Click the **Actions** icon.

7. Erase the existing button Action (GotoNextPage). (Use the Backspace or Delete keys on your keyboard just as you would in a word processor.)

8. Click the **Insert Action** button.

9. Select **GotoPrevPage** from the **Navigation** category.

10. Click **OK**.

You have created a second button!

Using the Style Palette

While we're still on the Master Page, you might also want to add ruling lines, boxes or other decorative graphical elements that you want to appear on all of your publication's pages.

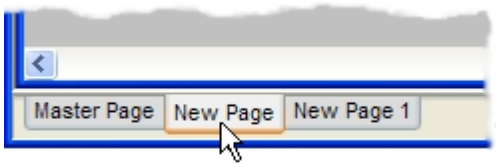
To do this, select either the **Line**, **Rectangle** or **Ellipse** tools from the Tool Palette. Then draw a shape on the page, just as you did to create your first button. If you make a mistake, use the **Edit** menu's **Undo** command to remove the change.

Using the controls on the Style Palette, you can change the selected object's Fill Color, Fill Pattern, Line Color, Line Width, and Line Style. If the selected object contains text, you can also select a new Font and Font Color.

Setting Page Properties

Now that we've added our basic navigation buttons to the Master Page, we're ready to work on our publication's first page.

1. Display the first page by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "New Page".



The two buttons we created on the Master Page will still be visible in the background of this page, so don't be alarmed if the screen doesn't appear to change very much.

Let's change the page's background to something other than plain white.

2. Select **Page Properties** from the **Page** menu.

The **Page Properties** screen will appear.

3. Click the small arrow button next to the **Solid Color** option under **Page Background**.

The **Color Selector** will appear.

4. Select a color from the palette. (Preferably a light color.)

5. Click the Color Selector's **OK** button.

6. Make sure a check mark appears next to the **Show items from master page** option. This enables the buttons we just created on the Master Page to be active on this page.

7. Enable the **Copy to all pages** option.

8. Click **OK**.

The Page Properties screen is replaced by the **Copy to all pages** screen.

9. Click the **Background** check box.

10. Click **OK**.

The background color we selected is copied to our publication's other pages. (The Master Page's background will remain white since it technically has no background.)

Adding Text

Since this is the first page readers will see, let's add a title that tells readers what this publication is all about. Captions, titles and other short text may be inserted using the Simple Text Tool.

1. Select the **Simple Text Tool** from the **Tool Palette**. 

2. Move the mouse pointer to the center of the page. Press and hold down the left mouse button and draw a rectangle about five inches wide and one half an inch tall. Once the rectangle is the correct size, release the mouse button.

The **Text Properties** screen will appear.

3. Type `Amalgamated Industries Annual Report` in the **Caption** field.

4. Click the **Center Alignment** icon. 

5. Click **OK**.

The Text object will appear surrounded by the selection handles.

6. On the Style Palette, click the small arrow button next to the **Fill Pattern** box.

7. Click the pattern square containing the letter "**H**". This will make our Text object hollow allowing the page's background color to show through.

8. Click the small arrow next to the Style Palette's **Line Width** box.

9. Select "**None**" for the line width.

10. Click the small arrow next to the Style Palette's **Font** box.

The **Font Selector** screen will appear.

11. Select **Arial**, **Bold**, **16 Point**.

12. Click **OK**.

If part of the text is missing or not centered, use the mouse to move or resize the object.

Importing a Picture

Now we'll dress up this page by importing an illustration created with another application.

1. Select the **Picture Tool** from the **Tool Palette**. 

2. Use your mouse to draw a rectangle on the page just below the Text object.

A Windows **File Selector** screen will appear.

3. Locate the `Program Files\VisualNEO for Windows 5` folder. (If you installed VisualNEO for Windows in a different folder, use that instead.)

4. Open the `Tutorial Files` folder.

5. Select the `Logo1.bmp` file.

6. Click **OK**.

The selected image file will be inserted into the Picture object and displayed on the page.

7. Place the mouse pointer over the Picture object and click the right mouse button.

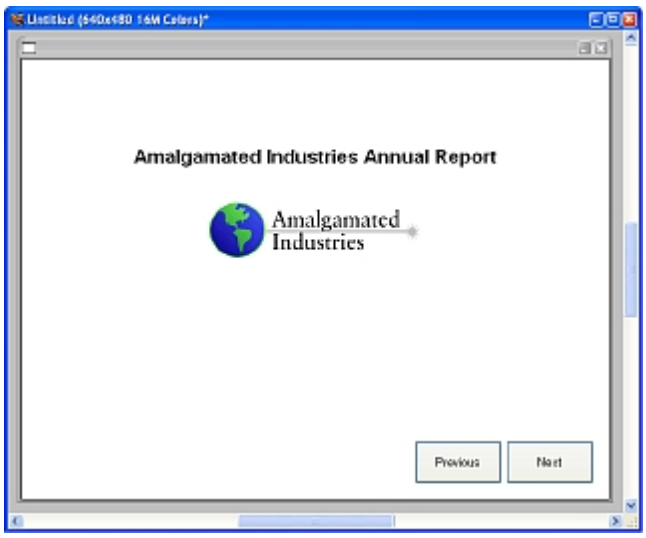
The **Picture Properties** screen will appear.

8. Click the **Appearance** icon.

9. Place a check mark in the box next to the **Resize object to match image dimensions** option.

10. Click **OK**.

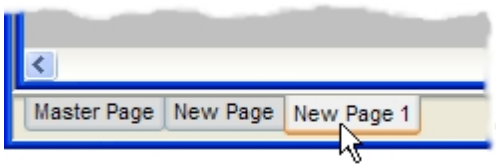
The Picture object will be automatically resized to match the dimensions of the image file. If needed, use your mouse to adjust the position of the Text and Picture objects to match the illustration below:



Importing a Text File

Now that our first page contains a title, we can concentrate on the main information page of our two page publication. First we need to go to the second page.

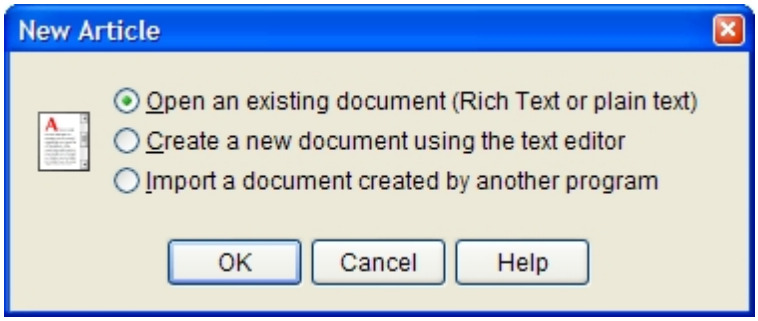
1. Display the second page by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "New Page 1".



2. Select the **Article Tool** from the **Tool Palette**. 

3. Use your mouse to draw a large rectangle in the top portion of the page. Don't worry if the rectangle isn't perfect, you can always resize it later.

The **New Article** screen will appear.



4. Select the **Open an existing document** option.

5. Click **OK**.

A Windows **File Selector** screen will appear.

6. Locate the `Documents\VisualNEO for Windows 5` folder. (If you installed VisualNEO for Windows's sample files in a different folder, use that instead.)

7. Open the `Tutorial Files` folder.

8. Select the `Sample1.rtf` file.

9. Click **OK**.

The text from the selected file will be inserted into the Article object and displayed on the page. Let's add a solid background and a border to our Article.

10. On the Style Palette, click the small arrow button next to the **Fill Pattern** box.

11. Click the pattern square containing the letter "**S**". This will make our Article object solid.

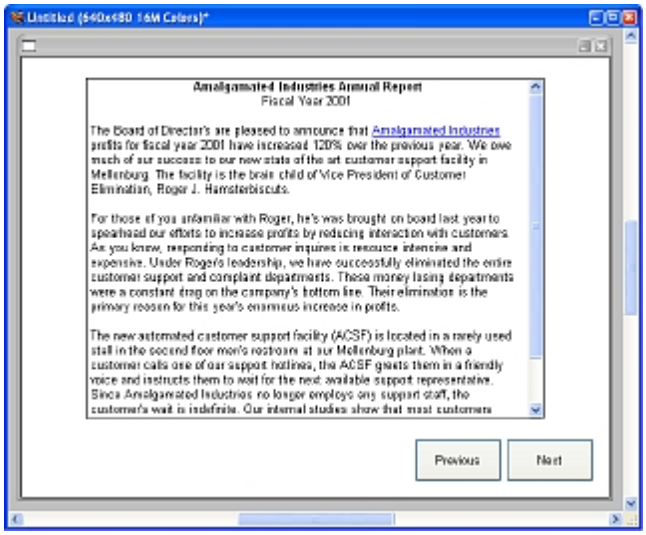
12. Click the small arrow next to the Style Palette's **Fill Color** box.

The **Color Selector** will appear.

13. Select "**white**" and click the Color Selector's **OK** button.

14. Click the small arrow next to the Style Palette's **Line Width** box.

15. Select the first line width (one pixel) just below "None".



If needed, use your mouse to adjust the position and shape of the Article object to match the illustration above. You can edit the Article's properties, by selecting **Object Properties** from the **Edit** menu. (You can also access the properties screen by clicking the object with the right mouse button.) The document displayed within the Article object can be edited by selecting the file's name from the Edit menu's Create/Edit command.

Disabling a Button from the Master Page

Since this is the last page of our publication, our Next Page button won't work here. To avoid confusing our readers, we'll use a simple Action command to disable this button while the page is visible. Another Action will re-enable the button before the reader leaves so it will continue to work on our other page.

1. Select **Page Properties** from the **Page** menu.

2. Click the **Actions** icon.

3. Click the **Insert Action** button.

4. Select **DisableObject** from the **Objects** category.

The **DisableObject Properties** screen will appear.

5. Type `PushButton1` in the **Object Name** field. (This is the name VisualNEO for Windows assigned to our Next Page button.)

6. Click **OK**.

The DisableObject Action will be added to the Page Enter event. This event is activated every time this page is displayed. Now, let's add another Action to re-enable the Next Page button.

7. Click the **Page Exit** tab at the bottom of the Page Properties screen.

8. Click the **Insert Action** button.

9. Select **EnableObject** from the **Objects** category.

The **EnableObject Properties** screen will appear.

10. Type `PushButton1` in the **Object Name** field.

11. Click **OK**.

The EnableObject Action will be added to the Page Exit event. This event is activated every time the reader leaves this page to go to another page.

12. Click the Page Properties screen's **OK** button.

Saving Your Work

Let's go ahead and save what you've done so far. It's a good idea to save your work often to avoid losing anything if there's a power failure or some other disruption.

1. Select **Save** from the **File** menu.

Since we haven't saved this publication before, the **Save As** screen will appear.

2. In the `VisualNEO for Windows 5` folder on your hard drive, open the `Tutorial Files` folder.

3. Type `Tutorial1.pub` in the **File Name** field.

4. Click **Save**.

Testing The Publication

Every so often it's a good idea to test drive your publication. VisualNEO for Windows's test mode provides you with a preview of your publication from a reader's perspective. Buttons, check boxes, animation, sound, and other elements will function just as they would in a finished publication.

1. Select **Run (From Start)** from the **App** menu.

2. Test the publication by clicking on the **Next** and **Previous** buttons, etc.

3. When you're finished testing, click the publication window's close button or press the **Esc** key on your keyboard.

Distributing The Publication

When your publication is complete and ready for distribution, you'll use VisualNEO for Windows's Compile option to package it into a stand-alone executable application (exe). This process is called compiling.

1. Select **Compile/Publish** from the **App** menu.

The **Compile/Publish App** screen will appear.

2. The **Compile To** field contains the location and name of the exe file that will be created. By default, this name is the same as the publication's file name except that the extension has been changed to "exe". You can leave this name as is or change it if you like.

3. Under **Type of compiled publication to create**, select **Standard Application (EXE)**.

4. Under **This compiled publication will be run from**, select **Hard Disk**.

5. Under **Compiler Options**, place check marks next to **Compress files embedded inside compiled publication** and **Compress and encrypt publication source code**. Make sure no check marks are next to the other two options.

6. Click **Compile**.

VisualNEO for Windows will convert the publication into a Windows application file (exe) using the name and location specified in the Compile To field. To launch the compiled application, double click this file name from the Windows Explorer.

Congratulations, you've just created your first VisualNEO for Windows publication!

Tutorial 2: Creating Hypertext Links

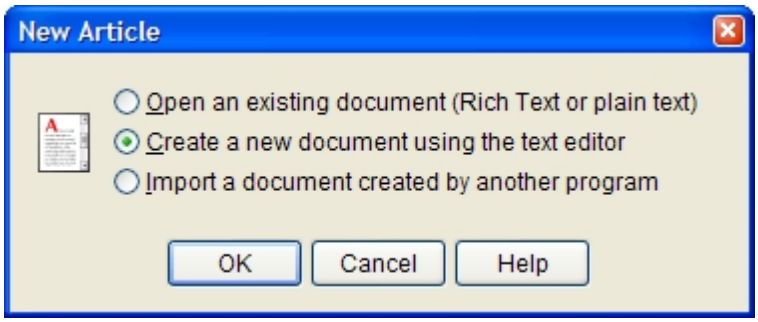
VisualNEO for Windows has great versatility and can be used to create offline applications that complement online content. Dynamic publications can be easily programmed to send email messages or linked to related web sites. One way of accomplishing this is through the use of Hypertext. A Hypertext document contains underlined words, or links, that can be clicked to jump to another page or display related information. VisualNEO for Windows provides some simple tools that allow you to incorporate Hypertext links into your publications. This tutorial will explain some of these concepts.

1. Launch VisualNEO for Windows and create a new publication by selecting **New** from VisualNEO for Windows's **File** menu.

2. Select the **Article Tool** from the **Tool Palette**.

3. Move the mouse pointer to the upper left portion of the page. Press and hold down the left mouse button and draw a rectangle about half the size of the page. Once the rectangle is the correct size, release the mouse button. Don't worry if your rectangle isn't perfect, you can move or resize the Article object later.

The **New Article** screen will appear.



4. Select the **Create a new document using the text editor** option.

5. Click **OK**.

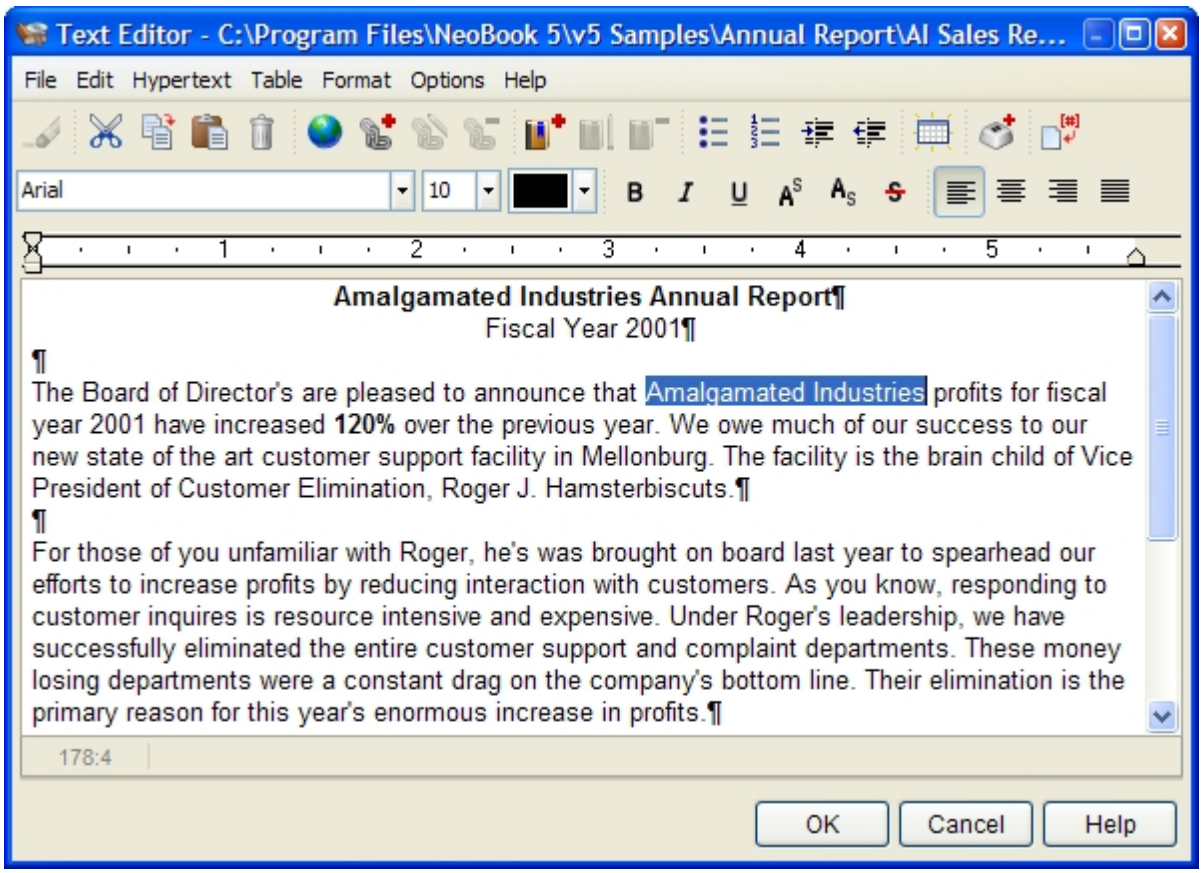
VisualNEO for Windows's **Text Editor** will appear.

To enter text into the editor, do one of the following:

6A. Use your keyboard to type directly into the Text Editor.

6B. Use another application to **Cut** or **Copy** text to the Windows Clipboard. Then select **Paste** from the Text Editor's **Edit** menu.

7. Once your text has been inserted into the Text Editor, highlight a word or phrase that you would like use as a Hyperlink.

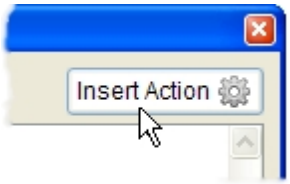


8. Select **Add Link** from the Text Editor's **Hypertext** menu (or click the **Create Hyperlink** button from the Speed Bar).

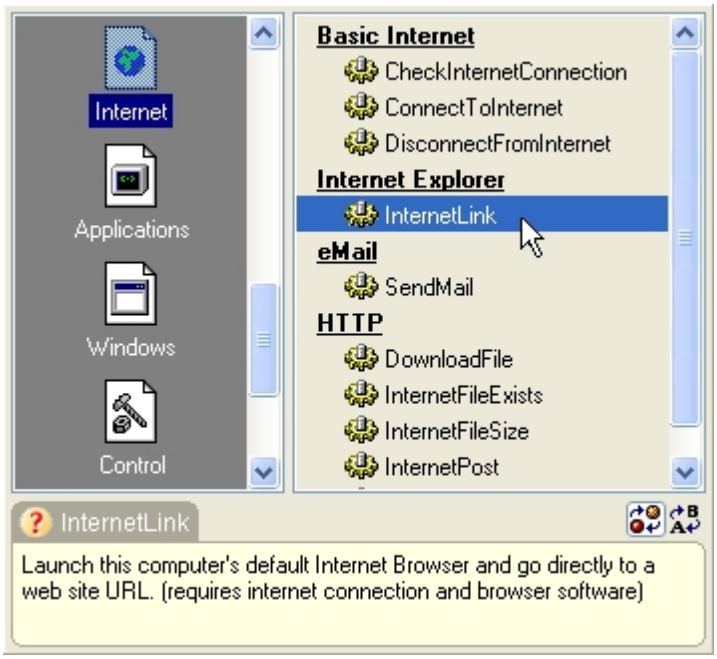
The **Hyperlink Editor** will appear.

9. Click the **Insert Action** button.

The **Insert Action** screen is divided into categories, indicated by the icon images on the left. Clicking an icon will display a list of Actions belonging to that category.



10. Select the **Internet** category. (You may need to scroll through the list of Action categories.)



11. If the Hyperlink is intended to launch the computer's default Browser and display a web site, select the **InternetLink** Action. If the Hyperlink is intended to send an email message, select the **SendMail** Action.

12A. If you selected InternetLink, the **InternetLink Properties** screen will appear.

12B. Type the address/URL of the web site you want to visit in the field provided.

12C. Click **OK**.

13A. If you selected SendMail, the **SendMail Properties** screen will appear.

13B. Fill in the **To**, **From**, **Subject** and **Message** fields just as you would when sending a normal email.

13C. Click **OK**.

VisualNEO for Windows will automatically add which ever Action you select to the Hyperlink Editor.

14. Click the Hyperlink Editor's **OK** button.

The Hyperlink will be created and the highlighted text will appear underlined. If you need to edit or remove a Hyperlink, place the text cursor on top of the underlined text and choose **Edit Link** or **Remove Link** from the **Hypertext** menu.

To add additional Hyperlinks, repeat Steps 7 through 14.

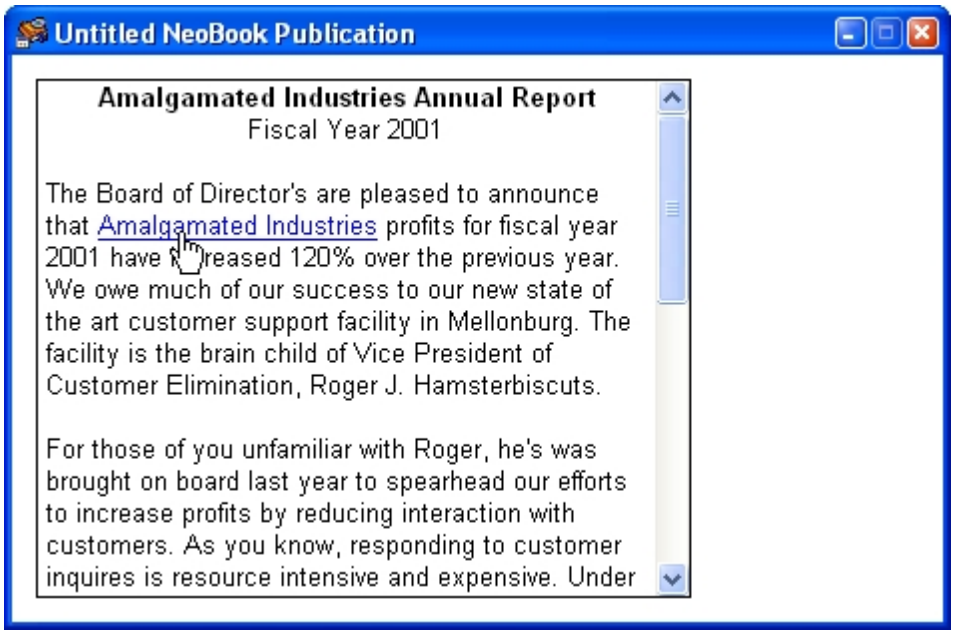
15. When you've finished editing your document, click the Text Editor's **OK** button.

Since this is a new document, the **Save As** screen will appear.

16. You may save the document using a file name of your choice. However, you should always use the ".RTF" extension when saving files created with the Text Editor. That way there will be no confusion about what type of file it is.

The document will be displayed in the Article object you created way back in steps 2 and 3. If needed, use your mouse to adjust the position and shape of the Article object.

17. You can test your hyperlinks by selecting the **Run** command from the **App** menu.



Tutorial 3: A Computer Based Training Example

VisualNEO for Windows provides teachers and corporate trainers with a wide array of tools that can be used to create powerful computer based training (CBT) and testing applications. VisualNEO for Windows's Action commands can be used to calculate scores and save or print the results. Fill in the blank, multiple choice and matching tests can be designed in a variety of ways using VisualNEO for Windows's Push Buttons, Text Entry Fields and Radio Button tools.

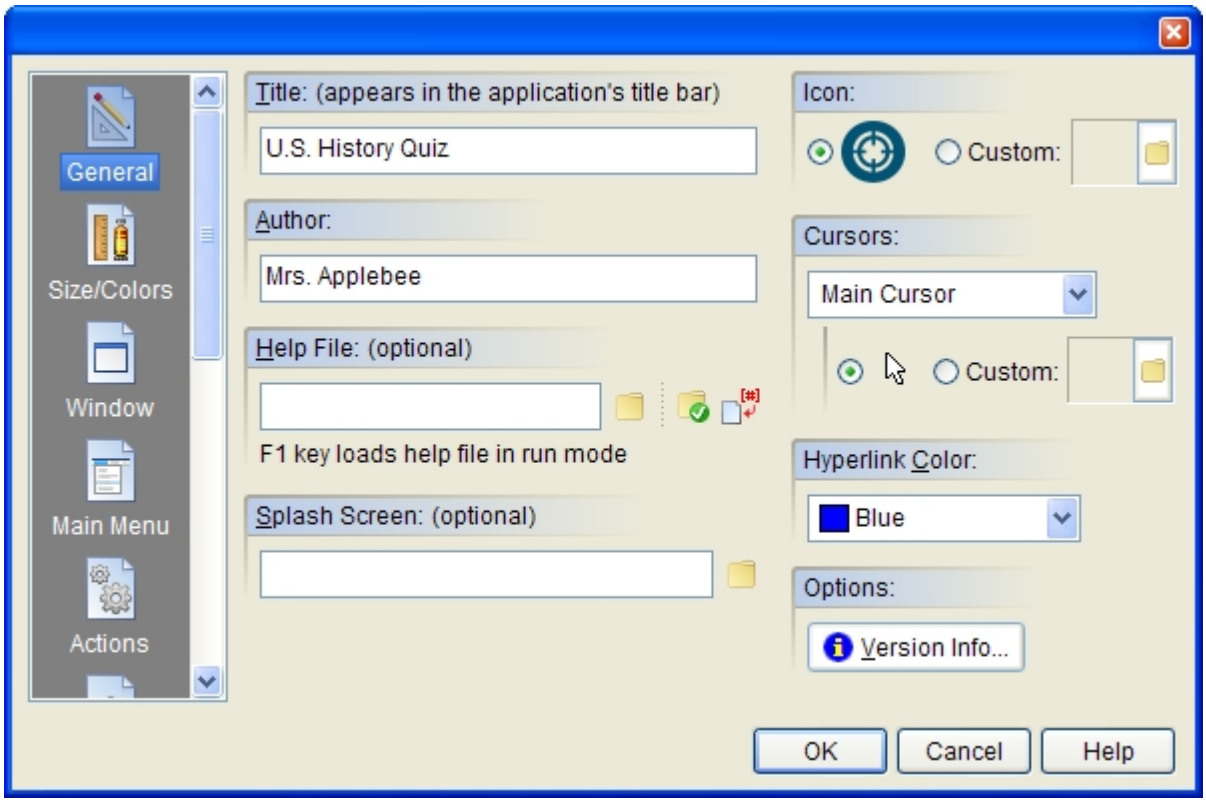
This tutorial demonstrates one method for creating a simple multiple choice test. Our sample test will display a series of multiple choice questions. Students will respond to questions by clicking a Radio Button next to the correct answer. The student's score will be reported at the end of the test.

Getting Started

1. Launch VisualNEO for Windows and create a new publication by selecting **New** from VisualNEO for Windows's **File** menu.

2. Select **App Properties** **> General** from the **App** menu.

The **App Properties** screen will appear.



3. Type `U.S. History Quiz` in the **Title** field.

4. Type your name in the **Author** field.

5. Click the **Access** icon.

6. Remove the check mark from the **Allow Page Up, Page Down, Home and End Keys to change pages** option. Since this is a test, we want to control how students move about the publication. Turning this option off will prevent students from using the keyboard to change pages.

7. Click **OK** to apply these changes to the publication.

Adding Pages

Our publication will contain one page for each question, plus a final page to display the student's score. Since our example test is small, we only need to add two additional pages.

1. Select **Add Page** from the **Page** menu.

The **Add Pages** screen will appear.

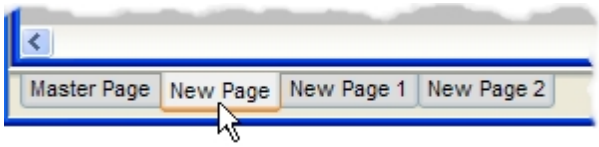
2. Enter "2" into the **Number of Pages to Add** field.

3. Click **OK**.

Question #1

Now we're ready to create our first question.

1. Display the first page by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "New Page".



2. Select the **Simple Text Tool** from the **Tool Palette**.

3. Move the mouse pointer to the center of the page. Press and hold down the left mouse button and draw a rectangle about five inches wide and one half an inch tall. Once the rectangle is the correct size, release the mouse button.

The **Text Properties** screen will appear.

4. Type Who was the first president of the United States? in the **Caption** field.

5. Click **OK**.

The Text object will appear surrounded by selection handles.

6. On the Style Palette, click the small arrow button next to the **Fill Pattern** box.

7. Click the pattern square containing the letter "H". This will make our Text object hollow allowing the page's background color to show through.

8. Click the small arrow next to the Style Palette's **Line Width** box.

9. Select "None" for the line width.

10. Click the small arrow next to the Style Palette's **Font** box.

The **Font Selector** screen will appear.

11. Select **Arial**, **Bold**, **12 Point**.

12. Click **OK**.

If part of the text is missing or not centered, use the mouse to move or resize the object.

Next, let's add some potential answers for question #1.

13. Select the **Radio Button Tool** from the **Tool Palette**.

14. Move the mouse pointer to the center of the page, below the Text object. Press and hold down the left mouse button and draw a rectangle about 2 inches wide and one quarter inch tall. Once the rectangle is the correct size, release the mouse button.

The **Radio Button Properties** screen will appear.

15. Type `George Washington` in the **Caption** field.

While we're here, let's take a look at the **Variable (to store button status)** field. You'll notice that it contains a strange looking entry:

This is called a variable. A variable is simply an area of the computer's memory that VisualNEO for Windows uses to store information while your publication is running. In VisualNEO for Windows, variable names should always be surrounded by brackets ([ ]). This is how VisualNEO for Windows knows that we're talking about a variable named [RadioGroup1] and not the word "RadioGroup1".

Radio Buttons always appear in groups of two or more. Since a complex publication may have multiple groups of Radio Buttons, VisualNEO for Windows uses this variable to determine which buttons go together. Radio Buttons that belong to the same group will all be assigned the same variable. For example, each of the three Radio Buttons that we'll create as potential responses to question #1 will use the variable [RadioGroup1]. Later, when we run our publication, this variable will contain the caption text of the Radio Button selected by the student. At the end of the test, we'll be able to tell which questions were answered correctly by examining the variables assigned to each group of buttons.

16. Click **OK**.

The Radio Button object will appear surrounded by selection handles.
17. Click the small arrow next to the Style Palette's **Font** box.

The **Font Selector** screen will appear.

18. Select **Arial**, **Regular**, **12 Point**.

19. Click **OK**.

If part of the Radio Button's text is missing, use the mouse to resize the object.

Repeat steps 12 through 18 two more times replacing the Caption in step 14 with the following names:

```
Abraham Lincoln
Thomas Jefferson
```

Use your mouse to position the two new Radio Buttons below the first.

Creating a Continue Button

After the student has chosen one of the Radio Buttons, we need to provide a method to proceed to the next question. A simple Push Button is just the ticket.

1. Select the **Push Button Tool** from the **Tool Palette**. 

2. Move the mouse pointer to the lower portion of the page below the Text and Radio Buttons. Press and hold down the left mouse button and draw a rectangle about one inch wide and one half an inch tall. Once the rectangle is the correct size, release the mouse button.

3. Type Continue in the **Caption** field.

When the student clicks this button we want to:

- Make sure one of the Radio Buttons has been selected.
- Go to the next page (which will contain question #2).

To accomplish this, we'll use some simple VisualNEO for Windows Actions.

4. Click the **Actions** icon

5. Type the following into the Action Editor:

```
If "[RadioGroup1]" ">" ""
  GotoNextPage
Else
  AlertBox "Whoops" "You forgot to select an answer. Try again."
EndIf
```

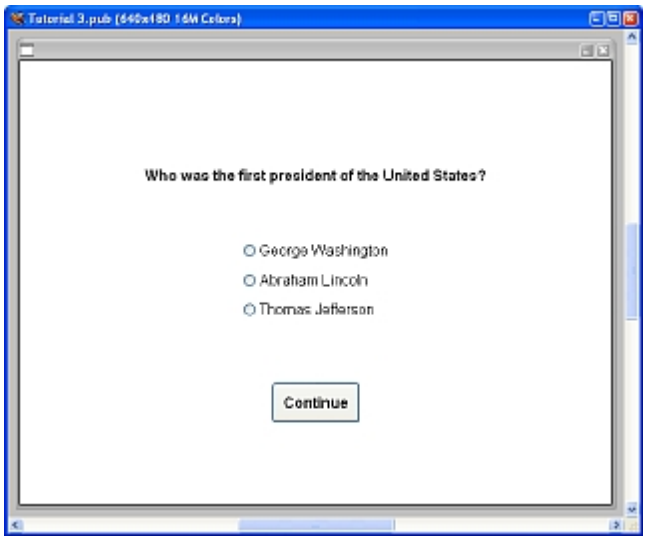
This simple Action script, first examines the contents of the [RadioGroup1] variable to make sure it contains an answer of some kind (correct or not). If it does, we tell VisualNEO for Windows to go to the next page, otherwise we display an error message instructing the student to try again.

6. Click **OK**.

The Push Button object will appear surrounded by selection handles.

7. Use the Style Palette to set the Text object's **Fill Pattern** to "Solid", **Fill Color** to gray and **Line Width** to 1 pixel.

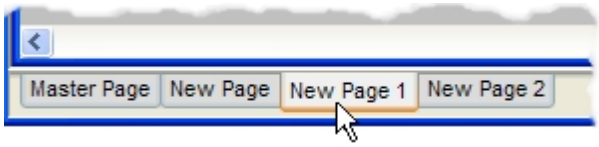
If needed, use your mouse to adjust the position of the objects to match the illustration below:



Question #2

Next, let's add another question.

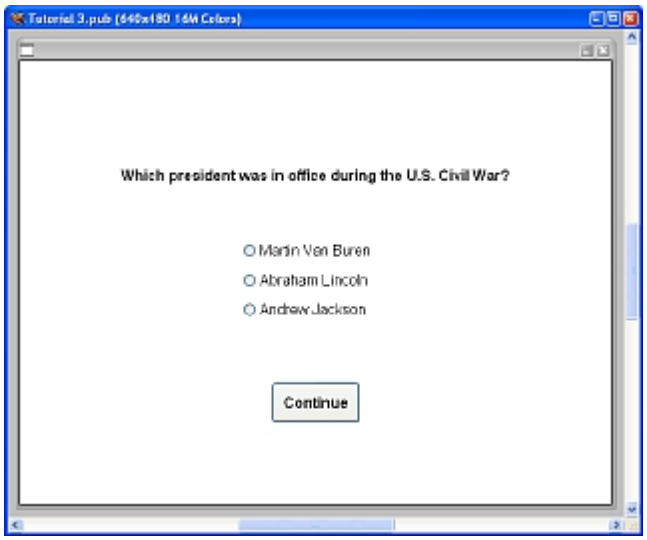
1. Display the second page by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "New Page 1".



2. Try this one on your own. Use the principals you learned above to add the following question and answers to the second page:

VisualNEO for Windows will automatically assign a variable called [RadioGroup2] to each of the Radio Buttons you add to this page. In order to make sure that Question #2 works correctly, let's take a look at the Actions assigned to this page's Continue button.

3. Select the Continue button, by clicking it with your mouse pointer.



4. Choose **Object Properties** from the **Edit** menu.

The **Push Button Properties** screen will appear.

5. Click the **Actions** icon.

6. Modify the contents of the Action Editor so that it looks like this:

```
If "[RadioGroup2]" ">" ""
  GotoNextPage
Else
  AlertBox "Whoops" "You forgot to select an answer. Try again."
EndIf
```

Scoring the Test

Now it's time to calculate our student's grade.

1. Display the third page by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "New Page 2".

2. Select the **Simple Text Tool** from the **Tool Palette**. 

3. Move the mouse pointer to the center of the page. Press and hold down the left mouse button and draw a rectangle about two inches wide and two inches tall. Once the rectangle is the correct size, release the mouse button.

The **Text Properties** screen will appear.

4. Type the following text in the **Caption** field:

```
Correct Answers: [Right]
Incorrect Answers: [Wrong]
```

5. Click **OK**.

The Text object will appear surrounded by selection handles.

6. Use the Style Palette to set the Text object's **Fill Pattern** to "Hollow" and **Line Width** to "None".

If part of the text is missing or not centered, use the mouse to move or resize the object.

7. Select **Page Properties** from the **Page** menu.

The **Page Properties** screen will appear.

8. Click the **Actions** icon

9. Type the following into the Action Editor:

```
SetVar "[Right]" "0"
SetVar "[Wrong]" "0"

If "[RadioGroup1]" "=" "George Washington"
  SetVar "[Right]" "[Right]+1"
Else
  SetVar "[Wrong]" "[Wrong]+1"
EndIf

If "[RadioGroup2]" "=" "Abraham Lincoln"
  SetVar "[Right]" "[Right]+1"
Else
  SetVar "[Wrong]" "[Wrong]+1"
EndIf
```

This Action script uses SetVar to set two variables, [Right] and [Wrong], to zero. The two If statements that follow examine the contents of [RadioGroup1] and [RadioGroup2] and increment [Right] for a correct answer or increment [Wrong] for an incorrect answer.

10. Click **OK**.

Saving The Publication

Let's save what we've done so far.

1. Select **Save** from the **File** menu.

Since we haven't saved this publication before, the **Save As** screen will appear.

2. In the VisualNEO for Windows 5 folder on your hard drive, open the Tutorial Files folder.

3. Type Tutorial3.pub in the **File Name** field.

4. Click **Save**.

Testing The Publication

We're ready to try out our test.

1. Select **Run (From Start)** from the **App** menu.

2. Test the publication by answering each question and clicking the **Continue** buttons.

3. When you reach the third page, your score should be displayed.

4. Click the publication window's close button or press the **Esc** key on your keyboard to return to authoring mode.

Possible Improvements

This example test is admittedly simple. However, the concepts learned here can be employed to create much more complex tests. One idea for improvement is to add an option to save the test results to a file. This can be done by adding a simple command to the end of the last page's Enter Action. For example:

```
FileWrite "Test.dat" "1" "Correct: [Right], Incorrect: [Wrong]"
```

Another improvement, would be to provide an opportunity at the beginning of the test for students to enter their name or student ID number. The possibilities are almost endless.

Tutorial 4: Creating a Resizable Publication

This tutorial will introduce you to the special techniques involved in creating a resizable publication. The first thing we need to do is decide what our publication's interface will look like. Generally, most Windows applications are divided into a few distinct interface components: a tool bar, status bar and work space. More complex applications may include additional elements, but the process is basically the same.

For this tutorial we will create a basic image viewer. Our publication will consist of a tool bar (containing a file selector button); a status bar (showing the name of the image file being viewed); and a work space (where the image will be displayed).

Getting Started

1. Launch VisualNEO for Windows and create a new publication by selecting **New** from VisualNEO for Windows's **File** menu. From the New Publication screen select the following: **640 x 480**, **16 Million Colors** and **Standard Application (EXE)**.

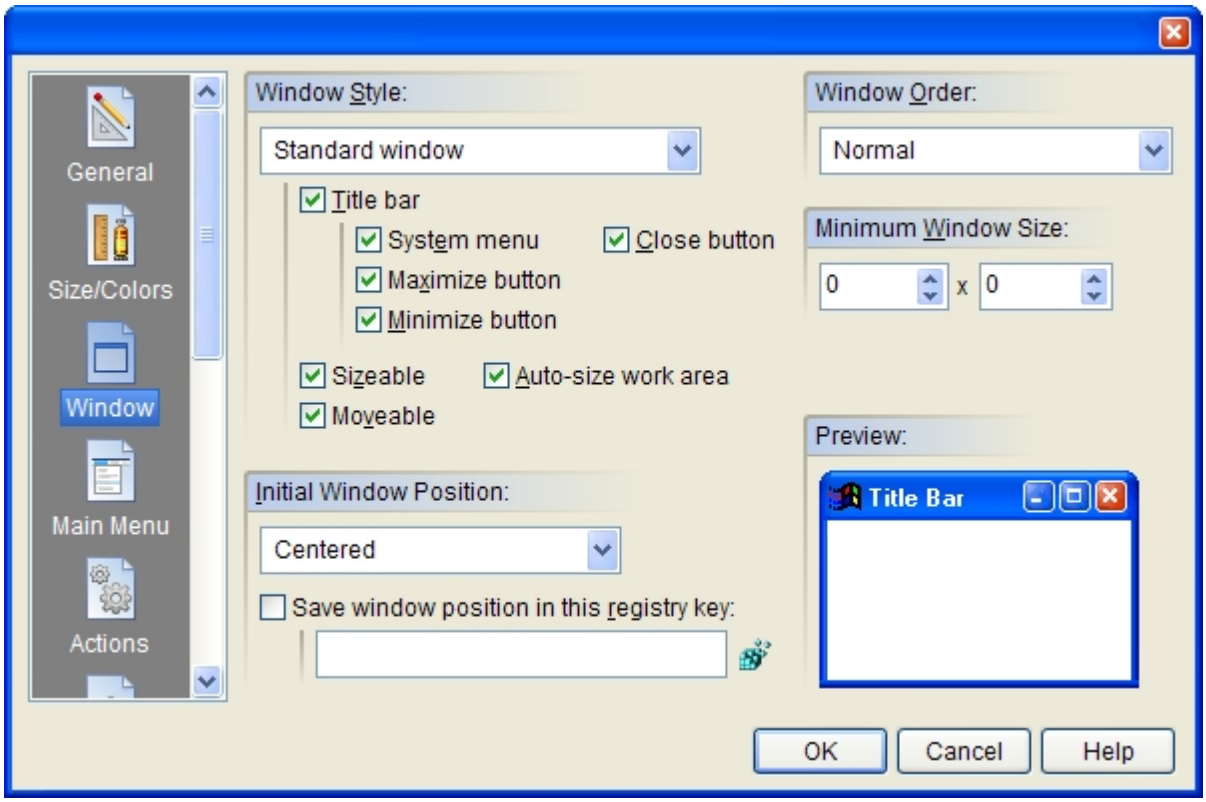
2. Select **App Properties > General** from the **App** menu.

The **App Properties** screen will appear.

3. Type `Image Viewer` in the **Title** field.

4. Type your name in the **Author** field.

5. Click the **Window** icon to display that section of the App Properties screen.



6. Under **Window Style**, make sure **Standard Window** is selected and the following options are checked: **Title bar**, **System menu**, **Maximize button**, **Minimize button**, **Sizeable**, **Moveable** and most important **Auto-size work area**.

7. Click **OK** to apply these changes to the publication.

The Interface Components

The first thing we need to do is add three Container objects - one for each of our main interface components. Assigning the correct properties to these Containers is the key to creating a sizeable publication.

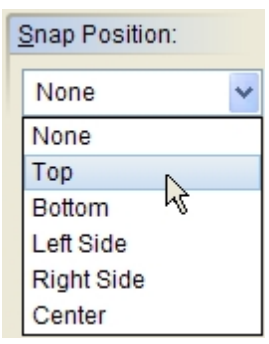
1. Select the **Container Tool** from the **Tool Palette**.

2. Move the mouse pointer to the upper portion of the page. Press and hold down the left mouse button and draw a rectangle about 2 inches wide and ½ inch tall. Once the rectangle is the correct size, release the mouse button.

3. Place the mouse pointer over the Container object and click the right mouse button.

The **Container Properties** screen will appear.

4. Set the **Snap Position** property to **Top**.



5. Click **OK** to save the changes.

The Container will reposition itself along the top of the page. This Container will serve as our publication's tool bar. The Snap Position setting will keep it docked along the top edge of the page even if our reader's later alter the publication window's shape.

The next Container will become our publication's status bar.

6. Select the **Container Tool** from the Tool Palette. 

7. Move the mouse pointer to the lower portion of the page. Press and hold down the left mouse button and draw a rectangle about 2 inches wide and ½ inch tall. Once the rectangle is the correct size, release the mouse button.

8. Place the mouse pointer over the new Container object and click the right mouse button.

The **Container Properties** screen will appear.

9. Set the **Snap Position** property to **Bottom**.

10. Click **OK** to save the changes.

Finally, let's create the most important Container of all - the work space.

11. Select the **Container Tool** from the Tool Palette. 

12. Move the mouse pointer to the center of the page. Make sure the pointer is **not** on top of any other objects. Press and hold down the left mouse button and draw a rectangle about 2 inches wide and 2 inches tall. Once the rectangle is the correct size, release the mouse button.

13. Place the mouse pointer over the new Container object and click the right mouse button.

The **Container Properties** screen will appear.

14. Set the **Snap Position** property to **Center**.

15. For this Container, let's also set both of the **Margin** properties to "8" pixels. This will give us some space between the edge of the work space Container and the Picture object we're going to add later.

16. Click **OK** to save the changes.

Adding a Picture Object

The publication we're creating is an image viewer, so we need to add a Picture object to handle that task.

1. Select the **Picture Tool** from the **Tool Palette**. 

2. We need to make sure that the Picture object is attached to the work space Container (the one in the center of the page). To accomplish this, simply position the mouse pointer on top of the work space Container. Press and hold down the left mouse button and draw a rectangle. The size isn't important because we're going to change it in a moment.

A Windows **File Selector** screen will appear.

3. In this case, we don't really need to select a particular file, but VisualNEO for Windows won't let us create a Picture object without one. To satisfy VisualNEO for Windows, just select any image file you like and click **OK**.

The selected image file will be inserted into the Picture object and displayed on the page.

4. Place the mouse pointer over the Picture object and click the right mouse button.

The **Picture Properties** screen will appear.

5. The name of the image you selected will appear in the **Image File** field. Backspace over the entire file name and enter the following (including the square brackets) in its place:

```
[ImageFile]
```

Your screen should look like this:

6. Set the Picture's **Snap Position** property to **Center**.

7. Click the **Appearance** icon.

8. Set the **Display Mode** property to **Actual Size**.

9. Place check marks in both the **Scroll bars** and **Allow mouse to pan image** options.

10. Click **OK**.

The Picture will snap to match the dimensions of the work space Container minus the margins we specified earlier. Because the Picture object is attached to the Container, it snaps to the bounds of the Container instead of the entire publication. Don't worry if the Picture is blank - that's what we want.

Adding Text to the Status Bar

Next, let's add the name of the image file we're viewing to the status bar.

1. Select the **Simple Text Tool** from the **Tool Palette**. 

2. We need to make sure that the Simple Text object is attached to the status bar Container (the one at the bottom of the page). To accomplish this, simply position the mouse pointer on top of the status bar Container. Press and hold down the left mouse button and draw a rectangle. The size isn't important because we're going to change it in a moment. Just make sure that the rectangle begins and ends within the bounds of the Container.

The **Text Properties** screen will appear.

3. Type [ImageFile] in the **Caption** field. (This is the same variable name we used in the Picture's Image File field above.)

4. Set **Horizontal Margin** to "2" and **Vertical Margin** to "0".

5. Set the **Snap Position** property to **Center**.

6. Click **OK**.

7. With the Text object selected, click the small arrow next to the Style Palette's **Line Width** box.

8. Select "None" for the line width.

9. Click the small arrow next to the Style Palette's **Font** box.

10. Select **Arial**, **Normal**, **10 Point**.

Adding a File Selector Button

Finally, we need to add a button to the tool bar Container so that our readers can select image files for viewing.

1. Select the **Push Button Tool** from the **Tool Palette**. 

2. We need to make sure that the Push Button object is attached to the tool bar Container. To accomplish this, simply position the mouse pointer on top of the tool bar Container (the one at the top of the page). Press and hold down the left mouse button and draw a rectangle. The size isn't critical, just make sure that the rectangle begins and ends within the bounds of the Container.

The **Push Button Properties** screen will appear.

3. Type Open in the **Caption** field.

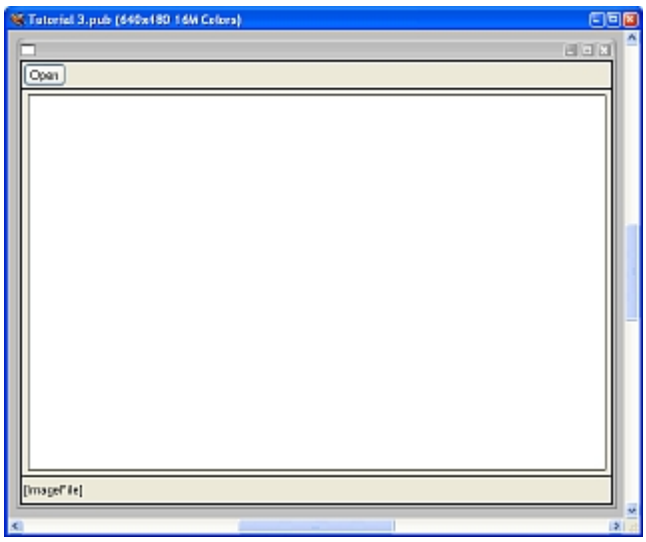
4. Click the **Actions** icon.

5. Type the following into the Action Editor:

```
FileOpenBox "Select an Image" "Images|*.jpg;*.bmp;*.png;*.gif" ""
"[ImageFile]" ""
```

6. Click **OK**.

If part of the Open button's text is missing, use the mouse to resize the object. Your publication should look something like this:



Saving The Publication

Before going any further, let's save what we've done so far.

1. Select **Save** from the **File** menu.

2. In the `VisualNEO for Windows 5` folder on your hard drive, open the `Tutorial Files` folder.

3. Type `Tutorial4.pub` in the **File Name** field.

4. Click **Save**.

Testing The Publication

We're ready to take our image viewer for a test drive.

1. Select **Run (From Start)** from the **App** menu.

2. Test the publication by clicking the **Open** button and selecting an image file.

3. The purpose of this tutorial was to create a resizable publication, so let's test that by using the mouse to change the size and shape of the window. As the window's shape changes, notice how the Containers automatically adjust to compensate while retaining their relative positions.

4. Click the publication window's close button or press the **Esc** key on your keyboard to return to design mode.

Congratulations, you've just created a resizable publication!

Tutorial 5: Creating an E-Book

In this tutorial you will learn how to use VisualNEO for Windows's Linked-Article object to automatically flow a text document across multiple pages.

Getting Started

1. Launch VisualNEO for Windows and create a new publication by selecting **New** from VisualNEO for Windows's **File** menu.

2. From the New Publication screen select the following: **640 x 480**, **16 Million Colors** and **Standard Application (EXE)**.

A new blank publication will appear.

3. Select **Page Properties** from the **Page** menu.

4. Select the **Wallpaper** option for the **Page Background**.

5. Click the folder icon to the right of the Wallpaper's **Image File** field.

A Windows **File Selector** screen will appear.

6. Locate the `Documents\VisualNEO for Windows 5` folder. (If you installed VisualNEO for Windows's sample files in a different folder, use that instead.)

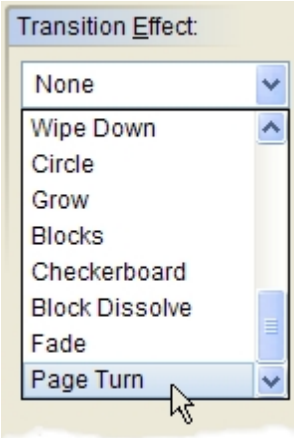
7. Open the `Tutorial Files` folder.

8. Select the `Empty Book.jpg` file.

9. Click **Open**.

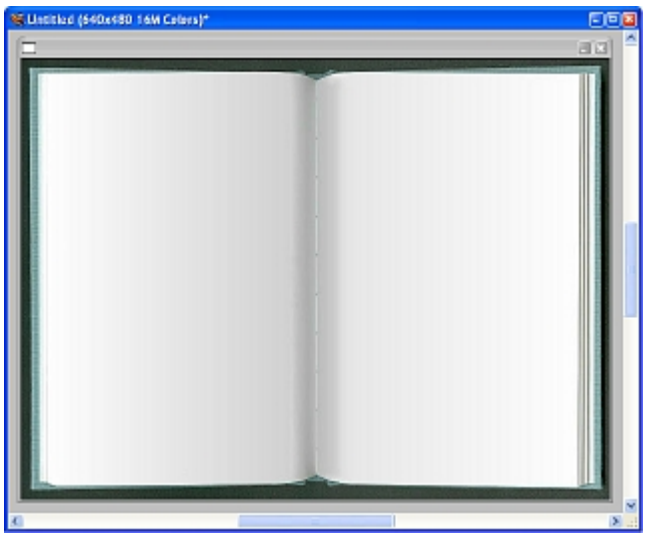
You will be returned to the Page Properties screen.

10. Select **Page Turn** from the list of **Transition Effects**.



11. Click **OK**.

Your publication's screen should now look like the picture at the top of the following page:



Importing Text

We'll use the **Linked-Article Tool** to import text for our e-book because it has the ability to automatically distribute text across multiple pages. Before importing our text, we first need to make some changes to VisualNEO for Windows's Style Palette.

1. On the Style Palette, click the small arrow button next to the **Fill Pattern** box.

2. Click the pattern square containing the letter "**H**". This will make the Linked-Articles we're about to create hollow.

3. Click the small arrow next to the Style Palette's **Line Width** box.

4. Select "**None**" for the line width.

5. Click the small arrow next to the Style Palette's **Font Color** box.

6. Select "**Black**" from the **Color Selector**.

Now we're ready to import our text.

7. Select the **Linked-Article Tool** from the **Tool Palette**. 

8. Use your mouse to draw a rectangle on the left side of the page. Don't worry if the rectangle isn't perfect, we're going to resize it later.

The **New Linked-Article** screen will appear.

9. Select the **Open an existing document** option.

10. Click **OK**.

A Windows **File Selector** screen will appear.

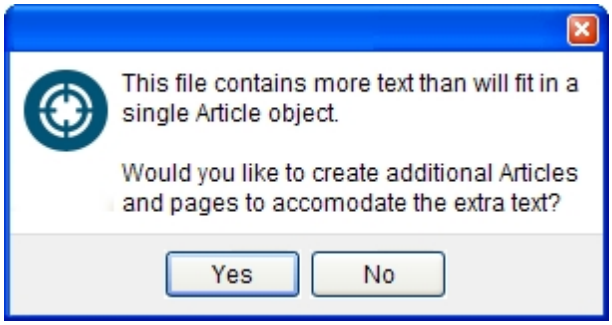
11. Locate the `Documents\VisualNEO for Windows 5` folder. (If you installed VisualNEO for Windows's sample files in a different folder, use that instead.)

12. Open the `Tutorial Files` folder.

13. Select the `Intro to Electronic Publishing.rtf` file.

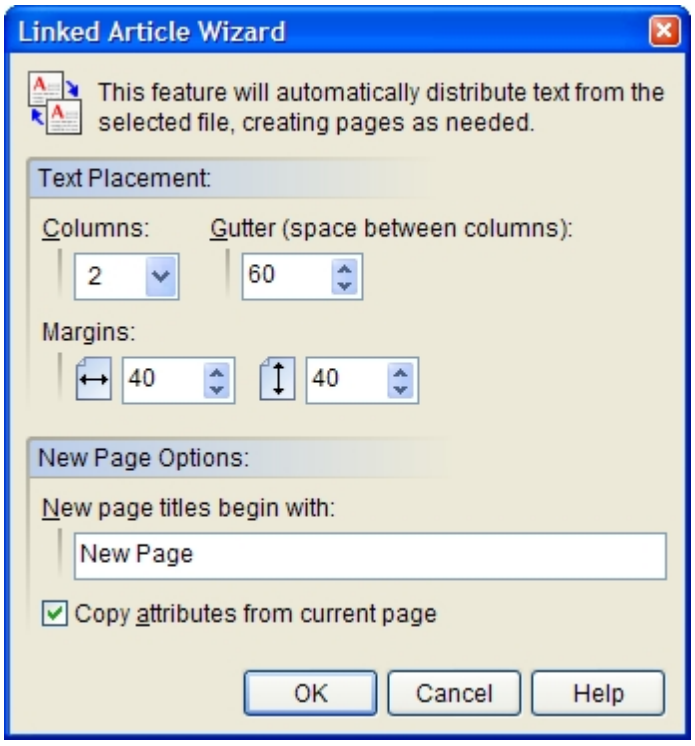
14. Click **OK**.

Since the text file we selected is too large to fit in a single Linked-Article, VisualNEO for Windows will display the message box below:



15. Click **Yes**.

The Linked-Article Wizard screen will appear.



16. Select "**2**" for the number of **Columns**.

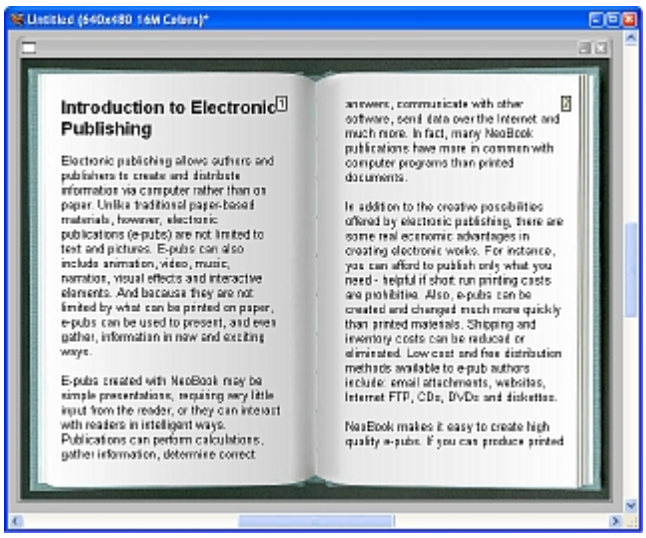
17. Enter "**60**" for the **Gutter** - the space between the Columns.

18. For both the Horizontal and Vertical **Margins**, enter "**40**".

19. Place a check mark in the box next to the **Copy attributes from current page** option.

20. Click **OK**.

VisualNEO for Windows will now distribute the text across the publication, creating additional pages as needed. When the process is complete, your publication should contain several new pages, and the screen should look like this:



Adding Navigation Buttons

We can use the Page Up and Page Down keys to navigate through our E-Book, but let's make life easier for our readers by adding some navigation buttons.

1. Display the **Master Page** by clicking the tab at the bottom of the VisualNEO for Windows screen labeled "Master Page".

2. Select the **Push Button Tool** from the **Tool Palette**. 

3. Move the mouse pointer to the lower right portion of the page. Press and hold down the left mouse button and draw a rectangle about ½ inch wide and ¼ inch tall. Once the rectangle is the correct size, release the mouse button.

The **Push Button Properties** screen will appear

4. Type `Next` in the **Caption** field.

5. Click the **Actions** icon.

6. Click the **Insert Action** button.

7. Click the **Navigation** category.

8. Select the **GotoNextPage** Action.

VisualNEO for Windows will automatically add the command to the button's Action Editor. This Action tells VisualNEO for Windows to advance to the next page whenever the button is pressed.

9. Click **OK**.

If the button's position or size don't look right, use your mouse to make any needed adjustments. You may also want to use the Style Palette's controls to alter the button's fill color, fill style, outline and font.

Our second navigation button is almost identical to the first so let's use VisualNEO for

Windows's Duplicate command to make a quick copy.

10. Make sure the button you just created is selected. (If it's not, click it with the mouse.)

11. Choose **Duplicate** from the **Edit** menu.

A second, identical button will appear on your screen.

12. Use your mouse to drag the duplicate button to the far left side of the page.

13. With the duplicate button still selected, choose **Object Properties** from the **Edit** menu.

14. Replace the button's current **Caption** with the word Prev.

15. Click the **Actions** icon.

16. Erase the existing button Action (GotoNextPage). (Use the Backspace or Delete keys on your keyboard just as you would in a word processor.)

17. Click the **Insert Action** button.

18. Select **GotoPrevPage** from the **Navigation** category.

19. Click **OK**.

Saving The Publication

Before going any further, let's save what we've done so far.

1. Select **Save** from the **File** menu.

2. In the VisualNEO for Windows 5 folder on your hard drive, open the Tutorial Files folder.

3. Type Tutorial5.pub in the **File Name** field.

4. Click **Save**.

Testing The Publication

Let's take our E-Book for a spin and see how it works.

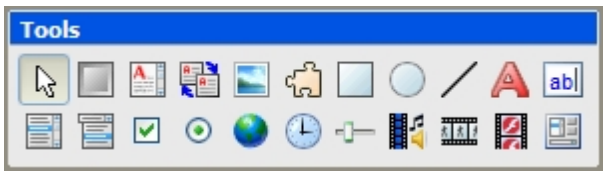
1. Select **Run (From Start)** from the **App** menu.

2. Test the publication by clicking the **Next** and **Prev** buttons.

3. When you're finished, click the publication window's close button or press the **Esc** key on your keyboard to return to design mode.

The Tool Palette

Most of the tools you will need to create and edit publications can be found in VisualNEO for Windows's Tool and Style Palettes. This Section will explain how these tools are used to build publications. For more information about a specific tool, click the picture below:

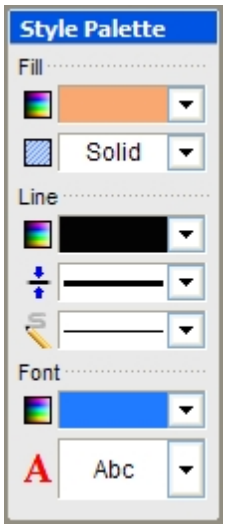


Tool Buttons

In VisualNEO for Windows, every element you add to your publication (text, pictures, buttons, etc.) is considered an object. Objects are created using the Object Tool Buttons above. Each button represents a different type of VisualNEO for Windows object. Each object has distinct capabilities and features. To create a new object, select a tool button (other than the Selection Tool) and use the mouse to draw a rectangle on your publication. For most types of objects, a properties screen will appear requesting additional information. For example, when creating objects with the Picture or Article Tools, VisualNEO for Windows will ask you to supply the name of an image or text file.

The Style Palette

Once created, the appearance of most types of objects can be modified using the Style Palette's controls. Before an object can be modified, however, it must be selected using VisualNEO for Windows's Selection Tool. Once selected, objects may be modified using the controls below:



Fill Color

Click the small arrow button to the right of this box to select a new Fill Color for the currently selected object. A special **Color Selector** window will appear, providing you with a selection of colors to choose from.

You may choose colors from the publication's **Color Palette** or from a selection of Windows' **System Colors**. Two small icons at the top of the Color Selector allow you to toggle between these choices.

For 256-color publications, your palette choices will match those assigned to the current page. (See the File > New or Page > Page Properties commands for more information on page palettes.) For 16-color publications, you may choose from the standard 16 Windows colors. For 16 Million color publications, you may select from a rainbow of colors or create your own shades by adjusting the color values using the RGB (red, green, blue) controls.

The System Colors are associated with various portions of the Windows interface (such as title bars, dialogs, buttons, etc.). System Colors can vary from computer to computer, since Windows allows users to customize these settings to suit their individual tastes. Select a System Color if you wish to use the same colors your readers have defined for their Windows interface.

Note: *The System Colors displayed on your computer most likely will not be the same as those on your reader's computer. Keep this in mind when designing your publication.*

Fill Pattern

Click the small arrow to the right of this box to select a Fill Style for the selected object's interior. Choose a fill pattern, click the "H" square for a **Hollow** object or click the "S" square to create a **Solid** object using the current Fill Color.

Enable the **Transparent Patterns** option to paint pattern-filled objects transparently, allowing items underneath to show through openings in the selected pattern.

Note: *Objects, like Lines, which do not have interiors, are not affected by changes to the Fill Color and Fill Pattern.*

Line Color

Click the small arrow to the right of this box to select a Line Color for the selected object. (See Fill Color above for information about using the Color Selector.) If your object doesn't appear to have a border, make sure the Line Width (below) is not set to "None."

Line Width

Click the small arrow to the right of this box to change the Line Width for the selected object. Selecting "None" will display the object without a border.

Line Style

Click the small arrow to the right of this box to choose a different Line Style for the selected object. Line styles include solid, dashed, dotted and 3D lines. If your object doesn't appear to have a border, make sure the Line Width (above) is not set to "None."

Note: *Some objects do not have borders or support line styles other than solid.*

Font Color

Click the small arrow to the right of this box to select a color for the selected object's text. (See Fill Color above for information about using the Color Selector.) Objects that do not contain text are not affected by this control.

Font

Click the small arrow to the right of this box to change the font used by the selected

object's text. The choices here depend on what fonts have been installed on your computer. Windows includes a basic selection of fonts and more are available from a variety of sources. Consult your Windows User Guide for information about installing new fonts on your computer. Objects that do not contain text are not affected by this control.

Note: *Some fonts are copyrighted and cannot be distributed freely without the permission of their authors or publishers. Before using fonts in your publication, make sure you have the legal authority to do so. There are many free fonts available on the Internet which you may distribute with your publications. Also, using a large number of different fonts can greatly increase the size of your publication.*

Selection Tool

 Use the Selection Tool to select, move and resize objects. Click on an object using the left mouse button to select it. Select multiple objects by holding down the Shift key while clicking on additional objects, or by dragging the mouse to surround the desired objects in a rectangle. Selected objects will be surrounded with small, box-like handles. For example:

Once selected, an object or group may be moved by clicking on the object with the left mouse button, then dragging the object or group to a new location before releasing the mouse button. To resize an object, simply click and drag one of the object's selection handles. Holding down the Shift key while resizing an object restricts the object's shape to a perfect square. Selected objects also can be cut or copied to the Windows Clipboard, or removed from the publication by pressing the Delete key.

Depending on the object selected, clicking it with the right mouse button will allow you to modify the object's appearance and define how the object behaves. Most types of objects can also be assigned Actions to execute when clicked on by the reader.

Common Properties

The following properties are common to most types of objects:

Object Name

VisualNEO for Windows will automatically assign new objects a unique name. You may change this name if you like, but no two objects in a single publication may have the same name. Object names are important because they are used to identify individual objects in a variety of places, including many of VisualNEO for Windows's Action Commands. For example, while your publication is running, you can hide an object by passing its name to the HideObject Action like this:

```
HideObject "PushButton1" "None" "0"
```

Hint

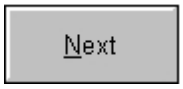
Many objects also include an optional Hint to be displayed whenever the reader's mouse pointer hovers over the object. A Hint is a word or phrase that indicates to the reader what the object is for and what might happen when it's clicked. For example, readers confronted with a Push Button containing a simple icon, but no descriptive caption might find a Hint like the one below extremely helpful:

If your object doesn't require a Hint, leave this field blank.

Short Cut Key

Readers can be given the choice of using the mouse or the keyboard to activate certain objects by entering a key into the Short Cut Key field. You can enter a single letter (like "A") or use a combination keystroke (such as CTRL+A). To enter a combination keystroke, click the Short Cut Key field and type the key combination on your keyboard. For example, to enter CTRL+A, hold down the Ctrl key and press the letter A at the same time.

It's usually a good idea to give your readers an indication that a short cut key is available. Many Windows programs identify a short cut key by underlining that letter in the button's caption. You can do this in VisualNEO for Windows by placing a caret character "^" in front of the letter in the caption to be underlined. For example: "^Next" will appear on screen as:



Snap Position

When creating publications that can be resized, you can use Snap Position to align an object to the edge or center of the workspace. Available options are: None, Top, Bottom, Left Side, Right Side and Center. When an option other than None is selected, the object will conform itself automatically to the dimensions of whatever side it's snapped to. When Center is selected, the object will grow to fill the portion of the workspace that is not occupied by other snapped objects. If the object has been placed on a Container object, then the selected Snap Position applies to the bounds of the Container instead of the entire workspace.

Note: *When the Snap Position property is enabled, you will not be able to move the object using the mouse. To move the object, you must set the Snap Position property back to None.*

Initial State

For most objects, you can specify an Initial State by modifying the Visible and Enabled options. These settings represent the object's state when your publication first starts. To hide an object, remove the check mark from the Visible option. To disable an object, remove the check mark from the Enabled option. A disabled object will not respond to mouse clicks or keyboard events.

Hint: *An object's state can be modified while your publication is running using the ShowObject, HideObject, EnableObject and DisableObject Actions.*

Lock Position

Enabling the Lock position option in the lower left corner of the screen will fix the object's position on the page and prevent it from being accidentally moved during the publication's editing phase.

Actions

Most objects have the ability to perform tasks in response to specific events. These tasks can be as simple as jumping to another page or as complex as calculating a test score. Objects perform these tasks based on special instructions entered by you. In VisualNEO for Windows, these instructions are called Actions.

Depending on its purpose, each object has its own set of events it can respond to. Common events include clicking a mouse button, moving the mouse pointer, pressing a key on the keyboard, etc.

The Action screen contains a large editor window and a tool bar. Action commands may be typed directly into the editor, but most authors prefer to use the **Insert Action** button instead. You may specify separate Actions for each of the object's events by selecting one of the tabs located at the bottom of the editor window.

Push Button Tool

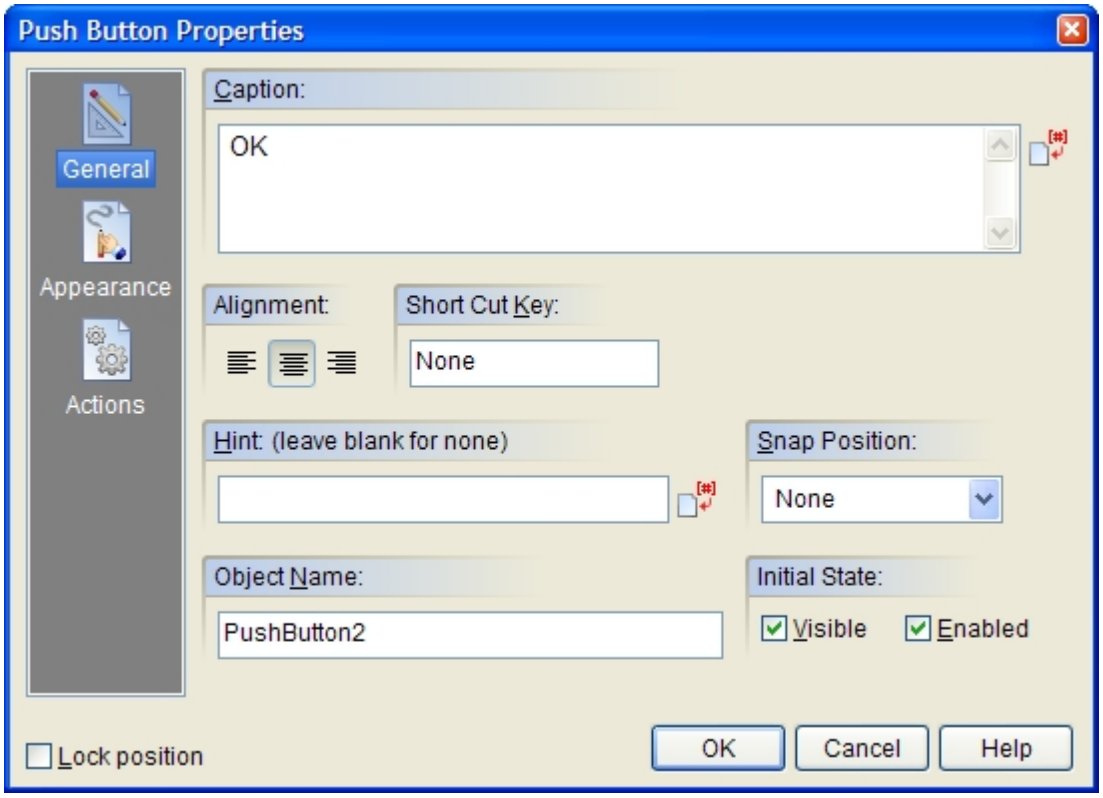
Push Buttons are probably the simplest and most easily understood method for readers to interact with your publications. Just about anyone who's ever used a computer or an airport kiosk will immediately know how to use one. Push Buttons can be used to navigate between pages, perform calculations, display messages, send email and perform a variety of other tasks.

To create a Push Button, use the mouse to draw a rectangle where you would like the button to appear. The Push Button Properties screen will be displayed allowing you to define the button's appearance and behavior.

The **Push Button Properties** screen is divided into three sections, indicated by the icon images on the left: General, Appearance and Actions. To view the settings for a section, click the corresponding icon.

General

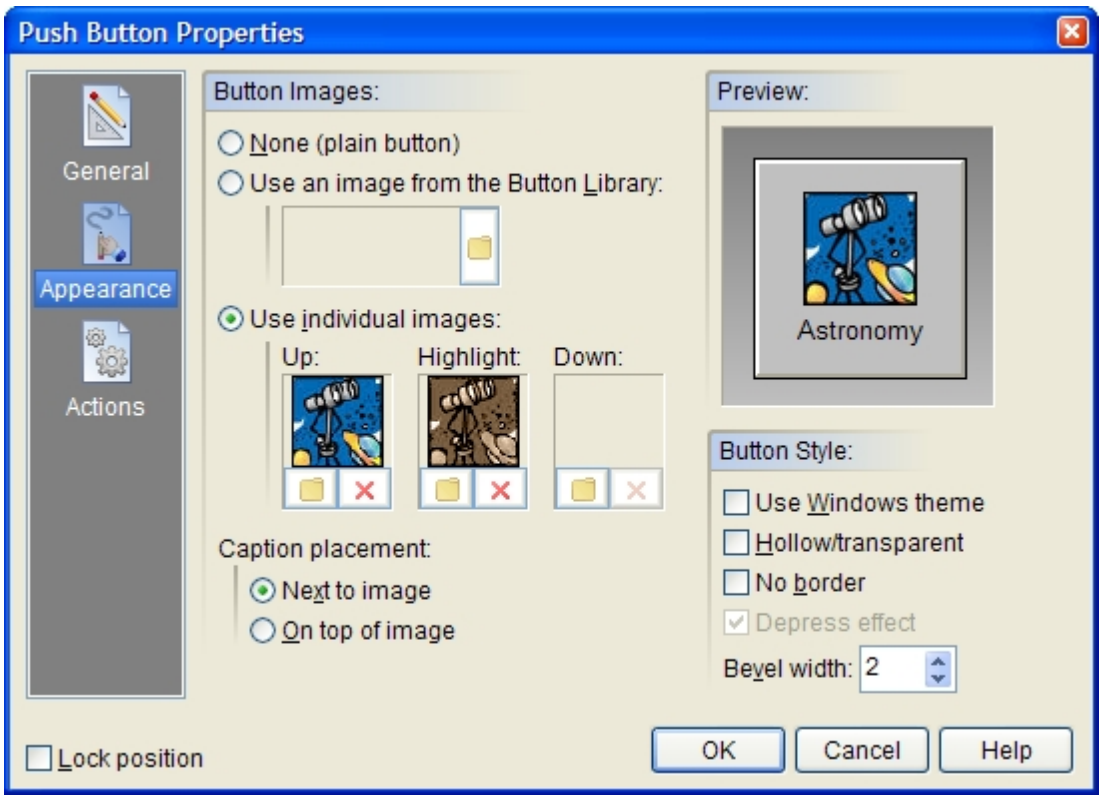
Text entered into the **Caption** field will appear inside your button. Helpful caption's generally indicate to the reader what will happen when the button is clicked. You may leave this field blank if your button does not require a caption.



You may set the alignment for the button's text using one of the three **Alignment** buttons: Left Aligned, Centered or Right Aligned. If the button contains an image (see Appearance below), it will be aligned using the method chosen here.

Appearance

In addition to text, Push Buttons may contain small images or icons to represent each of the button's three states: Up, Down and Highlight (mouse over). Interesting effects can be created by using slightly different versions of the same image for each state.



Under Button Images, select **None** to create a simple, plain, text-only button. Selecting **Use an image from the Button Library** allows you to choose from a list of special pre-made files containing images for all three button states. The **Use individual images** option allows you to select separate image files for each of the button states. You can use your favorite paint program to create your own custom button images. When importing individual images, VisualNEO for Windows will prompt you to select a color to serve as the transparent portion of the image. (The button's background will show through pixels in the image matching this color.) By default, VisualNEO for Windows will use the color of the pixel in the lower left corner of the image, but you may select another color if you like.

If your button contains both text and images, you may choose to place the caption **Next to image** or **On top of image**.

The **Preview** window on the right side of the screen will be updated to reflect how the button will look using the appearance settings you select. Move the mouse pointer over the preview to see the Highlight image or click the preview to see the Down image.

Enabling the **Use Windows theme** option will display the button using the Windows XP/Vista style. This feature only works when your publication is viewed under Windows XP and Vista. When viewed under older versions of Windows, the button will be drawn using the normal method.

Enable the **Hollow/transparent** option to set the button's Fill Style to Hollow. When this option is enabled, the button will appear to be transparent, allowing objects underneath to

show through. Click the **No border** option to create a button that does not have a border outline.

Hint: *Both the Fill Style and Border also can be changed using the controls on VisualNEO for Windows's Style Palette.*

You may specify a **Bevel width** for the button's three-dimensional border. The default setting is 2, but you may change this to any number between 0 and 10 pixels. The Preview window will show the results of changes made to this setting.

The **Depress effect** option causes the button image or text to shift to the right and down one pixel when the button is clicked. This option is only available when Bevel Width is set to zero and Button Images is set to None or Use individual images.

Hint: *You can create an invisible Hot Spot by setting the button's Fill Pattern to "H" (Hollow) and the Line Width to "None." To create an irregular shaped hotspot, see the Polygon/Hotspot object.*

Actions

Push Buttons support the following Action Events: **Click**, **Right Click**, **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

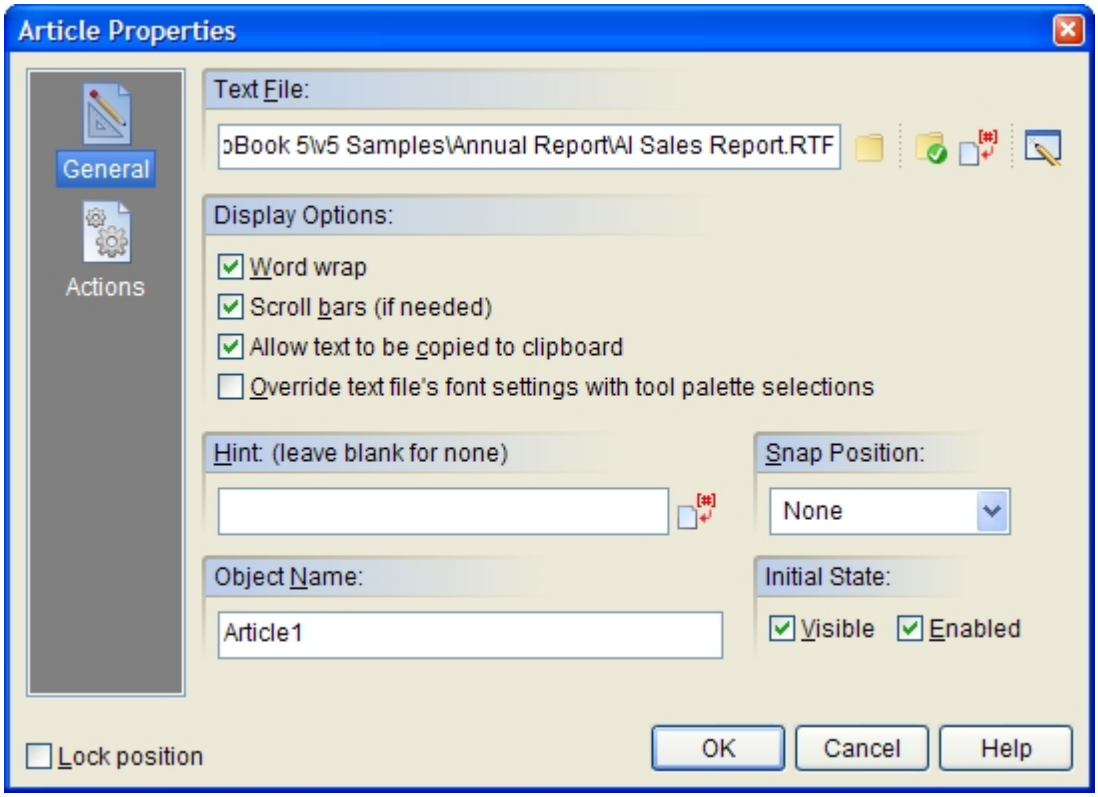
Article Tool

 Use this tool to place a formatted text document or Article into your publication. You may import files containing plain text (ASCII / ANSI), Rich Text (RTF) or convert files created by another program or word processor. Articles are generally used for large or complex blocks of text. Use the Simple Text tool for inserting short pieces of text for titles, captions, etc. To have a text file display across multiple pages, use the Linked-Article Tool.

To import a text document, use the mouse to draw a rectangle where you would like the Article object to appear. You will have the option of selecting an existing plain or Rich Text file, creating a new document, or converting a document created with another program. When creating a new document, VisualNEO for Windows's Text Editor will open with a blank screen ready for you to begin typing. (The Text Editor provides basic word processing functions, plus the ability to add hyperlinks and bookmarks to your text.If converting a file created with another program, you will be prompted to select the file and format using a standard file selector. If the correct format for the file is not listed under Files of Type, then no filter for that format is available and it cannot be imported. Once converted, the file will be displayed in VisualNEO for Windows's Text Editor for proofing. If the conversion is acceptable, click OK to save a copy of the file as Rich Text (VisualNEO for Windows's preferred format).

Note: *Converting files created with other programs requires that a compatible filter be installed and registered with Windows by the source program. The success of the conversion process depends of the quality of the installed filter. If you're having trouble importing a particular file into VisualNEO for Windows, see if your word processor can export to Rich Text (RTF) format, then import that file directly into VisualNEO for Windows. Also, if the word processor isn't the latest version, you can often improve the conversion by downloading an update from the publisher's website.*

Once a file is selected or created, the Article object will appear on your publication. You can modify the Article's properties by right clicking on the object. The **Article Properties** screen will be displayed, allowing you to modify the default behavior of the object.



The Article Properties screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

The **Text File** field contains the name of the file displayed in the Article object. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and the changes will appear in your publication. Only during the Compile stage will the file be bound inside the publication. To the right of the field are four small buttons: click the button to select a different file or click the button to open the current file in VisualNEO for Windows's Text Editor. Experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.

Enable the **Word wrap** option to force the document to conform to the boundaries of the object. When word wrap is on, resizing the Article object will reformat the document to fit the space available.

Turn on the **Scroll bars** if your document is too large to fit within the bounds of the object. Readers will need the scroll bars to bring additional portions of the document into view if it's larger than the space available.

The **Allow text to be copied to the clipboard** option provides readers with the ability to select and copy portions of the document to the Windows Clipboard. If enabled, the mouse pointer can be used to highlight portions of text. Text can be copied by clicking the right mouse button or pressing Ctrl + C. Turn this option off if you want to prevent readers from copying portions of your text without your consent. The items in the popup menu that appear when you click the right mouse button can be translated from the Language page of the App Properties screen.

Most Rich Text files contain specific font selections as part of the document's formatting. You can replace these internal font choices with your own by enabling the **Override text file's font settings with Tool Palette selections** option. Once enabled, you can use

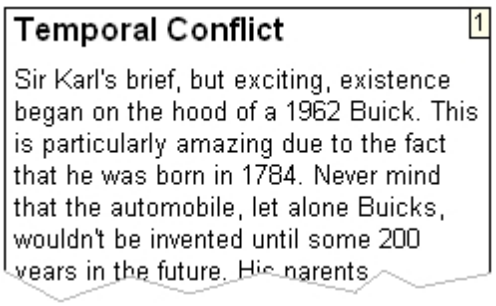
VisualNEO for Windows's Style Palette to select a new font for the text file.

Actions

Articles support the following Action Events: **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. You can define clickable **Hyperlinks** within the Article text using VisualNEO for Windows's Text Editor. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Linked-Article Tool

 The Linked-Article Tool is similar to the standard Article Tool, except that it allows text from large documents to be distributed automatically across multiple pages. All Linked Articles that reference the same text file are considered linked together in a chain. The chain starts with the Linked Article nearest the beginning of the publication and proceeds through the publication to the last Linked Article. In design mode, Linked Articles display a number indicating their position in the chain. For example:



When a single page contains more than one Linked-Article, you can change their position in the chain using the Bring to Front and Send to Back commands found in the Arrange Menu.

There are two methods you can use to create Linked-Articles - Automatic and Manual:

Automatic Mode:

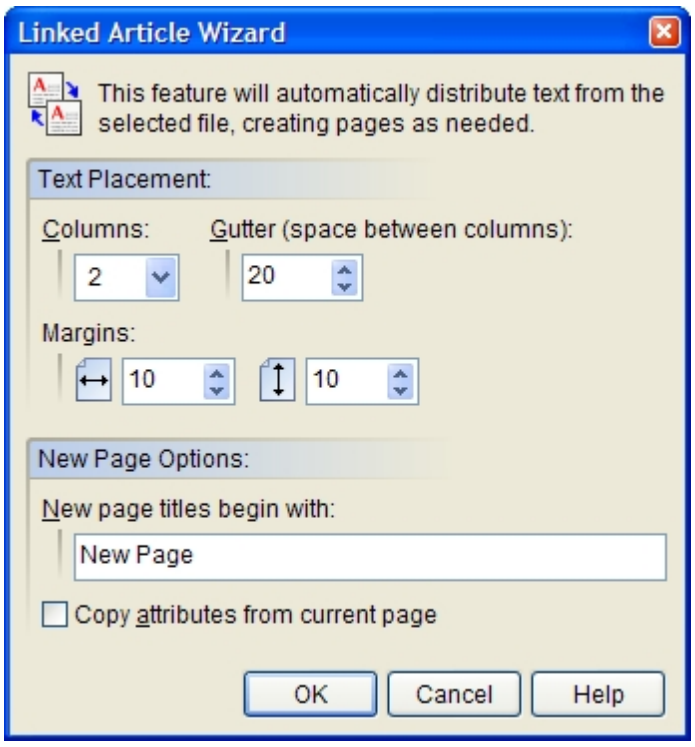
1. Navigate to the page where you want the first portion of the text file to appear.

2. Use the Page Properties screen to define the page's background and transition effect. If your text file is large enough to require more than one page, VisualNEO for Windows will use these properties for any pages it creates.

3. Using VisualNEO for Windows's Style Palette, select the fill and line styles you want to use for displaying the text. VisualNEO for Windows will use these settings for each of the Linked-Articles it creates during this step.

4. Select the Linked-Article icon from the Tool Palette and draw a rectangle on the publication.

5. Follow the instructions on the screen. If the text file you select is too large to fit in a single Linked Article, VisualNEO for Windows will display the **Linked-Article Wizard** (below) and offer to automatically flow the text across multiple pages.



The Linked-Article Wizard allows you to select the number of **Columns** and specify how much space you want between columns (the **Gutter**). You can specify horizontal and vertical page **Margins** to provide a buffer between the text and the edge of the publication. Use the **New page titles begin with** field to specify a base title for any new pages created during this process. For example, if you enter "Sample" as the new title, pages added will be named "Sample 1", "Sample 2", etc. Finally, the enable the **Copy attributes from current page** option if you want new pages to use the same background, transition effect, etc. as the current page (see step 2 above).

VisualNEO for Windows will use these settings when creating pages and distributing text across the publication as the illustration below demonstrates:

Note: *VisualNEO for Windows will offer to flow the text only if there are no Linked-Articles in the publication already using the text file you select.*

After VisualNEO for Windows automatically creates the Linked-Articles, you can make adjustments (position, size, color, font, etc.) to the articles individually, if needed.

Manual Mode

To manually create a group of Linked-Articles, follow steps 1 through 4 above. When asked if you want to automatically create additional Linked-Articles and pages to accommodate the additional text, answer "No." After the first Linked-Article is created, you can create additional Linked Articles using the duplicate, copy-paste, etc. commands. Remember, all Linked-Articles that point to the same file are considered linked no matter how they have

been created.

You can modify the Linked-Article's properties by right clicking the object. The Linked-Article Properties screen will appear, allowing you to modify the default behavior of the object.

Tips for Efficient Linked-Articles

1. Avoid using variables if possible. Changes to the contents of a variable may trigger a reformat, which can be time consuming with large files. Also, files that do not contain variables can be optimized by VisualNEO for Windows and will load much faster after your publication is compiled.

2. Break up extremely large text files into several smaller ones. This will speed up editing and loading of your publication.

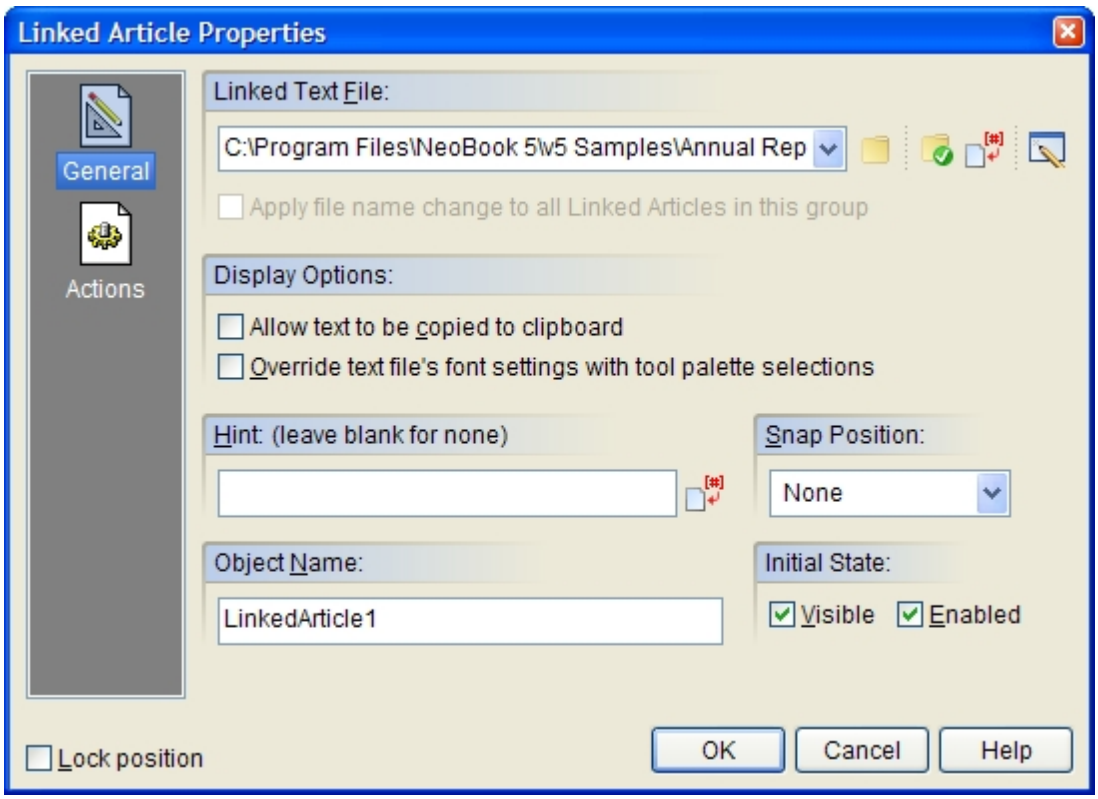
3. Avoid using excessive formatting. Only basic formatting should be used such as indents, bold, underline, italics, bullets, etc.

4. Don't use tables. The Linked-Article can't flow text that contains tables. If you need a table, insert it as a separate element.

The **Linked-Article Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

The **Text File** field contains the name of the file displayed in the Article object. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and the changes will appear in your publication. Only during the Compile stage will the file be bound inside the publication. To the right of the field are four small buttons: click the  button to select a different file or click the  button to open the current file in VisualNEO for Windows's Text Editor. Experienced authors can use the  button to perform advanced file options or the  button to replace the file name with a variable.



Enable the **Apply file name changes to all Linked-Articles in this group** option to update the file name property for all of the Linked-Articles that belong to this group.

The **Allow text to be copied to the clipboard** option provides readers with the ability to select and copy portions of the document to the Windows Clipboard. If enabled, the mouse pointer can be used to highlight portions of text. Text can be copied by clicking the right mouse button or pressing Ctrl + C. Turn off this option to prevent readers from copying portions of your text without your consent. The items in the popup menu that appear when you click the right mouse button can be translated from the Language page of the App Properties screen.

Most Rich Text files contain specific font selections as part of the document's formatting. You can replace these internal font choices with your own by enabling the **Override text file's font settings with Tool Pallet selections** option. Once enabled, you can use VisualNEO for Windows's Style Palette to select a new font for the text file.

Actions

Linked-Articles support the following Action Events: **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. You can define clickable **Hyperlinks** within the Article text using VisualNEO for Windows's Text Editor. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Picture Tool

Use the Picture Tool to incorporate graphics and illustrations into your publication. Just about any paint or drawing program can be used to create images that can be imported into VisualNEO for Windows. Compatible image formats include BMP, JPEG, GIF, PCX, PNG, TIFF, ICO and WMF.

Note: *If you want to create your own image files, there are a variety of programs available*

that are designed for this purpose. (One such program is PixelNEO® for Windows, from the makers of VisualNEO for Windows. A trial copy of PixelNEO can be found on your VisualNEO for Windows CD.)

To import a Picture, use the mouse to draw a rectangle where you would like the object to appear. A file selector will be displayed, allowing you to select a compatible image file. Once a file is selected, the Picture object will appear on your publication. You can modify the object's properties by right clicking the object. The Picture Properties screen will be displayed, allowing you to define the object's appearance and behavior.

The **Picture Properties** screen is divided into three sections indicated by the icon images on the left: General, Appearance and Actions. To view the settings for a section, click the corresponding icon.

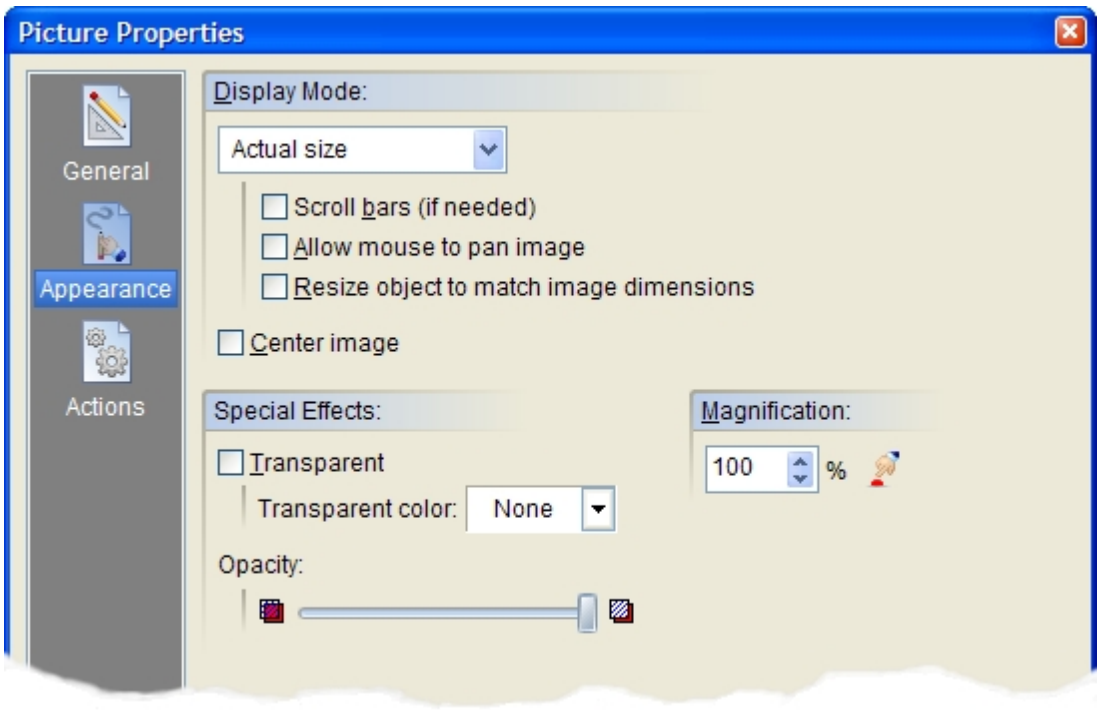
General

The **Image File** field contains the name of the file displayed by this object. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and changes will automatically appear in your publication. The file will be bound inside the publication only during the Compile stage. To the right of the field are three small buttons: click the button to select a different file; experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.

Appearance

You can control how the image will appear by selecting one of the following **Display Mode** options:

Actual Size	The image will be displayed exactly as it was drawn. If the image is larger than the space provided, you can add **Scroll bars** to the Picture object. When the **Allow mouse to pan image** option is enabled, readers can scroll images by clicking and dragging the center of Picture. Enable the **Resize object to match image dimensions** option to automatically s the Picture object to the width and height of the image file. If the image is smaller than th space available, you can enable the **Center image** option to center it within the Picture object.
Stretch	The image will be stretched to fit the dimensions of the Picture object. Enable the **Mainta aspect ratio** option to restrict the stretching so that the original width to height proportio of the image are maintained. For best results, enable the **Use high quality resampling** op to scale the image using interpolation and averaging. This process can be slightly slower th the normal stretch method, which simply expands the picture by enlarging some of the ima pixels.
Auto	The image will be displayed actual size when it will fit within the bounds of the Picture obje and stretched when it will not.
Tile	If the image is smaller than the space available, it will be duplicated (tiled) as many times needed to fill the Picture object. Both the **Horizontal** and **Vertical** options tile the image only one direction, while **Normal** option tiles the image in both directions.



The **Magnification** option (available in Actual Size mode only) can be used to enlarge or reduce the size of the image. A value of 100% displays the image in its original size. Values less than 100% make the image smaller, while values greater than 100% make the image larger.

You can designate a portion of the image as transparent by clicking the small arrow button next to the **Transparent color** box. VisualNEO for Windows will prompt you to select a color to serve as the transparent portion of the image. The background will show through pixels in the image matching this color.

Move the **Opacity** slider to the left to allow a portion of the background to show through the image. The further to the left you move the slider, the more of the background will be visible. This is often called an alpha channel transparency effect. To display the image in its normal solid form, move the slider all the way to the right. The opacity option is only available in 16 million color modes.

Note: *In order for the page background to show through the transparent portions of the image, you may also need to set the Picture object's Fill Pattern to Hollow using the controls on the Style Palette.*

Actions

Pictures support the following Action Events: **Click, Right Click**, **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Note: *When the "Allow mouse to pan image" option is enabled for a Picture object that contains a Click Action, the pan event will be handled before the Click Action is executed.*

Polygon / Hotspot Tool

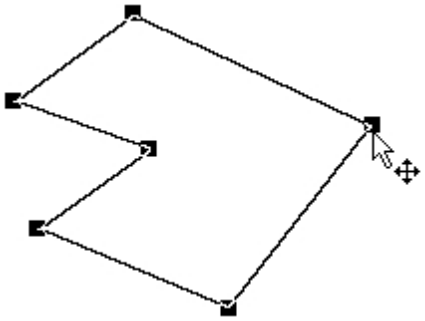
 The Polygon / Hotspot Tool can be used to create many different types of free-form shapes. The Polygon also has the ability to display a background image, and can be

configured to shape itself to match the contents of an image. A Polygon can be created manually using the mouse or automatically by importing an image file.

To **manually construct a Polygon using the mouse**, first select the Polygon/Hotspot icon from the Tool Palette. Move the mouse cursor to a point in the publication workspace where you want your Polygon to start. Hold down the left mouse button and drag to draw a line for the first side. When the line reaches the desired length and position, release the mouse button. Next, move the mouse to the point where the second side will terminate. A rubberband line will follow the cursor. When the second side has been correctly positioned, click once with the left mouse button to set the line. Repeat this step to construct the rest of your shape. (A Polygon must have at least three sides.)

To complete the Polygon, click on the point at which you started. You also can complete the Polygon by clicking the right mouse button anywhere on the page and VisualNEO for Windows will connect the last line to the first automatically.

After defining the initial shape, the Polygon will appear on the screen. When the Polygon object is selected, small boxlike handles will be added to the endpoint of each line. At this point, you can make changes to the polygon's shape.



To reposition one of the endpoints, place the mouse cursor over the endpoint's handle. (The cursor will include a directional arrow ✣ when it's in the correct position.) Hold down the left mouse button and drag the handle. As you move, the shape will stretch or shrink. When you are satisfied with the shape, release the mouse button.

Should you need to add another endpoint, position the mouse cursor over the point on the outline where you would like a new handle and click the left mouse button. (The cursor will include a plus sign + when it passes over a location where a new handle can be added.)

To delete a handle, position the mouse cursor over the unwanted handle. (The cursor will include a directional arrow ✣ when it is in the correct position.) Hold down the Ctrl key, click the left mouse button and the handle will disappear.

To **create a Polygon using an image file**, first select the Polygon/Hotspot icon from the Tool Palette. Move the mouse cursor to a point in the publication workspace where you want your Polygon to appear. Click and release the left mouse button once. A Windows File Selector will appear, allowing you to select an image file. Once you select a file, VisualNEO for Windows will attempt to convert the image into a Polygon shape.

When converting an image into a Polygon, VisualNEO for Windows expects the image to conform to some specific guidelines. During the conversion, VisualNEO for Windows uses the image's transparent color as a guide to determine the Polygon's outline. By default, VisualNEO for Windows assumes that the transparent color matches the pixel in the image's lower left corner. (If needed, you can specify a different transparent color from the Polygon/Hotspot Properties screen.)

When creating images for use with the Polygon object, it should be clear which portions of the image are to be transparent. In the example below, black is the transparent color for the original image on the left. When imported into VisualNEO for Windows, the black portion of

the image is trimmed off resulting in the clean Polygon shape below:

Original Image

Polygon

Once created, you can modify the Polygon's properties by right clicking on the object. The Polygon/Hotspot Properties screen will be displayed, allowing you to modify the object's appearance and behavior.

The **Polygon/Hotspot Properties** screen is divided into three sections indicated by the icon images on the left: General, Drag & Drop and Actions. To view the settings for a section, click the corresponding icon.

General

The **Background Image** field contains the name of the file displayed inside the Polygon. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and changes will appear in your publication. The file will be bound inside the publication only during the Compile stage. To the right of the field are three small buttons: click the button to select a different file. Experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.

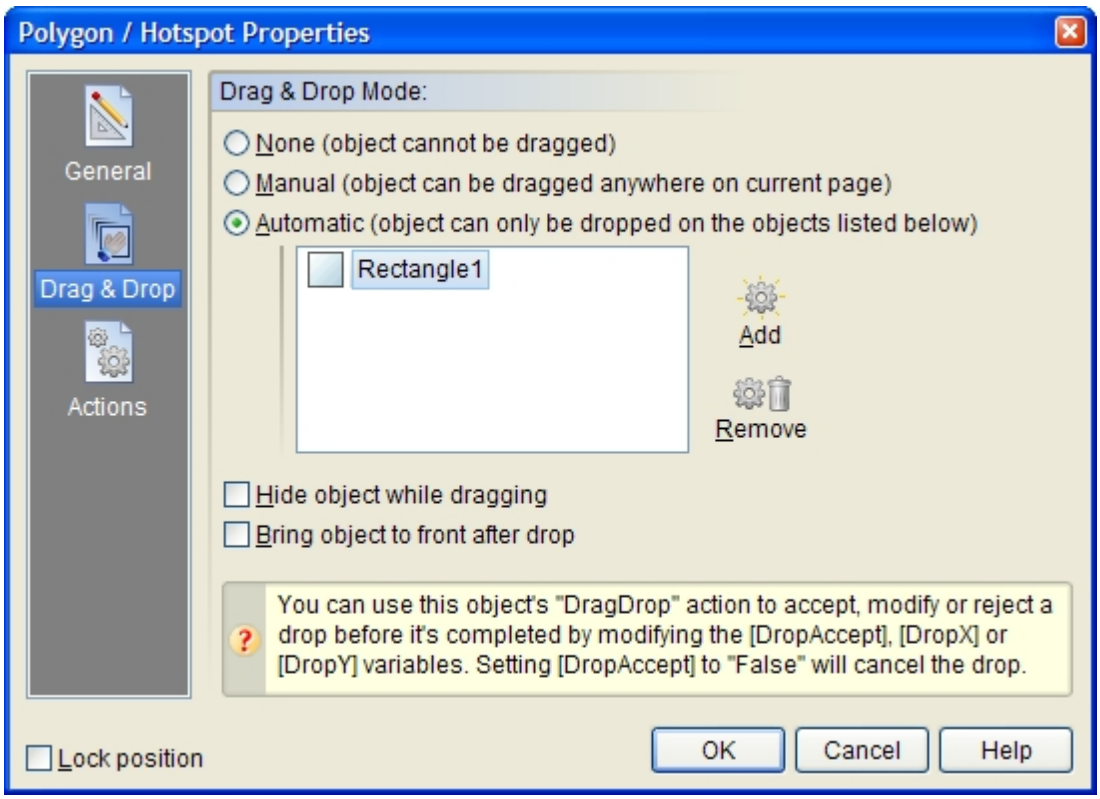
You can designate a portion of the image as transparent by clicking the small arrow button next to the **Transparent color** box. VisualNEO for Windows will prompt you to select a color to serve as the transparent portion of the image. The background will show through pixels in the image matching this color.

If the **Use transparent portion of image to define polygon** option is enabled, the transparent color will be used to determine the Polygon's shape as described in the previous section. When this option is disabled, the Polygon's shape must be defined manually.

Move the **Opacity** slider to the left to allow a portion of the background to show through the image. The further to the left you move the slider, the more background will be visible. This is often called an alpha channel transparency effect. To display the image in its normal solid form, move the slider all the way to the right. The opacity option is only available in 16 million color modes.

Drag & Drop

The Polygon object can be configured to support drag and drop functions, allowing readers of your publication to move the object using the mouse. Drag and drop modes include **None**, **Manual** and **Automatic**.



When None is selected, the object cannot be dragged. Manual allows the object to be dragged anywhere on the page. Automatic allows you to specify other objects to serve as drop targets. When using automatic mode, VisualNEO for Windows will only allow the dragged Polygon to be dropped on top of objects that appear in the drop target list. You can modify the behavior of both of these modes using the "Drag Drop" Action described below.

During drag and drop operations, VisualNEO for Windows doesn't actually move the object. Instead, a semitransparent ghost image representing the object follows the reader's mouse cursor around the screen. The original object remains visible during this process, unless you enable the **Hide object while dragging** option.

Enabling the **Bring object to front after drop** option will insure that the dragged object doesn't end up underneath the object it's dropped on.

Actions

Polygons support the following Action Events: **Click, Right Click**, **Mouse Enter**, **Mouse Exit** and **Drag Drop**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control.

If you've enabled either the Manual or Automatic **Drag and Drop** Modes above, you can use the special "Drag Drop" Action to control exactly what happens at the end of the drag and

drop operation. VisualNEO for Windows triggers the Drag Drop Action when the object is dropped. If you do nothing, VisualNEO for Windows will move the object to the new location provided the specified Drag Mode criteria has been met. However, just before the Drag Drop Action executes, VisualNEO for Windows will initialize the following special variables which can be used to influence how the drop operation is completed:

[DropAccept]	Setting this variable to "False" will cancel the drop operation. For example: `SetVar "[DropAccept]" "False"` Also, if you specified a drop target for this operation (Automatic mode), you can override that and allow the object to be dropped anywhere by setting this variable to "True". (See the SetVar Action for information on how to set variables.)
[DropX], **[DropY]**	This is the proposed new location for the dropped object. You can change these coordinates to affect where the object is placed.
[DropTarget]	If the drop occurred on top of one of your specified drop targets, this variable will contain that object's name.

Note: *For basic drag and drop operations, you do not need to worry about these variables or place any code in the Drag Drop Action.*

See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Rectangle Tool

 Use this tool to create solid or hollow rectangular or square shapes. Position the cursor over your publication where you want the top left corner of your rectangle to begin. Hold down the left mouse button and drag the cursor to where you would like the opposite corner. When your rectangle reaches the desired size and shape, release the mouse button. Set the attributes for the outline and interior colors using the Style Palette.

Holding down the CTRL key while drawing the rectangle will create a rectangle with rounded corners. You can draw perfect squares by enabling the Snap to Grid feature or by holding down the Shift key while drawing or resizing the object.

Rectangle objects are also used by some Plug-ins as place holders for special controls.

Ellipse Tool

 Use the Ellipse Tool to create circle or oval shapes. Position the cursor over your publication where you want the top left corner of your ellipse to begin. Hold down the left mouse button and drag the cursor to where you would like the opposite corner. When your ellipse reaches the desired size and shape, release the mouse button. Set the attributes for the outline and interior colors using the Style Palette.

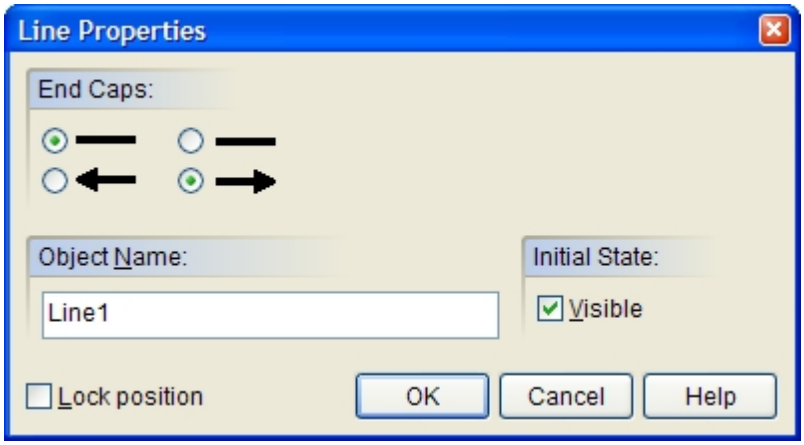
You can draw perfect circles by enabling the Snap to Grid feature or by holding down the Shift key while drawing or resizing the object.

Line Tool

 Use the Line Tool to create straight or diagonal lines with optional arrowheads on both ends. To draw a line, click and hold down the left or right mouse button where you want the line to begin. Drag the cursor to where you want the line to end and release the mouse button. A line connecting these two points will be drawn. Set the attributes for the line's color, width and style using the Style Palette.

You can draw perfectly horizontal or vertical lines by holding down the Shift key while drawing or resizing the line.

To add an arrowhead, right click the line to display the **Line Properties** screen and add arrowhead to one or both ends. The size of the arrowhead depends on the width of the line.

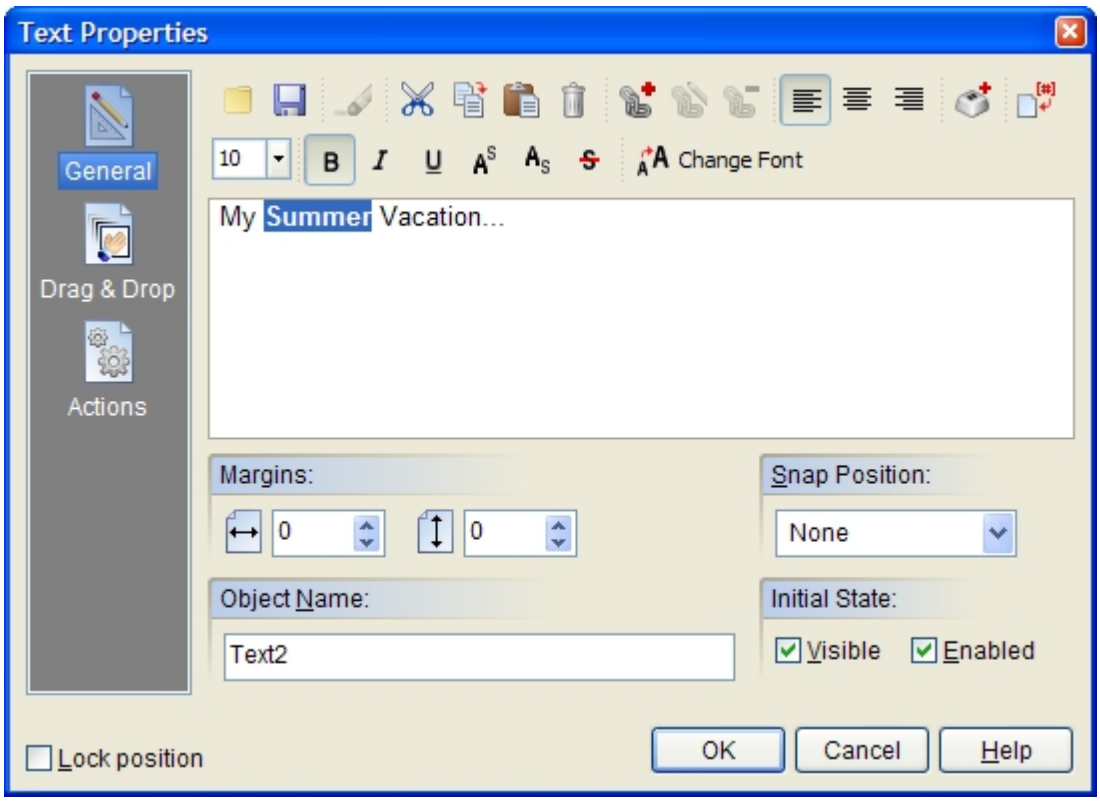


Simple Text Tool

 Use this tool to insert short blocks of text, such as a caption or title, into your publication. For larger blocks of text or to import a file created with a word processor, use the Article or Linked-Article Tools described earlier in this section.

To add text to your publication, use the mouse to draw a rectangle where you would like the text to appear. The **Text Properties** screen will be displayed, allowing you to enter and format your text.

The Text Properties screen is divided into three sections indicated by the icon images on the left: General, Drag & Drop and Actions. To view the settings for a section, click the corresponding icon.



General

The large space in the middle of this screen is the text editor where you will enter and format your text. The editor functions very much like a miniature word processor – just type the desired text into the editor window. Above the editor are several buttons that will assist you in formatting and editing your text. These buttons are described below:

- Insert text from a file.
- Export text to a file.
- Undo the last edit.
- Cut the selected text to the clipboard.
- Copy the selected text to the clipboard.
- Paste text from the clipboard into the editor.
- Delete the selected text.
- Create a Hyperlink.
- Edit the Hyperlink under the cursor.
- Delete the Hyperlink under the cursor.
- Align the text to the left.
- Align the text to the center.
- Align the text to the right.
- Insert an extended ASCII character into the text. (The characters available depend on the font
- Insert a VisualNEO for Windows variable into the text.
- Select a font size for the selected text.

B	Bold the selected text.
I	Italicize the selected text.
U	Underline the selected text.
A^S	Display the selected text as Superscript.
A_S	Display the selected text as Subscript.
~~S~~	Display the selected text as Strikethrough.
AA	Select a font to be used for the entire text block.

Hyperlinks containing hidden Action Commands can be inserted into your text. At runtime, Hyperlinks appear as underlined text, which readers can click to execute the associated Actions. Hyperlinks are often used to navigate between pages or display information related to the underlined text.

To define a Hyperlink, highlight a word or phrase and click the  button. VisualNEO for Windows's Action Editor will appear, allowing you to specify what will happen when the reader clicks this Hyperlink.

A **Variable** is an area of the computer's memory that you can use to temporarily store information (text or numbers) relevant to your publication. For example, you might request that your reader enter his or her name at the start of a publication. That variable could then be used to personalize the publication. For example, you might enter something like this into a Text Object:

```
Suddenly, [Name] saw a bright flash in the sky. "Could it be a
spaceship," [Name] asked? No, it's probably just an airplane.
```

Notice the word Name surrounded by square brackets - that's a variable. In VisualNEO for Windows parlance, the names of variables are always surrounded by [brackets]. That's how VisualNEO for Windows knows that you're referring to a variable called [Name] and not the word "Name." At runtime, VisualNEO for Windows will automatically update your paragraph with the contents of any variables you specify. For example, the same paragraph as the reader might see it:

```
Suddenly, Billy Johnson saw a bright flash in the sky. "Could it be a
spaceship," Billy Johnson asked? No, it's probably just an airplane.
```

See Understanding Actions and Variables for more information about using variables.

Adjust the Left/Right and Top/Bottom **Margins** to add space between your text and the border of the Text object. A margin of zero will butt the text up against the edge of the object.

Drag & Drop

Simple Text objects can be configured to support drag and drop functions, allowing readers of your publication to move the object using the mouse. Drag and drop modes include **None**, **Manual** and **Automatic**. When None is selected, the object cannot be dragged. Manual allows the object to be dragged anywhere on the page. Automatic allows you to specify other objects to serve as drop targets. When using automatic mode, VisualNEO for Windows only will allow the dragged Simple text to be dropped on top of objects that appear in the drop target list. You can modify the behavior of both of these modes using the "Drag Drop" Action. Please refer to the Polygon/Hotspot Tool topic for a more detailed discussion of Drag and Drop.

Actions

In addition to **Hyperlinks** embedded within the text, Simple Text objects also support the following Action Events: **Left Click**, **Right Click**, **Mouse Enter**, **Mouse Exit** and **Drag Drop**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Text Entry Field Tool

 VisualNEO for Windows's Text Entry Field allows you to provide your readers with a place to enter text or numeric information. Use Text Entry Fields to create fill in the blank forms, record answers to essay questions or collect just about any other type of information. The information may be stored in a variable, used in calculations or written to a file.

To create a Text Entry Field, use the mouse to draw a rectangle where you want the field to appear. The Text Entry Properties screen will be displayed, allowing you to define the field's appearance and behavior.

The **Text Entry Properties** screen is divided into three sections indicated by the icon images on the left: General, Style and Actions. To view the settings for a section, click the corresponding icon.

General

In the **Text** field, enter any text you want to appear in the Text Entry object when your publication first starts. This can be used to represent a default choice, where the user can accept the text already in the object and move on. Most of the time this field will be left blank, since the main purpose of a Text Entry object is to provide a space for readers to enter information of their own.

In order to keep track of the contents of a Text Entry object while your publication is running, you will need to assign the object a unique variable name. VisualNEO for Windows will automatically assign a variable name that matches the Object Name, but you may change this if you like by modifying the **Variable (to store Text Entry contents)** field. At runtime, the variable will contain whatever has been typed into the field. You can modify the contents of the Text Entry object by manipulating the variable using a simple Action Command. For example:

```
SetVar "[TextEntry1]" "Have a nice day!"
```

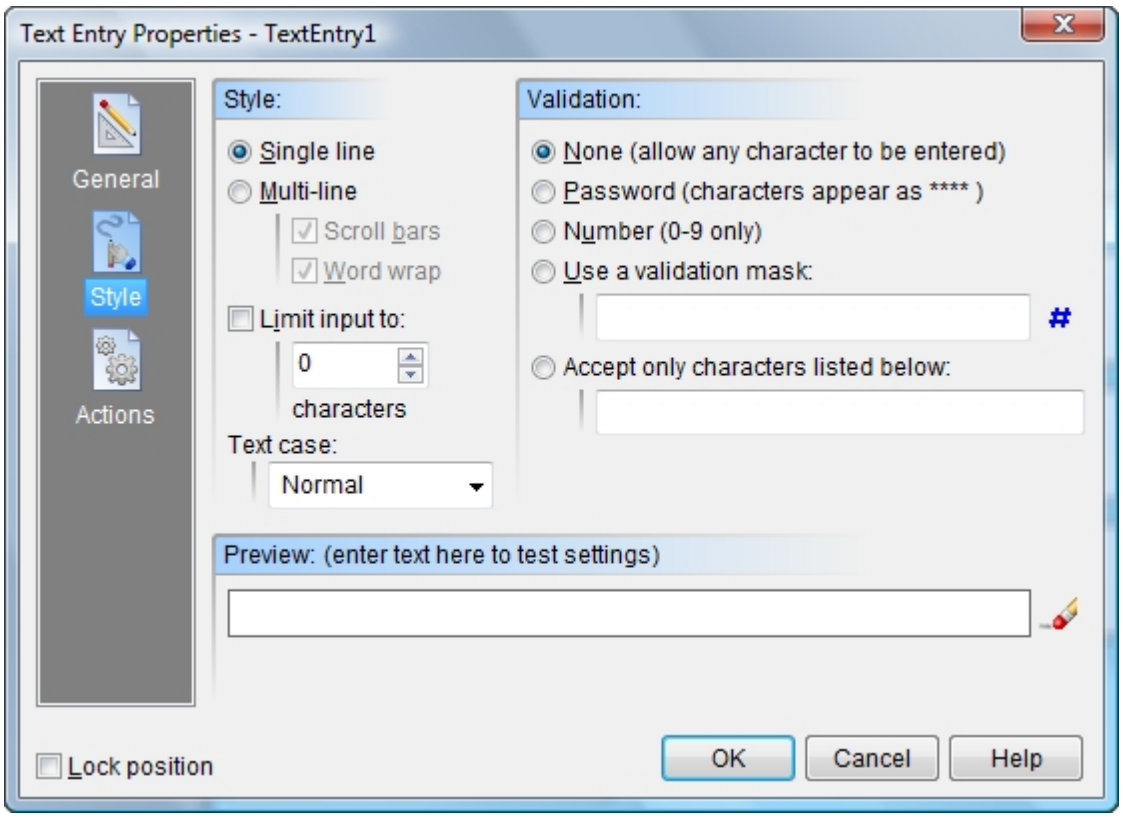
Similarly, you can clear the contents of the Text Entry object like this:

```
SetVar "[TextEntry1]" ""
```

You may also set the alignment for the Text Entry using one of the three **Alignment** buttons: Left Aligned, Centered or Right Aligned. The alignment affects how the text typed into the field is displayed.

Style

The options here allow you to control how much and what type of information may be typed into the Text Entry object.



You can limit the Text Entry to a **Single line** or allow a large block of text to be entered by selecting the **Multi-line** option. The **Word wrap** and **Scroll bar** features also may be enabled for Multi-line fields. You can limit how much text the reader may enter by typing a number into the **Limit input to** box. Specify 0 (zero) to set the maximum number of characters to 64,000 which is the Windows default.

You can control exactly what type of text may be entered by selecting a **Validation** method. Select **None** to allow any character (letters, numbers, punctuation, spaces, etc.) to be entered. Use the **Password** option to display only asterisks "*" in place of whatever text is entered. This is useful if the field is to serve as a place to enter a secret code or password. Select Number to limit the text entered to numbers (0...9). The **Accept only characters listed below** option allows you limit input to a specific list of acceptable letters, numbers, etc. The list is case sensitive, so if you want both upper and lower case letters you must include both. You can select **Use a validation mask** to restrict what the reader can enter to specific characters and formats. Any invalid characters entered by the reader are rejected. You can select some predefined masks by clicking the # button, or compose your own using codes from the table below:

L	Requires that a letter (A-Z or a-z) be entered.
l	Permits letters (A-Z or a-z) to be entered, but does not require it.
A	Requires that a letter or number (A-Z, a-z or 0-9) be entered.
a	Permits letters or numbers (A-Z, a-z or 0-9) to be entered, but does not require it.
C	Requires that a character be entered, but does not place any limitation on what that character mu
c	Permits any character to be entered, but does not require it.
0	Requires that a number (0-9) be entered.
9	Permits a number (0-9) to be entered, but does not require it.
#	Permits a number (0-9) or a plus (+) or minus (-) sign to be entered.
:	Used to separate hours, minutes and seconds. If the computer's regionalsettings specify a differe character will be used instead.
/	Used to separate months, days and years. If the computer's regionalsettings specify a different c

	will be used instead.
_	Used to represent spaces. When the reader types in the field, the cursor automatically skips the _
>	All characters following this symbol will be in uppercase until then end of the mask or a < characte
<	All characters following this symbol will be in lowercase until then end of the mask or a > character
<>	These two symbols together turn off case checking. Characters are formatted as entered by the r
\	The character that follows this symbol is treated as a literal character not a mask code. Use this if of the special characters above into your mask.

Any character present in the mask but not found in the table above will be interpreted as a literal character. Literal characters will be inserted automatically into the formatted text and the cursor will skip over them during data entry. For example a mask for a telephone number with area code would look like this:

```
(000)_000-0000
```

After a telephone number has been entered by the reader, the data stored in the Text Entry object's variable might look like this:

```
(123) 456-7890
```

The **Preview** shows you what your Text Entry object will look like and can be used to test any masks you create.

Actions

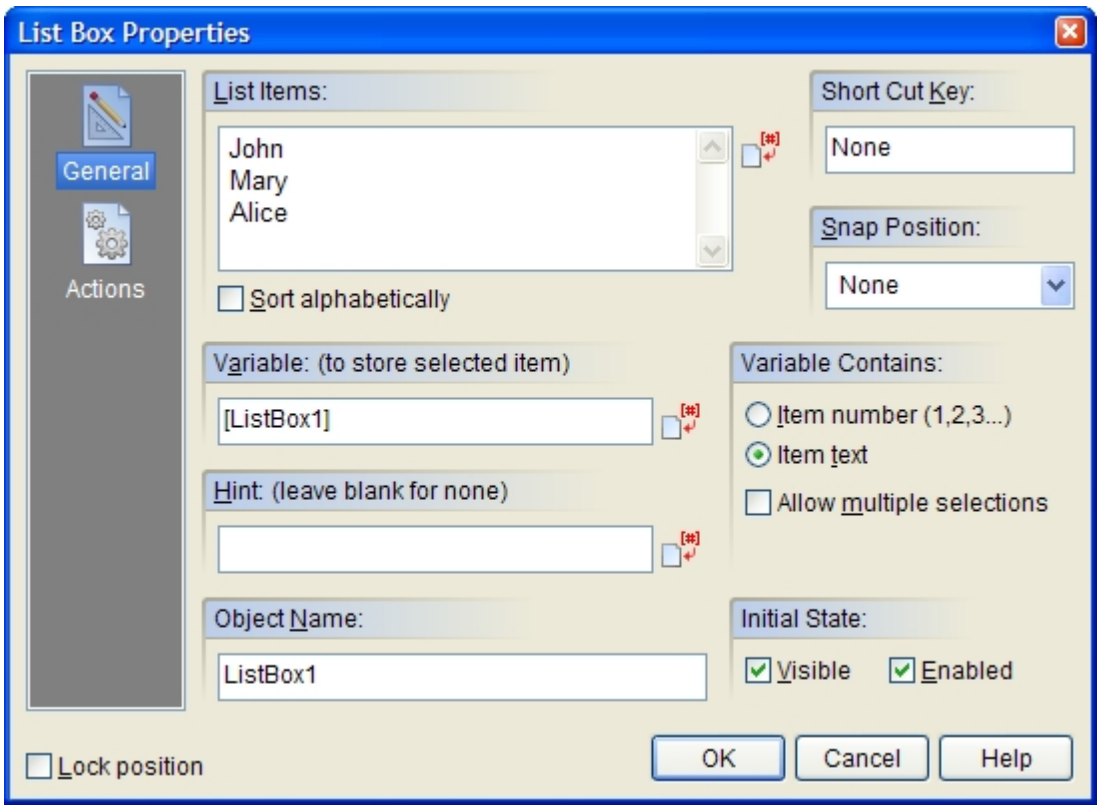
Text Entry Fields support the following Action Events: **Text Change**, **Mouse Enter, Mouse Exit, Gain Focus** and **Lose Focus**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

The **Text Change** Action is triggered each time a key is pressed while Text Entry object has the input focus. It's generally not the best place to process data, since the Action may execute many times before the reader finishes entering the requested information. The preferred method is to place a Push Button next to the field for the user to click when the data entry is complete.

List Box / Combo Box Tools

 Both the List Box and Combo Box objects allow readers to select from a list of items. These objects allow a large number of choices to be displayed in a relatively small space. List and Combo Boxes differ only in how they appear on screen. List Boxes display items in a rectangular window with a scroll bar (if needed). Combo Boxes are more compact - displaying only the selected item along with a small button that can be clicked to display the entire list. Other than that, both objects are created and used in the same manner.

To create a List or Combo Box, use the mouse to draw a rectangle where you would like the object to appear. The List Box or Combo Box Properties screen will be displayed allowing you to define the object's appearance and behavior.



Both the **List and Combo Box Properties** screens are divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

Create items to be displayed by entering them into the **List Items** field. As you type your list, use the Enter key to separate each item. Each item should appear on a line of its own. Enter as many items as you wish, keeping in mind that the size of the object and the font size determine how the list is presented.

Enable the **Sort alphabetically** option to organize the list into ascending order using the first letter of each item.

Enabling the **Use Windows theme** option (ComboBox Tool only) will display the object using the Windows XP/Vista style. This feature only works when your publication is viewed under Windows XP and Vista. When viewed under older versions of Windows, the ComboBox will be drawn using the normal method.

In order to keep track of the which item in the list is selected while your publication is running, you can assign the List/Combo Box a unique variable name. VisualNEO for Windows will automatically assign a variable name that matches the Object Name, but you may change this by modifying the **Variable (to store selected item)** field. At runtime, the variable can contain either the **Item number** or **Item text**. If the Item Number option is enabled, the variable will contain the line number of the selected item in the list. The first item is "1," the second item is "2," and so on. If Item Text is chosen, the variable will contain the actual text of the item selected. For example, if the second item is "Apple," and that item is selected, the variable will contain the word "Apple."

In addition, List Boxes include an extra check box under Variable Contains that you can use to specify if the object will **Allow multiple selections**. This can be useful in circumstances where you want to allow the reader to select more than one item. When this option is enabled, the variable assigned to the List Box may contain more than one item. Multiple items will be separated by carriage returns, so you will need to use an Action like StrParse to find

out which items were selected. For example:

```
StrParse "[ListBox1]" "[#13]" "[SelectedItems]" "[ItemCount]"
AlertBox "Hello" "You selected [ItemCount] items."
```

Note: *The [#13] code above is used by VisualNEO for Windows to indicate a carriage return.*

Actions

Both List and Combo Boxes support the following Action Events: **Selection Changed**, **Mouse Enter** and **Mouse Exit**. In addition, List Boxes support **Right Click** and **Double Click** actions. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Hint: *VisualNEO for Windows provides several Action Commands that can be used to add, delete, find or sort items in a List or Combo Box. This allows you to create dynamic lists that change as your publication progresses.*

Check Box Tool

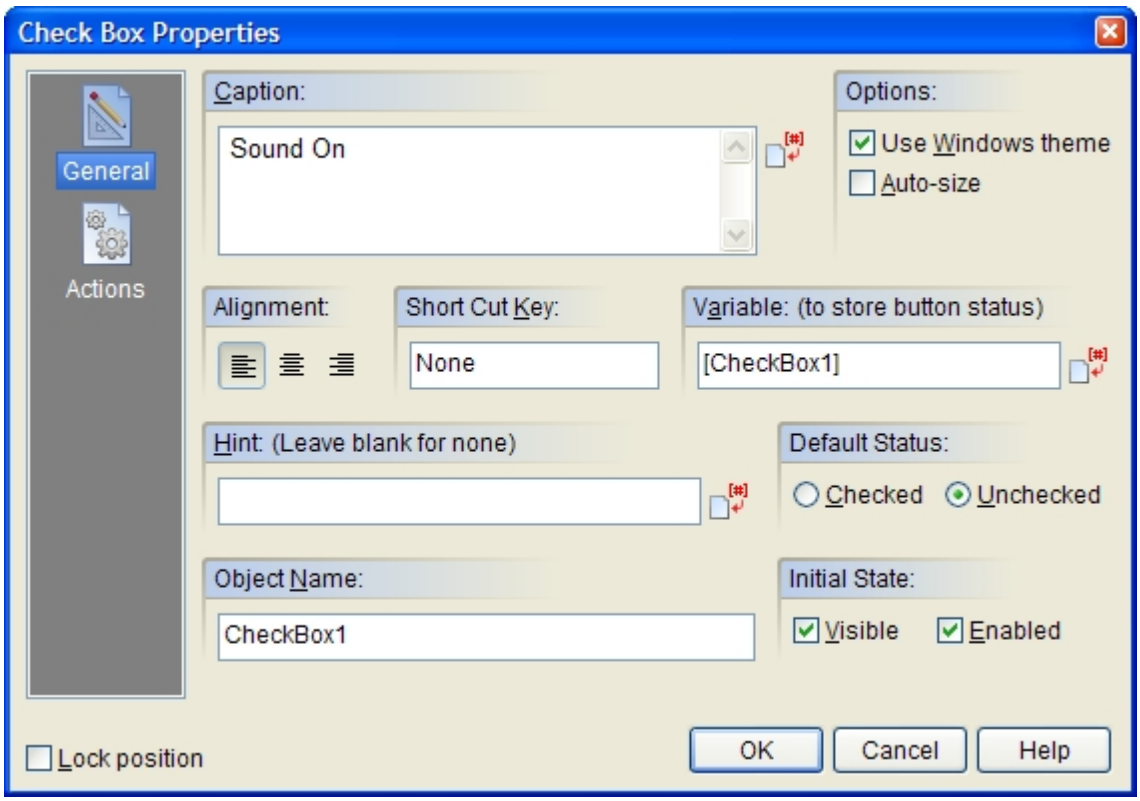
 The Check Box lets readers make a Boolean choice like yes / no, true / false. A Check Box consists of a small box and descriptive caption. A check mark appears in the box when the option is selected. The absence of a check mark indicates that the option is not selected. Check Boxes are often used on forms or configuration screens for selecting options or turning program features on and off.

To create a Check Box, use the mouse to draw a rectangle where you would like the object to appear. The Check Box Properties screen will be displayed, allowing you to define the button's appearance and behavior.

The **Check Box Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

Text entered into the **Caption** field will appear beside the Check Box. The caption should give the reader some indication of what the Check Box's purpose is.



Enabling the **Use Windows theme** option will display the Check Box using Windows XP/Vista style. This feature only works when your publication is viewed under Windows XP and Vista. When viewed under older versions of Windows, the Check Box will be drawn using the normal method.

When the **Auto-size** option is enabled, the object will automatically adjust its dimensions to match the size of the caption text.

You may also set the alignment for the Check Box using one of the three **Alignment** buttons: Left Aligned, Centered or Right Aligned. The alignment affects how the Check Box is positioned relative to its caption. For example:

In order to keep track of the status of a Check Box while your publication is running, you will need to assign the object a unique variable name. VisualNEO for Windows will automatically assign a variable name that matches the Object Name, but you may change this by modifying the **Variable (to store button status)** field. At runtime, the variable will contain the word "Checked" when the Check Box is selected; otherwise, the variable will be undefined or empty. You can use some simple Action Commands to detect if the box is checked. For example:

```
If "[CheckBox1]" "=" "Checked"
  do something...
Else
  do something else...
EndIf
```

You can select or unselect a Check Box at runtime like this:

```
SetVar "[CheckBox1]" "Checked"

SetVar "[CheckBox1]" ""
```

For more information about variables and Actions, click here.

You may specify that the Check Box start out **Checked** or **Unchecked** by selecting the appropriate Default Status option. When your publication starts, the Check Box will be initialized in this state.

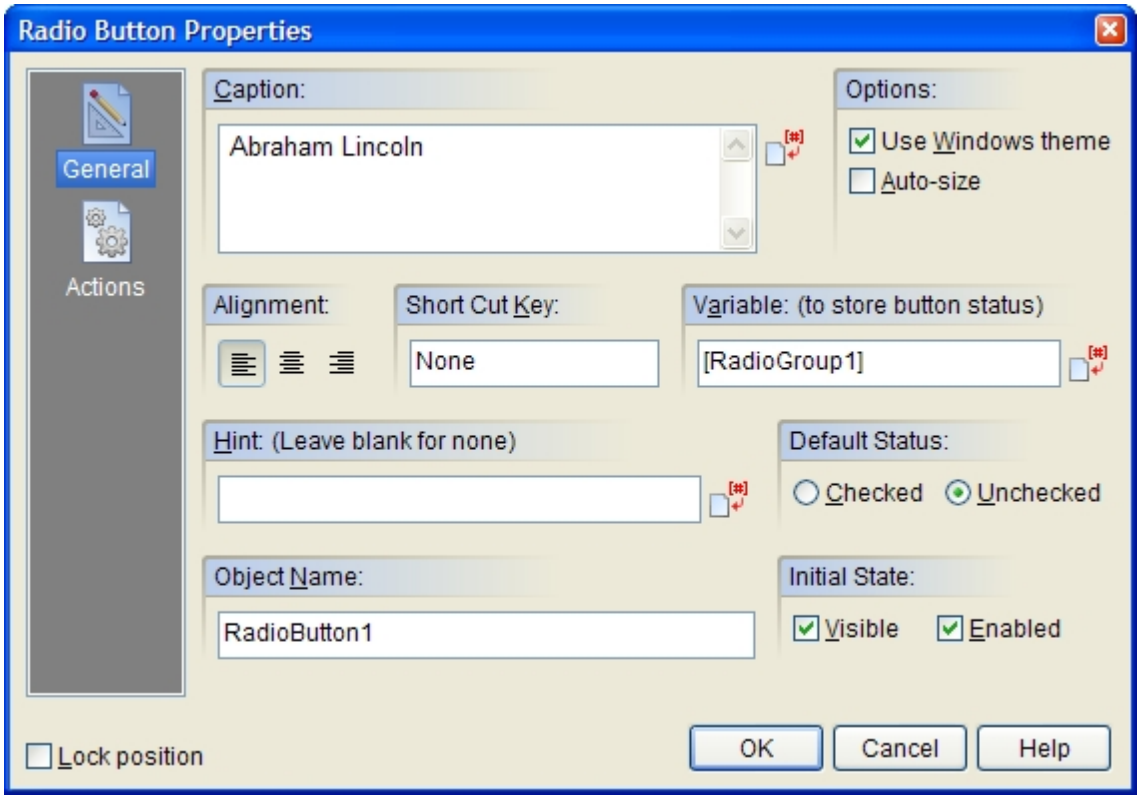
Actions

Check Boxes support the following Action Events: **Click**, **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Radio Button Tool

 Radio Buttons allow readers to select one option from a set of mutually exclusive choices. Radio Buttons always appear in groups of two or more. Only one button in the group may be selected at a time. A Radio Button consists of a small circle and descriptive caption. A dot appears in the circle of the button that is selected. The absence of a dot indicates that the button is not selected. Radio Buttons often are used on forms or tests when readers must make a single selection from multiple choices.

To create a Radio Button, use the mouse to draw a rectangle where you would like the button to appear. The Radio Button Properties screen will be displayed, allowing you to define the button's appearance and behavior.



The **Radio Button Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

Text entered into the **Caption** field will appear beside the Radio Button. The caption should give the reader some indication of the selection associated with each button.

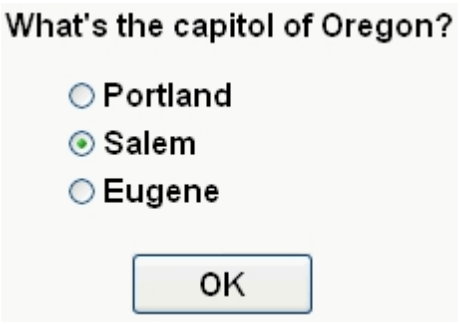
Enabling the **Use Windows theme** option will display the Radio Button using the Windows XP/Vista style. This feature only works when your publication is viewed under Windows XP and Vista. When viewed under older versions of Windows, the Radio Button will be drawn using the normal method.

When the **Auto-size** option is enabled, the object will automatically adjust its dimensions to match the size of the caption text.

You may set the alignment for the Radio Button using one of the three **Alignment** buttons: Left Aligned, Centered or Right Aligned. The alignment affects how the Radio Button is positioned relative to its caption.

In order to keep track of the status of a group of Radio Buttons while your publication is running, you will need to assign each button in the group the same variable name. However, no two groups in a publication should share the same variable. The variable name tells VisualNEO for Windows which Radio Buttons belong to a group.

VisualNEO for Windows will automatically group Radio Buttons created on the same page with a generic variable name based on the page number (such as [RadioGroup1]). If you need more than one group per page, you will need to manually change the variable name of the other groups by modifying the **Variable (to store button status)** field. At runtime, the group variable will contain the caption of selected Radio Button. You can determine which button is selected using simple Action Commands. For example:



In this example, clicking the OK button executes the following Action Commands, which provide feedback about the reader's choice:

```
If "[RadioGroup1]" "=" "Portland"
  AlertBox "Sorry" "That's not the right answer."
EndIf
If "[RadioGroup1]" "=" "Salem"
  AlertBox "Good Job" "That's correct!"
EndIf
If "[RadioGroup1]" "=" "Eugene"
  AlertBox "Sorry" "That's not the right answer."
EndIf
If "[RadioGroup1]" "=" ""
  AlertBox "Whoops" "You forgot to select an answer."
EndIf
```

You can check an individual Radio Button programmatically while your publication is running using the SetVar Action. To select the second item in the above example, insert that button's caption into the group's variable:

```
SetVar "[RadioGroup1]" "Salem"
```

Similarly, you can reset a Radio Button group so that none of the buttons are selected by clearing the group's variable like this:

```
SetVar "[RadioGroup1]" ""
```

For more information about using variables and Action commands, click here.

You may specify that a Radio Button start out **Checked** or **Unchecked** by selecting the appropriate Default Status option. When your publication starts, the Radio Button will be initialized to this state. Since each item in a Radio Button group is mutually exclusive, only one button in a group should be initially checked. For multiple choice tests, none of the buttons should start out checked, unless you want your readers to know the answers in advance.

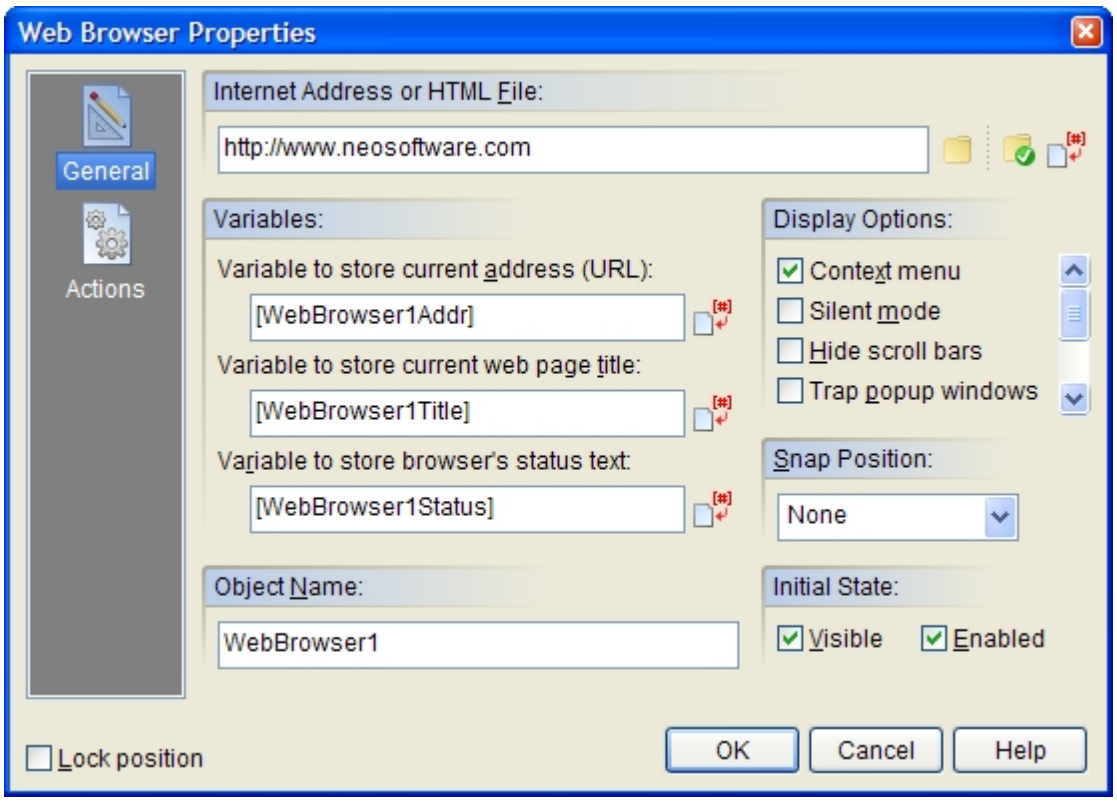
Actions

Radio Buttons support the following Action Events: **Click**, **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Web Browser Tool

 The Web Browser object allows you to display HTML documents and other Internet content inside your publication. Use the Web Browser to incorporate live Internet content, such as advertisements or a related web site, into your publication. You could even link to a secure online order system to allow readers to order products directly from your publication. It's also possible to establish two-way communication between Web Browser content and your publication.

To create a Web Browser, use the mouse to draw a rectangle where you would like the object to appear. The Web Browser Box Properties screen will be displayed, allowing you to define the object's appearance and behavior.



The **Web Browser Properties** screen is divided into two sections, indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

Enter the address of the web site or HTML document you want to display in the **Internet Address or HTML File** field. You can select a local HTML file (one located on your computer) by clicking the button to the right of this field; experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.

If you select a local file, it will remain external until the compile stage, when it will be bound inside the publication for distribution. During this process, the compiler will parse the document and include any required images. If the document includes links to other HTML files, those will be included and parsed as well.

There are three variables that can be assigned to the Web Browser to provide information about the currently displayed web site or HTML document. VisualNEO for Windows will assign variable names based on the Object Name, but you may change these names by modifying the appropriate fields. Unlike most other VisualNEO for Windows variables, these are read-only and cannot be modified to affect the state of the Web Browser. The variables are described below:

Variable to store current address (URL): This variable will be updated each time the contents of the Browser changes. Initially, this will be the same as the Internet Address specified above but will change if the reader is allowed to navigate to another web site. (To change the address at runtime, use the BrowserGoTo Action rather than modifying this variable directly.)

Variable to store current web page title: This variable will contain the title of the current HTML document displayed in the Browser. The title is specified within the HTML code by the document's author. Commercial browser's, such as Internet Explorer, Chrome and FireFox, usually display a document's title at the top of the screen. You can do that too by creating a Simple Text object containing this variable's name.

Variable to store browser's status text: This variable will be updated whenever the Browser has status information to present to the reader. Status information changes when downloading a new document or when the reader's mouse pointer passes over a hyperlink. You can share this information with readers by creating a Simple Text object containing this variable's name.

If desired, you can prevent readers from accessing the Browser's **Context menu**, which normally appears if you right click the Browser window. A standard browser feature, the Context Menu provides readers with navigational commands. If you want to control which browser commands your readers can access, turn this option off.

You may enable the **Silent mode** option to prevent the Browser from generating any messages or dialog boxes on its own. For example, when Silent Mode is disabled, the BrowserPrint Action will display the Windows Print Screen before printing. When Silent Mode is on, the document will print immediately, without showing the Print Screen.

Select **Hide scroll bars** if you want the Browser window to be displayed without scroll bars. This can be handy if you're displaying some type of special content that fits exactly within the boundaries of the Browser. (Internet Explorer's natural inclination is to add a scroll bar whether one is needed or not.)

When the **Trap popup windows** option is enabled, VisualNEO for Windows will attempt to capture any requests to open new Browser windows and redirect the content to a window that contains a VisualNEO for Windows Web Browser object. When this option is disabled, any requests to display additional windows will be handled by Internet Explorer.

Enabling the **Show tool bar** option instructs VisualNEO for Windows to add a simple navigation bar to the top of the Browser. The navigation bar includes buttons for moving Backward and Forward as well as Stop and Refresh. The images used for the tool bar can be found in the "VisualNEO for Windows 5\Buttons\Resources" folder. If desired, you can change the appearance of the navigation buttons by editing these files using your favorite paint program. You can change the content of these files, but do not change their height or width.

Select the **Show status bar** option to add a simple status bar to the bottom of the Browser. The status bar will display the Browser's current status text. A small progress bar will also appear whenever the Browser is waiting for files to download.

When the **Enhanced Security** option is enabled, VisualNEO for Windows will not allow potentially dangerous Actions embedded within HTML hyperlinks to be executed. (See Embedding VisualNEO for Windows Actions Inside an HTML Document.) Prohibited Actions include: Run, ExecuteAddOn, FileCopy, FileDelLine, FileErase, FileInsLine, FileRead, FileWrite, SendKeys, SaveVariables, CreateFolder, RemoveFolder, RegistryRead, RegistryWrite, SendMail, ExtractFile, Suspend, RunVisualNEO for Windows, ClickMouse and all plug-in based Actions. Disable the Enhanced Security option if you wish to allow the above Actions to be executed. **However, if this Web Browser object will have unrestricted access to the Internet, it is highly recommended that you leave the Enhanced Security option enabled.**

Actions

The Web Browser supports the following Action Events: **Before Navigate**, **Download Begin**, **Download Complete** and **Navigate Complete**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

The **Before Navigate** Action can be used to prevent readers from leaving the current web site or viewing sites that you do not want them to see. At runtime, you can prevent or redirect navigation by changing the contents of the variable containing the current URL

address from within the Before Navigate Action. For example, the following code placed in the Before Navigate Action will prevent the Browser from displaying any pages that are not part of Microsoft's web site:

```
SearchStr "microsoft.com" "[WebBrowser1Addr]" "[Found]"
If "[Found]" "=" "0"
  SetVar "[WebBrowser1Addr]" ""
  AlertBox "Sorry" "That's not an authorized web address."
EndIf
```

The Before Navigate Action could also be used to keep a log of which web sites were visited. For example:
FileWrite "activity.log" "Append" "[Time]: [WebBrowser1Addr]"

The [Time] variable above adds the current time to the activity.log file along with the address of the site visited.

The **Download Begin and Download Complete** Actions are triggered whenever the Browser downloads content over the Internet. These Actions can be used to display a message or play an animation indicating to the reader that the computer is busy.

The **Navigate Complete** Action is executed when the entire HTML document or web page and all of its elements (images, links, etc.) have finished downloading.

Embedding VisualNEO for Windows Actions Inside an HTML Document

Special hyperlinks can be added to HTML documents to execute VisualNEO for Windows Action commands. If you're using an HTML editor, such as Microsoft's FrontPage®, you can enter VisualNEO for Windows Actions instead of a traditional URL address when creating hyperlinks. Just precede the command with "VisualNEO for Windows:". For example:

```
neobook:GotoPage "Main"
```

Adding "VisualNEO for Windows:" to the beginning of the hyperlink tells the Web Browser that VisualNEO for Windows should process the command instead of Internet Explorer. This only works when the HTML document is displayed inside VisualNEO for Windows's Web Browser object. Stand-alone browsers like Internet Explorer, Chrome or FireFox cannot execute VisualNEO for Windows Actions.

If you prefer to edit your HTML documents manually, the code for a VisualNEO for Windows hyperlink would look like this:

```
Click this <a href="neobook:GotoPage %22Main%22">link</a> to jump to another
VisualNEO for Windows page.
```

Notice the special codes (%22) where you would expect to see quotation marks surrounding the action's parameter. Since quotation marks have special meaning in HTML, we can't use them here. Instead we place %22 wherever we would normally place a quotation mark. This is not necessary when using an HTML editor, as the editor will replace any special characters automatically.

If you want to add more than one Action to a hyperlink, you can separate them with another special code - %0D (%+zero+D). This is the hexadecimal code for a carriage return. For example:

```
Click this <a href="neobook:AlertBox %22Hello%22 %22About to switch pages.%22%
0DGotoPage %22Main%22">link</a> to jump to another VisualNEO for Windows page.
```

If your HTML editor doesn't allow you to enter the %0D code, you can use the ¶ character (ASCII #0182) instead. For example:

```
Click this <a href="neobook:AlertBox %22Hello%22 %22About to switch pages.%
22¶GotoPage %22Main%22">link</a> to jump to another VisualNEO for Windows
page.
```

In addition to the default "neobook:" prefix, you may optionally use the publication's title instead. However, if your title includes spaces, they must be removed. For example, if your publication's title is "My Super Cool Program", then the hyperlink should look like this:

```
mysupercoolprogram:GotoPage "Main"
```

For security reasons, an Action that could be potentially dangerous, such as FileWrite, FileErase, etc. cannot be executed from the Web Browser object unless the Enhanced Security option above is disabled.

Note: *VisualNEO for Windows's compiler will not detect external files used by Actions embedded inside HTML documents. If you use VisualNEO for Windows Actions here that reference text or media files, you will need to link those files elsewhere in your publication or make other arrangements to insure that all needed files are distributed with your compiled exe.*

Passing Information Between the Browser and VisualNEO for Windows

The Web Browser object includes support for three special external methods that can be used by JScript or VBScript programmers to interact with VisualNEO for Windows. These external methods (called nbGetVar, nbSetVar and nbExecAction) can be combined with VisualNEO for Windows's BrowserGetElement, BrowserSetElement and BrowserExecScript actions to establish two-way communication between the Web Browser's content and your publication. This provides VisualNEO for Windows publications with direct access to many powerful JavaScript and VBScript features. With these tools, you can essentially use JScript/VBScript to augment VisualNEO for Windows's own scripting language.

Note: *Some experience with HTML and JScript or VBScript are required to use these features.*

External methods supported by VisualNEO for Windows's Web Browser object:

nbSetVar

Purpose:	This method assigns a string value to a VisualNEO for Windows variable.
Syntax:	window.external.nbSetVar(variable name, value)
	variable name The name of a VisualNEO for Windows variable. The variable name should be surrounded by square brackets.
	value The value to assign to the variable. Must be of type string.
Example:	The following HTML JScript copies the contents of two Text Boxes called "FirstName" and "LastName" to the VisualNEO for Windows variables [FirstName] and [LastName]:

```
<script language="JScript">
function SetVars() {
  window.external.nbSetVar( '[FirstName]', mainform.FirstName.value );
  window.external.nbSetVar( '[LastName]', mainform.LastName.value );
  }
</script>
```

nbGetVar

Purpose:	This method returns a string containing the contents of a VisualNEO for Windows variable.
Syntax:	value = window.external.nbGetVar(variable name)

variable name
The name of the VisualNEO for Windows variable to retrieve. The variable name should be surrounded by square brackets.

value
The contents of the VisualNEO for Windows variable in string format.

Example: The following HTML JScript copies the contents of VisualNEO for Windows's [FirstName] and [LastName] variables to two Text Boxes called "FirstName" and "LastName":

```
<script language="JScript">
function GetVars() {
  mainform.FirstName.value = window.external.nbGetVar( '[FirstName]' );
  mainform.LastName.value = window.external.nbGetVar( '[LastName]' );
  }
</script>
```

nbExecAction

Purpose:	Use this method to execute VisualNEO for Windows actions.
Syntax:	window.external.nbExecAction(action script)

action script
The VisualNEO for Windows action to execute. Multiple actions may specified by separating them with a carriage return.

Example: This example executes VisualNEO for Windows's AlertBox action:

```
<script language="JScript">
function DoExec() {
  window.external.nbExecAction( 'AlertBox "Hello" "Hello from the Web Browser!"' );
}
</script>
```

Note: *These features require the user's Internet Security/Active Scripting option to be enabled.*

Timer Tool

 Use the Timer object to trigger an Action Command after a specific amount of time has elapsed. You can use Timer objects to create simple slide shows and timed tests.

To create a Timer object, use the mouse to draw a rectangle where you would like the object to appear. (The placement of the Timer object is not critical, since it will not be visible when running the publication.) The Timer Properties screen will be displayed, allowing you to define the object's behavior.

The **Timer Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

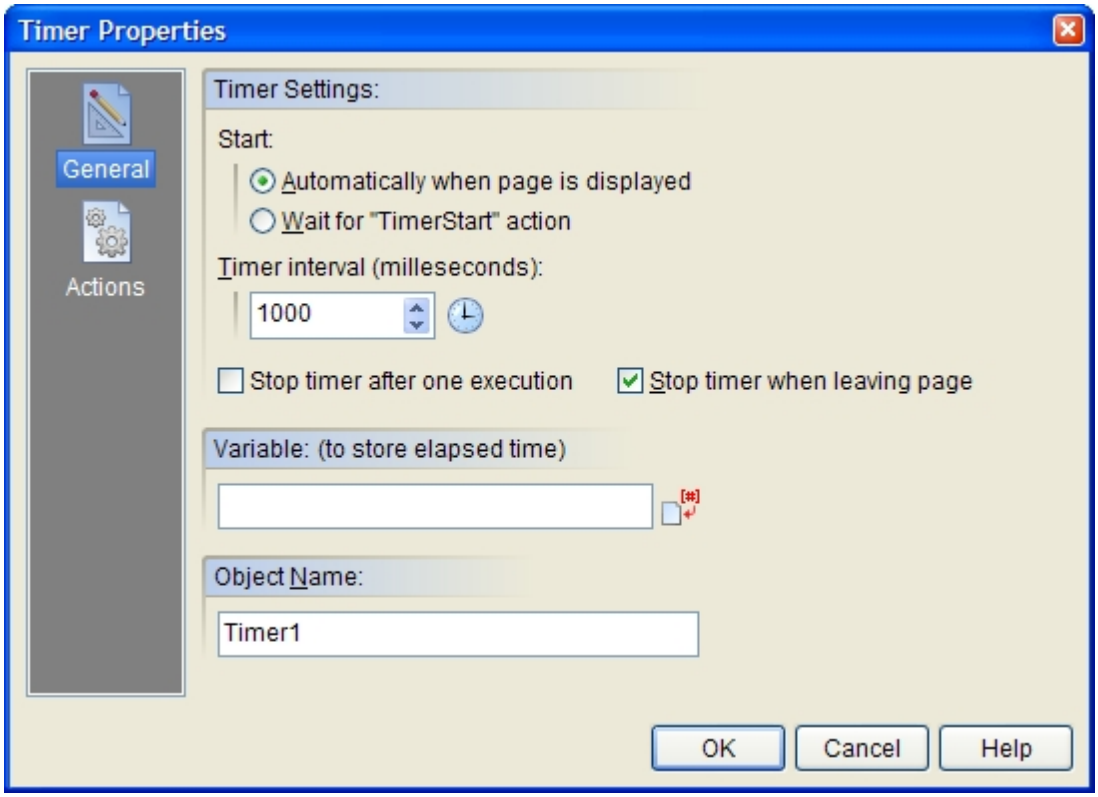
General

Timer objects can be activated either manually or automatically. Enable the **Automatically when page is displayed** option to activate the Timer immediately whenever the page it's on is displayed. Use this option when you want to give the reader a set amount of time to view a page before moving on. If **Wait for "TimerStart" Action** is selected, the Timer will remain

inactive until it receives a TimerStart command. You might choose to have the Timer sit dormant until the reader clicks a Push Button or responds to a MessageBox. For example:

```
TimerStart "Timer1" "2000"
```

The **Timer interval** is specified in milliseconds (one-thousandth of a second). For a one second interval enter 1000 milliseconds. For one minute interval enter 60000 milliseconds. Once activated, this is the amount of time that will elapse before the Actions associated with the Timer are executed.



Once activated, a Timer will execute repeatedly until manually stopped or the publication shuts down. The assigned Action commands will be executed each time the timer's interval is reached. If you only need the Timer's Action to execute once, enable the **Stop timer after one execution** option. Alternatively, enabling the **Stop timer when leaving page** option will allow the Timer to run until the reader navigates from the current page.

If neither of the above stop options are enabled, you can deactivate the Timer manually using the TimerStop Action. For example:

```
TimerStop "Timer1"
```

You can use the **Variable (to store elapsed time)** to keep track of how many milliseconds the Timer has been running. To use this option, enter a variable name into the field. At runtime, this variable can be examined to determine how long the Timer has been running. This feature could be used to create a rudimentary stopwatch. For example, you could track how long a reader took to complete a test:

```
.convert the elapsed time from milliseconds to minutes
Math "[ElapsedTime] / 60000" "2" "[Score]"
AlertBox "Hello" "You took [Score] minutes to finish the test."
```

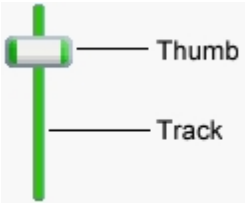
Actions

The Timer object supports only a single Action Event called **Timer Interval**. The Actions

entered here will be executed when the Timer's internal counter reaches the specified interval. (See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.) After the Actions have been executed, the Timer will be reset and begin counting from zero again unless it has been deactivated. If you do not want your Timer Action to execute more than once, include a TimerStop command in the Timer's Action or use the Stop timer after one execution option.

Track Bar Tool

 Track Bars, or sliders as they are sometimes called, allow readers to easily select a numeric value within a minimum and maximum range. A Track Bar is similar to your stereo's volume control. The volume control limits your choices to values preselected by the stereo manufacturer. Readers can adjust the Track Bar by grabbing the thumb button with the mouse and dragging it to a new location along the track. As the thumb button moves, the variable assigned to the Track Bar will be updated with the current position on the track. The position or value can be used in calculations, written to a file, etc.



To create a Track Bar, use the mouse to draw a rectangle where you would like the object to appear. The Track bar Properties screen will be displayed, allowing you to define the object's appearance and behavior.

The **Track Bar Properties** screen is divided into three sections indicated by the icon images on the left: General, Appearance and Actions. To view the settings for a section, click the corresponding icon.

General

In the **Minimum value** and **Maximum value** fields, enter the lowest and highest values to define the Track Bar's range. The minimum value must be less than the maximum value. The Track Bar will be initialized to the specified **Initial value** when your publication starts. The initial value must be within the minimum and maximum range.

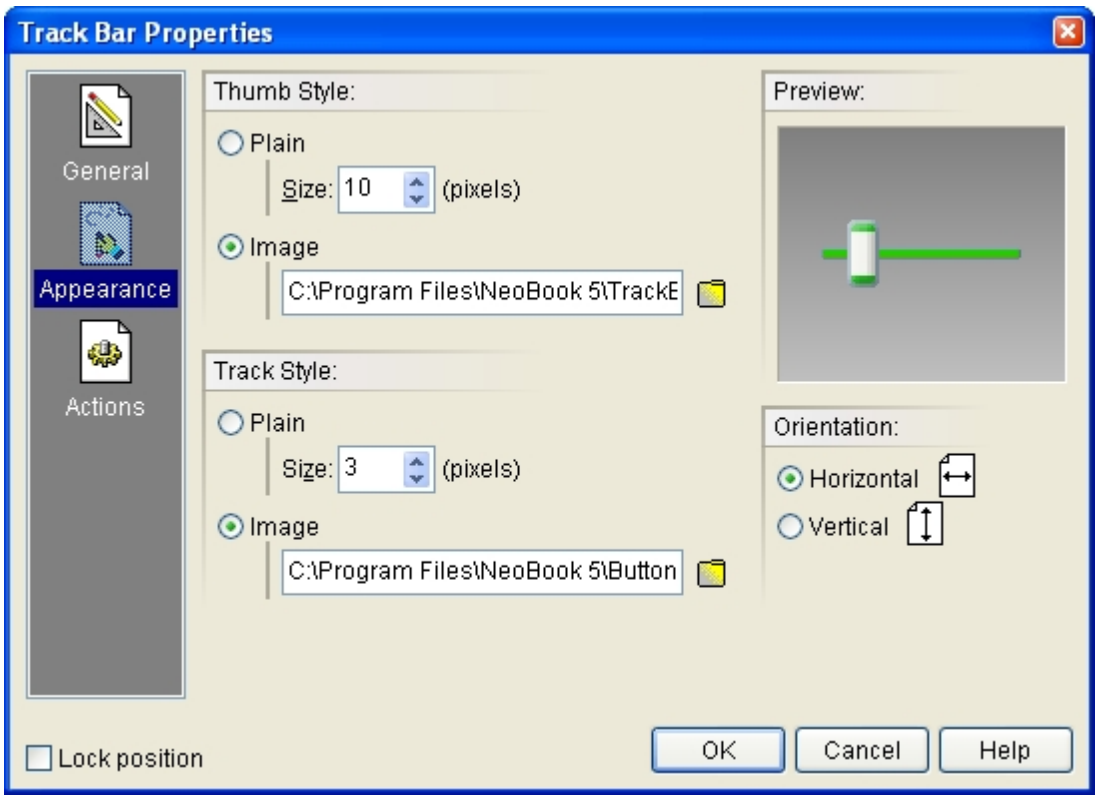
In order to keep track of the position of the Track Bar while your publication is running, you will need to assign the object a unique variable name. VisualNEO for Windows will assign a variable name that matches the Object Name, but you may change this by modifying the **Variable (to store track bar value)** field. At runtime, the variable will contain the value of the Track Bar. You can modify the value of the Track Bar programmatically by manipulating the variable using a simple Action command. For example:

```
SetVar "[TrackBar1]" "50"
```

Hint: *You can provide readers with additional feedback by placing a Simple Text object next to the Track Bar. Add the Track Bar's variable ([TrackBar1]) to the Text object and the text will update whenever the thumb button is moved.*

Appearance

The Track Bar's orientation may be set to **Horizontal** (left to right) or **Vertical** (top to bottom). The **Preview** on the right shows you how your changes affect the appearance of the Track Bar.



Both the **Thumb Style** and **Track Style** options allow you to choose between plain rectangular features and custom images for your Track Bar's components. The two components that comprise the Track Bar are the moveable thumb and the track that the thumb travels across.

If you choose to use custom images, they should be drawn to the exact sizes that you need. Also, the orientation of the images and the Track bar should be the same.

Actions

Track Bars support the following Action Events: **Value Changed**, **Value Changing**, **Mouse Enter** and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Media Player Tool

The Media Player provides you with the ability to play video and audio clips within your publication. The Media Player may contain a set of VCR-like controls allowing the reader to play, pause and stop the video clip at their own pace.

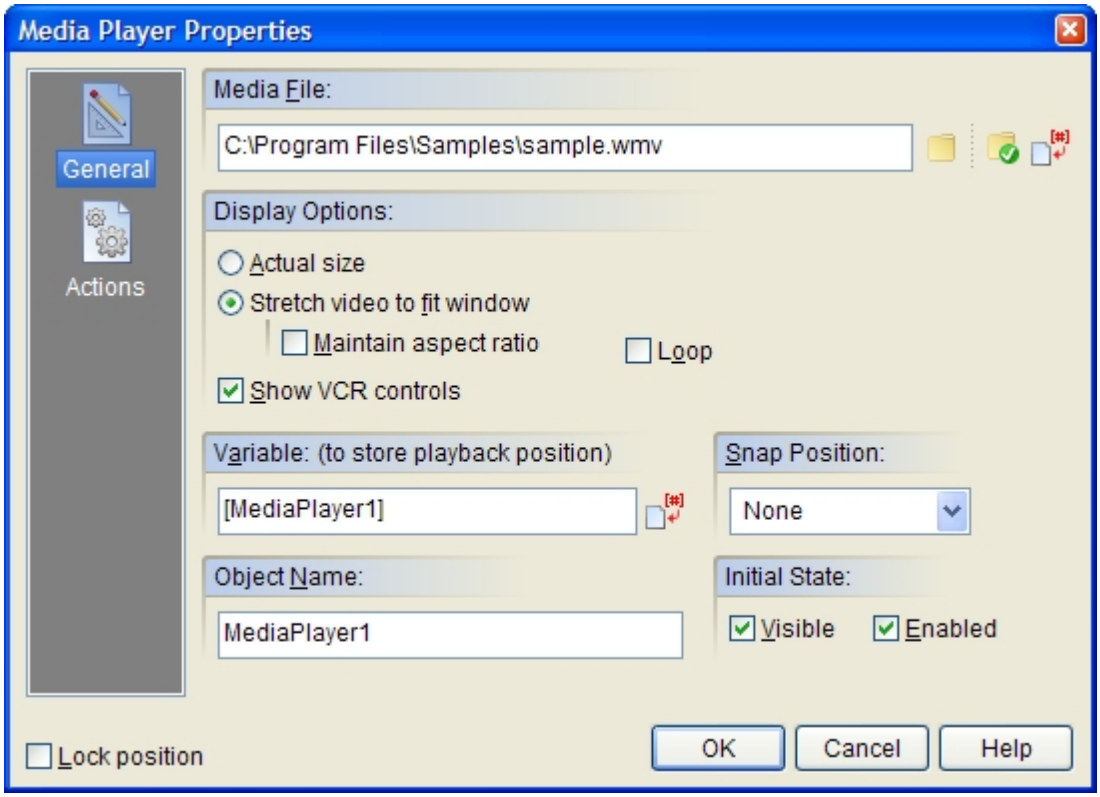
To add a Media Player, use the mouse to draw a rectangle where you would like the object to appear. A File Selector will display, allowing you to select a video or audio file. Once a file is selected, a Media Player object will be added to your publication. You can modify the Media Player by right clicking the object. The Media Player Properties screen will be displayed, allowing you to define the object's appearance and behavior.

The Media Player Properties screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

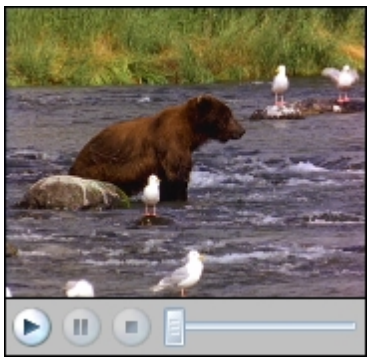
The **Media File** field contains the name of the video or audio file to be played. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and changes will appear in your publication. The file will be bound inside the publication only during the Compile stage. To the right of the field are three small buttons: click the button to select a different file; experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.

Note: *AVI files are generally the best choice if you plan on including video in your publications. The necessary drivers for playing most types of AVI files are incorporated into Windows, while many other video formats require that special software be installed. For example, before playing MPEG videos, you (and your readers) may need to install Microsoft's ActiveMovie™ driver. ActiveMovie is included with most newer versions of Windows, but for older installations, you may need to download the driver from www.microsoft.com.*



You can control the appearance of the Media Player using the Display Options. Enable the **Actual size** option to display the video in its original dimensions, or select **Stretch video to fit window** to distort the video image so that it fills the Media Player. Enable the **Maintain aspect ratio** option to restrict the stretching so that the original width to height proportions of the video are maintained. Of course, if the file you've selected contains audio only, these options will have no effect.

When the **Show VCR controls** option is enabled, the Media Player will include a small palette of VCR-like controls. These controls allow the reader to start, pause or stop the video playback. The palette also includes a small track bar that indicates the position of the current frame within the video. When the VCR controls are turned off, you must use VisualNEO for Windows's MediaPlayerPlay and MediaPlayerStop Action commands to start and stop the video playback.



Media Player with VCR controls

The images used for the VCR controls can be found in the "VisualNEO for Windows 5\Buttons\Resources" folder. You can change the appearance of the buttons and the Track Bar by editing these files using your favorite paint program.

Enabling the **Loop** option will play the file continuously until the Media Player is stopped manually.

You can keep track of the playback position of the Media Player by entering a variable name in the **Variable (to store playback position)** field. At runtime, the variable will contain the position of the current frame within the video. You can change the position of the video by manipulating the variable using a simple Action Command. For example:

```
SetVar "[MediaPlayer1]" "75"
```

Actions

The Media Player supports the following Action Events: **Clicked**, **Mouse Enter**, **Mouse Exit**, **Begin Playing**, **Finished Playing** and **Canceled**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Animated GIF Tool

Use this tool to display Animated GIF files within your publication. An Animated GIF is a special file containing a series of bitmap images that are displayed in rapid sequence like a cartoon. These are the same types of files used to display animated advertisements and logos found on many web sites.

To add an animation to your publication, use the mouse to draw a rectangle where you would like the image to appear. A Windows File Selector will appear, allowing you to locate and select an Animated GIF file. After you select the file, an Animated GIF object will be created. The animation will not be active while you are in Edit mode. To see the animation, switch to Run mode by pressing F8.

Hint: *If you would like to create your own Animated GIF files, there are a variety of utility programs available for this purpose. (One such program is PixelNEO® for Windows from the makers of VisualNEO for Windows. A trial copy of PixelNEO can be found on your VisualNEO for Windows CD.)*

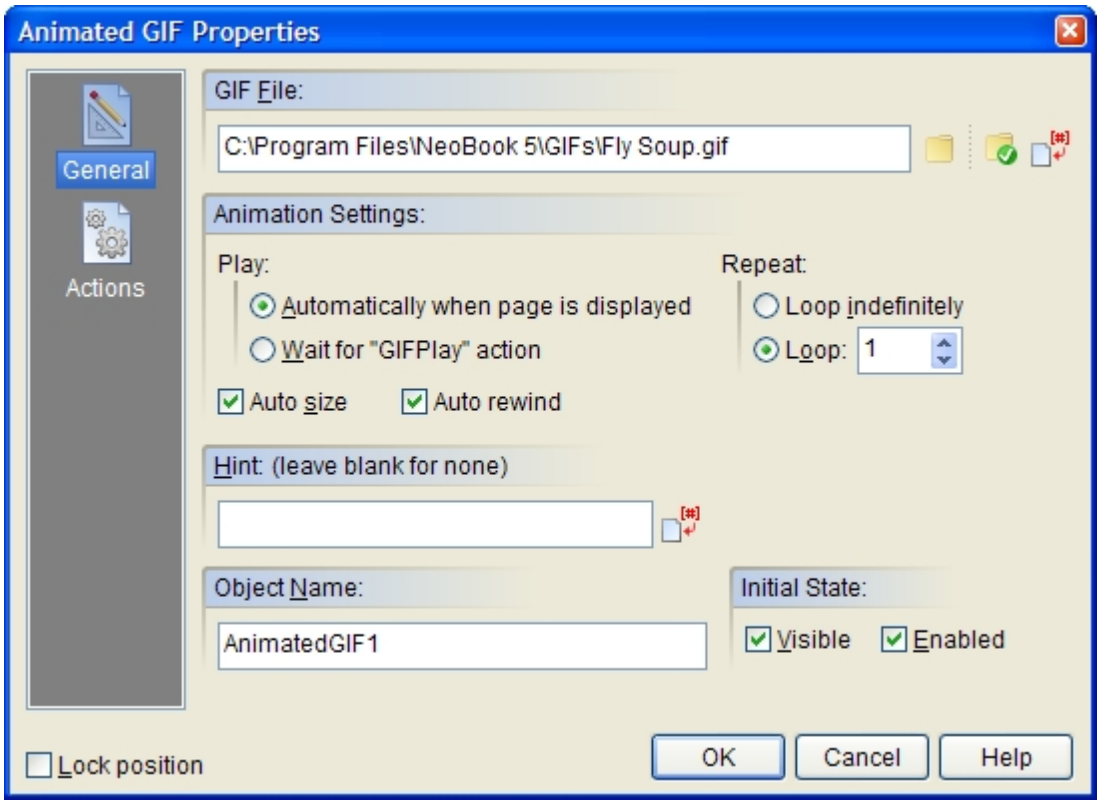
Once created, you can modify the Animated GIF's properties by right clicking the object. The Animated GIF Properties screen will be displayed, allowing you to alter the default behavior of the object.

The **Animated GIF Properties** screen is divided into two sections indicated by the icon

images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

The **GIF File** field contains the name of the animation file to be played. Even though you've imported the file into your publication, it's still an external file. Edit the file outside of VisualNEO for Windows and changes will automatically appear in your publication. The file will be bound inside the publication only during the Compile stage. To the right of the field are three small buttons: click the button to select a different file; experienced authors can use the button to perform advanced file options or the button to replace the file name with a variable.



You can specify when the animation will begin playing using the Animation Settings options. By default, the animation will start playing **Automatically when page is displayed**. This is handy if you always want the animation to play when its page is visible. Alternatively, you can have the animation remain motionless and **Wait for "GIFPlay" Action**. The GIFPlay and GIFStop Actions can be used to control a GIF object more precisely. Using these Actions, an animation can be started and stopped in response to a button click or as feedback for performing a task correctly. More information about these Actions can be found here.

Enabling the **Auto size** option instructs VisualNEO for Windows to match the dimensions of the GIF object with those of the file you selected.

The animation may be set to repeat over and over by enabling the **Loop indefinitely** option or set to **Loop** a specific number of times.

If you've selected an option other than Loop indefinitely, you can enable the **Auto rewind** button to restore the animation's first frame when finished. Disabling Auto rewind will cause the last frame to remain on screen after the animation is complete.

Actions

Animated GIF objects support the following Action Events: **Click**, **Mouse Enter**, **Mouse Exit**

and **Finished Playing**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Note: *The Finished Playing Action is ignored for animations set to loop indefinitely, since such animations technically never finish.*

Flash Player Tool

 The Flash Player allows you to incorporate Adobe®/Macromedia® Flash™ SWF content into your publication. The Flash content may be a local file or an Internet address pointing to a remote file. Some common types of Flash content include web site user interfaces, interactive online advertising and animation.

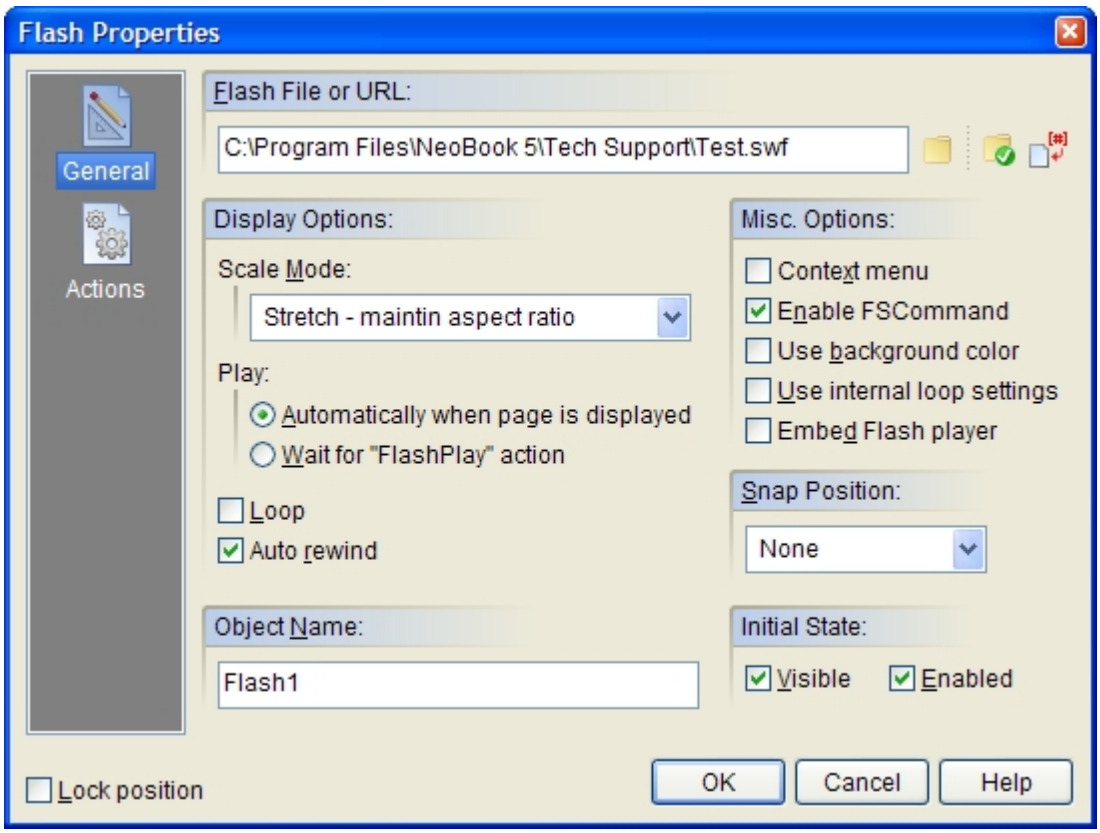
Note: *In order to use the Flash Player object, you and your readers must have version 5 or higher of the Adobe/Macromedia Flash Player installed. The latest version of the Flash Player can be downloaded from www.adobe.com. Adobe/Macromedia claims that 97% of all Windows PCs have some version of Flash installed, so unless your readers are using very old equipment, this should not be an issue.*

To add a Flash Player, use the mouse to draw a rectangle where you would like the object to appear. A File Selector will display, allowing you to select a SWF file. Once a file is selected, a Flash Player object will be added to your publication. You can modify the Flash Player by right clicking the object. The Flash Player Properties screen will be displayed, allowing you to define the object's appearance and behavior.

The **Flash Player Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

The **Flash File or URL** field contains the name of the Flash file to be played. You can select a local file (one located on your computer) by clicking the  button to the right of this field; experienced authors can use the  button to perform advanced file options or the  button to replace the file name with a variable. If you select a local file, it will remain external until the compile stage when it will be bound inside the publication for distribution.



You can control how the Flash file will appear by selecting one of the following **Display Mode** options:

Actual Size	The Flash content will be displayed using the original dimensions specified by the file's creator. If the content is larger than the space provided, it will be clipped at the object's border.
Stretch	The content will be stretched to fit the dimensions of the Flash Player object. The content may appear distorted if the object's dimensions are dramatically different than those of the content. To preserve the content's original width to height proportions, select the **Stretch - maintain aspect ratio** option instead.
No Border	If the Flash author included a border as part of the content, you can use this option to eliminate it. Otherwise, No Border is the same as the Actual Size option above.

You can specify when the Flash content will start playing using the Animation Settings options. By default, the content will play **Automatically when page is displayed**. This is handy if you always want the content to play whenever its page is visible. Alternatively, you can have the content remain motionless and **Wait for "FlashPlay" Action**. The FlashPlay, FlashPause and FlashStop Actions can be used to control a Flash Player more precisely. Using these Actions, a Flash file can be started and stopped in response to a button click or as feedback for performing a task correctly. More information about these Actions can be found in here.

The Flash file may be set to repeat over and over by enabling the **Loop** option.

Enable the **Auto-rewind** option to restore the Flash file's first frame when finished. Disabling Auto rewind will cause the last frame to remain onscreen after the Flash file is complete.

Note: *Some Flash files do not distinguish between static and play modes. In this situation, Loop and Auto-rewind may be irrelevant and attempting to play or stop the file may have no effect.*

You can allow or prevent readers from accessing the Flash file's **Context menu**, which normally appears if you right click the Flash Player. The context menu provides readers with commands that allow them to modify the Flash file's appearance. If you want to prevent readers from accessing these commands, turn this option off.

The **Enable FSCommands** option allows special VisualNEO for Windows Action commands embedded inside the Flash file to be executed. The Flash FSCommand feature is an advanced programming tool used by Flash authors to communicate with the host application - in this case VisualNEO for Windows. Using FSCommand, Flash authors can execute any of VisualNEO for Window's Action commands. Using FSCommand is beyond the scope of this manual, but experienced Flash authors need to direct the command at VisualNEO for Windows. For example:

```
on (release) {fscommand ("VisualNEO for Windows", "AlertBox \"A Message From
VisualNEO for Windows!\" \"Hello there,|This is an Alert box From VisualNEO
for Windows.\"");}
```

In addition to VisualNEO for Windows specific FSCommands, you can manually intercept FSCommands intended for other applications. To do this, define a special subroutine called ObjectName_FSCommand. Replace ObjectName with the name of your Flash Player object. Before calling the subroutine, VisualNEO for Windows will create two variables called [ObjectName.fsCommand] and [ObjectName.fsArgs] containing the information passed from the SWF file to VisualNEO for Windows. Again, replace ObjectName with the name of your Flash Player. The contents of these variables can be almost anything, so some knowledge of the SWF file will be helpful when deciding how to interpret this information. The example subroutine below displays an AlertBox in response to FSCommands:

```
:Flash1_FSCommand
  AlertBox "FSCommand" "[Flash1.fsCommand], [Flash1.fsArgs]"
Return
```

See Book Properties > Actions for more information on creating subroutines.

Enable the **Use background color** option if you want the Flash Player to use the object's Fill Color (selected from the Style Palette) instead of the background color stored inside the Flash file.

Enabling the **Embed Flash Player** option will include a copy of the Flash Player (flash.ocx) installed on your PC inside the compiled publication. Later if someone viewing your publication does not have a Flash Player installed, VisualNEO for Windows will use the compiled internal player to display SWF files.

Actions

The Flash Player supports the following Action Events: **Load Complete**, **Finished Playing** and **Canceled**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Note: *The Finished Playing Action will be ignored if the Loop option is enabled.*

Container Tool

 The Container object is primarily intended to be used as a surface for other objects. It's similar to a group, except that objects placed on a container can be moved and edited easily. This makes the Container an excellent tool for creating sizeable publications and custom windows.

To add a Container to your publication, use the mouse to draw a rectangle where you would like the object to appear. Other objects can be attached to a Container by selecting a tool from the Tool Palette and drawing directly onto the Container's surface. (You can even attach other Containers.) Existing objects can be moved onto a Container by cutting and pasting them from the clipboard. When using this method, make sure that the container is selected before choosing paste. To move an object off a container, select the object, cut it to the clipboard, make sure no other objects are selected, then paste. Once an object is attached to a Container, the two become linked. Moving the Container also moves any attached objects. Similarly, deleting a Container also deletes any attached objects.

You can modify a Container by right clicking the object. The Container Properties screen will be displayed, allowing you to define the object's appearance and behavior.

The **Container Properties** screen is divided into two sections indicated by the icon images on the left: General and Actions. To view the settings for a section, click the corresponding icon.

General

If you're planning to attach other objects to your Container, you can adjust the horizontal and vertical **Margins** settings to create space between those objects and the edge of the Container. In order for the margin settings to affect the placement of attached objects, you must specify a Snap Position other than None for those objects.

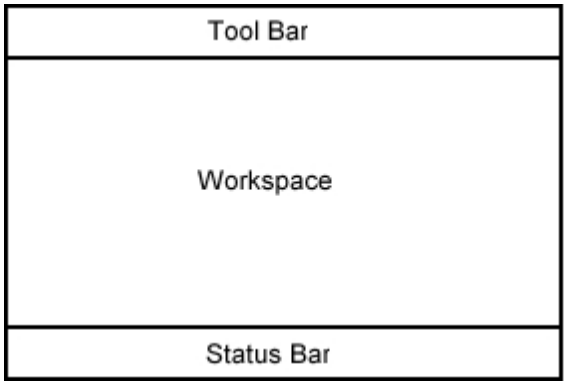
Actions

Container objects support the following Action Events: **Resize**, **Mouse Enter**, and **Mouse Exit**. Click the appropriate tab at the bottom of the Action Editor to create or edit Actions for the events you want to control. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Creating Sizeable Publications with Containers

There are some special techniques involved in creating sizeable publications, but with careful planning the process is actually quite easy. The first thing you need to do is decide what your publication's interface will look like. Generally, most Windows applications are divided into a few distinct interface components: a tool bar, status bar and workspace. More complex applications may include additional elements, but the process is basically the same.

Once you've decided on the elements that will make up your publication's interface, you need to decide where they should go. A typical layout might look like this:



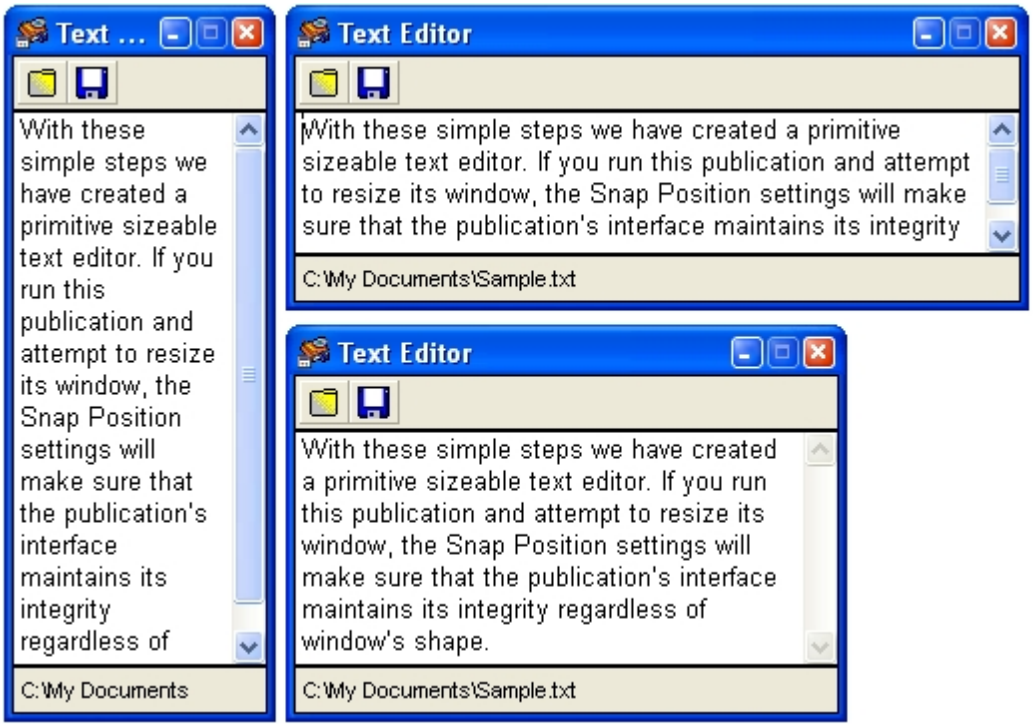
You can reproduce this layout in VisualNEO for Windows by creating three Container objects. Create the Tool Bar Container first and set its Snap Position to "Top". Next, create the Status Bar Container and set its Snap Position to "Bottom". Finally, create the Workspace Container and set its **Snap Position** to "Center." If needed, adjustments can be made to the heights of the top and bottom Containers, but the snap settings will prevent them from being

moved.

To complete the publication, buttons, pictures and other elements can be attached to each of the three base Containers. However, the Workspace Container is special because it will be most affected when the publication is resized at runtime. In a traditional Windows application, the workspace usually consists of a single component. For example, a word processor's workspace is the area where you enter text. When the word processor window is resized, the text area expands or shrinks, but the formatting controls, status bar and other components retain their basic size and shape.

To create a word processor with VisualNEO for Windows, you would attach a multi-line Text Entry Field object to the workspace Container and set its **Snap Position** to "Center." When Center is selected, the object will fill the portion of the workspace that is not occupied by other snapped objects. If the object has been placed on a Container object, as done here, then the selected Snap Position will apply to the bounds of the Container instead of the entire workspace. This is the key to making the whole thing work.

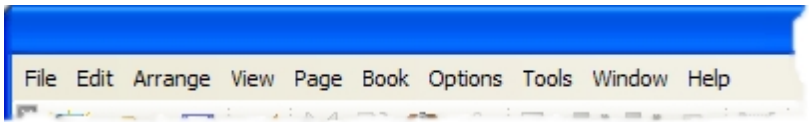
With these simple steps you have created a primitive sizeable text editor. If you run this publication and attempt to resize its window, the **Snap Position** settings will make sure that the publication's interface maintains its integrity regardless of the window's shape. For example:



A short tutorial describing the steps involved in creating a sizeable publication can be found here.

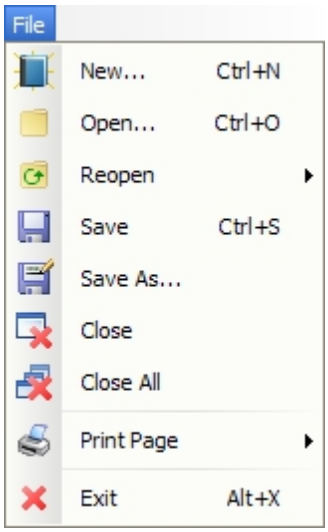
Menu Functions

This section contains descriptions of commands found in the Menu Bar at the top of VisualNEO for Windows's screen. Many menu commands also have short cut keys that can be used to access them more quickly from the keyboard. Click one of the menu topics below for more information on how to use the commands in the menu:



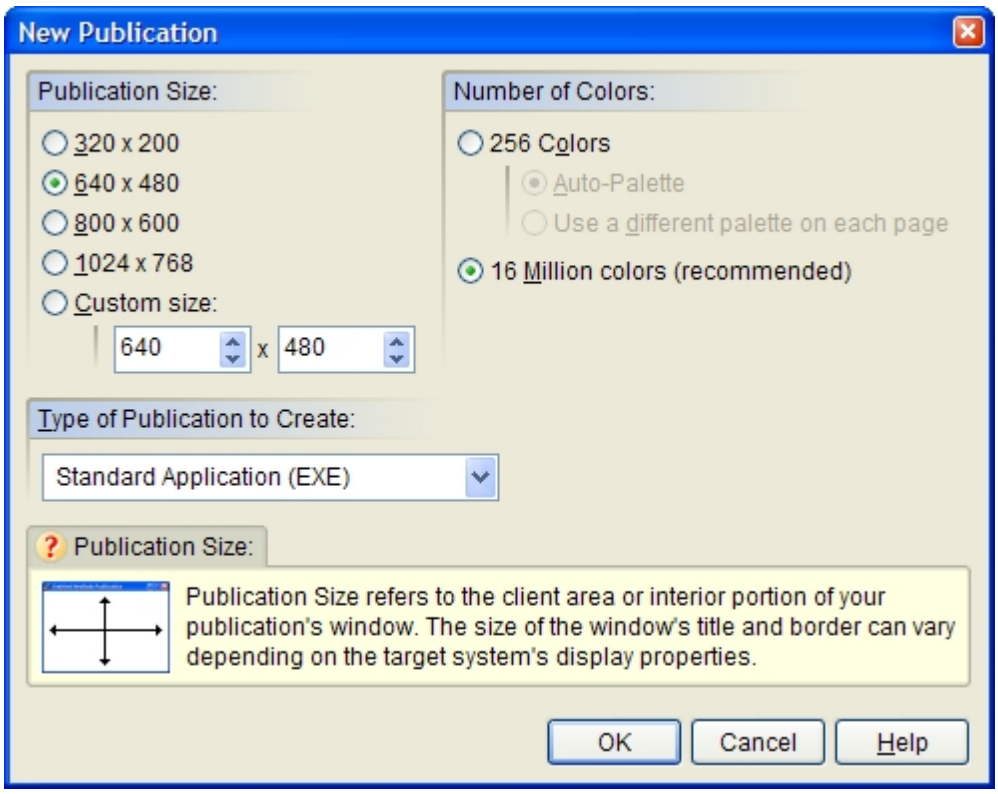
The File Menu

This section contains descriptions of the commands found in VisualNEO for Windows's File menu. For more information, select a command from the picture below:



New

Use this command to create a new, blank publication. Specify the desired screen size, color resolution and application type. An empty publication containing two blank pages will be created. One of the blank pages is the Master Page. The other is the first page of your publication. Additional pages can be added at any time by selecting Add Page from the Page menu. Use the Save command to store the new publication on your disk. You will be prompted to give your publication a file name the first time you save it.



The **Publication Size** you choose depends on what type of publication you are creating and what type of equipment your audience is likely to have. Most computer systems today have monitors capable of displaying a minimum of 640 x 480 pixels and 256 colors. This is generally considered the lowest common denominator – the minimum capabilities you can safely assume your reader will have. Therefore, publications of this size or smaller will be able to run without problems on the largest number of computers. However, if your publication is to be used in a location where you know that your readers will all have the same type of equipment (a school computer lab for example), the lowest common denominator may be much higher – perhaps 800 x 600 pixels and 16 million colors.

Depending on what type of publication you are creating, you also may need to consider a few other factors when deciding on a size. If your publication will run in full screen mode, then you will have the entire screen area to work with. However, if your publication window will include a title bar, menu bar and border (see App Properties), you will need to allow for this space when selecting a size. Since the height of the title bar, menu bar and border can vary depending on how the user has configured Windows, you'll need to estimate the amount of space to reserve for these elements. The size you select defines the client area or interior portion of the publication's window – not the area used by the title bar and border.

VisualNEO for Windows also supports the creation of sizeable publications that can automatically adjust to whatever size screen is available. When creating a sizeable publication, the dimensions you select represent an optimal size rather than a strict limitation.

Selecting the **Number of Colors** for your publication is fairly straight forward. The 16 million color option gives you a virtually limitless selection of colors and provides the best rendering of photographic images. However, as discussed earlier, some older computers may not be capable of running in this mode. If that's the case, VisualNEO for Windows will do the best it can to display your publication given the limitations of the reader's computer.

For 256 color publications, each page may be assigned its own unique 256 color palette. VisualNEO for Windows also provides an Auto-Palette option that uses a generic color palette for the entire publication. The Auto-Palette contains a full spectrum of general colors including the 20 standard colors used by Windows.

VisualNEO for Windows's Compiler is capable of producing finished applications in four different formats. These include: Standard Application, Screen Saver, System Tray Application and ActiveX Control. Each of these **Publication Types** have different properties, so which one you choose depends on the needs of your particular project. See Compile for more details about Publication Types.

Open

This command allows you to load an existing VisualNEO for Windows publication for editing. A standard Windows file selector will appear allowing you to select a publication from your hard drive. When opening the publication, VisualNEO for Windows will check to see if any required files are missing and display a message if any cannot be located.

Reopen

This command displays a list of recently opened publications. To load one of these publications, click on its title.

Save

Choose this command to save changes made to your publication. (Closing a modified publication will automatically prompt you to save your work.) If you choose not to Save your work, it will be lost once the editing window is closed.

If this is a new publication that has not previously been saved, you will be prompted to specify a file name.

Note: *VisualNEO for Windows can import files from VisualNEO for Windows for DOS and earlier Windows versions. However, once these files are saved in version 5 format, they will no longer be readable by the older version. If you import publication files from an older version, please use the Save As function to save the new publication under a different name.*

Save As

Use this command to save an existing publication using a different file name. If you select a file name which already exists, VisualNEO for Windows will ask for confirmation before replacing the existing file.

Note: *VisualNEO for Windows publication file names should end with the file extension ".PUB".*

Close

This command closes the currently-selected editing window. You may also shut down a publication by clicking the close button  on the right side of the window's title bar. If any changes have been made to the publication, you will be prompted to save your work before the window is closed.

Close All

This command closes all open publications.

Print Page

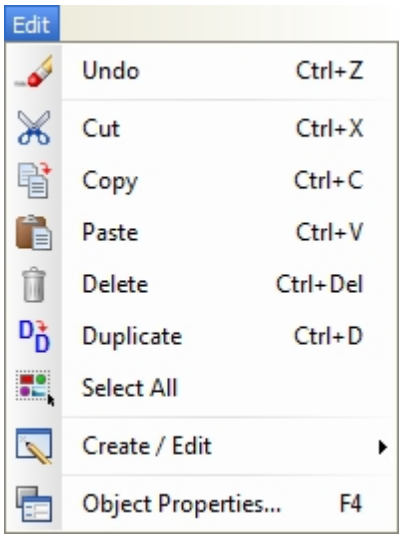
This command prints the active publication's current page. Two quality options are available - **Draft** and **Final**. Draft quality produces an exact copy of the page that matches the resolution of the screen, which is lower than that of the printer. Final quality produces a much higher resolution representation of the page, but some elements may not reproduce exactly as they appear on the screen.

Exit

This command closes VisualNEO for Windows.

The Edit Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Edit menu. For more information, select a command from the picture below:



Undo

This command removes the last change you made to the publication.

Cut

Use this command to remove the currently selected objects and place them onto the Windows Clipboard. The cut objects will remain on the Clipboard until replaced by another Cut or Copy command. Use the Paste command to place objects from the Clipboard onto a page. The cut objects also may be pasted into another publication by switching to that publication's window and choosing Paste.

Copy

Use this command to place copies of the selected objects onto the Windows Clipboard. The copied objects will remain on the Clipboard until replaced by another Cut or Copy command. Use the Paste command to place objects from the Clipboard onto a page. The copied objects may also be pasted into another publication by switching to that publication's window and choosing Paste.

Paste

The Paste command places a copy of the Clipboard contents onto the current page. After pasting, you may use your mouse to drag the objects to a new location. In addition to items copied from VisualNEO for Windows, you may paste most types of plain text and bitmap images copied from other programs.

Note: *The Paste command will be disabled if the contents of the Clipboard are not compatible with VisualNEO for Windows.*

Delete

Use this command to remove the selected objects from the current page. Deleted objects are not placed onto the Windows Clipboard and cannot be pasted back into the publication. If you delete objects by mistake, you can use the Undo command to recover them.

Duplicate

This command creates copies of all selected objects. The copied objects are placed to the right of the originals on the current page. Duplicate is essentially the same as copying and pasting in a single action.

Select All

Use this command to select all objects on the current page. Once selected, you may use your mouse to drag the objects to a new location, to another open publication, or another page; or you may use the Copy or Cut commands to place the items onto the Clipboard.

Create / Edit

This command displays a submenu allowing you to select and edit an external file associated with any currently selected objects. If no objects are selected, you may create a new external file using one of your assigned editor programs. Editors for a variety of file types can be specified on the Tools page of the Set Preferences screen.

Object Properties

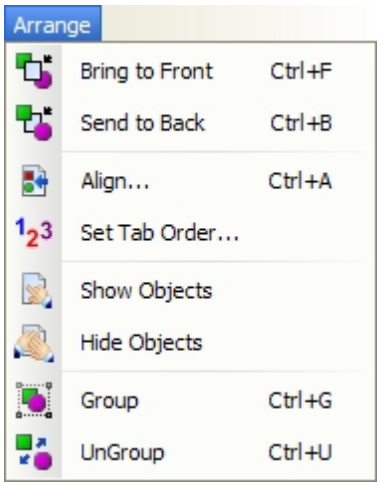
When a single object is selected, use this command to modify its properties and behavior. The contents of the properties screen depends on the type of object selected. For an

explanation of the tools used to create VisualNEO for Windows objects and their properties, please see The Tool Palette.

Note: *You also can access Object Properties by placing the mouse pointer over an object and clicking the right mouse button. You also can jump directly to the Action page of an object's properties screen by double clicking the object.*

The Arrange Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Arrange menu. For more information, select a command from the picture below:



Bring to Front

Use this command to move the selected object or group to the foreground, on top of other objects on the current page. When you move a group of objects to the front, they will remain in the same order, but on top of the other objects that are not part of the group.

Send to Back

Use this command to move the selected object or group to the background, behind other objects on the current page. When you move a group of objects to the back, they will remain in the same order, but behind other objects that are not part of the group.

Align

This command allows you to position objects relative to the page boundary or, if multiple objects are selected, relative to each other. For example, if you selected three objects, chose Align, and checked the Relative to Page option, all three objects would be placed at the very top of the page. However, if you checked the Align Relative to Other Selected Objects option, they would be aligned to the top of the object with the highest vertical position.

Also, both the horizontal and vertical alignments may be adjusted independently. For example, selecting Horizontal/Left and Vertical/No Change will adjust the horizontal position of the objects while leaving their vertical position unchanged.

Set Tab Order

Use this command to set the order in which readers may select objects on your VisualNEO for Windows page. When viewing your finished publication, users may move between some types of objects using the Tab key. The Set Tab Order screen allows you to specify the order in which objects will be selected. In Run mode, an object is selected when it responds to keys typed on the keyboard (it has the input focus.) The object at the top of the Tab List will have the input focus first, followed by the second object and so on. Setting the Tab Order is useful for publications that support keyboard access, such as text entry forms. See App Properties / Access to enable or disable tabbing.

Hint: *Not all types of objects can accept input. These objects will not appear in the Tab List. Push Buttons, Check Boxes, Radio Buttons and Text Entry Fields are examples of objects that can accept input. You also may see Rectangles appear in the Tab List. This is because plug-ins sometimes use rectangles to host special objects. Even though rectangles appear in the Tab List, Windows will not include them in the tab order unless a plug-in requires it.*

Show Objects / Hide Objects

User experience can be enhanced by showing and hiding certain objects during a presentation. A print button that remains hidden until an onscreen form is properly filled in is one example. Use these commands to change the initial visibility of selected objects. For example, if an object is to be hidden at the start of your publication, use the HideObject command to make it invisible before switching to Run mode. When the publication is running, you can use the ShowObject Action to display the object at the appropriate time.

Hint: *A hidden object will not be visible on screen, but you still can select it using VisualNEO for Windows's Object List. See the View menu.*

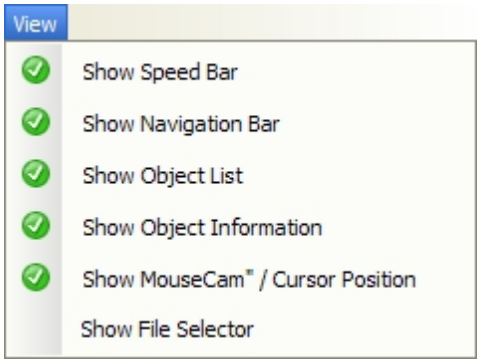
Group / Ungroup

Use this command to combine several selected objects into a group. Grouped objects may be copied, aligned, moved, etc. as if they were a single object. Use the Ungroup command to divide a group back into its individual components. The Group command is only available when more than one object is selected. Ungroup is only available when a previously created group is selected.

The View Menu

The View Menu contains commands for showing and hiding VisualNEO for Windows's palette windows. You can customize your VisualNEO for Windows environment by positioning the palette windows anywhere on the screen, or by docking them to the left, right, top or bottom edges of the main program window. Docked palettes can be repositioned, or undocked, by dragging the small bar at the top or left of each palette window.

For more information, select a command from the picture below:



Show Speed Bar

The Speed Bar contains buttons that provide single click access to several often-used VisualNEO for Windows commands. The Speed Bar may be hidden or displayed by selecting the Show Speed Bar option.

Show Navigation Bar

Use this option to show or hide the Page Navigation buttons. The Navigation bar provides a VCR-style control panel for quickly moving between pages. If you hide the Navigation Bar, you still can move between pages using the keyboard's page up, page down, home and end keys, as well as the Page Tabs at the bottom of the screen.

Show Object List

This option displays a palette window containing a list all of the objects (buttons, text fields, pictures, etc.) found on the current page of your publication. Click on an item in the Object List to select that item in the work space. Right click on the object's name to display a menu of options. Menu choices include Object Properties, Bring to Front, Send to Back, Show Object and Hide Object. These are shortcuts to commands also found in VisualNEO for Window's Edit menu.

Show Object Information

The Object Information palette allows you to monitor and adjust the position and size (in pixels) of a selected object. Enter the desired position and size into the palette's fields (left, top, width and height). The object's left and top positions are measured from the top/left corner (0 x 0) of the publication's work area. After entering the object's new coordinates, click on the Apply button to view the effect the new settings have on the object.

Show MouseCam / Cursor Position

Enabling this option will display a palette window containing a magnified view of the screen beneath the mouse pointer, along with the coordinates of your mouse cursor. The coordinates are in screen pixels as measured from the top/left corner of the publication's work area. You can turn the cursor position indicator on or off, and adjust the magnification settings by right clicking on the middle of the MouseCam window. The MouseCam is for editing purposes only and will not appear in your compiled publications.

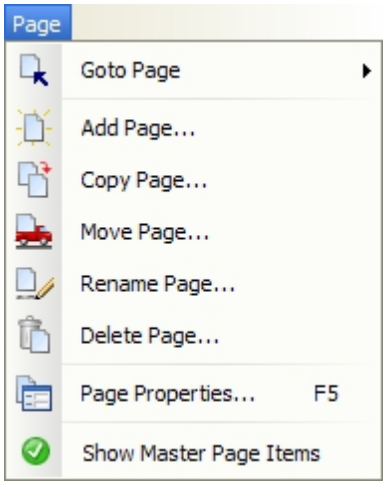
Show File Selector

This option displays a small window containing a list of folders and files on your computer. You may select and drag files directly from this window onto your publication. The contents of the file will be placed automatically onto the current page. This is an easy way to quickly build a publication. Any type of file supported by VisualNEO for Windows (text, picture, sound, animation, video, etc.) can be added to a publication using the File Selector window.

You can navigate to different drives or folders by double clicking those items in the file list. Double clicking on the "\" folder takes you to current drive's root directory. Double clicking on the ".." folder takes you back one level to the immediate parent of the current folder.

The Page Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Page menu. For more information, select a command from the picture below:



Go To Page

Use this command to jump to another page in the publication. To view a specific page, click on that page's title in the submenu. If there are more than a few pages in the publication, an option will be available to list additional pages that will not fit in the menu. You also can move between pages using the Page Tabs at the bottom of the VisualNEO for Windows screen. Pressing F7 will display a list of pages to select from.

Add Page

This command allows you to add additional pages to your publication. You will be prompted to specify the number of pages to add, and whether to insert them before or after the current page or at the end of the publication.

You also have an opportunity here to specify a title for the new page. If multiple pages are being added, new page titles will be based upon the name you specify. For example, if you specify "Sample" as the new title, pages added will be named "Sample 1", "Sample 2", etc.

Copy Page

Use this command to create an exact duplicate of the current page, including all of the objects it contains. The duplicate page may be edited, renamed or relocated using the Move function. This feature is handy when you need to create several pages that contain the same elements.

Hint: *You can copy pages from one publication to another by opening both publications side by side, then dragging the tab of the page you want to copy onto the other window.*

Move Page

Use this command to relocate the current page to another point in your publication. Select a page near where you would like the current page to be moved. Then decide if you would like the current page to be placed Before or After the selected page.

Hint: *You also can move pages by dragging the Page Tabs at the bottom of the screen, or by dragging the small thumbnail images found the Page Layout screen.*

Rename Page

Use this command to change the title of the current page. Type the new title into the space provided, then click OK to save the change or Cancel to return to editing without renaming the page. Remember, each page in a publication must have a unique title. VisualNEO for Windows will let you know if you attempt to use a title that's already taken.

Hint: *When renaming a page, VisualNEO for Windows will automatically update any links or page actions that reference the old page title.*

Delete Page

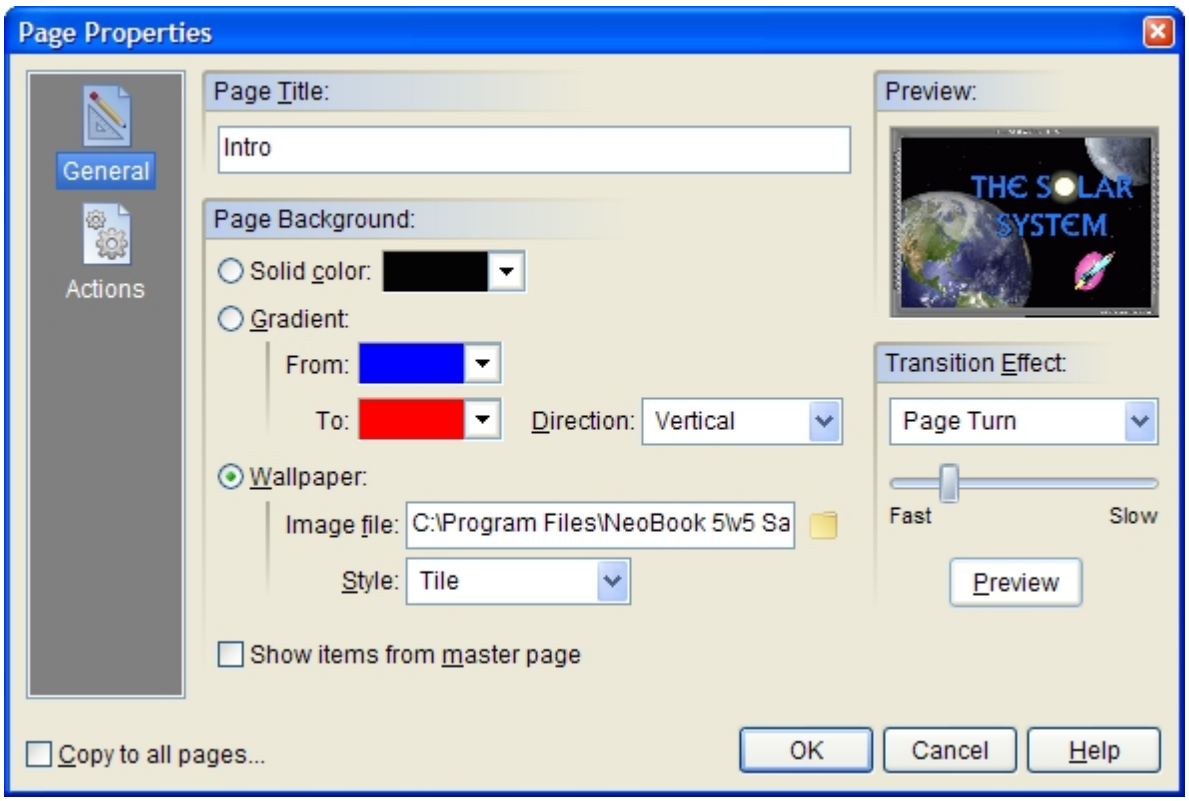
This command will remove the current page from your publication. This command cannot be undone, so use it with care!

Page Properties

Use this command to change the attributes of the current page. The **Page Properties** screen is divided into sections indicated by the icon images on the left. General, Actions and (for 256 color publications) Palette. To view the settings for a section, click the corresponding icon.

General

Use **Page Title** to assign a unique name or number to the current page. This title will appear on the page's tab at the bottom of the VisualNEO for Windows screen. Remember: you can change the title assigned to a page, if you like, but no two pages in a publication can have the exact same title.



You can enhance your publication by selecting an interesting background for each page. You may select a **Solid Color**, create a **Gradient** (the blending of two colors), or choose an image file to use as a **Wallpaper**-type background. To select a plain background, click the **Solid Color** option, then click the arrow to the right of this option, and choose the desired background color from the drop-down color box. (The selection of colors is determined by the resolution of the publication and palette assigned to this page.) To create a Gradient Color background, select the **Gradient Color** option, then click the arrow buttons next to the two color boxes and choose your colors. Choose Horizontal or Vertical to change the **Direction** of the gradation. Select **Wallpaper** to use an image as a background. Click the small folder icon next to the Image File box, then select the desired file.

If you enable the **Show Items from Master Page** box, controls, images and other objects which have been placed on the Master Page will be displayed on this page as well.

A variety of animated special effects are available to smooth the transitions that your readers will see when moving between pages. Click on the arrow button next to the **Transition Effect** option to select dissolves, slides, page turn, explode/implode, splits, weaves or wipes. The small **Preview** box will display a sample of the effect you select. Click the Preview button (below Transition Effects) so see it again. You can control the speed of the effect by moving the slider control between Fast and Slow.

Hint: *If you require additional effects not available in VisualNEO for Windows's built-in selection, a plug-in is available from www.billeniumsoft.com that adds a number of professional effects to VisualNEO for Windows.*

Actions

You can define special Actions to be taken each time the reader enters or leaves a page. These Actions are particularly useful for slide shows and testing publications. Actions can be used to play animation or music, display messages, calculate scores, verify input, etc.

The Action screen contains a large editor window and a tool bar. Action commands may be typed directly into the editor, but most authors prefer to use the **Insert Action** button instead. Clicking the Insert Action button will display a list of available commands. The

commands are organized into groups. Simply locate and click the title of the command you want. For most commands, VisualNEO for Windows will display a simple form that can be filled in to complete the Action. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

You may specify separate Actions to perform when the reader enters and leaves the page using the **Page Enter** and **Page Exit** tabs located just below the editor window.

If you're creating a slide presentation, two commands that you may wish to use in your Page Enter Action are Delay and GotoNextPage. This is a quick method of constructing a self-running, timed slide show type-presentation. For example:

```
Delay "3000"
GotoNextPage
```

Palette

You may find the Palette Editor useful when designing publications to be viewed on computers running in 256 color mode. Many older computers can display only a limited number of colors at one time. This limited selection of colors is called a palette. VisualNEO for Windows allows you to decide which colors will appear on the palette. If desired, each page may be assigned its own unique 256 color palette. VisualNEO for Windows also provides an Auto-Palette option that uses a generic color palette for the entire publication. The Auto-Palette contains a full spectrum of general colors including the 20 standard colors used by Windows. The Auto-Palette feature can be turned on or off from the Size/Colors section of the App Properties screen.

Hint: *The Palette Editor is not available when creating publications that use more or less than 256 colors since those resolutions don't use palettes. The Palette screen will not appear if the Auto-Palette option is turned on.*

The left side of the Palette Editor contains a selection of colors in the palette assigned to the current page. Select a color to modify by clicking one of the color boxes in the palette. The Red, Green, Blue and Brightness sliders on the right will change to reflect the color values that comprise the selected color: you may use your mouse to modify the selected color by changing the position of each slider. The top three sliders represent the amount of Red, Green and Blue (RGB) pigment that comprise the selected color. The Brightness slider adjusts all three color sliders simultaneously to lighten or darken the selected color. A preview showing the effects of your changes to the selected color appears below the palette.

If you make a mistake, use the **Reset Color** or **Reset Palette** buttons to restore either the currently selected or all palette colors to their original values. To load a new palette from a previously saved palette (PAL) file, click the Load Palette button. To save your edited palette as a PAL file for use with other pages or publications, click the Save Palette button and assign the palette a file name.

Hint: *When designing a publication to be viewed by the general public, it is always a good idea to try running a copy of your compiled publication on other systems which display only 256 colors. This will give you a better idea of how your readers will see your work.*

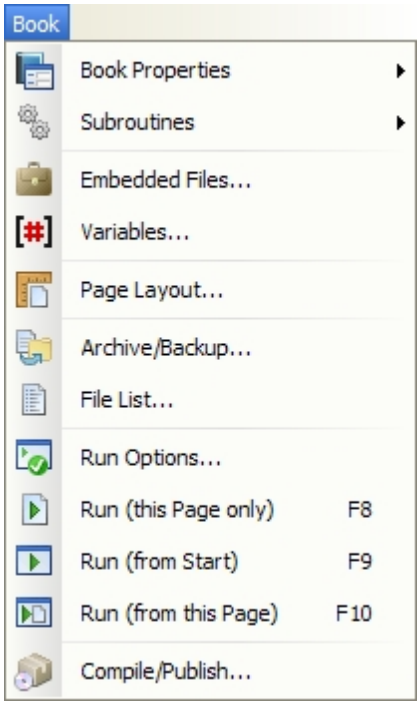
If you want the page attributes you've selected to be used for all the pages in your publication, check the **Copy to All Pages** box in the lower left side of the screen.

Show Master Page Items

Enabling this option will display items from the Master Page on the current page.

The App Menu

This section contains descriptions of the commands found in VisualNEO for Windows's App menu. For more information, select a command from the picture below:

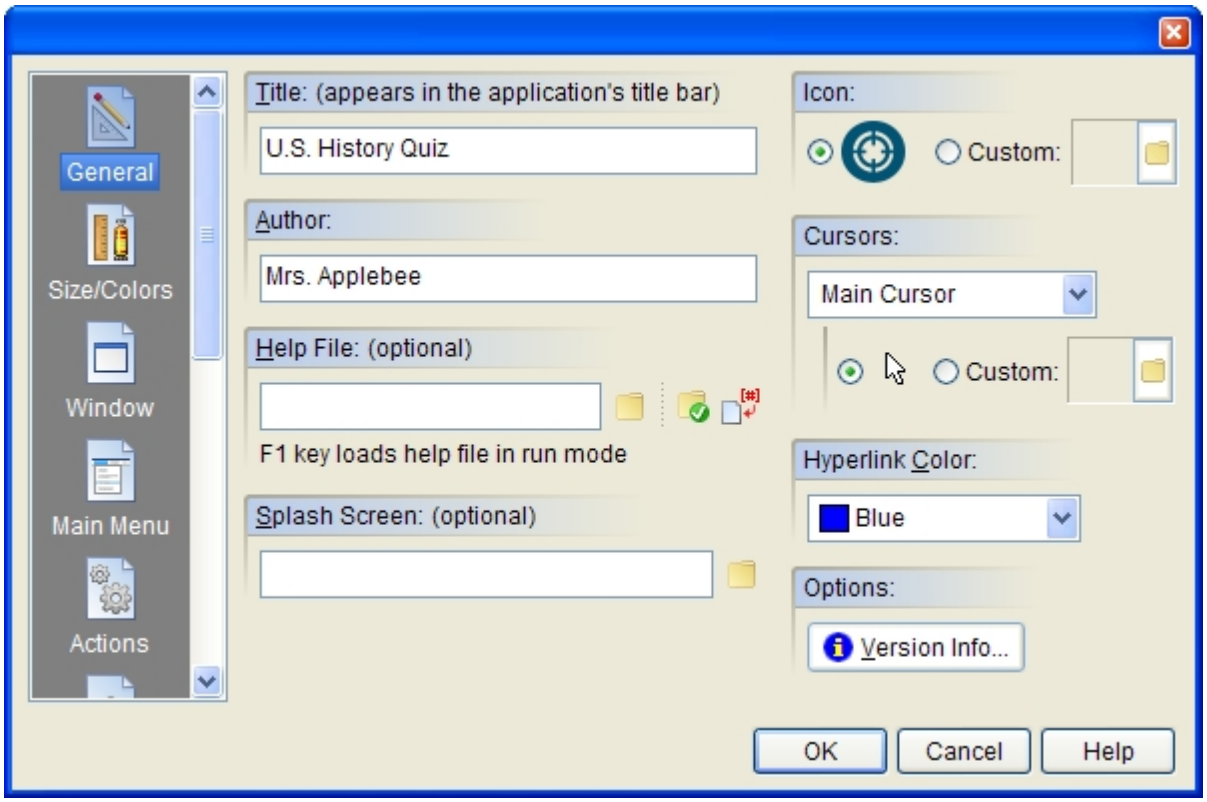


App Properties

Use this option to configure general publication-wide properties. The **App Properties** screen is divided into the following sections:

General

Enter the publication's **Title** and **Author's** name here. The Title will be displayed in the compiled publication window's title bar. The author's name is for your use only and does not appear in the publication.



Hint: *The publication's title also can be modified programmatically while your publication is running by changing the contents of the global [PubTitle] variable.*

You may enter the name of a Windows **Help File** (HLP or CHM) if you've created one to go along with your publication. The Help File is optional, but if your publication is complex, it may be beneficial to offer one to your readers. The Help File can be accessed from the compiled publication by pressing the F1 key. To create a Help File, you will need to use a special program called a Help Compiler.

The optional **Splash Screen** is a custom image that appears onscreen while your publication is loading. Large publications may take several seconds to load. A well-designed splash screen provides readers with something to look at while they wait. In addition to the image that you specify here, a small gauge will indicate the progress of the loading process. A splash screen can contain anything from a pretty picture to information about your product or company. To create a splash screen, simply draw a picture using your favorite paint program, then insert the picture's file name into the space provided here. You may also choose to draw a 3D **Border** around the splash screen or include a **Progress bar** to provide an estimate of your application's load time.

You can assign a custom icon image (ICO) to your VisualNEO for Windows publication by clicking on the **Custom Icon** button. When compiled, the icon will be incorporated into your finished publication. Multi-resolution icons are supported.

Note: *If you wish to create your own ICO files, there are a variety of utility programs available designed for this purpose. (One such program is PixelNEO™ from the makers of VisualNEO for Windows. A trial copy of PixelNEO can be downloaded from visualneo.com)*

You may select the mouse cursors that will be used in your publication. The **Hotspot Cursor** appears when the mouse is over an object that accepts user input (like a button or track bar control); the **Drag Cursor** is used for draggable objects; the **Busy Cursor** is used when the system is busy; and the **Main Cursor** is used everywhere else. Cursors must be in Windows CUR file format. You can also use most types of ANI files (which are animated cursors). A variety of utilities are available that will allow you to create your own CUR files. (See PixelNEO for Windows.)

The **Hyperlink Color** specified will be used to highlight any hypertext links you add to Article or Simple Text objects.

Click the **Version Info** button to change the version number, description, company name, and copyright information that will be embedded inside your compiled publication's exe file. The information entered here will only be visible when your compiled exe's properties are viewed from Windows Explorer. Enabling the Auto increment build number option will instruct VisualNEO for Windows to add one to the build number each time your publication is compiled. You may leave the version information blank if you choose.

Advanced authors may wish to change the default **Requested Execution Level** setting for certain types of Vista/Windows 7 applications, such as auto-run CDs or installers. This feature affects the compiled publication's Vista manifest as it pertains to User Access Control (UAC). Versions of Windows prior to Vista will ignore this setting. According to Microsoft:

> *"User Account Control is a new security component in Windows Vista. UAC enables users to perform common tasks as non-administrators, called standard users in Windows Vista, and as administrators without having to switch users, log off, or use Run As. A standard user account is synonymous with a user account in Windows XP. User accounts that are members of the local Administrators group will run most applications as a standard user. By separating user and administrator functions while enabling productivity, UAC is an important enhancement for Windows Vista...*
>
> *"The primary difference between a standard user and an administrator in Windows Vista is the level of access the user has over core, protected areas of the computer. Administrators can change system state, turn off the firewall, configure security policy, install a service or a driver that affects every user on the computer, and install software for the entire computer. Standard users cannot perform these tasks and can only install per-user software."*

Clear as mud, right? **Fortunately, most VisualNEO for Windows authors can safely ignore this feature and simply use the default "As Invoker" setting.** For advanced authors who need to customize this setting, the available choices are described below:

None	The application will be compiled without a Vista manifest. Windows Vista (and h restricting access to sensitive areas of the file system and registry, while allowi access to these areas - this is called virtualization. (This option appears to wor
As Invoker	The application runs with the same access privileges as the parent process. Mic standard applications.
Highest Available	The application runs with the highest access privileges the current user can obt
Require Administrator	The application can only be used by users with administrator level access privile

Additional technical information about this topic is available from Microsoft.

Size / Colors

Use this screen to change your publication's size, color resolution and application type. You may select from among several standard publication sizes or enter a custom width and height. See New Publication for more information about screen size and colors.

VisualNEO for Windows's Compiler is capable of producing finished applications in four different formats. These include:

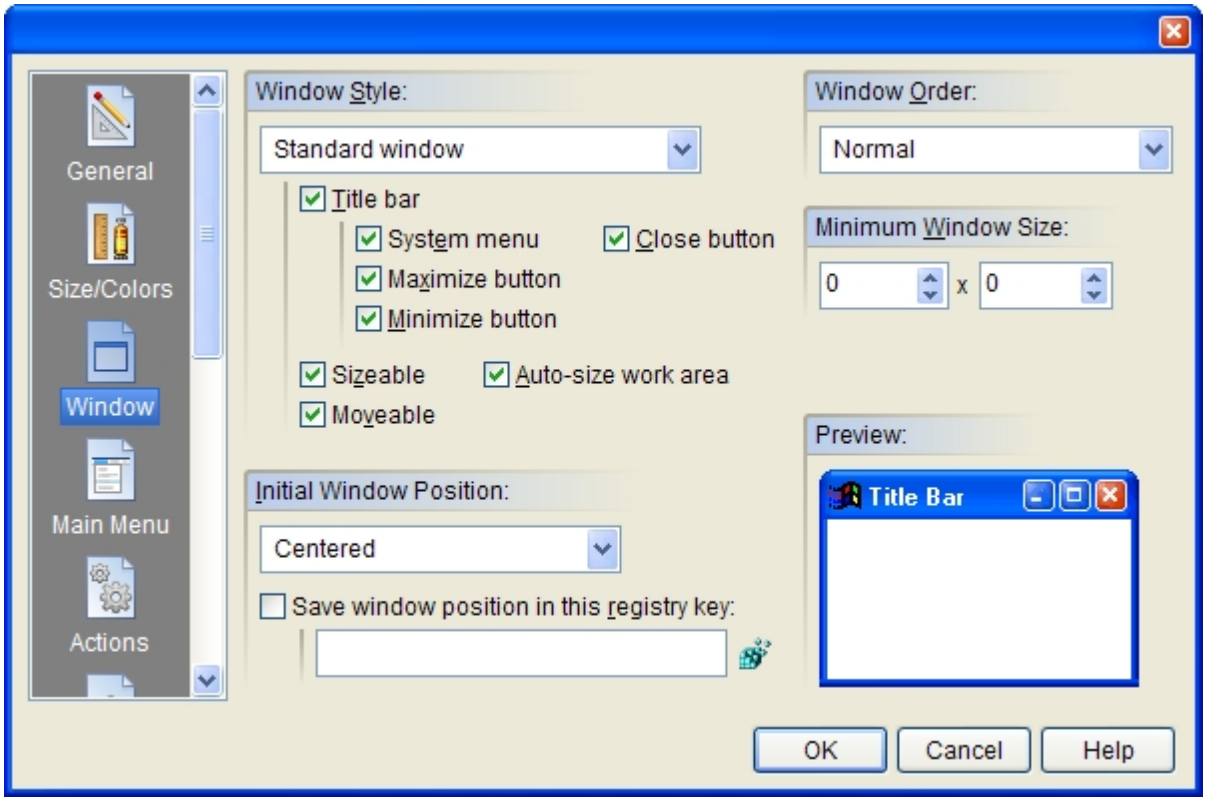
- Standard Application
- Screen Saver
- System Tray Application

- ActiveX Control

Each of these **Publication Types** have different properties, so which one you choose depends on the needs of your particular project. See Compile for more details about Publication Types.

Window

This section allows you to define how your publication will be presented to your readers. You can have your publication take over the entire screen, look like a standard Windows application, or define a custom shape of your own. As you make changes, the results will be reflected in the Preview box in the lower right corner of this screen.



A **Standard Window** emulates the look and feel of a traditional Windows program. You can specify whether your publication window includes a Title bar, Minimize/Maximize and Close buttons. You can specify whether your publication window can be sized or moved around the screen. Enable the **Auto-Size work area** option if you want to create a publication that will automatically resize itself to match the dimensions of the window. See Creating a Resizable Publication and the Container Tool topics for a discussion about the special techniques involved in creating sizeable publications.

A publication using the **Full Screen Window** option will occupy the entire screen. There will be no title bar or other recognizable Windows elements visible - only the contents of your publication. This option is best for presentations, slide shows, screen savers or any type of publication where you want to command the reader's full attention. If your publication size happens to be smaller than the reader's display, the unused border area will be painted with the **Border Color**. Otherwise, enable the **Auto-Size work area** option if you want to create a publication that will automatically resize itself to match the dimensions of the reader's screen.

For full screen publications, you also have the option of **Covering the Windows Task Bar** and physically switching the computer into a video mode that matches the dimensions and

colors you defined for your publication. The Task Bar appears at the bottom of your Windows desktop and is used to open and navigate your computer. Covering the Task Bar removes that distraction from your reader's view. Enabling the **Switch to Compatible Video Mode** option instructs VisualNEO for Windows to change the computer's display settings to match your publication's size and number of colors. When your publication is closed, VisualNEO for Windows will restore the original display settings. If the reader's computer hardware does not support the exact display mode required by your publication, VisualNEO for Windows will use the closest available mode.

VisualNEO for Windows authors also have the option of creating a completely unique **Custom Shaped Window** for their publications. This is done by creating a special image file that will serve as a mask and define the shape of your publication's window. Want a round window? Use your favorite paint program to draw a circle. Make the area outside the circle a different color than the circle's interior. Save the image and use the **Define Shape** button to import the file into VisualNEO for Windows. When asked to specify a transparent color, click the area outside the circle. You can enhance the effect further by drawing something interesting inside the circle and using that same image as a wallpaper background for each of your pages. To make your Custom Shaped Window moveable, check the **Moveable** option.

The **Initial Window Position** option can be used to define an area of the screen where you want your publication to appear when it first starts. If desired, you can also enable the **Save window position in this registry key** option to have VisualNEO for Windows automatically save the publication's screen position between sessions. In addition to enabling this option, you will need to enter a location (or key) in the Windows Registry where you want the window's position to be saved. The registry key entered by you should look similar to this:

```
HKEY_CURRENT_USER\Software\MyPub\WindowPos
```

Replace MyPub above with the name of your publication. Unless you are knowledgeable about the Windows Registry, you should not attempt to modify the above key other than to insert the name of your publication.

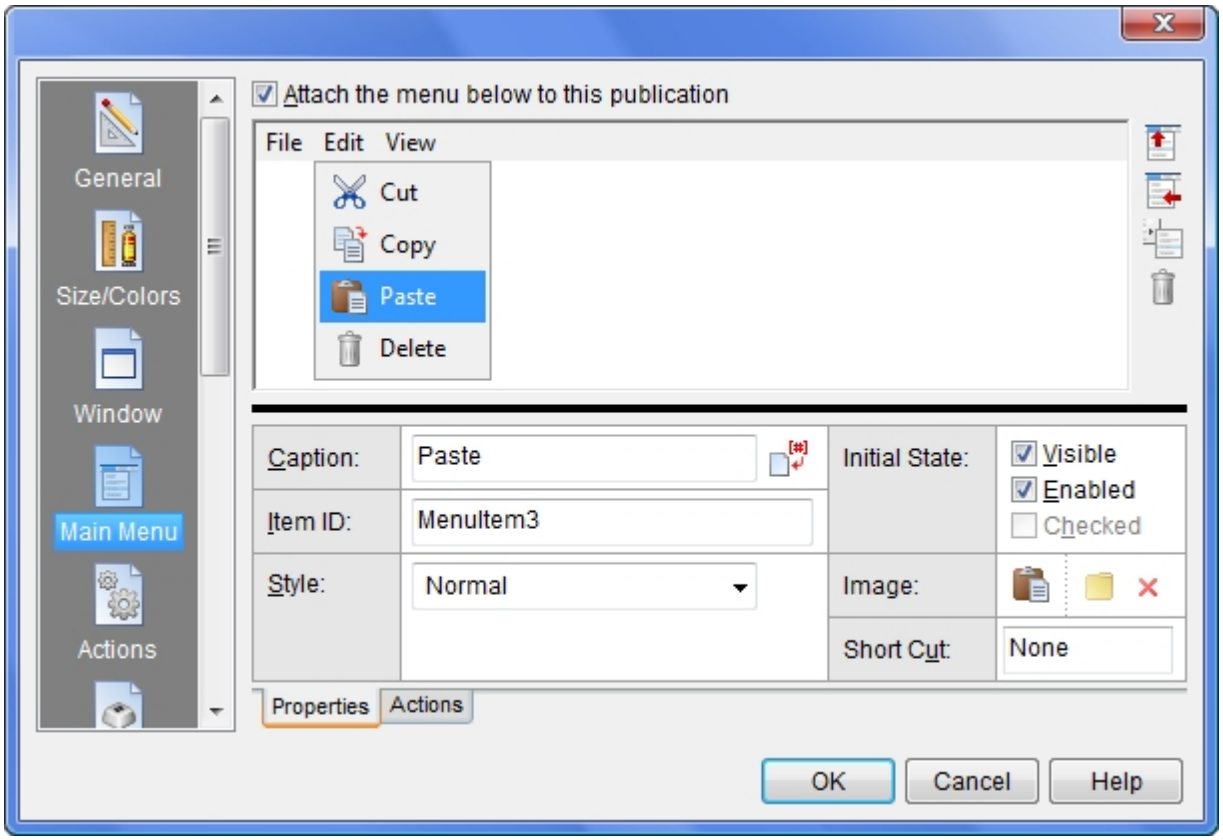
Note: *The Registry save option is not available for full screen publications.*

Use the **Window Order** if you want your publication to always Stay on Top or Beneath other running Windows programs. Most publications will leave this set to Normal, but some interesting tool bar applications can be created using the other two choices.

For standard windows with the sizeable option enabled, you can also specify a **Minimum Window Size** to prevent the reader from shrinking the publication below a certain size.

Main Menu

You can use the tools here to add a custom **Menu Bar** to your publication. The Menu Bar is a standard Windows component which appears at the top of many applications. Unless your publication already contains a menu, the Menu Designer Screen will be empty. Along the right side of the screen are four small buttons that can be used to add or remove items from your menu.



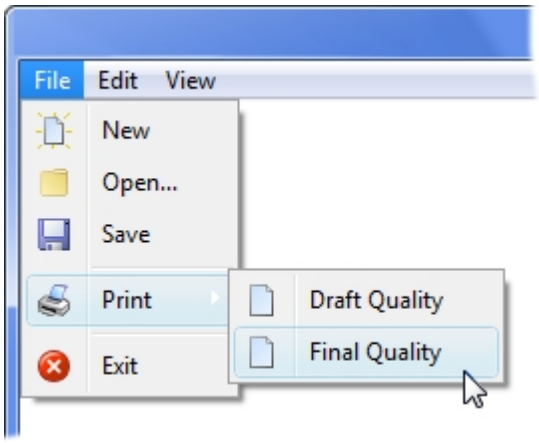
Items added to the menu will appear in the top portion of the Designer screen, and properties for the selected item in the lower portion. The tabs at the bottom of the Designer can be used to switch between the item's properties and its assigned Action commands. Assigned actions will be executed at runtime whenever the reader selects that menu item. The menu is organized in a hierarchical structure with top level heading items across the top and sub items under each heading.

To create a menu, start by adding one or more top level headings. For example, VisualNEO for Windows's menu includes the following menu headings: File, Edit, Arrange, View, Page, App, Options, Tools, Window and Help. Of course, your menu headings will be different depending on what your publication does.

To add a menu heading, click the **Add Menu Heading** button . VisualNEO for Windows will automatically assign the new heading a generic name like "MenuHeading1". Click the Caption property field and replace this with your own text. The Item ID property is used by VisualNEO for Windows to identify this menu heading. You can leave the ID as "MenuHeading1" or change it to something more descriptive, as long as no two menu items have the same name. Only the caption will be visible to readers of your publication.

To add an item below a heading, make sure the heading is highlighted and click the **Add Menu Item** button . A small box will appear below the heading. VisualNEO for Windows will assign the new item a generic name like "MenuItem1". As described above, the Caption and Item ID properties can be edited to replace "MenuItem1" with your own text.

You can create sub-menus by selecting a menu item and clicking the **Create Sub-menu** button . The picture below shows an example of a sub-menu:



With items and sub-items, you also have the option of selecting one of the following **Style** properties:

Normal	A standard menu item.
Check Box	A menu item that can be either checked or unchecked. When checked, a small check mark appears next to the item's text.
Divider	The item is a divider used to separate groups of items.

You also can specify whether headings and items are initially **Visible**, **Enabled** and (for Check Box items) **Checked**. All of these settings can be changed later while the publication is running using the ShowMenuItem, HideMenuItem, DisableMenuItem and EnableMenuItem actions.

An optional **Short Cut** key can be assigned to menu headings and items. When your publication is running, pressing the Short Cut key has the same affect as clicking the Menu Item - any Action commands assigned to the item will be executed.

When the **Check Box** style if selected, a **Variable** property is available for keeping track of the item's checked state while the publication is running. This Variable's name initially matches the Item ID, but can be changed if needed. When your publication is running, you can use the If Action to examine the variable and determine whether the menu item is checked or not. For example:

```
If "[MenuItem1]" "=" "Checked"
  AlertBox "Whoopee" "The item IS checked."
Else
  AlertBox "Boo Hoo" "The item is NOT checked."
EndIf
```

You can optionally add a small **Image** (sometimes called a glyph) to menu items. Click the button to add or replace an image. Click the button to remove an image. For best results, your menu images should all be the same size. Smaller images (20 x 20 pixels) work best, but menu images can be as large as 64 x 64 pixels. Transparent PNG images are recommended, but any image format supported by VisualNEO for Windows can be used. When importing non-transparent images, VisualNEO for Windows will prompt you to select a color to serve as the transparent portion of the image. (The menu's background will show through pixels in the image matching this color.)

Note: *When assigning a image to a menu item that has been designated a check box, the image will only appear when the item is checked. The image replaces the default Windows check mark graphic.*

Finally, you can define special actions to be taken whenever the reader clicks a Menu Item or Heading. Clicking on the **Actions** tab below the menu properties will display the Action Editor

and any commands assigned to the selected menu item.

Action commands may be typed directly into the editor, but most authors prefer to use the **Insert Action** button instead. Clicking the Insert Action button will display a list of available commands. The commands are organized into groups. Locate and click the title of the command you want. For most commands, VisualNEO for Windows will display a simple form that can be filled in to complete the Action. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Note: *Although it's unusual to assign actions to headings, advanced users may find it useful in certain situations. However, doing so may have unintended side effects, as these actions will be executed anytime an item below the heading is selected.*

Actions

If needed, you can define special Actions to be taken each time your publication starts, shuts down, is activated or deactivated, changes pages or after a period of inactivity. These actions can be used to access files, play animation or music, display messages, calculate scores, verify input, etc.

The Action page contains a large editor window and a tool bar. Action commands may be typed directly into the editor, but most authors prefer to use the **Insert Action** button instead. Clicking the Insert Action button will display a list of available commands. The commands are organized into groups. Locate and click the title of the command you want. For most commands, VisualNEO for Windows will display a simple form that can be filled in to complete the Action. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

Use the tabs below the editor window to switch between the different actions. If you do not have a need for a particular Action, leave it blank. The different actions are described below:

StartUp: Actions placed here execute when your publication is first launched. This is a good place to put any initialization code or functions your publication requires.

ShutDown: These actions will execute just before your publication closes. The ShutDown Action often is used to save any publication settings or information entered by readers that you want to retain for future use. Optionally, you can abort the shutdown by changing the value of the global [ShutdownStatus] variable. For example:

```
MessageBox "MyPub" "Want to quit?" "Yes|No" "[Result]"
If "[Result]" "=" "2"
  SetVar "[ShutdownStatus]" "Don't do it!"
EndIf
```

In addition to [ShutdownStatus], you can also examine the global [ShutdownSource] variable to determine what triggered the publication to close. [ShutdownSource] may contain one of the following:

NeoBook	The shutdown request was triggered by VisualNEO for Windows's Exit action.
Windows	The shutdown request originated with Windows. There are several Windows functions that could trigger a shutdown request, including: selecting "Turn off computer" from the Start Menu; the Task Manager's End Task command; or manually closing the application's icon from the System Tray. You should not normally refuse to close the publication when the requested to by Windows.
CloseButton	The user clicked on the publication window's close button, selected close from the system menu, or pressed Alt+F4.

For example, to minimize the publication instead of closing it when the source of the shutdown is the close button, do the following:

```
If "[ShutdownSource]" "=" "CloseButton"
  SetVar "[ShutdownStatus]" "False"
  SetVar "[WindowState]" "Minimized"
EndIf
```

Note: *It's certainly possible with this feature to create a publication that cannot be closed, so use it with caution. The words surrounded by square brackets are variables.*

Activate: These actions are executed each time your publication window becomes active, such as after using another program and switching back to VisualNEO for Windows.

Deactivate: Actions placed here are executed when your publication window becomes inactive, such as when the reader switches to another running program.

Page Change: These actions are executed whenever the reader attempts to navigate to another page. Optionally, the page change can be redirected or canceled by altering the value of the global [PageChangeName] variable. To redirect the reader to a different page, insert the new page's title into the variable. For example:

```
SetVar "[PageChangeName]" "ErrorPage"
```

You can also cancel the page change by setting the variable to any empty title. For example:

```
If "[PageChangeName]" "=" "Goodies"
  If "[User]" "<>" "Register User"
    AlertBox "Sorry" "Only registered users can view that page."
    SetVar "[PageChangeName]" ""
  EndIf
EndIf
```

Idle Event: These actions are executed if no keyboard or mouse activity has been detected during the time interval specified. Enter the time interval (in seconds) that the system must remain idle before the Action is triggered.

For a kiosk publication, you could use the Idle Event to automatically return to a main page after a specified period of inactivity. This would prevent new visitors from becoming confused by a presentation left open in the middle by a previous reader. For example:

```
GotoPage "Start"
```

Subroutines: You can think of this as a repository for often used groups of actions. Instead of typing the same commands in multiple places, you can enter them once here and then reference them using the GoSub Action. Command blocks, or subroutines, begin with a unique descriptive label and end with the special command "Return". For example:

```
:SubroutineOne
  AlertBox "Hello" "This is my subroutine"
Return

:OtherSubroutine
  AlertBox "Hello" "This is my other subroutine."
Return
```

Unlike the other actions in this section, subroutines are not executed automatically in response to some specific event. Instead, subroutines must be executed explicitly with the GoSub Action. For example, to execute "SubroutineOne" above, we would execute the

following Action somewhere else:

```
GoSub "SubroutineOne"
```

Resized: Actions placed here are executed when the reader manually changes the width or height of your publication's main window. Under most circumstances, the Resized action should execute only once at the end of the resizing process. The Resized action can only be used when the publication's Window Style is set to the "Standard" option.

Access

This section allows you to define which keys your readers may use to navigate through your publication. You may wish to activate some of these key options for those users who do not have a mouse, or simply for the convenience of your readers. The Keyboard Control options are described below:

Allow ESC key to exit publication: Disable this option to prevent your readers from closing your publication by pressing the Esc key. If you disable this feature, be sure to provide your readers with another way to shut down your publication, such as an exit button.

Allow Page Up, Page Down, Home and End keys to change pages: Enable this option to allow your readers to move about your publication using these keys. In some situations, you may want to disable these keys in order to control which pages readers can visit and in what order. Many authors choose to disable these keys and instead provide a set of navigation buttons. Unlike the keyboard method, navigation buttons can include Action commands to intelligently determine if changing pages is OK. For example, a publication designed to administer a classroom quiz could make sure that an answer has been selected before allowing the student to move on to the next topic.

Display visible marquee to indicate item with keyboard focus: For publications that include text entry fields, check boxes and other input devices, enabling this option will allow the reader to access these items by pressing the keyboard's Tab key. The active object will be surrounded by a rectangle, or marquee, indicating that it has the input focus. See Arrange > Set Tab Order to specify the order in which objects will be selected.

Disable Ctrl+Alt+Del and Alt+Tab keys: This option allows you to prevent readers from switching to other programs (Alt+Tab) or using the Crtl+Alt+Del key combination to shut down your publication. This option is intended only for use with special types of publications, like kiosks, where it can be disastrous to allow the reader unrestricted access to the computer. It's generally unwise for normal Windows programs to disallow access to these functions. This feature is not available in Windows Vista.

Disable Print Screen key: This option will prevent readers from using the Print Screen key to copy images from your publication's screen to the Windows clipboard. There are other methods that can be used to copy screen images, so this option will not prevent someone intent on stealing images. The Print Screen key is the most common method of capturing screen images, so disabling it is helpful.

Note: *It's not possible to disable both Ctrl+Alt+Del and Print Screen under Windows 95, 98 or ME. If both these options are selected when running this publication under Windows 95, 98 or ME, then only Print Screen will be disabled. This problem does not affect Windows NT, 2000 or XP.*

Prevent multiple copies of this publication from running at the same time: Enable this option to prevent more than one copy of your publication from running at the same time. This is a good idea for most publications, since multiple instances of the same program waste system resources and can confuse users.

You can detect when a user attempts to launch a second instance of your publication by creating a special subroutine called **CommandLine_OnChange**. VisualNEO for Windows will automatically execute this subroutine whenever the user attempts to start more than one copy of your publication. For example:

```
:CommandLine_OnChange
  AlertBox "Warning" "An attempt was made to launch a second copy of this
app."
Return
```

The CommandLine_OnChange subroutine is particularly useful if your publication uses the global [CommandLine] variable to load external files at startup. Prior to executing CommandLine_OnChange, VisualNEO for Windows will automatically update the contents of the [CommandLine] variable with the command line of the second instance. This is useful if you have registered a file type association with Windows, which allows users to start your publication by double clicking on a file matching the association. Windows passes the name of clicked file to your publication via the command line which you can intercept using the [CommandLine] variable.

See App Properties > Actions for more information on creating subroutines.

Note: *The CommandLine_OnChange subroutine will not be executed unless the Prevent Multiple Copies... option is enabled.*

Disabling the **Display error messages** option allows advanced VisualNEO for Windows authors to trap and respond to errors programmatically rather than having VisualNEO for Windows display the error in a dialog box. When this option is off, all error messages are placed in the global [LastError] variable. You can use this variable in your Action scripts to determine when an error occurs and respond appropriately. For example:

```
FileWrite "parts.dat" "Append" "[PartNum], [Desc], [Price]"
If "[LastError]" ">" ""
  AlertBox "Error" "Unable to write data to Parts.dat!"
EndIf
```

You can turn the error display on and off programmatically using the ShowErrors Action.

Turning off the **Display the Windows print setup screen for all print actions** will disable the Print Setup screen, causing actions that use the printer (PrintPage, PrintTextFile, etc.) to begin printing without first asking for confirmation. When this feature is off, the Print Setup screen can still be accessed manually using the PrintSetup Action. For example:

```
PrintSetup "[Result]"
If "[Result]" "=" "True"
  PrintPage "Contents"
  PrintPage "Help"
EndIf
```

Enabling the **Automatically update objects containing time sensitive variables** option instructs VisualNEO for Windows to update the screen at regular intervals. This is useful if your publication contains objects which display the current time or date. If this option is turned off, these objects will only be updated when entering or leaving a page.

The **Image Cache Size** may be adjusted to optimize the performance of publications that contain a large number of images. For publications with many small images, you may be able to improve performance by increasing the size of the cache. For publications that consist primarily of very large images, performance can be improved by reducing the cache size. Incorrectly adjusting the cache size can have the opposite effect and actually degrade performance, especially on older PCs. For this reason we recommend that you leave the cache size set at the default value of 30 unless you're having performance problems.

Security

You may also require readers to correctly enter a **Password** before entering or exiting your publication. Check the appropriate boxes to enable these features and type the passwords in the fields to the right. This is useful to prevent unauthorized users from accessing or shutting down a publication. An example of how this might be useful is for a kiosk or network utility in which you do not want unauthorized users to exit the publication.

Language

If you are creating a publication in a language other than English, this screen allows you to translate error messages and prompts generated by VisualNEO for Windows. Enter your translations (on the right) in place of their English equivalents (on the left). Click the **Save** button to export the translated messages to a file. Click the **Load** button to load translations which you've previously saved. Click the **Reset** button to restore English text. Check the **Default** option if you want to use the current settings each time you load VisualNEO for Windows.

Use the **Bidirectional Language Mode** option to control the direction in which text appears (left-to-right or right-to-left), the placement of vertical scroll bars and the default alignment of text when your publication runs under Middle Eastern versions of Windows. Unless you are designing a publication for this locale, the language direction should be set to "Left to Right". (This option has no effect when used with non-Middle Eastern versions of Windows.)

Note: *For languages that require special character sets, (Arabic, Japanese, etc.), you may also need to change the font using the Custom Font setting.*

Interface

This screen allows you to customize the appearance of dialog boxes displayed by your publication. Dialog boxes are used to request information or to inform the reader that an error has occurred. You may select a custom image to be displayed inside VisualNEO for Window's AlertBox, MessageBox and Exit dialog boxes. Leave the image fields blank to use VisualNEO for Window's default dialog box images.

Use the **Custom Font** button to select a font face and size for the text within the dialog boxes. This font also will be used by any popup Menus or StickyNotes generated by the publication.

Screen Saver

These features are useful when you intend to compile your finished publication as a Windows Screen Saver.

The **Control Panel Preview Image** is a small, static BMP format picture that will appear in the Windows Control Panel when selecting from installed screen savers. The Control Panel Image should be 152 x 112 pixels, 16 or 256 colors for best results.

Enable the **Mouse and keyboard events exit publication** option to make your screen saver exit when the user moves the mouse or presses a key - just like other screen savers do. If this feature is turned off, your screen saver only will exit when the Esc key is pressed, or if

you provide a button (or other object) that executes the Exit Action.

Tray Menu

If you choose to compile you publication as a System Tray Application, this screen can be used to create the popup menu that appears when you right click on your publication's tray icon. (See the Compile for more information on System Tray Applications.)

The Tray Menu Designer works just like the Main Menu Designer, except that tray menus do not have headings. Tray menus are composed entirely of items and sub items. For information on using the Menu Designer see Main Menu.

Subroutines

The Subroutines option provides quick access to the Subroutine Action page of the App Properties screen. The submenu displays a list of subroutines created by you. Clicking one of the items will take you directly to that section of the subroutine code. You also can jump directly to the subroutine page by pressing the F12 key.

Embedded Files

The Embedded Files option allows you to create a list of additional files to be compiled inside your pub EXE. Use the **Add** and **Remove** buttons on the right to manage the file list.

You can use embedded files most places in your publication just as you would in a normal disk-based file. Simply replace the file's original path with the special embedded file variable. For example:

```
[Embedded]Sample.jpg
```

When referencing an embedded file, the file's originalpath should be replaced with the special [Embedded] variable.

Note: *The Embedded Files option is intended to be used primarily by advanced VisualNEO for Windows developers. Novice and intermediate users will rarely need to use this option since VisualNEO for Windows automatically takes care of embedding most types of files contained in a publication.*

Variables

This option can be used to display a list of all the variables used by your publication. The two tabs at the top of the screen (**Author Defined** and **Global**) allow you to switch between lists of variables created by you (the author) and ones created automatically by VisualNEO for Windows (global). If you've modified your publication, pressing the **Refresh** button will scan the publication and make sure that the list is up to date.

 The Variable List can also be accessed from many of VisualNEO for Window's property screens by clicking on the **Insert Variable** button. Then, a specific variable can be selected from the list and inserted into the property field associated with the button.

Page Layout

Use this command to display an overview of the pages in your publication. Pages in the current publication are displayed as small thumbnail images on the left side of the screen. Click on a page to select it. If you check the **Show Images** box, images within each page also will be displayed (though this may be slow on some systems).

You can modify the publication using the Add, Copy, Move, Rename, Delete and Properties buttons. (These commands are identical to those in the Page menu.) The two Arrow buttons allow you to move the selected page up or down in the page order. You can also use the mouse to drag individual pages within the list.

Archive / Backup

Use this command to copy all of the source files used in the current publication to another folder or drive. Just specify the destination folder, or click on the folder icon on the right, to select from existing folders on your computer. If the folder you specify does not yet exist, VisualNEO for Windows will create it before copying the files.

Aside from being a great way to backup your work, this is a fast way to gather all the files used in your publication into one spot on your system.

File List

This command displays a list of all the source files used to build the current publication. It's a quick way to locate the name of a file, print out a record of the files you've used, or look into how a sample publication was constructed. Use the Print button to send the information to your printer.

Run Options

Test Parameters

This option allows you to specify parameters to be passed to your publication when running in test mode using one of the Run commands. The parameters are passed to the publication via the Windows command line. Windows applications sometimes use the command line as a way of accepting input from the user. For example, when you double click on a .doc file in Explorer and Microsoft™ Word appears, Windows uses the command line to pass the file's name to Word.

A running publication can access the contents of the command line by examining the global [CommandLine] variable. The script below parses the [CommandLine] variable and separates the different parameters into variables [Param1], [Param2], [Param3], etc. The first variable ([Param1]) will always contain the path and file name of the publication. The variables after that ([Param2], [Param3], etc.) will contain any other command line items. The following script separates the command line elements and then displays a message for each one:

```
StrParse "[CommandLine]" "[#13]" "[Param]" "[Count]"
Loop "1" "[Count]" "[X]"
  AlertBox "Result" "Parameter [X] is [Param[X]]"
EndLoop
```

Debug Options

When the **Keep a log of executed actions** option is enabled, the debugger will record

everything that happens while your publication is running in test mode. Use the default log file name or enter one of your own in the space provided. After running your publication, using one of the App menu's Run options, view the log file in Windows Notepad or similar editor. The log file can be very useful for debugging complex scripts.

Note: *These options are for testing purposes only and have no effect when running compiled publications outside of VisualNEO for Windows.*

Run

Use these commands to test drive your publication. In test mode, you can preview your publication from the reader's point of view. Buttons, check boxes, animation, sound and other elements will function just as they would in a compiled publication. Select **Run (this Page only)** to test the current page only. Use **Run (from Start)** to test the entire publication from the beginning (page one). Similarly, **Run (from this page)** can be used to test the entire publication starting execution on the current page.

Both **Run (From Start)** and **Run (from this page)** will execute the Startup and Shutdown Actions (App Properties > Actions) assigned to the publication which **Run (this Page only)** does not.

To assist you in tracking down potential problems with your publication, VisualNEO for Windows includes a handy **Debugger** that is displayed automatically whenever you enter test mode. The Debugger allows you to monitor variables and Action commands as they are processed by VisualNEO for Windows.

Compile / Publish

When your publication is complete and ready for distribution, use this option to package it into a stand-alone executable application. This process is called compiling. The **Compile/Publish** screen is divided into five sections indicated by the icon images on the left:

General
Files
Fonts
Advanced
Setup

To view the settings for a section, click the corresponding icon. Once you are satisfied with these settings, click the **Compile** button to build a stand-alone application.

Command Line Compiler Option (Advanced)

For advanced users, VisualNEO for Windows also includes a Command Line Compiler option. This can be useful in certain situations, provided that you do not need to make any changes to the publication or the compiler settings prior to compiling. To use the command line compiler, simply start VisualNEO for Windows with the /C switch followed by the full path and file name of the publication to compile. For example:

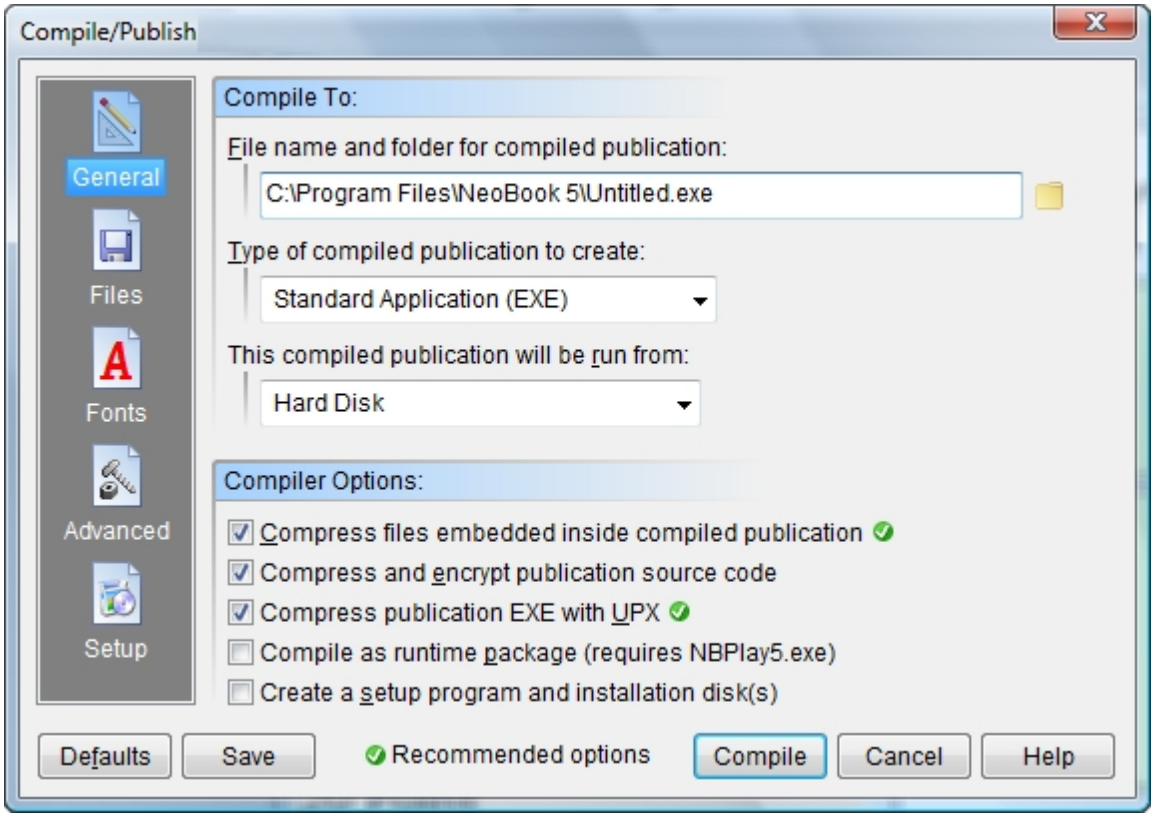
```
visualneo.exe /C "C:\MyDocuments\VisualNEO for Windows\Sample Apps\Quick Tour\Quick Tour.pub"
```

The command line compiler will only accept one publication file at a time. The publication will be compiled using the publication's last saved compiler configuration. When the compile is complete, VisualNEO for Windows will exit. Make sure VisualNEO for Windows is not already

running before using the command line compile option.

General

Enter a folder and file name for your finished application in the **Compile To** field. By default, this field contains the same base name as the publication's uncompiled PUB file. You may enter a different name here if you like. The file name's extension depends on what type of application you decide to create.



VisualNEO for Windows's Compiler is capable of producing finished applications in four different formats described below:

Standard Application (EXE)

A Standard Application closely resembles a typical Windows program and can be displayed in a moveable window or use the entire screen, depending on the setting you select under App Properties. This is the most common type of application produced with VisualNEO for Windows.

Screen Saver (SCR)

A screen saver is a special program that is designed to run after your computer has been idle for a specified period of time. Screen savers can reduce the wear and tear on your monitor and provide basic security for an unattended computer. Effective screen savers generally include a continuously changing display of graphic images or animation.

When compiling a screen saver, VisualNEO for Windows automatically enables the Esc key to allow users to exit and return to Windows. In addition, screen savers always run in Full Screen mode, so use of other window settings in the App Properties screen will be ignored. The start and exit password options are also disabled, since Windows already includes a password feature especially for screen savers.

If you want your screen saver to have a configuration option that can be accessed from the Windows Control Panel, create a page titled "Config." On this page, place controls for adjusting your screen saver's properties. Use VisualNEO for Windows's File or Registry actions to save changes to your settings. Config must not be the publications first page.

Once compiled, copy your screen saver's SCR file to the Windows or Windows\System32 folder and access it from the Windows Control Panel.

Hint: *Most newer monitors no longer require a screen saver for protection against screen burn-in. Screen savers are now generally used for entertainment or promotional purposes.*

System Tray Application (EXE)

A system tray application works exactly like a Standard Application, except that it appears as an icon in the system tray (usually located in the extreme lower right of your Windows desktop).

Normally, when a compiled system tray application is launched, it will automatically appear as an icon in the tray. If you prefer to have your tray application open in a window instead, you can place the following code in the App Properties Startup Action:

```
SetVar "[StartInSystemTray]" "False"
```

Once open, minimizing the window will send it to the system tray. Remove this code to start the publication normally as an icon in the system tray.

See also, App Properties > Tray Menu.

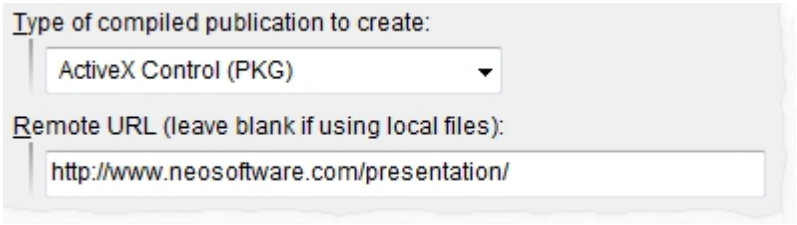
ActiveX Control (PKG)

This compiler options allows programmers to incorporate compiled VisualNEO for Windows publications into their applications. VisualNEO for Windows's NB5ActiveX.ocx control can be installed as a component in other programming environments that support ActiveX controls, such as Delphi, Visual Basic, C++, etc. Please see the ActiveX Programmer's Guide topic for more information.

Additionally, a publication compiled to this format can also be viewed with Microsoft's Internet Explorer browser*. When compiling as an ActiveX control, a sample HTM (HTML) file is created in addition to a special package (PKG) data file. The sample HTM file is very basic, but you can modify it or cut and paste the VisualNEO for Windows code into another file. Open the HTM file in Internet Explorer to display your VisualNEO for Windows publication.

Due to increased security concerns, ActiveX controls are not recommended for web sites

that will be viewed by the general public. However, advanced users can compile publications using the ActiveX option for limited distribution over the Internet or corporate intranets. To compile for distribution over the Internet, enter your web server's URL address in the field provided in VisualNEO for Windows's compiler (below). After compiling, you will need to upload the *.HTM, *.PKG and VisualNEO for Windows's NB5ActiveX.ocx file to your server. NB5ActiveX.ocx is the viewer required to display the compiled publication inside Internet Explorer.



For added security, VisualNEO for Windows's NB5ActiveX.ocx control will prompt users before downloading a compiled publication pkg file for the first time. The prompt alerts users to the potential risk and allows them to OK or Cancel the download. The warning only appears when downloading a new pkg file over the Internet. Pkg files that are already installed on the system will not generate a warning.

***Note:** In order to have VisualNEO for Windows's NB5ActiveX.ocx control function properly, you must have the security settings in Internet Explorer set to allow unsigned ActiveX controls to download and run. If the ActiveX feature is disabled, the control will not function and your publication will not be visible. Currently, Internet Explorer is the only browser with built-in support for ActiveX controls.*

Enabling **Compress files embedded inside compiled publication** will instruct VisualNEO for Windows's compiler to attempt to reduce the size of the text, pictures, sound, video and other elements used in your publication. Compressing these files can greatly reduce the overall size of your finished product. However, in some situations, this can also slow down the execution of your publication, since files must be uncompressed before they can be displayed.

Similarly, the **Compress and encrypt publication source code** option will shrink the size of your final product by compressing the instructions that describe the contents and behavior of your publication. This option also encrypts the "source code" to inhibit anyone who might try to hack your publication.

The **Compress publication EXE with UPX** option will greatly reduce the size of your finished application by compressing it with the UPX utility. UPX is a high-performance executable packer for Windows (and other operating systems). It provides excellent compression with no memory overhead or other performance drawbacks. The only reason not to use this option is if you want to compress your EXE using an alternative utility or copy protection product. *(UPX, Copyright (c)1996-2011 Markus Oberhumer, Laszlo Molnar & John Reiser. Additional information about UPX can be found at: http://upx.sourceforge.net)*

The **Compile as Runtime Package** option essentially compiles your publication without any of VisualNEO for Windows's brains. The result is a much smaller file, but one that must be distributed with VisualNEO for Windows's Runtime Player (NBPlay5.exe) in order to function. On the reader's computer, NBPlay5.exe can be installed in the same folder as the package EXE or in the Windows, System or Temp folders. The Compile as Runtime Package option is not available when compiling a publication as a Web Browser Plug-In.

If you are distributing a group of compiled publications, using this feature can greatly reduce the overall size of your product. You can also use this feature for publications that require frequent updates. The cost of distributing updates can be reduced, because you only need to send your customers the new package rather than the entire program.

Hint: *Like VisualNEO for Windows, the Runtime Player is assigned an internal version number. For example, if you are using VisualNEO for Windows 5.0.0, then the runtime player's version number also will be 5.0.0. The version number of the Runtime Player must be the same or higher than the version number of VisualNEO for Windows used to compile the package, or the publication will not run.*

Enable the **Create setup program and installation disk(s)** option if you want VisualNEO for Windows to create a special utility program that your readers can use to install your publication on their computer. In addition to being a convenience, this option also allows you to span very large publications across more than one diskette. See Setup for more information on implementing an installation utility.

Files

This screen gives you the option of storing multimedia files inside the final compiled EXE or leaving these files outside the EXE. There are several types of multimedia files that must be made visible to Windows before they can be played. These files types are AVI, MOV, MPG, WAV, MID and MP3. Normally, when a publication needs to play one of these files, it first must be extracted temporarily from the compiled EXE to the reader's hard drive. If the file is particularly large, there may be a slight delay before the file can be played.

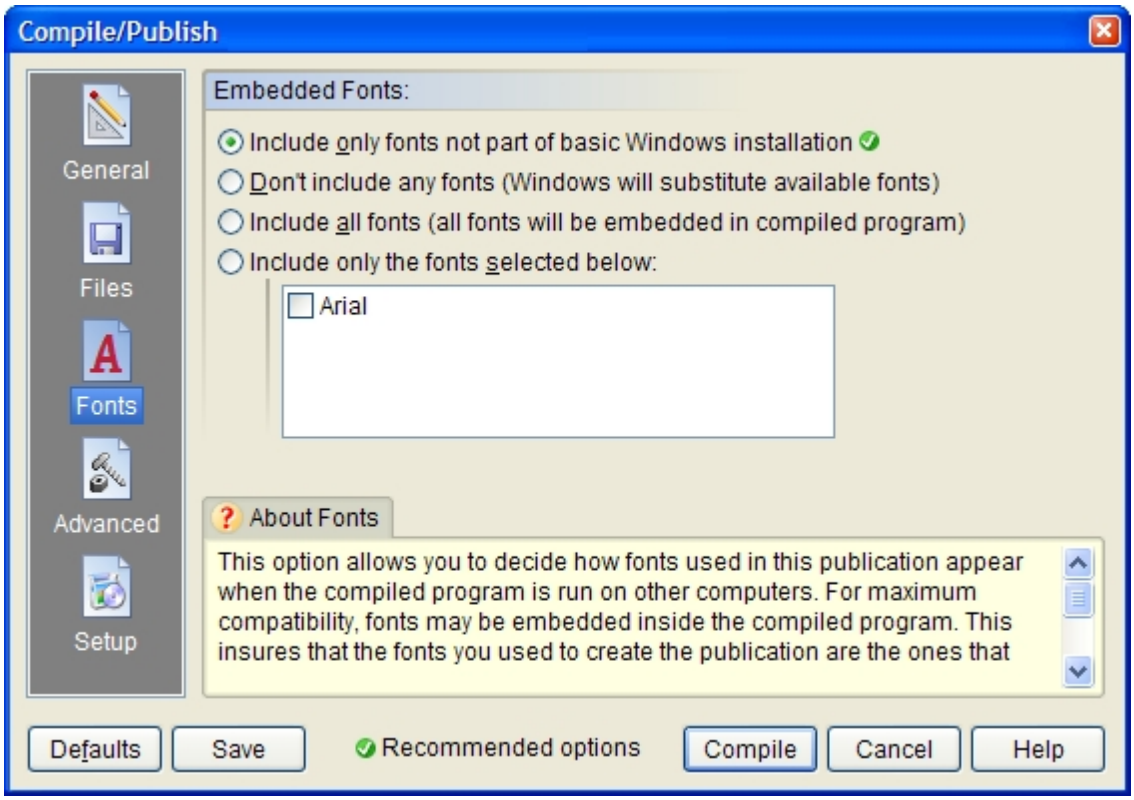
The advantage of compiling all files inside the publication is that it makes it much more difficult for someone to modify or steal elements of your publication. Also, many types of files compiled inside an EXE can be compressed, reducing the disk space required by your program. The advantage of leaving multimedia files outside the compiled publication is that they may play faster when your publication is viewed on drives with slow access speeds like older CD-ROMs. Also, external files won't take up space on your reader's hard drive.

If you select the **Compile only files that do NOT need to be extracted** option, you will be given an opportunity to specify a separate folder to place copies of these files. We recommend that you use an empty folder for this option to avoid including files not needed by the compiled publication. When distributing your publication, you must include these files in the same folder as your compiled EXE. However, if you choose to have VisualNEO for Windows create a Setup program for your publication, then you do not need to specify this folder. VisualNEO for Windows include any required external files in the setup automatically.

Note: *Any files extracted by your publication will be removed upon shutdown. See Advanced Compiler Settings.*

Fonts

Something most people don't realize is that many fonts are copyrighted and cannot be freely distributed without the permission of their authors or publishers. Some vendors may not allow their copyrighted fonts to be used within electronic publications (even though these fonts are not accessible to the reader). Since vendor policies vary, and since they can affect the size of your final publication, we have included several options for handling fonts. When fonts are managed correctly, your publication should look the same no matter where it's displayed.



Hint: *There are many free fonts available on the Internet which you may distribute with your publications. Consult your Windows User Guide for information on installing new fonts on your computer.*

The **Include only fonts not part of basic Windows installation** option only compiles fonts that are not included with the standard Windows installation. The advantage of this option is that your publication may be smaller (depending on whether or not you used any of the basic Windows fonts, since most readers will already have them). The disadvantage is that, if the reader has removed some of the basic Windows fonts, your text may not appear exactly as you intended. Also, some foreign language editions of Windows may include different fonts than those found in the English version.

You should only use the **Don't include any fonts** option with publications containing little or no text. The advantage of this option is that your publication will be smaller and load a bit faster. The disadvantage is that you will have no control over the look of the fonts in your publication, as it will be up to Windows to choose substitutes for any missing fonts. Windows does a notoriously poor job of selecting substitute fonts. In some cases, substitute fonts might contain symbols rather than letters, rendering your publication unreadable.

The **Include all fonts** option will compile all of the fonts used in your publication. The advantage of this option is that your fonts will display exactly as you intended. The disadvantage is that your compiled program will be larger (and if you use lots of fonts, it can be much larger). Disadvantages aside, this is the best option if you want your publication to appear exactly as it does on your computer.

The **Include selected fonts** option allows you to choose which fonts to compile inside the publication. Simply click the small box next to the fonts you want to include. A check mark will appear in the box next to the fonts to be included. Click the font a second time to remove the check mark. The advantage is that you can choose to include fonts used for prominent text, while allowing VisualNEO for Windows to substitute other fonts for all other text. The disadvantage is that some text may not appear exactly as you intended.

Advanced

As with most Windows applications, compiled publications will sometimes need to temporarily store files on the reader's hard drive. Windows provides a Temp folder for this purpose, but VisualNEO for Windows authors may store these files in another location if desired. Temporary files may be stored in one of the following locations:

The **Windows temporary folder** is the recommended location for temporary files. This folder is used to store temporary files by Windows and most Windows applications. This is the preferred option when your publication is intended to be placed on a CD-ROM, DVD-ROM or other write-only media.

Selecting the **Windows folder** option places temporary files into the same folder as Windows. This is useful if running from a network, as users should always have read/write access to this folder.

The **Same folder as compiled publication** option places temporary files into the same folder as the publication EXE. This option works well if you are certain your publication always will be placed in a folder that has read/write access. Also, temporary files stored here can be easily identified and cleaned up should something go wrong (like a power failure) while a publication is running. If your publication will be used on a network/intranet or accessed from CD-ROM or other read-only media, you cannot use this option. If VisualNEO for Windows is unable to write to this folder, your publication will not run properly.

The **Custom folder** option allows you to specify another folder for storing temporary files. It is important that this directory name be valid for the end-user's system or your publication may not execute correctly. This directory will be created if it does not exist. Use this option with care - many network/intranet users are not permitted to create folders without permission. If VisualNEO for Windows is unable to create the custom folder, your publication will not run properly.

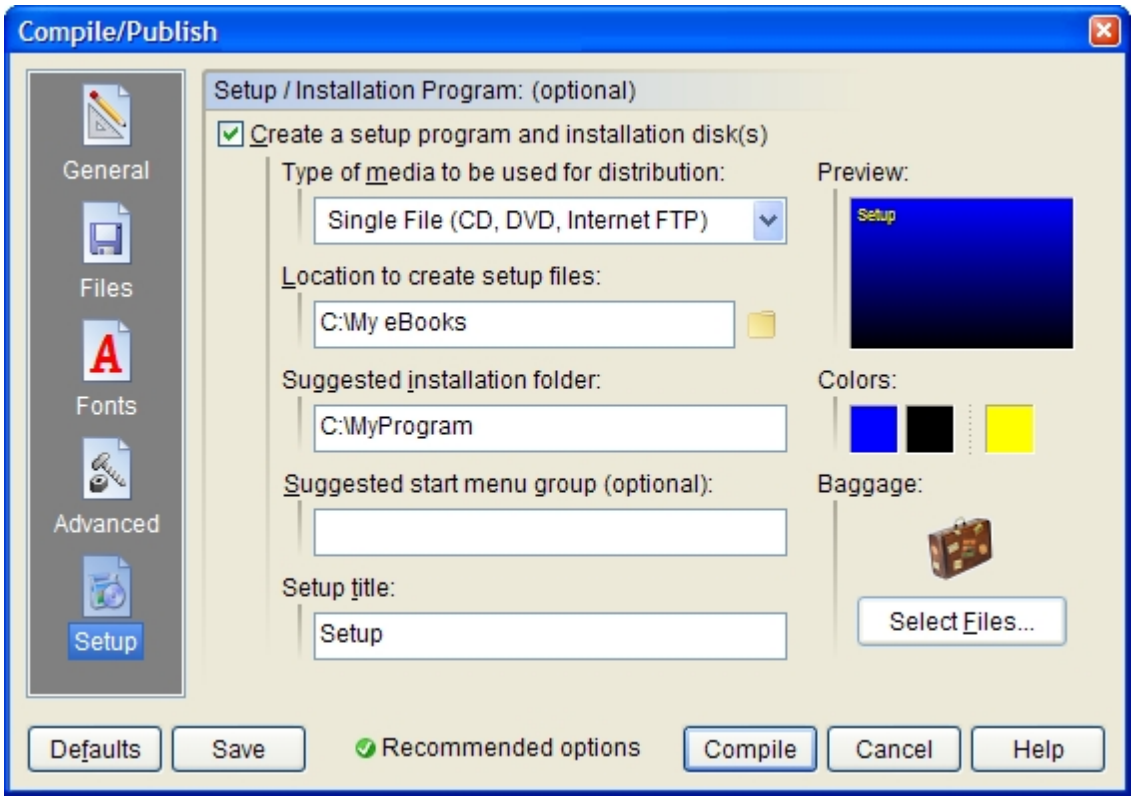
Enable the **Hide extracted files** option to mark extracted files as hidden. This will render the files invisible to most Windows programs and prevent many computer users from tampering with the files.

The **Remove extracted files when publication terminates** option should be checked, unless you want to leave the extracted files on the hard drive after your publication terminates. We strongly recommend that this option be enabled, unless you have a specific reason for leaving these files on the user's system. Also, never use the Hide Extracted Files option if you don't intend to remove the files.

Hint: *If your publication created a Custom Folder (see above), it also will be removed if this option is enabled.*

Setup

Enable the **Create setup program and installation disk(s)** option to create a special utility program that your readers can use to install your publication on their computer. Most Windows applications that are designed to be run from the hard drive include a simple SETUP.EXE program to assist the user with this task. A setup program creates a folder and copies the application's files from a compressed archive into the folder. Most setup programs also add an application icon/shortcut to the Windows Start Menu. VisualNEO for Windows's simple setup utility can do this for your publication. The options you select here will determine how your setup program will function.



Make a selection from the **Type of media to be used for distribution** list to specify how your setup files will be packaged. Available options include Single File or 1.44MB, 720K and 1.2MB diskettes. The single file option is preferred if you intend to distribute your publication on CD-ROM or over the Internet. Select the appropriate diskette type if you need your setup program broken down into multiple disk-sized pieces.

In the **Location to create setup files** field, enter the name of a folder on your hard drive where you want VisualNEO for Windows to create the setup files. When creating a diskette-based setup, you can specify a drive letter here followed by a colon (A:, B:, etc.) to have VisualNEO for Windows write directly to the diskettes. You must have a sufficient number of blank, formatted diskettes ready beforehand. If you specify a folder on your hard drive, VisualNEO for Windows will create subfolders called DISK1, DISK2, etc. - one for each diskette required. Then you can manually copy the contents of each folder to its own diskette. For single file installations, VisualNEO for Windows will create a single SETUP.EXE file containing everything needed to install your publication.

Note: *You can NOT use this option to write directly to a CD-ROM. Most CDR drives require special software for writing files onto CD-ROMs. VisualNEO for Windows will prepare the publication file, which you may then place on the media using the appropriate software. See your CDR manual for instructions.*

The **Suggested installation folder** should contain the name of the directory where you recommend your publication be installed on the reader's computer. For example: "C:\MyProgram." During the installation, the reader can accept the this folder or specify one of their own.

The name entered in the **Suggested start menu group** field will be used as the default heading for your publication in the Windows Start Menu. The reader will have the opportunity to specify a different menu heading during the installation. Leave this field blank if you do not wish to have your publication listed in the Start Menu.

The **Setup title** appears at the top of the Setup program's screen. For example, "My Excellent Publication Setup."

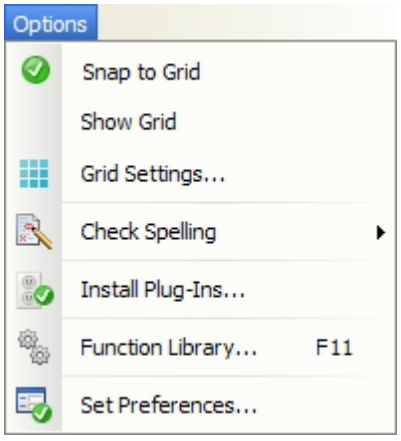
On the right side of the screen, you will see a small preview window that shows what the background of your Setup program will look like. Just below the preview window, there are three small buttons that control the background gradient colors (top and bottom) and the text color of your Setup screen. Click the buttons to select a different color for each of these items. Your changes will be reflected in the preview. For a solid background color, select the same color from both the top and bottom gradient buttons.

Hint: *You can translate the messages and prompts displayed by your Setup program by selecting the Language option from the App Properties screen.*

The final option on the Setup screen allows you to specify extra files to be transported along with your publication setup. These extra files are called **Baggage**. Files specified here can be just about anything from external programs to database files. This can be useful if you need to include additional files with your publication that are not handled by VisualNEO for Windows's compiler. The Setup program will deposit baggage files in the same folder as the publication EXE.

The Options Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Options menu. For more information, select a command from the picture below:



Snap to Grid

Enabling this option causes VisualNEO for Windows's grid to magnetically attract objects as they are being drawn, moved or resized. See Show Grid and Grid Settings.

Show Grid

Enabling this option will place a grid over the editing window, making it easier to align objects on the screen. The grid will not appear in the compiled publication.

Hint: *The grid becomes magnetic (objects are drawn to the nearest grid point) when Snap to Grid is enabled. Grid spacing can be set using Grid Settings.*

Grid Settings

This feature allows you to specify the horizontal and vertical grid spacing (in screen pixels). When Snap to Grid is enabled, tools and objects will be magnetically attracted to the nearest

point on the grid, helping you line up objects when drawing, resizing or moving. Enable the Show Grid option to make the grid visible.

Check Spelling

This function checks your text and allows you to identify and correct misspelled words. You can choose to check the spelling of the Entire Publication, or limit checking to the current page or just the selected objects on the current page. All objects and Action scripts containing valid text within the limits specified will be checked.

Misspelled or unrecognized words will be displayed in the **Unknown Word** field. The **Context** field shows how the word is used in the text. The **Change To** field displays the most likely correct spelling, with other possible spellings listed below.

Click the **Change** button to accept the suggested spelling. Click the **Change All** button to correct all occurrences of the Unknown Word throughout the text. If the spelling suggested in the Change To field is not the word you wish to use, you may click on one of the alternatives displayed in the area below this field. If none of the suggested alternatives are acceptable, you may type the correct word into the Change To field before clicking the Change button.

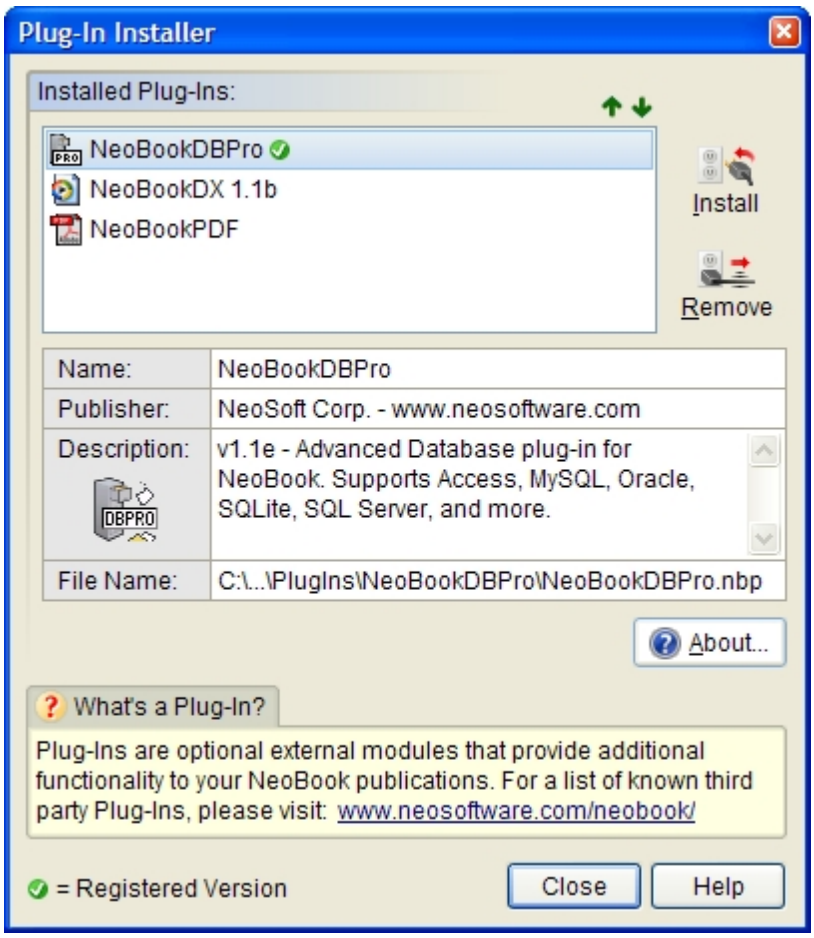
To leave the Unknown Word as it was originally spelled, click the **Skip** button. Choose **Skip All** to leave all occurrences of the Unknown Word in their original form.

If the Unknown Word is a proper name or a correct spelling for a word which you use frequently, you can choose **Add to Dictionary** to place the word into VisualNEO for Windows's dictionary. This will prevent the word from being flagged as unknown in future spell checks.

To end the spell check early, click the Close button.

Install Plug-Ins

Use this feature to register third party Plug-Ins with VisualNEO for Windows. Plug-ins are specially designed components that can be integrated into VisualNEO for Windows to provide additional functions not found in the basic program. Properly designed plug-ins are extremely fast and their use virtually transparent. Once installed, the plug-in's functions will appear in the Insert Action list along with VisualNEO for Windows's native commands. To use a plug-in, simply select one of its commands from the list.



Note: *Some plug-ins may require a special installation or setup process before they can be registered in VisualNEO for Windows. Consult your plug-in's documentation for any special requirements prior to using this function.*

To register a plug-In, click the **Install** button. A standard Windows file selector will appear. Locate and select the desired plug-in file (*.nbp). Most plug-ins will be installed in the VisualNEO for Windows for Windows\Plug-Ins folder. If the installation works correctly, the plug-in will appear in the list of Installed Plug-Ins. The plug-in's name, publisher, description and file name will be listed under Plug-In Information.

To remove the plug-in, highlight the plug-in's name in the Installed Plug-In list and click **Remove**. This will completely remove any functions associated with the plug-in from VisualNEO for Windows. Any files associated with the plug-in still will be present on your hard drive. You will need to use the Windows Explorer or the plug-in vendor's un-install utility to remove the files. Be careful not to remove any files belonging to other plug-ins or programs.

An up-to-date list of available VisualNEO for Windows plug-ins is available at the web site below:

visualneo.com

Note: *If you experience problems associated with a specific plug-in, please consult with the plug-in publisher or author. It is the plug-in vendor's responsibility to provide support for their product.*

Function Library

The Function Library is a place where you can store groups of often-used Action commands.

These groups, or functions, then can be accessed from your publications at any time using VisualNEO for Windows's Call Action. Functions can make your publications smaller and easier to update. For example, suppose you created a publication with 25 buttons, each performing basically the same task. Instead of entering the same series of Action commands 25 times, you could create a function and simply "call" it from each button. Also, if your function needs to be modified, you only need to edit it once instead of 25 times.

Functions can be organized into sub folders to make finding the one you want easier. Use the **New Folder** button to add a sub folder. To move a function to another folder simply drag its icon to the desired location.

You can modify an existing function by clicking the function's name in the Available Functions list, then clicking the **Edit** button. The Action Editor will appear allowing you to make changes to the function.

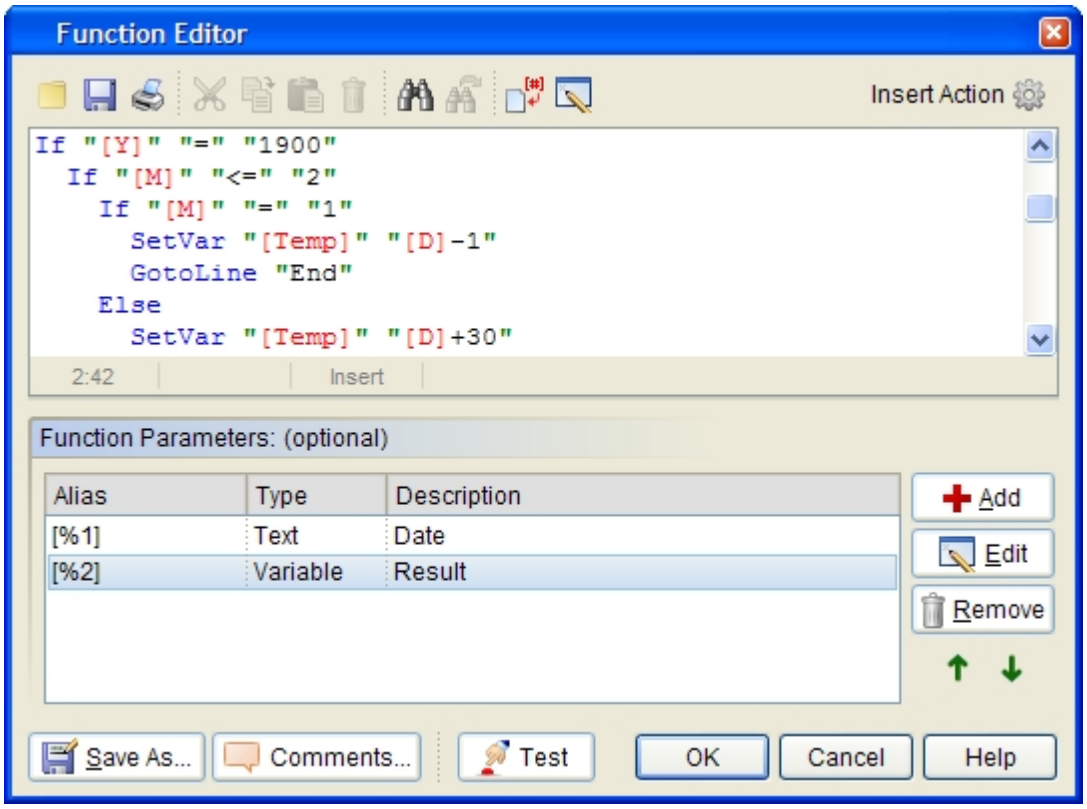
The **Rename** button can be used to change the name of an existing function or folder. However, any publications that use the function's old name will need to be edited to take advantage of the change.

A function or folder that is no longer needed can be removed from the list using the **Delete** button. Be sure that functions you delete are not used by any of your publications. If a publication attempts to "call" a function that has been deleted, you will receive an error message.

To create a new function, click the **New** button. A small menu will appear allowing you to select a **scripting language** for your new function. You can use VisualNEO for Windows's native scripting language, VBScript or JScript. Both VBScript and JScript are complex scripting languages that generally require an advanced knowledge of programming. However, there are many VBScript and JScript examples on the Internet that can usually be adapted to work with VisualNEO for Windows. See Advanced Scripting below for more information on using VBScript and JScript in VisualNEO for Windows.

Once you select a scripting language, the Action Editor will appear, allowing you to define what your function will do. The Action Editor contains a large editor window and a tool bar. Action commands may be typed directly into the editor, but most authors prefer to use the Insert Action button instead. Clicking the Insert Action button will display a list of available commands. The commands are organized into groups. Locate and click the title of the command you want. For most commands, VisualNEO for Windows will display a simple form that can be filled in to complete the Action. See Understanding Actions and Variables and Action Command Reference for a complete discussion of the Action Editor and Action Commands.

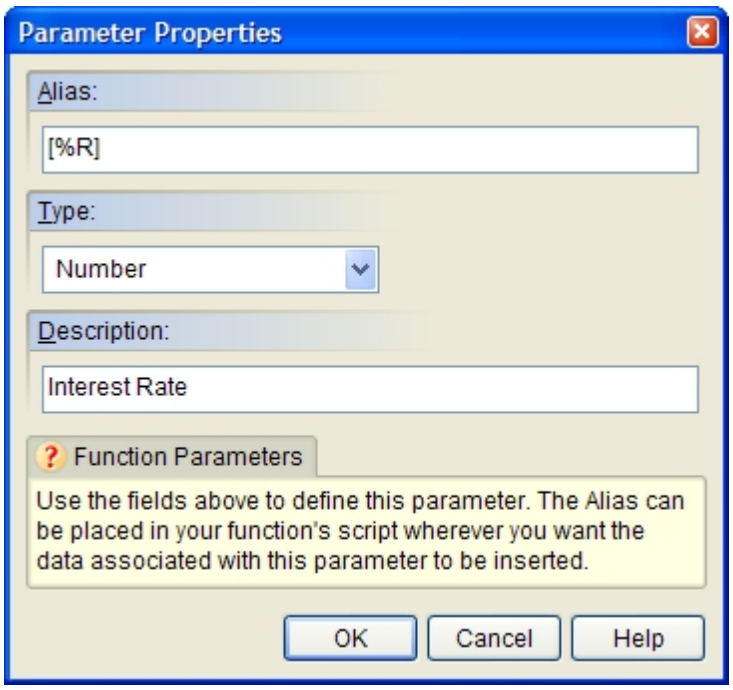
Note: *The Insert Action button is only available when editing scripts created in VisualNEO for Windows's scripting language.*



The **Save As** button can be used to save a copy of your function under a different name. Use the **Comments** button to record a short description of your function. This description will appear in the Call action's properties screen whenever your function is selected. The **Test** button can be used to run the current publication allowing you to test your function without leaving the editor.

The **Function Parameters** panel below the Action Editor can be used to create advanced functions. An advanced function can accept up to nine parameters, allowing information to be passed from anywhere in the publication. The parameters can contain whatever data your function requires to complete its work. For example, a function that calculates compound interest might use parameters to input the loan's present value, interest rate and length.

To create a parameter, click the **Add** button. A **Parameter Properties** screen like the one below will appear:



The **Alias** is a place holder for the parameter's data. The alias is essentially a special variable that can be used in any of your function's actions. Like a variable, an alias is simply an area of the computer's memory that can be used to store information. Each function alias should be given a unique name and surrounded by square brackets ([ ]). When creating a new parameter, the default alias assigned by VisualNEO for Windows corresponds to the parameter's number (for example: [%1]). However, you may wish to change this name to something that better describes what this particular parameter represents. In the case of our sample Compound Interest function above, we might change the first parameter's alias to something like [%PV] for Present Value. The % symbol is optional, but including it can be helpful in telling the difference between aliases and regular variables.

Parameters can contain different types of data. To prevent the wrong type of data from being entered into a parameter, use the **Type** field to select one of the following options:

Text	Parameter is a character string. May contain alpha characters, numbers, punctuation, etc.
Number	Parameter is a number.
Mixed	Parameter may be either numeric or alpha. May contain mathematical expression.
File Name	Parameter is a file name.
Variable	Parameter is a variable name.

Finally, use the **Description** field to indicate what type of data the parameter expects. The description will appear in the Callaction's wizard as a reminder to you what the parameter is for.

Click OK to close the Parameter Properties screen and add the new parameter to the function.

When writing the script that comprises the function, simply insert the alias in the locations where you want the data they represent to appear. For example, our sample function that computes compound interest includes the following parameters:

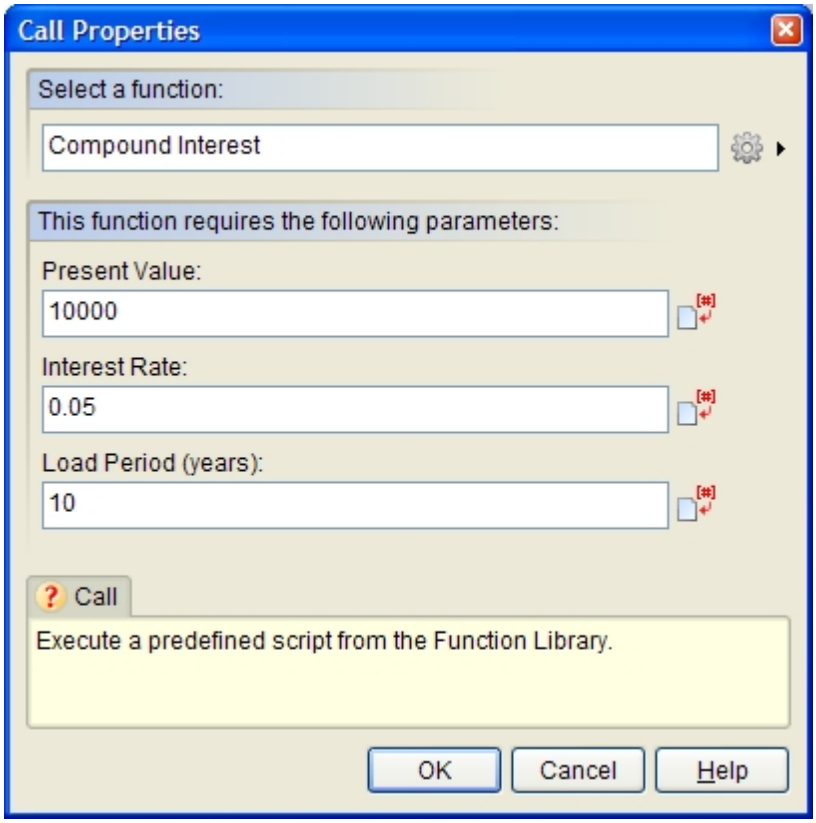
Alias	Type	Description
[%PV]	Number	Present Value
[%R]	Number	Interest Rate
[%N]	Number	Load Period (years)
[%FV]	Variable	Variable to store result (Future Value)

The script to calculate the compound interest consists of only a single Math Action, but you can see how the aliases play an important role:

```
Math "[%PV]*((1+[%R])^[%N])" "2" "[%FV]"
```

Once the function is complete, click the OK button to save it. You will be prompted to give the function a Name. You should select a name that adequately describes what the function does. "Compound Interest," for example. The name will appear in the Function Library and will be used when calling your function.

When you later use the **Call** Action and select the "Compound Interest" function, the Action Wizard will display fields for the Present Value, Interest Rate, Loan Period and Result parameters. The values you enter here will be used by the Compound Interest function.



Advanced Scripting: Using VBScript and JScript with VisualNEO for Windows

VisualNEO for Windows provides three special methods that VBScript and JScript programmers can use to communicate with VisualNEO for Windows. These methods (called nbGetVar, nbSetVar and nbExecAction) can be combined with native VBScript and JScript methods to establish two-way communication between the a function and your publication. This provides VisualNEO for Windows publications with direct access to many powerful JScript and VBScript features, which you can use to augment VisualNEO for Windows's own scripting language.

Note: *Some experience with and VBScript or JScript are required to use these features.*

Special VBScript/JScript methods supported by VisualNEO for Windows:

nbSetVar

Purpose: This method assigns a value to a VisualNEO for Windows variable.

Syntax: publication.nbSetVar "variable name", "value"

variable name
The name of a VisualNEO for Windows variable. The variable name should be surrounded by square brackets.

value
The value to assign to the variable.

Example: The following VBScript example passes a simple string to VisualNEO for Windows:

```
publication.nbSetVar "[Result]", "Hello from VBScript!"
```

Here is an example showing how function aliases can be used with nbSetVar:

```
Function Celsius( fDegrees )
  Celsius = (fDegrees - 32) * 5 / 9
End Function

publication.nbSetVar "[%2]", Celsius( [%1] )
```

nbGetVar

Purpose: This method returns a string containing the contents of a VisualNEO for Windows variable.

Syntax: value = publication.nbGetVar("variable name")

variable name
The name of the VisualNEO for Windows variable to retrieve. The variable name should be surrounded by square brackets.

value
The contents of the VisualNEO for Windows variable in string format.

Example: The following VBScript example copies the contents of VisualNEO for Windows's [PubTitle] variable to a VBScript variable and displays it in a message box:

```
title = publication.nbGetVar( "[PubTitle]" )
MsgBox title
```

nbExecAction

Purpose: Use this method to execute VisualNEO for Windows actions.

Syntax: publication.nbExecAction("action script")

action script
The VisualNEO for Windows action to execute. Multiple actions may specified by separating them with a carriage return.

Example: This example executes VisualNEO for Windows's AlertBox action:

```
publication.nbExecAction( "AlertBox ""Hello"" ""Hello from the Web Browser!""" )
```


Set Preferences

Use this option to configure how your copy of VisualNEO for Windows works. These options

only effect your interaction with VisualNEO for Windows; they do not affect your compiled publications (use App Properties to configure your publication). The Preferences screen is divided into three sections indicated by the icon images on the left: General, Tools and Editor. To view the settings for a section, click the corresponding icon.

General

Under Startup Options, you may specify what actions VisualNEO for Windows should take each time it is loaded. Choose **Open a new blank publication** if you want to start each VisualNEO for Windows session with a fresh publication. Select **Open last edited publication** to reopen the publication you were working on last time you used VisualNEO for Windows. Finally, you may choose to have VisualNEO for Windows **Display a file open dialog** to start each session by selecting an existing publication from your hard drive.

Enable the **Undo on** option to activate VisualNEO for Windows's undo feature. When active, you can remove the last modification made to a publication by selecting Undo from the Edit menu.

Note: *Disabling Undo may improve performance on slower computers.*

Enable the **Create backup files** option to have VisualNEO for Windows retain a copy of the previous version of your publication whenever you select Save from the File menu. The backup file will have .BAK as its file name extension. While creating backup files will consume extra hard disk space, this may prove invaluable should your publication file become corrupted (by Windows, disk compression, power failure, bad disk sectors, etc.) To access a backup file, use Windows Explorer to change the backup file's extension from BAK to PUB. Then load the renamed file into VisualNEO for Windows. This way, at most you'll only lose the edits you made since you last edited the publication.

Check the **Automatically update text and image files modified by other programs** option to have VisualNEO for Windows periodically check to see if any of the elements in your publication have been modified. If you leave this option unchecked, VisualNEO for Windows will display files as they were when VisualNEO for Windows loaded the publication (which can speed performance on slower computers). Leaving this option unchecked does not affect compiled publications, as the files always will be updated before compiling.

Tools

This is where you can specify which programs will be used to edit the many types of media files supported by VisualNEO for Windows. Select your favorite editors for text, images, animated GIF, cartoons, sound, music, video and Flash™ files. VisualNEO for Windows automatically will launch the appropriate program when you select a file from the Edit menu's Create/Edit command. With most file types, you may choose to let Windows automatically locate the appropriate editor, or you may specify a specific application installed on your computer.

Note: *It is recommended that you use VisualNEO for Windows's built-in word processor/text editor for editing text files. The built-in editor provides many formatting options and has been designed specially for use with VisualNEO for Windows.*

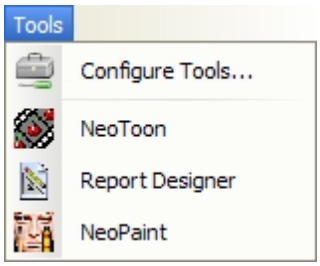
Editor

Use this screen to select which font and colors to use for VisualNEO for Windows's Action Editor. Choose an item from the list of **Elements**, then select a color and attributes for that item. The preview window shows the effect of your selections. Be sure to select compatible colors for each of the elements. Don't select white text on a white background, for example.

The **Editor Font** combo box contains a list of compatible fonts available on your computer. Because the editor supports syntax highlighting, the selection is limited to mono-spaced fonts.

The Tools Menu

The Tools menu is a place where you can add shortcuts to programs and utilities you find useful when working in VisualNEO for Windows. Programs added to the Tools menu can be accessed with a single mouse click.



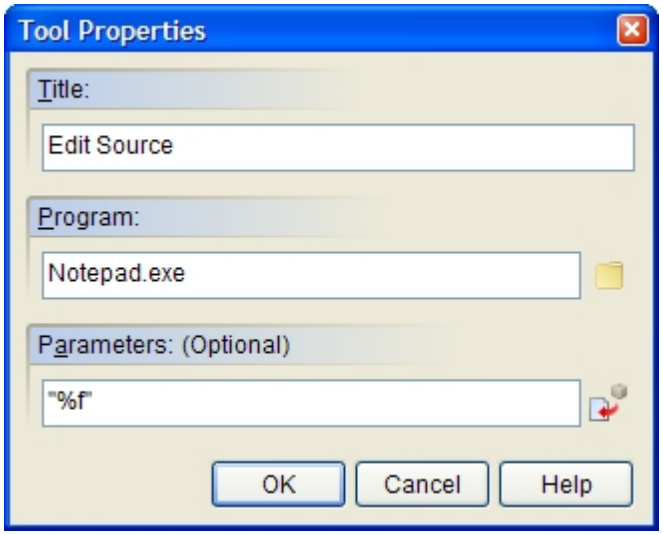
Configure Tools

Use this option to add and remove programs and utilities from the Tools menu.

To add a new tool, click the **Add** button. A Tool Properties screen will appear allowing you to define the program or utility to be added. Enter the name of the program in the Title field. The title will appear as a choice under the Tools menu. Next, enter the path and file name in the Program field. You may find it easier to click the small folder to the right of the Program field and select the program's EXE file using a file selector. Finally, if your program requires any special command line switches, enter those in the Parameters field. In addition to program specific command line options, you may incorporate any of the special codes below into the Parameter field:

%o The path and file name of the currently selected object (if any). For example, if the selected object is a Picture, then %o will represent the name of the image file assigned to the Picture.
%f The Full path and file name of the current publication.
%p The drive and folder where the current publication resides.
%n The file name only of the current publication.

For example, to load the current publication into Windows Notepad, your Tool Properties screen would look like this:



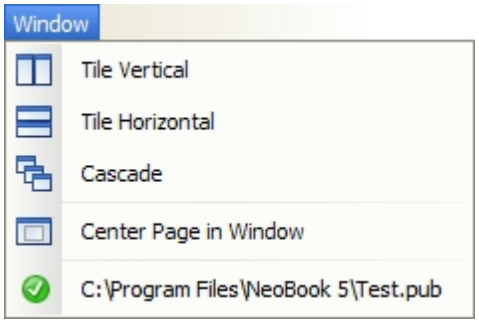
Note: *For most applications you must surround file names, or in this case the special codes, with double quotes. Otherwise, spaces in the file name will confuse the application and your file won't load.*

You can modify an existing tool by clicking the tool's name in the list, then clicking the **Edit** button. The Tool Properties screen will appear, allowing you to make changes.

A tool that is no longer needed can be removed from the list using the **Delete** button. This function doesn't actually remove the program from your computer, just from VisualNEO for Window's Tools menu.

The Window Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Window menu. For more information, select a command from the picture below:



Tile Vertical / Horizontal

Select these commands to arrange open publication windows side by side in the VisualNEO for Windows work space. These commands are most useful when more than one publication window is open.

Cascade

Select this command to overlap open publication windows so that each title bar is visible.

Center Page in Window

Use this function to center the publication inside the current window.

The Help Menu

This section contains descriptions of the commands found in VisualNEO for Windows's Help menu.

VisualNEO Help

Choose this option to display VisualNEO for Windows's Help file, which you are viewing now. The help file contains instructions for using VisualNEO for Windows. The help file can be used to search for information about specific topics. For information on using help files in general, please consult your Windows documentation.

VisualNEO Tutorial

Displays Tutorial section of VisualNEO for Windows's Help file.

VisualNEO Quick Tour

Displays the Quick Tour section of VisualNEO for Windows's Help file. The Quick Tour provides a brief overview of important VisualNEO for Windows features.

VisualNEO Home Page

Selecting this option takes you to VisualNEO web site using your default Internet Browser. The web site contains product information, upgrade patches, answers to common questions, news and trial copies of SinLios Soluciones Digitales products.

VisualNEO Support Forum

You can find answers to many VisualNEO for Windows questions by visiting our online support forum. The forum also contains tips, tricks and update information. You can even post questions of your own and get answers from advanced VisualNEO for Windows users and SinLios Soluciones Digitales support staff.

Contact Technical Support

If you run into trouble or have a question, use this option to send an email message to SinLios Soluciones Digitales Technical Support.

About VisualNEO

Displays information about your version of VisualNEO for Windows and contact information for SinLios Soluciones Digitales.

Understanding Actions and Variables

As you've learned in the preceding sections, each element added to your publication (text, pictures, buttons, etc.) is considered an object. In addition to their visual qualities, most types of objects can also be used to perform tasks. These tasks can be as simple as jumping to another page or as complex as calculating a test score. Each object performs its task based on special instructions which tell VisualNEO for Windows what to do – these are called Actions.

Actions assigned to objects are executed in response to specific events, including clicking

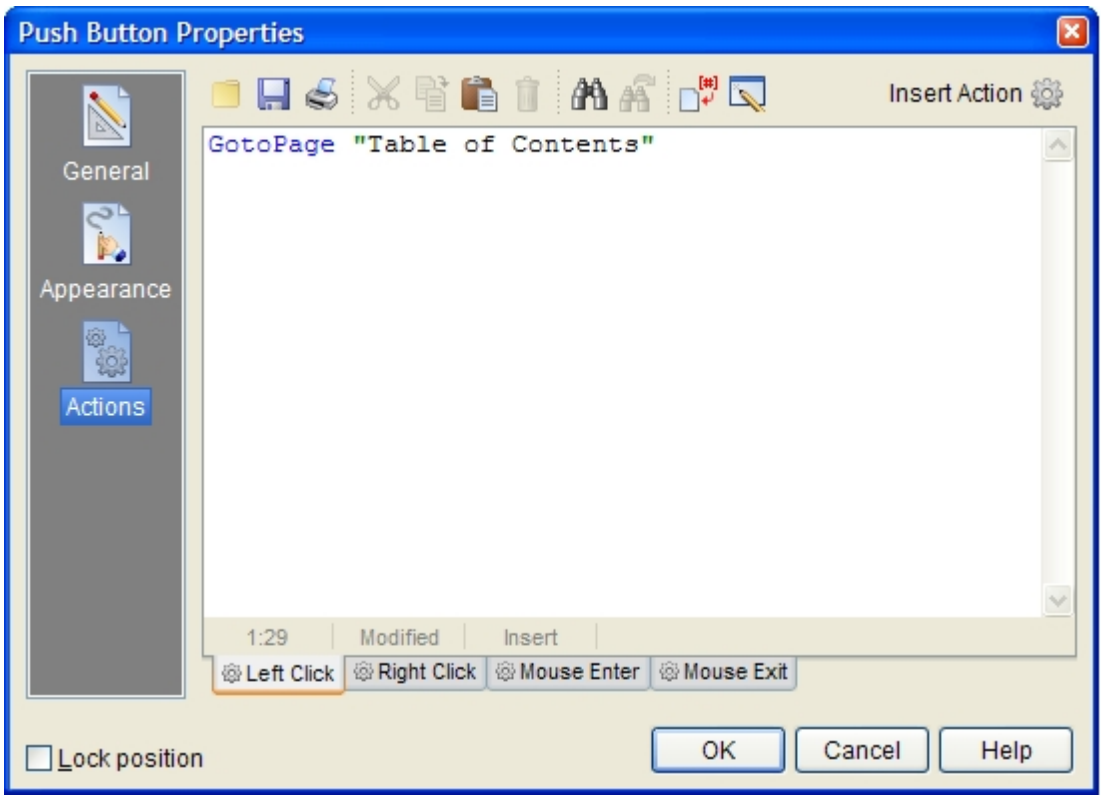
the mouse, moving the mouse, pressing a key on the keyboard, etc. Not all objects are sensitive to all types of events. For example, Push Buttons can execute Actions when clicked or when the mouse passes over the object, while Timer objects are oblivious to mouse activity.

Pages can contain Actions that execute whenever the reader enters or leaves (see Page Properties). Actions can be assigned to execute in response to specific publication events like Startup and Shutdown (see App Properties). Actions also can be assigned to hypertext links placed inside Simple Text and Article objects.

Adding Actions to Objects and Pages

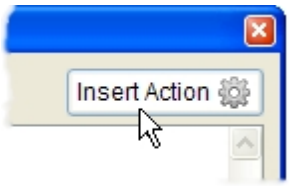
Fortunately, you don't need to learn a sophisticated programming language, like Java or C++, to use VisualNEO for Windows's Actions. Although VisualNEO for Windows's syntax resembles a traditional programming language, it's significantly less complex and much easier to learn. In fact, most Actions can be inserted by selecting the desired command from a list and filling out an easy to understand questionnaire.

To create Actions for your publication, display the Properties screen for the item you want to assign the Action to. For example, to add an Action to the current page, select the Page Properties command from VisualNEO for Windows's Page menu. To add an Action to an object, select the object and choose the Object Properties command from the Edit menu. (You can also display an object's Properties screen by placing the mouse pointer over the object and clicking the right mouse button.) Once a page or object's Properties screen is displayed, select the Actions icon located along the left side of the screen.

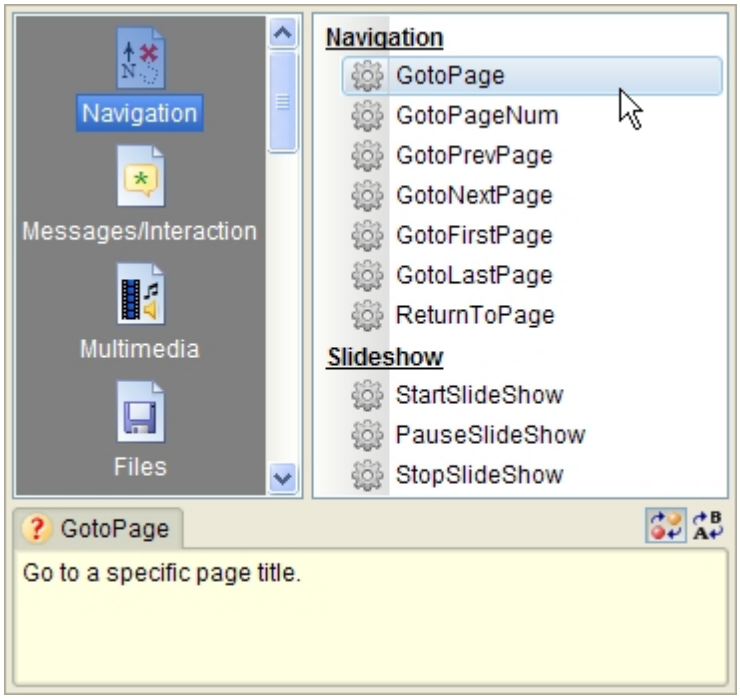


The Action screen contains a large editor window, a tool bar and set of tabs just across the bottom. If the item you've selected has the ability to respond to multiple types of Action events, the tabs will contain the names of these events. If needed, you may specify a separate set of Actions for each event by clicking the tabs.

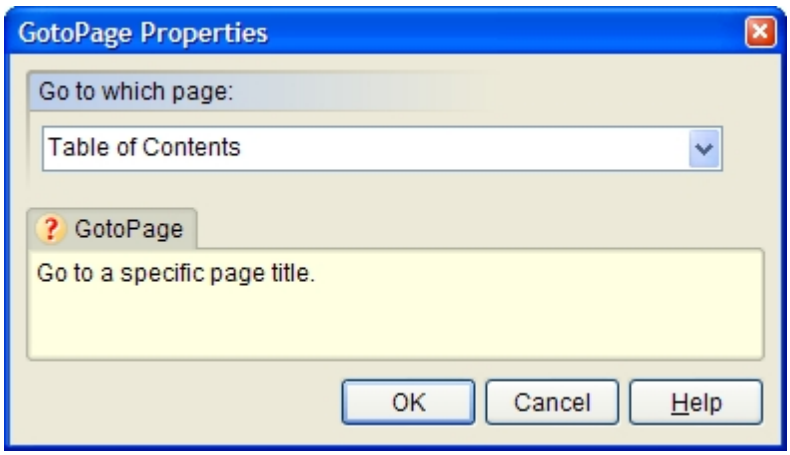
Action commands may be typed directly into the editor, but most authors prefer to use the **Insert Action** button instead. The Insert Action button (located at the right just above the editor), will display a list containing all of VisualNEO for Window's Action commands, plus commands belonging to any third-party Plug-Ins that have been installed. (See Install Plug-Ins)



Actions are separated into categories indicated by icons along the left side of the screen. Clicking an icon will display Actions belonging to that category on the right half of the screen. A short description of each Action will be displayed in a small window as your mouse pointer passes over it. To select an Action, click the desired item in the list.



If the Action you've selected requires additional information (such as a page title or a file name), a short questionnaire will appear, allowing you to provide the necessary details. These additional details are called parameters. For example, an Action called GotoPage requires a parameter that tells VisualNEO for Windows the title of the page you want to go to. Once you've completed the questionnaire, VisualNEO for Windows will format and insert the Action code into the editor.



The questionnaire for the GotoPage Action lets you select from a list of pages.

You may insert additional Actions by repeating the steps above. New Actions will be inserted into the editor at the cursor position. Multiple Actions will be executed in the order that they are listed in the editor. Once inserted, you can edit Actions manually using your keyboard or display the questionnaire again by double clicking the line containing the Action's code.

When Actions appear in the editor, they must be formatted using a specific syntax in order to be understood by VisualNEO for Windows. When using the Insert Action button, this formatting is handled automatically. However, if you enter or edit Actions manually, you'll need to make sure that you adhere to a few simple formatting rules.

The Action's name must appear first, followed by the required parameters, if any. Each parameter must be surrounded by "quotes" and separated by spaces. Each Action and its parameters must fit on a single line in the editor. If more space is needed, the editor window can be scrolled horizontally to accommodate additional text. For Actions that can display multiple lines of text (like AlertBox and MessageBox), line breaks must be marked within parameters using a special character called a pipe (|) and not by pressing the keyboard's Return key.

For example, an Action called AlertBox displays a Windows-style dialog box with a custom message. AlertBox requires two parameters: one for the dialog box's title and one for the message. The proper syntax for this command looks like this:

```
AlertBox "Greetings" "Welcome to my publication."
```

When executed, this Action will display a dialog box like this:

Any lines appearing in the editor that begin with a period "." are ignored by VisualNEO for Windows. These are considered comments. For example:

```
.this is a comment
AlertBox "Greetings" "Hello world."
```

You can use this feature to add comments to your Action scripts or to temporarily disable commands for debugging purposes.

Using Variables

A variable is an area of the computer's memory that you can use to store information temporarily while your publication is running. Variables can contain text, numbers, names, addresses, dates or just about anything else. The contents of variables can be used in calculations, inserted into text, read from files, written to files, etc.

Many objects, such as Check Boxes, Radio Buttons and Text Entry Fields, use variables to store information about their status or contents. You can use these variables, and others you create, within Action parameters in addition to or in place of literal text. For example, you might require readers to enter their name at the start of a publication. A Text Entry Field created for this purpose records the name into a variable. Later, that variable could be used to document a test score or to personalize the publication by adding the name to various screens.

Each variable used within a publication should be given a unique descriptive name. Like a pet dog, variables can be given almost any name. When used in VisualNEO for Windows, however, variable names should always be surrounded by brackets ([ ]). This is how VisualNEO for Windows knows that you're talking about a variable named [Spot] and not the word "Spot". Some examples of valid variable names are:

```
[Answer]  [Name]  [Price]  [Score]  [X]  [Y]
```

Referencing a variable within Action parameters is simple: just insert the variable's name into the text - remembering to enclose it in brackets, of course. For example:

```
AlertBox "Greetings" "Hello [Name]. Welcome to my publication."
```

Some advanced Actions use variables to pass information back to you. In the example below, the FileRead Action extracts a line of information from a file and places it into a variable called [Data]:

```
FileRead "Sample.txt" "1" "[Data]"
```

Variables may be inserted anyplace within your publication where text is required: object captions, Action parameters, file names, etc. For example, a publication that required readers to enter their name might also include a Push Button with the following caption:

```
[Name], please press this button to continue.
```

When viewing the publication, a reader named Sally would see:

Creating and Defining Variables

Creating variables in VisualNEO for Windows is easy, since you're not required to allocate memory or explicitly define the scope of a variable prior to using it - although you can do that too if you want. In VisualNEO for Windows, anytime you make a reference to a variable, it's created automatically. However, there are many situations where you might want to initialize a variable prior to using it. Suppose you have several default settings that you want to initialize when your publication first starts. VisualNEO for Windows's SetVar Action is designed for this purpose. For example:

```
SetVar "[Name]" "Unknown"
SetVar "[Busy]" "No"
SetVar "[Amount]" "1.00"
```

It's not necessary to delete variables since VisualNEO for Windows will do this for you when your publication closes. However, if your publication uses a large number of temporary variables, you may be able to improve performance by manually removing variables from memory when you no longer need them. You can do this using the SetVar Action to set the variable's value to null or empty. Empty variables are automatically removed from memory by VisualNEO for Windows. For example:

```
SetVar "[Name]" ""
SetVar "[Busy]" ""
SetVar "[Amount]" ""
```

Defining Variables

In addition to the SetVar Action, VisualNEO for Windows provides another Action called DefineVar that can be used to create variables and restrict their contents to specific types of data. Generic, undefined variables, created with SetVar, don't care what type of information they contain. It's perfectly acceptable for a generic variable to contain text one minute and a number the next.

However, there are instances where experienced authors might want to limit the contents of a particular variable to text or numbers. For example, you might want to limit what can be entered into the "Age" field of a fill-in-the-blank form to numbers. By using the DefineVar Action to declare an [Age] variable as an "Integer" at the beginning of your publication, you can do exactly that. For example:

```
DefineVar "[Age]" "Integer" "" "Global" ""
```

If the variable [Age] is assigned to a Text Entry Object, then entering anything other than a number will generate an error message.

Other variable types that can be created with DefineVar are:

Undefined	The variable's content is not limited. This is the same as a generic variable created with SetVar.
String	The variable can contain any characters either alpha or numeric.
Integer	The variable is limited to whole numbers.
Currency	The variable is limited to numbers and will be formatted as currency according to the regional settings in the Windows Control Panel.
Decimal	The variable is limited to numbers and will be formatted with the number of decimals specified.
Boolean	The variable is limited to "True" or "False" values.
Date	The variable is a date and will be displayed using the specified date format (m/d/y, etc.).

In addition to limiting a variable's content, DefineVar can be used to format certain types of variables so that they are more readable. Currency and Decimal types will be formatted with a specific number of decimal places. Boolean variables will always be either "True" or "False." Date variables can be displayed in a variety of ways depending on the formatting option specified in the DefineVar statement.

Variable Arrays

VisualNEO for Windows also allows the use of compound variable names to produce something called an array. An array is a collection of related variables. Each variable in the array is referred to by the name of the array followed by its position within the array. For example, a ten item array called *Name* would contain variables [Name1], [Name2], [Name3], [Name4], and so on, up to [Name10].

Arrays can be defined explicitly like [Name1] or by using compound variables like [Name[X]], where [X] is an incremented numeric value. Such an array could be used to store lines read from a file like this:

```
FileLen "sample.txt" "[FLen]"
Loop "1" "[FLen]" "[X]"
  FileRead "sample.txt" "[X]" "[Name[X]]"
EndLoop
```

After executing these Actions, the number of items in the array should be equal the number of lines in the file ([FLen]). (The FileLen Action counts the number of lines in the file and stores that number in the variable [FLen]. More information about these Actions can be found here.) Individual items in the array could be accessed directly like this:

```
AlertBox "Results" "Line 5 = [Name5]"
```

or the entire array can be processed like this:

```
Loop "1" "[FLen]" "[X]"
  AlertBox "Results" "Line [X] = [Name[X]]"
EndLoop
```

Like individual variables, it's not necessary to delete arrays. If needed, however, you can delete array elements individually using SetVar. You can remove an entire array using the DeleteArray Action. For example:

```
DeleteArray "[Name]" "[FLen]"
```

If you don't know the bounds or size of an array, you can use the GetArrayInfo Action to find out:

```
GetArrayInfo "[Name]" "[FirstItem]" "[LastItem]" "[ArraySize]"
```


Predefined Global Variables

There are a number of important variables that are created and updated automatically by VisualNEO for Windows. These variables contain information about the status of your publication, the reader's computer, the current date and time, etc. These may be inserted wherever normal variables are accepted.

Read-Only Variables

The first group of global variables are **Read-Only**, meaning that they can be examined and displayed, but not modified, using the SetVar or other Action commands.

App Properties

[CommandLine]	Command line parameters passed to the publication. Multiple parameters are separated with carriage returns [#13]. The first parameter is always the name of the publication exe.

[PubAuthor]	The publication's author as specified in App Properties.
[PubColors]	The publications native color resolution as specified in App Properties.
[PubVersion]	The publication's version number as specified in App Properties > Version Info.
[PubLeft] [PubTop]	The position of the upper left corner of the interior of the publication window (the client area) in screen coordinates. These values can be added to the coordinates of an object to determine its absolute position relative to the entire screen.

Page Properties

[PageTitle]	The title of the current page.
[PageNumber]	The number of the current page. See also [PageNumberOffset].
[P]	Same as [PageNumber].
[PageNumberLeft] [PageNumberRight]	These variables can be used to simulate left and right page numbers on publications that are designed to appear as if each screen is composed of two facing pages. See also [PageNumberOffset].
[PageCount]	The total number of pages in the publication, excluding the master page.

Date/Time

[Time]	The current time (H:M:S AM/PM).
[Time24]	The current time in 24-hour format.
[Hour]	The current hour.
[Minute]	The current minute.
[Second]	The current second.
[DateShort]	The current date in Windows' short format (10/22/15).
[DateLong]	The current date in Windows' long format (October 22, 2015).
[Month]	The current month in text form (October).
[MonthNum]	The current month in number form (10).
[Day]	The current day of the week in text form (Monday).
[DayNum]	The current day of the months in number form (1-31).
[Year]	The current year (2015).

Windows/Hardware

[CDRomDrive]	Drive letter of the first CD-ROM drive or "error" if none.
[HDSerialNum]	Returns the serial number for the system's C: drive.
[NetworkDrive]	Drive letter of the first network drive or "error" if none.
[ScreenColors]	The number of colors supported by the current Windows video mode.
[ScreenHeight]	The height the entire Windows screen in pixels.
[ScreenWidth]	The width of the entire Windows screen in pixels.
[SystemLanguage]	Returns the active system locale (language). For example: "English (United States)"
[SystemLanguageExt]	Returns the standard three-letter extension for the active system locale (language). For example: "ENU" for English (United States).
[UserName]	The name of the currently logged in user (if a network is installed).
[WindowsPlatform]	The current Windows platform installed (Windows 95, 98, ME = 1, Windows NT, 2000 XP and higher = 2).
[WindowsVer]	The major and minor Windows version number (4.0 for Windows 95, 5.1 for Windows XP, 6.0 for Vista, 6.1 for Windows 7, 6.2 for Windows 8, 6.3 for Windows 8.1, 10 for Windows 10).
[WindowsVerName]	The published name of the installed version of Windows. For example: "Windows 10 Home".

Folders

[MyDocuments]	The location of the current user's "My Documents" folder.
[ProgramFiles]	The locations of the system's "Program Files" folder. This folder's name may be different depending on the system's default language.
[PubDir]	The folder where the publication EXE resides.
[SystemDir]	The location of the Windows system directory (usually c:\windows\system).
[TempDir]	The location of the Windows temp folder.
[WindowsDir]	The folder where Windows itself is installed (usually c:\windows).

Messages/Errors

[MCIResult]	Used by the MCICommand Action to report errors. If MCICommand is successful, [MCIResult] will contain a 0 (zero), otherwise [MCIResult] will contain an error number.
[MCIResponse]	Used by the MCICommand Action to store information return by the device. The contents of this variable (if any) depends entirely on the MCI command string used.

[LastError]	Disabling the Display Error Messages option on the Misc. page of the App Properties screen allows experienced VisualNEO for Windows authors to trap and respond to errors programmatically rather than having VisualNEO for Windows display the error in a dialog box. When this option is off, all error messages are placed in the [LastError] variable. You can use this variable in your Action scripts to determine when an error occurs and respond appropriately. For example:
	
	`FileWrite "parts.dat" "Append" "[PartNum]"`
	`If "[LastError]" ">" ""`
	`  AlertBox "Error" "Unable to Parts file."`
	`EndIf`

Advanced

[Embedded]	Special embedded file variable indicates that the file name that follows has been embedded inside exe. See Embedded Files.
[FocusedObject]	The name of the object that has the input focus. ***Note:*** *This variable is primarily intended to be used in scripts to identify which object is active. It is not a "live" variable and will not automatically update the screen whenever an object is clicked.*
[VisualNEO for WindowsVersion]	Returns the VisualNEO for Windows version number. The variable will contain the major and minor version number followed by a period and the build number which corresponds to the version letter (if any). For example, version 5.0.0 would be "500.0" and version 5.1.1a would be "511.1". (Since this variable was not available prior to version 4.1.1d, [VisualNEO for WindowsVersion] will be empty for previous versions of VisualNEO for Windows.)
[NBMode]	Contains "D" if publication is running from VisualNEO for Windows's design mode or "R" if running from compiled/runtime mode.
[NBType]	Returns type of compiled pub - A = application, S = screen saver, T = tray application, W = web plug-in
[Self]	The name of the object that executed the currently running Action script. For scripts not executed by objects (such as page enter/exit), [Self] will be empty.
[WinHandle]	The handle (HWND) of the publication's windows (Generally of use only to plug-in authors).
[HyperlinkClickedText]	This variable contains the text of the most recently clicked hyperlink. This applies to the Article, Linked-Article and Simple Text objects.
[AppID.ProcessID] [AppID.ProcessHandle] [AppID.WinHandle]	These variables contain technical information about applications launched with the Run or RunInRectagle actions. The first part of the these variables are based on the Run action's *unique identification number* (AppID) variable. For example, if the AppID variable is [NotePad] then the corresponding ProcessID variable would be [NotePad.ProcessID]. These variables will generally only be of interest to plug-in developers.
[AppID.ExitCode]	This variable will contain the exit code of an application launched with the Run or

	RunInRectagle actions. Some utilities use exit codes for returning information or to indicate that an error has occurred. For most Windows applications, however, the exit code will be zero.
[Object.WinHandle]	Use this variable to access the handle (HWND) of the window created by the CustomWindow action. The first part of this variable is based on the name of object used to create the window. For example, a custom window created from an object named Container1 will result in a variable called [Container1.WinHandle]. (Generally of use only to plug-in authors).

Read-Write Variables

The global variables below are **Read-Write** meaning they can be read from as well as written to:

App Properties

[AppTitle]	The title of the application which appears in the tray icon and the Windows Task Ma [AppTitle] and [PubTitle] are the same.
[PubTitle]	The caption in the title bar of the publication window.
[PubWidth]	The width of the interior of the publication window (the client area).
[PubHeight]	The height of the interior of the publication window (the client area).

[PageChangeName]	This variable can be used to abort or redirect a page change by modifying it from wi Properties > Page Change Action. Upon entering the Page Change Action, this varia the name of the page the reader wishes to navigate to. You can abort the page cha the variable like this:
	
	`SetVar "[PageChangeName]" ""`
	
	The page change can be redirected by setting the variable to the name of another
	
	`SetVar "[PageChangeName]" "Error Page"`
	
	Setting this variable anywhere other than in the Page Change Action has no effect.

[ShutdownStatus]	This variable can be used to abort a publication exit by setting it to "False" from wit Properties > Shutdown Action. For example: `SetVar "[ShutdownStatus]" "False"` Setting this variable anywhere other than in the Shutdown Action has no effect.
[ShutdownSource]	In addition to [ShutdownStatus] above, you can also examine the global [Shutdown from within the App Properties > Shutdown Action to determine what caused the pu close. [ShutdownSource] may contain one of the following:

VisualNEO for Windows	The shutdown request was triggered by VisualNEO for Windows' action.
Windows	The shutdown request originated with Windows. There are sever Windows functions that could trigger a shutdown request, includi selecting "Turn off computer" from the Start Menu; the Task Mana End Task command; or manually closing the application's icon fro System Tray. You should not normally refuse to close the publicat when the requested to by Windows.
CloseButton	The user clicked on the publication window's close button, selecte close from the system menu, or pressed Alt+F4.

For example, to minimize the publication instead of closing it when the source of the close button, do the following:

```
If "[ShutdownSource]" "=" "CloseButton"
  SetVar "[ShutdownStatus]" "False"
  SetVar "[WindowState]" "Minimized"
```

```
EndIf
```

[StartInSystemTray] Normally, when a compiled system tray applications is launched it will automatically icon in the tray. If you would prefer to have your tray application open in a window place the following code in the App Properties Startup Action:

```
SetVar "[StartInSystemTray]" "False"
```

Once open, minimizing the window will send it to the system tray. Remove this code publication normally as an icon in the system tray.

[WindowLeft] The screen Y location of the upper left of the publication window.

[WindowTop] The screen X location of the upper left of the publication window.

[WindowWidth] The width of the publication window including the border and scroll bars (if any).

[WindowHeight] The height of the publication window including the title bar, border and scroll bars (

[WindowState] The display state of the publication window (Normal, Minimized or Maximized).

[WindowOrder] The publication window's order as set in App Properties (Normal, OnTop or OnBotto

[WSHTimeOut] Set this variable to modify the timeout value (in seconds) used when executing JScript functions. Use "-1" to disable the timeout feature. (WSH stands for Win Host.)

Page Properties

[PageNumberOffset] Set this variable to offset page numbering sequence used to calculate the [PageNumber], [PageNumberLeft] and [PageNumberRight] variables. The offset will be subtracted from the page number. This is for display purposes only. The physical page number is not affected.

Drag & Drop

[DropAccept]
[DropX]
[DropY]
[DropTarget]

These variables can be used to influence how drag and drop operations are performed. See Polygon/Hotspot Tool for information about using these variables.

Folders

[CurrentDir] Contains the current active folder. Use in conjunction with the Run Action to specify an application's working directory. This will become the active directory when the application is launched. For example:

```
SetVar "[CurrentDir]" "c:\windows"
Run "C:\MyPrograms\MyApp.exe" "" "Normal"
```

To disable the Run action's use of the working directory, simply set [CurrentDir] to null. For example:

```
SetVar "[CurrentDir]" ""
```

Windows

[Clipboard] The contents of the Windows Clipboard (text format only). You can place text onto the Windows Clipboard using the SetVar Action. For example:

```
SetVar "[Clipboard]" "Place this on the clipboard."
```

[DecimalSymbol] The character stored in this variable is used to separate the integer part from the fractional part of a number. The default value is determined by the settings in the Windows Control Panel. Changing this variable only affects the current publication. It has no effect on Windows or other running applications. Plug-ins are also not affected unless they have been specifically designed to check for changes to this variable.

Mail/HTTP

[MailServer]	The name of the user's SMTP email server. Used for sending messages via the SendMail Action. If left blank, readers must enter the server address manually before sending email since it can be different on each computer system.
	
	The [MailServer] variable can be used to manually set the name of the reader's SMTP email server. By default, VisualNEO for Windows will attempt to detect the name of the server before sending an email message, and then ask the reader to confirm if the detected server name is correct. Setting [MailServer] variable to "Detect" prior to executing SendMail will skip the user confirmation if VisualNEO for Windows was able to detect the server name. For example:
	
	`SetVar "[MailServer]" "Detect"`
	`SendMail...`
	
	However, if VisualNEO for Windows is unable to detect the server name, the user will still be prompted to enter it manually.

[MailPort]	If your mail server requires that you use a port other than 25 (the default), you can use this variable to specify a port number. For example:
	
	`SetVar "[MailPort]" "80"`
	`SendMail...`

[MailUserID]
[MailUserPassword]
	Use these two variables to specify the user ID and password when using the SendMa Action with servers that require authentication. For example:
	
	`SetVar "[MailUserID]" "nancy123"`
	`SetVar "[MailUserPassword]" "applesauce"`
	`SendMail...`

[HTTPUserID]
[HTTPUserPassword]
	Use these two variables to specify the user ID and password when using any of the HTTP Actions with protected servers or folders. For example:
	
	`SetVar "[HTTPUserID]" "nancy123"`
	`SetVar "[HTTPUserPassword]" "applesauce"`
	`DownloadFile...`

[HTTPTimeOut]	Use this variable to specify the number of milliseconds VisualNEO for Windows should wait for an HTTP action to return before giving up. For example, to set the timeout to 400 milliseconds:
	
	`SetVar "[HTTPTimeOut]" "400"`
	`InternetGet...`
	
	To restore the default setting (wait forever), set [HTTPTimeOut] to null like this:
	
	`SetVar "[HTTPTimeOut]" ""`

[HTTPPort]	If your HTTP server requires that you use a port other than 80 (the default), you can use this variable to specify a port number. For example:
	
	`SetVar "[HTTPPort]" "1000"`
	`InternetGet...`

[HTTPAgent]	Advanced. Use this variable to specify the name of the application making the HTTP request. This name is used as the user agent in the HTTP protocol. The default value for the user agent is the publication title.
[HTTPReferrer]	Advanced. Use this variable to specify a URL to be passed to the server to identify the source (referrer) of the HTTP request. By default, the referrer is blank.
[DownloadProgress]	This variable contains a number (0-100) that represents the progress of the current HTTP Action. Long HTTP Actions can be aborted by changing the contents of this variable to "Cancel". For example: `SetVar "[DownloadProgress]" "Cancel"`

Using Special Characters

As you know, VisualNEO for Windows requires an Action's parameters to be surrounded by "quotes" and variables to be placed inside [brackets]. This makes it easy to write and understand VisualNEO for Windows Actions, but impossible to display one of those characters as part of a AlertBox message or write them to a data file. You can overcome this limitation using a special variable that lets you specify characters using their ASCII codes. For example, the quote character is ASCII #34, which you would specify in VisualNEO for Windows as [#34]. The # symbol tells VisualNEO for Windows that this variable represents an ASCII character. Using this feature within an Action command would look like this:

```
AlertBox "Hello" "Look [#34]quotes.[#34]"
```

Other special characters and their ASCII values include:

"	[#34]
[	[#91]
]	[#93]
|	[#124]
Carriage Return	[#13]
Line Feed	[#10]
Tab	[#9]

Action Command Reference

This section contains descriptions of Action commands that can be assigned to objects and pages. Many of these Actions are straightforward and easy to use, others are designed to perform more complex tasks and are intended for experienced authors. In order to better understand the concepts discussed here, it is recommended that you read Understanding Actions and Variables first.

Alphabetical list of Action commands:

A
AlertBox
ArticleJumpTo

B
Balloon
BringAppToFront
BrowserBack
BrowserExecScript
BrowserExport
BrowserFind
BrowserForward
BrowserGetElement
BrowserGoTo
BrowserHome
BrowserLoadFromStr
BrowserPrint

G
GetArrayInfo
GetMousePos
GetObjectHandle
GetObjectInfo
GetVolume
GetWindowPos
GIFPlay
GIFStop
GoSub
GotoFirstPage
GotoLastPage
GotoLine
GotoNextPage
GotoPage
GotoPageNum
GotoPrevPage

R
Random
RefreshObject
RegistryRead
RegistryWrite
RemoveFolder
Return
ReturnToPage
Run
RunInRectangle
RunNeoBook

S
SaveVariables
SearchStr
SendAppToBack
SendKeys

Navigation

Navigation

GotoPage

Purpose: Jump to a specific page in the publication.
Category: Navigation
Syntax: GotoPage "page title"
page title
The title of the page to display.
Example: `GotoPage "Contents"`

GotoPageNum

Purpose: Jump to a specific page number.
Category: Navigation
Syntax: GotoPageNum "page number"
page number
The number of the page to display. The first page is "1".
Example: `GotoPageNum "20"`

GotoPrevPage

Purpose: Jump to the previous page in the publication.
Category: Navigation
Syntax: GotoPrevPage
Example: `GotoPrevPage`

GotoNextPage

Purpose: Jump to the next page in the publication.
Category: Navigation
Syntax: GotoNextPage
Example: `GotoNextPage`

GotoFirstPage

Purpose: Jump to the publication's first page.
Category: Navigation
Syntax: GotoFirstPage
Example: `GotoFirstPage`

GotoLastPage

Purpose: Jump to the publication's last page.
Category: Navigation
Syntax: GotoLastPage

Example: `GotoLastPage`

ReturnToPage

Purpose: Return to the last page viewed by the reader.

Category: Navigation

Syntax: ReturnToPage

Example: `ReturnToPage`

Slide Show

StartSlideShow

Purpose: Display a series of pages as a slide show. Use the StopSlideShow Action to stop a playing slide show.

Category: Navigation

Syntax: StartSlideShow "pages" "delay" "options" "subroutine"

pages
A group of page titles separated with the pipe character "|". Pages will be displayed in the order listed. Use "All" instead to include the entire publication in the slide show.

delay
The number of milliseconds to pause between each page. A millisecond is one thousandth of a second. For a one second interval enter 1000 milliseconds. For one minute interval enter 60000 milliseconds.

options
Use "Loop" to play the slide show continuously, or leave blank to play the slide show once

subroutine (optional)
The name of a subroutine block to execute at the end of the slide show. Leave this blank if you do not wish to use a subroutine. (See App Properties > Actions for more information about subroutines.)

Example: `StartSlideShow "Jenny|Alice|Kenny" "5000" "Loop" ""`

PauseSlideShow

Purpose: Pause a playing slide show or resume a paused slide show. Slide shows are initiated with the StartSlideShow Action.

Category: Navigation

Syntax: PauseSlideShow

Example: `PauseSlideShow`

StopSlideShow

Purpose: Stop a running slide show initiated with StartSlideShow.

Category: Navigation

Syntax: StopSlideShow

Example: `StopSlideShow`

Messages/Interaction

Dialog Boxes

AlertBox

Purpose: Display a Windows style dialog box containing a title, message and an OK button.

Category: Messages/Interaction

Syntax: AlertBox "title" "message"

title
The dialog box title.

message
The dialog box message. Line breaks can be entered using the pipe character "|".

Example:
```
AlertBox "Hello World" "This is my first AlertBox."
```

Hint: *You can change the font and icon used by the AlertBox, MessageBox, InputBox and Exit dialogs in the Interface section of App Properties.*

MessageBox

Purpose: Display a Windows style dialog box containing a message and a custom set of buttons.

Category: Messages/Interaction

Syntax: MessageBox "title" "message" "buttons" "variable"

title
The dialog box title.

message
The dialog box message. Line breaks can be entered using the pipe character "|".

buttons
The captions of the custom buttons separated by the pipe "|" character.

variable
The name of the variable to store the number of the selected button. The first button = 1, the second = 2, and so on. The variable will contain a 0 (zero) if the reader closes the dialog box without making a selection.

Example: The following example displays a MessageBox asking readers to rate their mood. An AlertBox with an appropriate message based on their response is displayed.

```
MessageBox "Hello" "How do you feel?" "Good|Fair|Poor" "[Mood]"
If "[Mood]" "=" "1"
  AlertBox "Hello" "That's wonderful!"
EndIf
If "[Mood]" "=" "2"
  AlertBox "Hello" "I hope your mood improves."
EndIf
If "[Mood]" "=" "3"
  AlertBox "Hello" "That's too bad."
EndIf
If "[Mood]" "=" "0"
```

```
  AlertBox "Hello" "Wow, you must really be in a bad mood!"
EndIf
```

InputBox

Purpose: Display a simple input dialog box and request that the reader enter a single line of text.

Category: Messages/Interaction

Syntax: InputBox "title" "message" "variable"

title
The dialog box title.

message
The dialog box message. Line breaks can be entered using the pipe character "|".

variable
The name of the variable to store text entered by the reader.

Example:

```
InputBox "Welcome" "Enter your access code" "[Code]"
If "[Code]" "=" "JB5276H"
  GotoPage "Start"
Else
  AlertBox "Sorry" "That's not a valid access code."
  Exit "" ""
EndIf
```

FileOpenBox

Purpose: Allow the reader to select a file name using a standard Windows File Open dialog box.

Category: Messages/Interaction

Syntax: FileOpenBox "title" "file mask" "initial path" "variable" "options"

title
The dialog box title.

file mask
A file mask indicating the types of files that may be selected. A mask consists of two portions: the text that the reader will see and the code that tells Windows which types of files to display. The two portions must be separated by the pipe character "|". Typically the code portion contains an asterisk, a period and a three letter file extension matching the file type you want the reader to select. For example, to display only files ending in txt, your mask would look like this:

```
Text Files|*.txt
```

You can also use an asterisk instead of a file extension to display all types of files. For example:

```
Any File|*.*
```

initial path

The folder that will be displayed when the File Open dialog box first opens.

variable
The name of the variable to store the name of the selected file(s).

options
Leave this parameter empty to select a single file or enter "Multiple" here to allow readers to select more than one file.

Example:
```
FileOpenBox "Open" "Any File|*.*" "c:\" "[File]" ""
```

If you include the Multiple option, readers will be allowed to select more than one file. When multiple files are selected, each file name will be placed into the specified variable separated by semicolons. To extract the file names, you will need to use the StrParse Action. For example:

```
FileOpenBox "Open" "Any File|*.*" "c:\" "[Files]" "Multiple"
StrParse "[Files]" ";" "[Names]" "[Count]"
```

When executed, StrParse will create an array based on the [Names] variable ([Names1], [Names2], etc.). The number of elements in the array will be equal to the number of files selected.

FileSaveBox

Purpose: Allow the reader to specify a file name using a standard Windows File Save dialog box.

Category: Messages/Interaction

Syntax: FileSaveBox "title" "file mask" "initial directory" "variable"

See FileOpenBox for a description of these parameters.

Example:
```
FileSaveBox "Save" "Any File|*.*" "c:\" "[File]"
If "[File]" ">" ""
  FileWrite "[File]" "1" "[CustomerInfo]"
EndIf
```

FolderBox

Purpose: Display a Windows Folder Selector and allow the reader to make a selection.

Category: Messages/Interaction

Syntax: FolderBox "title" "variable"

title
The dialog box title.

variable
The name of the variable to store the selected folder.

Example:
```
FolderBox "Select a Folder" "[Folder]"
```

Popups

Balloon

Purpose: Display a comic book-style speech balloon containing a short message.

Category: Messages/Interaction

Syntax: Balloon "message" "left" "top" "delay"

message
The message you want to appear in the balloon.

left, top
The coordinates of the balloon's left, top corner relative to the publication window. To position the balloon at the location of the mouse pointer, use "-1" for both.

delay
The number of milliseconds to display the balloon or 0 (zero) to leave the balloon on screen until the reader presses an key or clicks the mouse.

Example: `Balloon "Here is a balloon message!" "-1" "-1" "0"`

StickyNote

Purpose: Display a yellow sticky note with a custom message.

Category: Messages/Interaction

Syntax: StickyNote "left" "top" "message" "delay"

left, top
The coordinates of the sticky note's left, top corner relative to the publication window. To center the sticky note on the screen, enter "-1" for both.

message
The message that will appear in the sticky note. Line breaks can be entered using the pipe character "|".

delay
The number of milliseconds to display the sticky note or 0 (zero) to leave the sticky note on screen until the reader presses an key or clicks the mouse.

Example: `StickyNote "-1" "-1" "Thank you for viewing my program." "2000"`

PopUpImage

Purpose: Display an image file with an optional special effect.

Category: Messages/Interaction

Syntax: PopUpImage "left" "top" "file name" "delay" "effect" "speed"

left, top
The coordinates of the image's left, top corner relative to the publication window. To center the image on the screen, enter "-1" for both.

file name
The name of the image file to display.

delay
The number of milliseconds to display the image or 0 (zero) to leave the image on screen until the reader presses an key or clicks the mouse.

effect
One of the following:

None, Dissolve, Slide Left, Slide Right, Slide Up, Slide Down, Explode, Implode, Weave Horizontal, Weave Vertical, Split Horizontal, Split Vertical, Wipe Left, Wipe Right, Wipe Up, Wipe Down, Circle, Grow, Blocks, Checkerboard, Block Dissolve, Fade, Page Turn or Transparent. (Plug-ins are also available to add additional effects.)

speed
The effect's speed (0 = fastest, 10 = slowest).

Example: The following example displays an image using the dissolve effect. The image will remain on screen until the mouse is clicked or a key is pressed.

```
PopUpImage "-1" "-1" "c:\samples\arrow.bmp" "0" "Dissolve" "3"
```

Menus

Menu

Purpose: Display a popup menu allowing the reader to make a selection from a list of items. A single Action for each menu choice must appear immediately following the Menu statement. For example, if the menu contains three items, then three Actions must follow the Menu Action. Only the Action corresponding to the menu choice will be executed. If nothing is selected from the menu, the all of the corresponding menu Actions will be skipped.

Category: Messages/Interaction

Syntax: Menu "left" "top" "items" "width" "lines"

left, top
The coordinates of the menu's left, top corner relative to the publication window. To center the menu on the screen, enter "-1" for both.

items
A list of menu choices separated by the pipe "|" character.

width
The desired with (in pixels) for the menu. To let VisualNEO for Windows pick the optimal width , enter "-1".

lines
The number of items to display before adding a vertical scroll bar to the menu. This can be useful with very large menus. To display as many items as will fit on-screen, enter "-1".

Example: The example below displays a menu containing three choices, then displays an AlertBox corresponding to the item selected:

```
Menu "-1" "-1" "Apples|Oranges|Grapes" "-1" "-1"
AlertBox "" "You have selected Apples"
AlertBox "" "You have selected Oranges"
AlertBox "" "You have selected Grapes"
```

MenuEx

Purpose: Display a popup menu, and stores the number of the selected item in a variable.

Category: Messages/Interaction

Syntax: MenuEx "left" "top" "items" "variable" "width" "lines"

left, top
The coordinates of the menu's left, top corner relative to the publication window. To center the menu on the screen, enter "-1" for both.

items
A list of menu choices separated by the pipe "|" character.

variable
The name of the variable to store the selected item. The first item = 1, the second = 2, and so on. The variable will contain a 0 (zero) if no selection is made.

width
The desired with (in pixels) for the menu. To let VisualNEO for Windows pick the optimal width , enter "-1".

lines
The number of items to display before adding a vertical scroll bar to the menu. This can be useful with very large menus. To display as many items as will fit on-screen, enter "-1".

Example:
```
MenuEx "-1" "-1" "Apples|Oranges|Grapes" "[Result]" "-1" "-1"
```

```
If "[Result]" "=" "1"
  AlertBox "" "You have selected Apples"
EndIf
If "[Result]" "=" "2"
AlertBox "" "You have selected Oranges"
EndIf
If "[Result]" "=" "3"
AlertBox "" "You have selected Grapes"
EndIf
```

Search Functions

Find

Purpose: Display a Find dialog box and allow the reader to search for words or phrases. The search may be limited to a group of pages, the current page, or the entire publication.

Category: Messages/Interaction

Syntax: Find "page title"

page title
A page title or group of titles separated with the pipe character "|". Pages will be searched in the order listed. Use "All" instead of a page title to search the entire publication, or "Current" to search only the current page.

Example: `Find "Contents|Introduction|index"`

FindFirst

Purpose: Search for the first occurrence of a word or phrase without displaying the Find dialog box. Use FindNext to repeat the search more than once.

Category: Messages/Interaction

Syntax: FindFirst "search text" "page title"

search text
The word or phrase you want to find.

page title
A page title or group of titles separated with the pipe character "|". Pages will be searched in the order listed. Use "All" instead of a page title to search the entire publication, or use "Current" to search only the current page.

Example: `FindFirst "apple" "Contents|Introduction|Index"`

FindNext

Purpose: Continue a search started with FindFirst.

Category: Messages/Interaction

Syntax: FindNext

Example: `FindNext`

Help Functions

HelpTopic

Purpose: Display a topic in the publication's help file. You must specify a help file in App Properties before using this command.

Category: Messages/Interaction

Syntax: HelpTopic "topic title"

topic title
The title of the help topic to display.

Example: `HelpTopic "Index"`

Shut Down

Exit

Purpose: Exit the publication with an optional confirming dialog box.

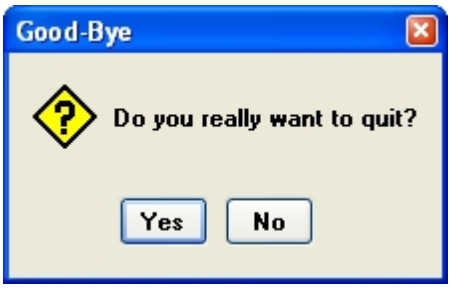
Category: Messages/Interaction

Syntax: Exit "title" "message"

title
The dialog box title.

message
The dialog box message. Line breaks can be entered using the pipe character "|". Leave the title and message fields blank to exit immediately without requesting confirmation.

Example: `Exit "Good-Bye" "Do you really want to quit?"`



Note: *You can change the font and icon used by the Exit box in the Interface section of the App Properties screen.*

Multimedia

Media Files

PlayCartoonFile

Purpose: Play an animation file created with VisualNEO for Windows's NeoToon utility.

Category: Multimedia

Syntax: PlayCartoonFile "start x" "start y" "end x" "end y" "file name" "times" "delay" "mode"

start x, start y, end x, end y
The screen coordinates where the animation will begin and end. The animation will travel between the two points. Coordinates are relative to the upper left corner of the publication window's.

file name
The name of the cartoon file (CAR) to play.

times
The number of times the animation will cycle between the start and end points.

delay

Number of milliseconds to pause between each frame.

mode

One of the following:

Normal	Begin playing the cartoon and continue executing the current script.
Wait	Play the cartoon and suspend VisualNEO for Windows until the cartoon finishes.
Loop	Play the cartoon continuously in the background.

You can use the StopMedia Action to halt a cartoon in progress.

Example: The following example plays a cartoon file that follows a trajectory from the top left of the publication to the lower right:

```
PlayCartoonFile "10" "10" "630" "470" "c:\samples\fish.car" "30" "25" "Normal"
```

You can create a cartoon file that remains stationary (does not follow a trajectory) by making the start and end coordinates the same. For example:

```
PlayCartoonFile "25" "10" "25" "10" "c:\samples\fish.car" "30" "25" "Loop"
```

PlaySoundFile

Purpose: Play a digital audio (WAVE) or music (MIDI) sound file.

Category: Multimedia

Syntax: PlaySoundFile "file name" "mode"

file name

The name of the sound file to play.

mode

One of the following:

Normal	Begin playing the sound and continue executing the current script.
Wait	Play the sound and suspend VisualNEO for Windows until the sound finishes.
Loop	Play the sound continuously in the background.

You can use the StopMedia Action to halt a sound in progress.

Example:
```
PlaySoundFile "c:\windows\media\ding.wav" "Loop"
```

PlayVideoFile

Purpose: Play a video (AVI/MPEG/etc.) file.

Category: Multimedia

Syntax: PlayVideoFile "left" "top" "width" "height" "file name" "mode"

left, top

The coordinates of the video's left, top corner relative to the publication window. To center the video on the screen, enter "-1" for both.

width, height

The desired with and height of the video. Substitute "-1" for both to use the video's actual dimensions.

file name

The name of the video file to play.

mode
One of the following:

Normal	Begin playing the video and continue executing the current script.
Wait	Play the video and suspend VisualNEO for Windows until the video finishes.
Loop	Play the video continuously in the background.

You can use the StopMedia Action to halt a video in progress.

Example: `PlayVideoFile "-1" "-1" "-1" "-1" "c:\samples\sales.avi" "Wait"`

Note: *AVI files are generally the best choice if you plan on including video in your publications. The necessary drivers for playing most types of AVI files are incorporated into Windows, while many other video formats require that special software be installed. For example, before playing MPEG videos, you (and your readers) may need to install Microsoft's ActiveMovie™ driver. ActiveMovie is included with most newer versions of Windows, but for older installations, you may need to download the driver from www.microsoft.com.*

StopMedia

Purpose: Stop a currently playing media file (sound, video or animation).

Category: Multimedia

Syntax: StopMedia "file name"

file name
The name of the media file to stop. Leave file name blank to stop all playing media. Use "Sound" instead of a file name to stop all audio (WAV, MID) files or "Video" to stop all video (AVI, CAR, etc.) files.

Example: `StopMedia "c:\sounds\frogs.wav"`

Windows MCI

MCICommand

Purpose: Send a command to the Windows Media Control Interface (MCI). This Action provides high-level access to multimedia devices attached to your computer. (See PlaySoundFile and PlayVideoFile for alternative methods of playing multimedia files.)

Category: Multimedia

Syntax: MCICommand "command string"

command string
A formatted MCI command string. Possible MCI commands include:

Play *filename/device* {start position} {stop position}
Pause *device*
Stop *device*
Open *device/filename* {alias}
Close *device*
Seek *device position*
Record *device*
Save *device filename*
Set *device* {function}

Device indicates the type of multimedia player the command is intended for. Items

surrounded in braces { } are optional. For more information about MCI commands, refer to the Windows SDK documentation available from Microsoft.

If the MCICommand is successful, the global variable [MCIResult] will contain a 0 (zero), otherwise [MCIResult] will contain an error number.

Example: The following example plays a simple wave file and displays an AlertBox if an error occurs:

```
MCICommand "play c:\windows\media\ding.wav"
If "[MCIResult]" "<>" "0"
  AlertBox "Error" "An error occurred: [MCIResult]"
EndIf
```

This example plays the sixth track on an audio CD:

```
MCICommand "open cdaudio"
MCICommand "set cdaudio time format tmsf"
MCICommand "play cdaudio from 6 to 7"
```

This example plays an AVI video in a window:

```
MCICommand "open demo.avi alias avi1"
MCICommand "play avi1 wait window"
MCICommand "close avi1"
```

Simple Sounds

PlayTone

Purpose: Play a tone (specified in Hertz) using the internal PC speaker.

Category: Multimedia

Syntax: PlayTone "frequency"

frequency

The frequency in Hertz of the tone to play. Middle "C" is approximately 440 Hertz.

Example: `PlayTone "440"`

SoundBuzzer

Purpose: Play an error-type tone on the internal PC speaker.

Category: Multimedia

Syntax: SoundBuzzer

Example: `SoundBuzzer`

SystemSound

Purpose: Play a default system sound defined in the Windows Registry. System sounds can vary from computer to computer since Windows allows users to customize these settings to suit their individual tastes. These sounds may be changed using the Windows Control Panel.

Category: Multimedia

Syntax: SystemSound "sound type"

Sound type

One of the following:

SystemDefault, SystemAsterisk, SystemExclamation, SystemStop or SystemQuestion.

Example: `SystemSound "SystemExclamation"`

Volume Control

GetVolume

Purpose: Get the speaker volume level for a specific device. Windows provides separate volume controls for each audio device. Device types that can be accessed with this Action include: Digital Audio (Wave), Musical Instrument (Midi), CD, Line Input (LineIn), Microphone, Auxiliary (AUX) and Master Volume (Master).

Category: Multimedia

Syntax: GetVolume "device" "variable"

device
The name of an audio device: Wave, Midi, CD, LineIn, Microphone, Aux or Master.

variable
The name of the variable to store the device volume. The volume level retuned will be between 0 (off) and 255 (maximum).

Example: `GetVolume "WAVE" "[Vol]"`

SetVolume

Purpose: Set the speaker volume level for a specific device. Windows provides separate volume controls for each audio device. Device types that can be controlled with this Action include: Digital Audio (Wave), Musical Instrument (Midi), CD, Line Input (LineIn), Microphone, Auxiliary (AUX) and Master Volume (Master).

Category: Multimedia

Syntax: SetVolume "device" "volume level"

device
The name of an audio device: Wave, Midi, CD, LineIn, Microphone, Aux or Master.

volume level
The new volume level - a number between 0 (off) and 255 (maximum).

Example: `SetVolume "WAVE" "100"`

Files

File Management

FileCopy

Purpose: Copy an external file.

Category: Files

Syntax: FileCopy "source file" "destination"

source file
The name of an existing external file on the reader's computer.

destination
The drive and folder on the reader's computer where the copied file is to be saved. You may optionally include a file name as part of the destination to save the copied file under a different name.

Example: `FileCopy "[PubDir]test.doc" "c:\my documents"`

FileErase

Purpose: Erase an external file.

Category: Files

Syntax: FileErase "file name"

file name
The name of an existing external file.

Example:
```
FileErase "c:\sample files\mydata.bak"
```

FileExists

Purpose: Determine if an external or embedded file exists.

Category: Files

Syntax: FileExists "file name" "variable"

file name
The name of the file (including drive and path) to check.

variable
The name of the variable to store the result of the search. If the file exists, the variable will be set to “True”, otherwise, it will be set to “False”.

Example: The example below displays an alert box if the file mydata.dat exists:

```
FileExists "c:\sample files\mydata.dat" "[Result]"
If "[Result]" "=" "True"
  AlertBox "Status" "The file was found!"
EndIf
```

FileSize

Purpose: Get the size (in bytes) of an external or embedded file.

Category: Files

Syntax: FileSize "file name" "variable"

file name
The name of the file (including drive and path) to check.

variable
The name of the variable to store the file's size. If the file does not exist, the variable will be cleared and an error message displayed.

Example: The example below displays an alert box containing the size of the file sample.txt:

```
FileSize "[PubDir]sample.txt" "[Size]"
AlertBox "Result" "The file's size is [Size]"
```

FolderExists

Purpose: Determine if an external folder exists.

Category: Files

Syntax: FolderExists "path" "variable"

path
The name of the folder to check.

variable
The name of the variable to store the result of the search. If the folder exists, the variable will be set to “True”, otherwise, it will be set to “False”.

Example: `FolderExists "C:\Program Files" "[Result]"`

CreateFolder

Purpose: Create a new folder.

Category: Files

Syntax: CreateFolder "folder name"

folder name
The path and name of the new folder.

Example: `CreateFolder "C:\MyFolder"`

RemoveFolder

Purpose: Remove an existing empty folder from the computer's hard drive.

Category: Files

Syntax: RemoveFolder "folder name"

folder name
The name of the folder to remove.

Example: `RemoveFolder "c:\myfolder"`

ExtractFile

Purpose: Extract a file from your compiled publication.

Category: Files

Syntax: ExtractFile "source file" "destination"

source file
The name of the source file to be extracted. This file will be stored inside your compiled publication.

destination
The drive and folder on the reader's computer where the extracted file is to be saved. You may optionally include a file name to save the extracted file under a different name.

Example: `ExtractFile "c:\samples\template.doc" "c:\my documents"`

FileList

Purpose: Generate a list of files and/or folders found in the specified path.

Category: Files

Syntax: FileList "file mask" "options" "variable"

file mask
The path and file name mask to search. The mask can include wildcard characters. For example, "C:\Sample*.*" will list all files in the C:\Sample folder.

options
Can include any combination of the following:

Files	Include files in the list.
Folders	Include folders in the list
NoExt	Remove extensions from file names before adding them to the list.

variable
The name of the variable to store the list. Multiple files will be separated by carriage returns.

Example: The following example compiles a list of all JPEG images in the "C:\My Photos" folder and stores them in a variable called [Pics]:

```
FileList "C:\My Photos\*.jpg" "Files+NoExt" "[Pics]"
```

File I/O

FileRead

Purpose: Read data from an external text file.

Category: Files

Syntax: FileRead "file name" "line number" "variable"

file name
The name of an existing external file.

line number
The number of the line in the file to read. The first line in the file is 1, the second is 2 and so on. You can specify "All" here instead of a line number to read the entire file at once.

variable
The name of the variable to store the data.

Example: The following example uses FileRead to build a menu from the contents of a file called "Parts.dat". The file is read one line at a time and the contents added to [MenuChoices] to build the menu. Finally, the menu is displayed using the MenuEx Action command.

```
SetVar "[MenuChoices]" ""
FileLen "parts.dat" "[MaxLine]"
Loop "1" "[MaxLine]" "[Count]"
  FileRead "parts.dat" "[Count]" "[MenuItem]"
  If "[MenuChoices]" ">" ""
    SetVar "[MenuChoices]" "[MenuChoices]|"
  EndIf
  SetVar "[MenuChoices]" "[MenuChoices][MenuItem]"
EndLoop
MenuEx "-1" "-1" "[MenuChoices]" "[Result]"
```

Another way of doing the same thing would be to use "All" in place of a line number, which instructs FileRead will load the entire file at once:

```
FileRead "parts.dat" "All" "[MenuChoices]"
MenuEx "-1" "-1" "[MenuChoices]" "[Result]"
```

FileWrite

Purpose: Write data to an external text file.

Category: Files

Syntax: FileWrite "file name" "line number" "data"

file name
The name of an existing external file. You can insert the name of a communications port (COM1, COM2, etc.) here instead of a file name to send data to an external device instead.

line number
The number of the line in the file to write to. The first line in the file is 1, the second is 2 and so on. You can specify "Append" here instead of a line number to add the data to the end of the file. Use "All" here to replace the entire file at once.

data
The data to be written to the file.

Example: The example below replaces the first line of the parts.dat file:

```
FileWrite "parts.dat" "1" "Calvin Klein Blazer"
```

By default, FileWrite will convert any pipe characters "|" contained within the text into carriage returns/line breaks. To prevent this, add an exclamation point character "!" to the beginning of the data parameter. For example:

```
FileWrite "filename" "All" "![Data]"
```

FileInsLine

Purpose: Insert a blank line into an external text file. The external file must be in plain text/ASCII format.

Category: Files

Syntax: FileInsLine "file name" "line number"

file name
The name of an existing external file.

line number
The number of the line in the file where the new line should be inserted. All lines from this point forward will be moved down to make room. The first line in the file is 1, the second is 2 and so on.

Example: The following example inserts a new first line in the file parts.dat:

```
FileInsLine "parts.dat" "1"
```

FileDelLine

Purpose: Delete a specific line from an external text file. The external file must be in plain text/ASCII format.

Category: Files

Syntax: FileDelLine "file name" "line number"

file name
The name of an existing external file.

line number
The number of the line in the file to delete. The first line in the file is 1, the second is 2 and so on.

Example: In the following example, readers are asked if they want to delete an item which is stored on a line in the parts.dat file. Their response is stored in a variable named "[Ans]". If the response is Yes (1), the line is removed from the file.

```
MessageBox "Delete" "Delete #[Item]?" "Yes|No" "[Ans]"
If "[Ans]" "=" "1"
  FileDelLine "parts.dat" "[Item]"
EndIf
```

FileLen

Purpose: Count the number of lines contained in an external text file.

Category: Files

Syntax: FileLen "file name" "variable"

file name
The name of an existing external file.
variable
The name of a variable to store the line count.

Example: The example below calculates the size of the parts.dat file and stores the number of lines in the [Lines] variable:

```
FileLen "parts.dat" "[Lines]"
```

FileToVar

Purpose: Read the contents of a file into a variable. This is similar to using FileRead with the "All" option, but unlike FileRead, FileToVar can be used with Embedded files.

Category: Files

Syntax: FileToVar "file name" "variable"

file name
The name of an existing external or embedded file.
variable
The name of the variable to store the contents of the file.

Example: `FileToVar "[Embedded]dealers.txt" "[TextEntry1]"`

File Name Utiltities

ExtractFilePath

Purpose: Extract the drive and folder portion of a file name.

Category: Files

Syntax: ExtractFilePath "file name" "variable"

file name
The name of a file.
variable
The name of the variable to store the file's path. If the file doesn't have a path, the variable will be empty.

Example: This example places "c:\samples" in the [Path] variable:

```
ExtractFilePath "c:\samples\test.doc" "[Path]"
```

ExtractFileName

Purpose: Extract the name and extension portion of a file name and store the result in a variable.

Category: Files

Syntax: ExtractFileName "file name" "variable"

file name
The name of a file.
variable
The name of the variable to store the file's name.

Example: This example places "test.doc" in the [Name] variable:

```
ExtractFileName "c:\samples\test.doc" "[Name]"
```

ExtractFileExt

Purpose: Extract the extension portion of a file name. See also ChangeFileExt.

Category: Files

Syntax: ExtractFileExt "file name" "variable"

file name
The name of a file.

variable
The name of the variable to store the file's extension.

Example: This example places ".doc" in the [Ext] variable:

```
ExtractFileExt "c:\samples\test.doc" "[Ext]"
```

ExtractFileDrive

Purpose: Extract the drive portion of a file name and store the result in a variable.

Category: Files

Syntax: ExtractFileDrive "file name" "variable"

file name
A complete file name including drive and path.

variable
The name of the variable to store the drive information.

Example: This example places "c:" in the [Drive] variable:

```
ExtractFileDrive "c:\samples\test.doc" "[Drive]"
```

ChangeFileExt

Purpose: Change a file name's extension and store the result in a variable.

Category: Files

Syntax: ChangeFileExt "file name" "extension" "variable"

file name
A file name

extension
The new file extension including a period. For example: ".txt"

variable
The name of the variable to store the modified file name.

Example:
```
ChangeFileExt "sample.dat" ".txt" "[FileName]"
```

Registry Utilities

RegistryRead

Purpose: Read a value from the Windows System Registry database. The Registry contains information about the hardware and software installed on the computer.

Category: Files

Syntax: RegistryRead "key" "section" "variable"

key
The Registry key that contains the data you want to read. The Registry database is divided into the following key groups:

HKEY_CURRENT_USER

HKEY_CLASSES_ROOT
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
HKEY_DYN_DAT

Most software applications store their information in the HKEY_CURRENT_USER key.

section
The Registry section that contains the actual value to read.

variable
The name of the variable to store the retrieved data.

Example: `RegistryRead "HKEY_CURRENT_USER" "Software\MyPub\UserName" "[User]"`

RegistryWrite

Purpose: Write a value to the Windows System Registry database. The Registry contains information about the hardware and software installed on the computer.

WARNING: Exercise caution when modifying the Registry. If there is an error in the Registry, your computer (or your reader's computer) may become nonfunctional.

Category: Files

Syntax: RegistryWrite "key" "section" "value"

key
The Registry key that contains the section you want to modify. See RegistryRead for a list of available keys.

section
The Registry section to modify.

value
The data to write to the Registry.

Example: `RegistryWrite "HKEY_CURRENT_USER" "Software\MyPub\UserName" "Joe"`

Printing

Printing

PrintPage

Purpose: Print a specific page from this publication.

Category: Printing

Syntax: PrintPage "pages" "mode"

page title
A page title or group of titles separated with the pipe character "|". Pages will be printed in the order listed. Use "All" instead of a page title to print the entire publication, or "Current" to print only the current page.

mode
One of the following:

Draft Pages are printed in low resolution mode. Produces a very accurate representation of the screen, but because the screen resolution is relatively low, some elements may appear rough.

Final Pages are printed in high resolution mode. Not as accurate as draft

mode, but most elements (especially lines and text) appear crisp and clean.

Example: `PrintPage "Contents|Introduction|Help" "Final"`

PrintImageFile

Purpose: Print an image file (BMP/PCX/GIF/etc.). For more control over the image size, use PrintImageFileWH instead.

Category: Printing

Syntax: PrintImageFile "file name" "scale factor"

file name

The name of the image file to print.

scale factor

This number determines how large the image will appear when printed. You can use the scale factor to compensate for differences in resolution between the screen and the printer. To print an image 1:1 using its normal resolution enter "100%" as the scale factor. To enlarge the printout, enter a number greater than 100. To automatically stretch the image to fit the dimensions of the paper, set the scale factor to "-1".

Example: `PrintImageFile "c:\samples\map.bmp" "300%"`

PrintImageFileWH

Purpose: Print an image file (BMP/PCX/GIF/etc.) using a specific output size. This Action allows you to precisely control the size of the printed image regardless of the resolution of the destination printer.

Category: Printing

Syntax: PrintImageFileWH "file name" "output width" "output height"

file name

The name of the image file to print.

output width, output height

The desired output width and output height of the printed image in inches. You can optionally set either the output width or output height to 0 (zero) to automatically calculate that dimension based on the aspect ratio of the image.

Example: `PrintImageFileWH "c:\samples\map.bmp" "3.75" "2.5"`

PrintTextFile

Purpose: Print a plain text (ASCII/ANSI) or formatted Rich Text (RTF) file.

Category: Printing

Syntax: PrintTextFile "header" "footer" "file"

header, footer

The text to appear at the top (header) and bottom (footer) of each page. The following codes may be incorporated into the header and footer parameters:

&f File name
&p Page number
&t Current time
&d Current date
&n Total page count
&& Prints the & symbol

file name
The name of the file to print.

Example: `PrintTextFile "Help" "Page &p" "c:\samples\intro.txt"`

PrintDataFile

Purpose: Print an external plain text (ASCII) data file (usually created with the FileWrite Action).

Category: Printing

Syntax: PrintDataFile "header" "footer" "file name"

header, footer
The text to appear at the top (header) and bottom (footer) of each page. The following codes may be incorporated into the header and footer parameters:

&f File name
&p Page number
&t Current time
&d Current date
&n Total page count
&& Prints the & symbol

file name
The name of the external text file to print.

Example: `PrintDataFile "XYZ Corp. Parts List &t" "Page &p" "parts.dat"`

Setup

SetPrinterOrientation

Purpose: Set the default printer orientation (Portrait or Landscape).

Category: Printing

Syntax: SetPrinterOrientation "orientation"

orientation
The paper orientation. Use "Portrait" to print vertically or "Landscape" to print horizontally

Example: `SetPrinterOrientation "Landscape"`

PrintSetup

Purpose: Display the Windows Printer Setup dialog box and allow the reader to OK or cancel a print request.

Category: Printing

Syntax: PrintSetup "variable"

variable
The name of the variable to store the reader's response. If the reader clicks the dialog's "OK" button, the variable will contain "True", meaning it's OK to go ahead an begin printing. If the reader clicks the dialog's "Cancel" button, the variable will contain "False" meaning don't print.

Normally, print Actions (PrintPage, PrintTextFile, etc.) automatically display the Windows Printer Setup screen and ask the reader for confirmation before printing. There are times when a publication's author may wish to control when the Printer Setup screen appears. For this purpose, VisualNEO for Windows provides the PrintSetup Action and an option

under App Properties to disable the automatic display of the Print Setup screen.

Example: The example below assumes that the Print Setup screen has been turned off in App Properties:

```
PrintSetup "[Result]"
If "[Result]" "=" "True"
  PrintPage "Contents"
EndIf
```


Strings

String Utilities

StrIns

Purpose: Insert characters into a string.

Category: Strings

Syntax: StrIns "source string" "dest string" "insert position" "variable"

source string
The characters to be inserted.

dest string
The destination string.

insert position
The position in dest string where the characters will be inserted.

variable
The name of a variable to store the modified string.

Example: The following example will insert the contents of the variable [Name] into a string and store the result in a variable called [Greeting]:

```
StrIns "[Name]" "Hello . How are you?" "7" "[Greeting]"
```

StrDel

Purpose: Delete characters from a string.

Category: Strings

Syntax: StrDel "source string" "start position" "length" "variable"

source string
The original string containing the characters to delete.

start position
The position of the first character to delete.

length
The number of characters to delete.

variable
The name of a variable to store the modified string.

Example: The example below searches for spaces in the string assigned to the variable [Name] and if found removes all characters that follow. The modified string is placed back into the [Name] variable:

```
SearchStr " " "[Name]" "[SpacePos]"
If "[SpacePos]" ">" "0"
  Strlen "[Name]" "[Len]"
```

```
  StrDel "[Name]" "[SpacePos]" "[Len]-[SpacePos]+1" "[Name]"
EndIf
```

StrLen

Purpose: Calculate the length of a string.

Category: Strings

Syntax: StrLen "string" "variable"

string
The string to examine.

variable
The name of a variable to store the string's length.

Example: In the example below, the variable [Name] contains the string "Bernard". Since this string contains seven characters, the example Action will place "7" into the variable [Size]:

```
StrLen "[Name]" "[Size]"
```

SubStr

Purpose: Copy a portion of a string.

Category: Strings

Syntax: SubStr "source string" "start position" "length" "variable"

source string
The original string containing the characters to copy.

start position
The position of the first character to copy.

length
The number of characters to copy.

variable
The name of a variable to store the copied text.

Example: The example below, copies this president's middle name (Quincy) into a variable called [MiddleName]:

```
SubStr "John Quincy Adams" "6" "6" "[MiddleName]"
```

SearchStr

Purpose: Search for characters within a text string.

Category: Strings

Syntax: SearchStr "search for" "string" "variable" "options"

search for
The characters to find.

string
The text string to search.

variable
The name of a variable to store the position of the found characters. The variable will contain 0 (zero) if the characters are not present in the string.

options
Enter "**CaseSensitive**" here to search using the exact characters entered. Leave this parameter blank to perform a case insensitive search (upper and lower case characters are treated the same).

Example: The following example searches for and removes all spaces from the variable [Name]:

```
SearchStr " " "[Name]" "[SpacePos]"
While "[SpacePos]" ">" "0"
  StrDel "[Name]" "[SpacePos]" "1" "[Name]"
  SearchStr " " "[Name]" "[SpacePos]" ""
EndWhile
```

StrUpper

Purpose: Convert the contents of a string to upper case. Numbers and punctuation within the string are not affected.

Category: Strings

Syntax: StrUpper "string" "variable"

string
The string to convert.

variable
The name of a variable to store the modified string.

Example: `StrUpper "good dog" "[Result]"`

StrLower

Purpose: Convert the contents of a string to lower case. Numbers and punctuation within the string are not affected.

Category: Strings

Syntax: StrLower "string" "variable"

string
The string to convert.

variable
The name of a variable to store the modified string.

Example: `StrLower "GOOD DOG" "[Result]"`

StrReplace

Purpose: Replace all occurrences of a character or substring within a string.

Category: Strings

Syntax: StrReplace "string" "old chars" "new chars" "variable" "options"

string
The original string.

old chars
The characters to replace.

new chars
The replacement characters.

variable
The name of a variable to store the modified string.

options
Enter "**CaseSensitive**" here to search using the exact characters entered. Leave this parameter blank to perform a case insensitive search (upper and lower case characters are treated the same).

Example: The following example replaces each occurrence of the word "dog" in the source string with "cat":

```
StrReplace "My dog is a good dog." "dog" "cat" "[Result]" ""
```

You can also use this Action to remove a character by leaving the replacement characters parameter empty. For example:

```
StrReplace "My doggy is a good doggy." "gy" "" "[Result]" ""
```

StrParse

Purpose: Separate a string into multiple parts using a delimiter character.

Category: Strings

Syntax: StrParse "source string" "delimiter" "array variable" "count variable"

source string
The string containing the information to parse.

delimiter
The character (or characters) used to separate the elements of the string.

array variable
The name of the variable array to store the parsed elements.

count variable
The name of a variable to store the number of elements stored in the array.

Each element in the string must be separated by the delimiter character (semicolon, comma, hyphen, etc.). StrParse will use the delimiter to divide the source string into individual elements which are placed into an array based on the array variable. The count variable will be set to the number of elements stored in the array.

Example: The example below takes a list of file names from the FileOpenBox Action and separates them into individual files:

```
FileOpenBox "Open" "Any File|*.*" "c:\" "[Files]" "Multiple"
StrParse "[Files]" ";" "[Names]" "[Count]"
```

When executed, an array based on the [Names] variable ([Names1], [Names2], etc.) will be created. Each element in the array will contain one file name. The number of elements in the array ([Count]) will be equal to the number of files selected.

PopulateStr

Purpose: Populate variables contained within a string. VisualNEO for Windows populates most strings automatically. The exception being strings that come from an external source such as a file loaded with the FileRead or FileToVar actions. After reading, if the file contained VisualNEO fo Windows style variables, you can use PopulateStr to replace all of the variables in the string with the contents of those variables.

Category: Strings

Syntax: PopulateStr "string" "variable"

string
The source string containing variables to populate.

variable
The name of a variable to store the modified string.

Example:

```
FileToVar "[PubDir]sample.html" "[RawHTML]"
PopulateStr "[RawHTML]" "[PopulatedHTML]"
BrowserLoadFromStr "WebBrowser1" "[PopulatedHTML]"
```

Objects

All Objects

ShowObject

Purpose: Show a hidden object with an optional animated effect.

Category: Objects

Syntax: ShowObject "object name" "effect" "speed"

object name
The name of an existing object.

effect
One of the following:

None, Dissolve, Slide Left, Slide Right, Slide Up, Slide Down, Explode, Implode, Weave Horizontal, Weave Vertical, Split Horizontal, Split Vertical, Wipe Left, Wipe Right, Wipe Up, Wipe Down, Circle, Grow, Blocks, Checkerboard, Block Dissolve, Fade, Page Turn or Transparent. (Plug-ins are also available to add additional effects.)

speed
The effect's speed (0 = fastest, 10 = slowest).

Example:
```
ShowObject "PushButton1" "Dissolve" "3"
```

HideObject

Purpose: Hide an object with an optional animated effect.

Category: Objects

Syntax: HideObject "object name" "effect" "speed"

object name
The name of an existing object.

effect
One of the following:

None, Dissolve, Slide Left, Slide Right, Slide Up, Slide Down, Explode, Implode, Weave Horizontal, Weave Vertical, Split Horizontal, Split Vertical, Wipe Left, Wipe Right, Wipe Up, Wipe Down, Circle, Grow, Blocks, Checkerboard, Block Dissolve, Fade, Page Turn or Transparent. (Plug-ins are also available to add additional effects.)

speed
The effect's speed (0 = fastest, 10 = slowest).

Example:
```
HideObject "PushButton1" "Dissolve" "3"
```

EnableObject

Purpose: Enable an object. An enabled object can respond to mouse and keyboard events. Use in conjunction with DisableObject.

Category: Objects

Syntax: EnableObject "object name"

object name
The name of an existing object.

Example:
```
EnableObject "PushButton1"
```

DisableObject

Purpose: Disable an object. Disabled objects ignore all mouse and keyboard events. Disabled objects appear shaded indicating to readers that they are not available for use.

Category: Objects

Syntax: DisableObject "object name"

object name
The name of an existing object.

Example:
```
DisableObject "PushButton1"
```

GetObjectInfo

Purpose: Obtain information about an object such as its current size, position, visible state, etc. This is useful if you have changed the state of an object and you wish to determine its current condition.

Category: Objects

Syntax: GetObjectInfo "object name" "info type" "variable"

object name
The name of an existing object.

info type
One of the following: Visible, Enabled, Left, Top, Width, Height, FileName, SelectedText, CursorPosition, ImageWidth, ImageHeight, MediaLength, Caption, AbsLeft, AbsTop, FillColor, FillStyle, FillTransparent, LineColor, LineStyle, LineWidth, FontColor, FontName, FontSize, FontStyle, FontCharSet.

Note: *Use Left and Top to obtain the object's position relative to its parent. If the object is attached to a Container then the position returned will be relative to the Container's upper left corner. If the object is on the page surface, then the position returned will be relative to the upper left corner of the publication window's client area. Use AbsLeft and AbsTop to obtain the object's position relative to the publication's window regardless of its parent.*

variable
The name of the variable to store the requested information. If the info type is Visible or Enabled, the variable will contain either "True" or "False". Left, Top, Width, Height, CursorPosition, ImageWidth, ImageHeight and MediaLength return numeric values. FileName, SelectedText and Caption return text.

Not all objects will respond to every type of information request. For example, FileName only works for objects that display files (Picture, Article, etc.). SelectedText can be used to copy highlighted from Text Entry or Article objects. CursorPosition returns the position of the cursor for Text Entry objects only. ImageWidth and ImageHeight only work with Picture objects. MediaLength is limited to Media Player and Flash Player objects.

Example:
```
GetObjectInfo "PushButton1" "Visible" "[Status]"
If "[Status]" "=" "False"
  Showobject "PushButton1" "Dissolve" "3"
EndIf
```

MoveObject

Purpose: Change an object's position on the screen.

Category: Objects

Syntax: MoveObject "object name" "x position " "y position"

object name
The name of an existing object.

x position, y position
The new left/top coordinates for the object. Coordinates are relative to the upper left corner of the publication window. If the object is attached to a container then the coordinates are relative to the container's upper left corner.

Example: This example moves object Rectangle1 to the upper left corner of the publication:
```
MoveObject "Rectangle1" "5" "5"
```

MoveObjectAlongPath

Purpose: Move an object along a predefined trajectory.

Category: Objects

Syntax: MoveObjectAlongPath "object name" "trajectory" "speed" "mode" "subroutine"

object name
The name of an existing object.

trajectory
The path that the object will follow as it moves. The trajectory contains a list of screen coordinates separated by commas. Coordinates must be listed in X/Y pairs. For example: "X1,Y1,X2,Y2,X3,Y3,etc.". Coordinates are relative to the upper left corner of the publication's window which is 0,0. If the object is attached to a container then the coordinates are relative to the container's upper left corner.

speed
The speed at which the objects moves along the trajectory. The actual speed is somewhat dependant on the speed of the readers computer and graphics card.

mode
One of the following:

Normal	Start the move and continue executing the current script.
Wait	Start the move and suspend VisualNEO for Windows until the move is complete.
Loop	Repeat the move continuously in the background.

Additionally, you may combine one of the choices above with:

Restore	Move the object back to its original location once the move is complete.
Center	Align the object's center along the trajectory. This is the default.
LeftTop	Align the object's upper-left corner along the trajectory.

subroutine (optional)
The name of a subroutine block to execute after the move is complete. (See App Properties > Actions for more information about subroutines.)

Example:
```
MoveObjectAlongPath "Picture1" "589,135,74,303,405,365" "10" "Normal+Restore+Center" ""
```

StopMovingObject

Purpose: Stop a moving object. The object's movement must have been initiated with the MoveObjectAlongPath Action.

Category: Objects

Syntax: StopMovingObject "object name"

object name
The name of an existing object. Leave the object name blank to stop all moving objects.

Example:
```
StopMovingObject "Picture1"
```

SizeObject

Purpose: Change the width and height of an object.

Category: Objects

Syntax: SizeObject "object name" "width" "height"

object name
The name of an existing object.

width, height
The object's new width and height. Use "-1" to keep the object's current dimensions.

Example: `SizeObject "Rectangle1" "150" "100"`

ObjectToFront

Purpose: Move an object to the foreground in front of all other objects on the page.

Category: Objects

Syntax: ObjectToFront "object name"

object name
The name of an existing object.

Example: `ObjectToFront "Picture1"`

ObjectToBack

Purpose: Move an object to the background behind all other objects on the page.

Category: Objects

Syntax: ObjectToBack "object name"

object name
The name of an existing object.

Example: `ObjectToBack "Picture1"`

FocusObject

Purpose: Set the keyboard input focus to a specific object. This Action is primarily used to activate Text Entry objects. Objects that don't accept keyboard input are not affected by this Action.

Category: Objects

Syntax: FocusObject "object name"

object name
The name of an existing object.

Example: `FocusObject "TextEntry1"`

RefreshObject

Purpose: Update the contents of an object. Use this Action if the contents of an object change while your publication is running. Objects that don't support refreshing will ignore this Action.

Category: Objects

Syntax: RefreshObject "object name"

object name
The name of an existing object.

Example: `RefreshObject "WebBrowser1"`

GetObjectHandle

Purpose: Get an object's Windows Handle (HWND). This Action is intended primarily for use by Plug-In developers.

Category: Objects

Syntax: GetObjectHandle "object name" "variable"

object name
The name of an existing object.

variable
The name of the variable to store the object's handle.

Example:
```
GetObjectHandle "ListBox1" "[Handle]"
```

SetObjectFill

Purpose: Set an object's fill color and style.

Category: Objects

Syntax: SetObjectFill "object name" "fill color" "fill style" "transparent"

object name
The name of an existing object.

fill color
The fill color. See Defining Colors.

fill style
The desired fill style. Use "Solid" for an opaque object, "Hollow" for a transparent object, or a number between "2" and "48" to use one of VisualNEO for Windows's built-in patterns

transparent
If a fill pattern is selected above, use "True" to draw the pattern transparently, or "False" to draw it normally.

Example:
```
SetObjectFill "PushButton12" "248,192,160" "15" "False"
```

SetObjectLine

Purpose: Set an object's line color, width and style.

Category: Objects

Syntax: SetObjectLine "object name" "color" "width" "style"

object name
The name of an existing object.

color
The line color. See Defining Colors.

width
The line width or 0 (zero) for none.

style
The line style. Use 0 (zero) for solid, or numbers 1 through 6 for other styles.

Example:
```
SetObjectLine "Rectangle1" "Red" "3" "0"
```

SetObjectFont

Purpose: Set an object's font name, size and style. This only affects objects that have caption or text properties.

Category: Objects

Syntax: SetObjectFont "object name" "color" "name" "size" "style" "charset"

object name
The name of an existing object.

color
The font color. See Defining Colors.
name
The font name.
size
The size of the font in points.
style
The font style. This can be "Normal" or any combination of the following: Bold, Italic, Underline or Strikeout.
charset
The preferred character set for the font. The default setting is "DEFAULT_CHARSET".

Example: `SetObjectFont "PushButton1" "Black" "Arial" "10" "Bold+Italic" "DEFAULT_CHARSET"`

SetObjectCaption

Purpose: Replace an object's caption text. This only affects objects that have caption properties.

Category: Objects

Syntax: SetObjectCaption "object name" "caption"

object name
The name of an existing object.
caption
The text to be assigned to the object's caption.

Example: `SetObjectCaption "PushButton1" "Click Me"`

SetObjectFileName

Purpose: Set an object's file name. This only affects objects that have file name properties.

Category: Objects

Syntax: SetObjectFileName "object name" "file name"

object name
The name of an existing object.
file name
The name of the file to be loaded into the object.

Example: `SetObjectFileName "Picture1" "[PubDir]customer.jpg"`

Animated GIF

GIFPlay

Purpose: Begin playing an Animated GIF object.

Category: Objects

Syntax: GIFPlay "object name" "loop count"

object name
The name of an existing Animated GIF object.
loop count
The number of times to repeat the animation sequence or 0 (zero) to play the animation continuously.

Example: `GIFPlay "AnimatedGif1" "0"`

GIFStop

Purpose: Stop a playing Animated GIF object.
Category: Objects
Syntax: GIFStop "object name"
object name
The name of an existing Animated GIF object.
Example: `GIFStop "AnimatedGIF1"`

Media Player

MediaPlayerPlay

Purpose: Play a media player object.
Category: Objects
Syntax: MediaPlayerPlay "object name"
object name
The name of an existing Media Player object.
Example: `MediaPlayerPlay "MediaPlayer1"`

MediaPlayerStop

Purpose: Stop a playing Media Player object.
Category: Objects
Syntax: MediaPlayerStop "object name"
object name
The name of an existing Media Player object.
Example: `MediaPlayerStop "MediaPlayer1"`

MediaPlayerPause

Purpose: Pause a playing Media Player object.
Category: Objects
Syntax: MediaPlayerPause "object name"
object name
The name of an existing Media Player object.
Example: `MediaPlayerPause "MediaPlayer1"`

MediaPlayerRewind

Purpose: Rewind a Media Player object.
Category: Objects
Syntax: MediaPlayerRewind "object name"
object name
The name of an existing Media Player object.
Example: `MediaPlayerRewind "MediaPlayer1"`

Timer

TimerStart

Purpose: Manually start a Timer object.

Category: Objects

Syntax: TimerStart "object name" "interval"

object name
The name of an existing Timer object.

interval
The number of milliseconds to wait before executing the Timer's Action Event. A millisecon is one-thousandth of a second.

Example:
```
TimerStart "Timer1" "2000"
```

TimerStop

Purpose: Manually deactivate an active Timer object.

Category: Objects

Syntax: TimerStop "object name"

object name
The name of an existing Timer object.

Example:
```
TimerStop "Timer1"
```

ListBox and ComboBox

ListBoxGetItem

Purpose: Retrieve the text of an item in a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxGetItem "object name" "line number" "variable"

object name
The name of an existing List Box or Combo Box object.

line number
The line number of the item to retrieve. The first item in the list is 1, the second item 2 and so on.
Use "all" instead of a number to retrieve all items.

variable
The name of the variable to store the item's text.

Example:
```
ListBoxGetItem "ListBox1" "5" "[Result]"
```

ListBoxAddItem

Purpose: Add an item to a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxAddItem "object name" "line number" "item text"

object name
The name of an existing List Box or Combo Box object.

line number
The line number where the new item is to be inserted or 0 (zero) to append the item to the end of the list.

item text
The text of the item to be added.

Example: `ListBoxAddItem "ListBox1" "0" "George Washington"`

ListBoxMoveItem

Purpose: Move an item in a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxMoveItem "object name" "source line" "destination line"

object name
The name of an existing List Box or Combo Box object..

source line
The line number of the item to move. The first item in the list is 1, the second item 2 and so on.

destination line
The line number of the item's new position.

Example: `ListBoxMoveItem "ListBox1" "1" "5"`

ListBoxDeleteItem

Purpose: Remove an item from a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxDeleteItem "object name" "line number"

object name
The name of an existing List Box or Combo Box object.

line number
The line number of the item to delete. The first item in the list is 1, the second item 2 and so on. All items after the deleted item will move up one. You can delete the entire list by using "All" instead of a specific line number.

Example: `ListBoxDeleteItem "ListBox1" "3"`

ListBoxChangeItem

Purpose: Modify an existing item in a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxChangeItem "object name" "line number" "new item text"

object name
The name of an existing List Box or Combo Box object.

line number
The line number of the item to change. The first item in the list is 1, the second item 2 and so on.

new item text
The item's new text.

Example: `ListBoxChangeItem "ListBox1" "3" "Abraham Lincoln"`

ListBoxSize

Purpose: Count the total number of items in a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxSize "object name" "variable"

object name
The name of an existing List Box or Combo Box object.

variable

The name of the variable to store the list size.

Example:
```
ListBoxSize "ListBox1" "[Count]"
AlertBox "Info" "The list contains [Count] item(s)."
```

ListBoxFindItem

Purpose: Search for an item in a List Box or Combo Box object.

Category: Objects

Syntax: ListBoxFindItem "object name" "item text" "variable"

object name
The name of an existing List Box or Combo Box object.

item text
The text of the item to find.

variable
The name of the variable to store the number of the found item. If no match is found, variable will contain 0 (zero).

Example: The example below searches for a specific item and, if found, deletes the item:

```
ListBoxFindItem "ListBox1" "Herbert Hoover" "[Found]"
If "[Found]" ">" "0"
  ListBoxDeleteItem "ListBox1" "[Found]"
EndIf
```

ListBoxSort

Purpose: Arrange the items in a List Box or Combo Box object in alphabetical order.

Category: Objects

Syntax: ListBoxSort "object name" "sort state"

object name
The name of an existing List Box or Combo Box object.

sort state
Specify "True" to turn the sort feature on, or "False" to turn it off.

Example:
```
ListBoxSort "ListBox1" "True"
```

TrackBar

TrackbarSetMax

Purpose: Set the maximum range for a Track Bar object.

Category: Objects

Syntax: TrackBarSetMax "object name" "maximum value"

object name
The name of an existing Track Bar object.

maximum value
The value to assign as the Track Bar's maximum range.

Example:
```
TrackBarSetMax "Trackbar1" "100"
```

TrackbarSetMin

Purpose: Set the minimum range for a Track Bar object.

Category: Objects

Syntax: TrackBarSetMin "object name" "minimum value"

object name
The name of an existing Track Bar object.

maximum value
The value to assign as the Track Bar's minimum range.

Example: `TrackBarSetMin "TrackBar1" "0"`

Web Browser

BrowserGoTo

Purpose: Display an HTML file or website in a Web Browser object.

Category: Objects

Syntax: BrowserGoTo "object name" "url or file name"

object name
The name of an existing Web Browser object.

url or file name
The address of a website or path to a local HTML file.

Example: `BrowserGoTo "WebBrowser1" "www.visualneo.com"`

BrowserBack

Purpose: Navigate to the previous item in a Web Browser's navigation history.

Category: Objects

Syntax: BrowserBack "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserBack "WebBrowser1"`

BrowserForward

Purpose: Navigate to the next item in a Web Browser's navigation history.

Category: Objects

Syntax: BrowserForward "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserForward "WebBrowser1"`

BrowserHome

Purpose: Display the system's default home page in a Web Browser object.

Category: Objects

Syntax: BrowserHome "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserHome "WebBrowser1"`

BrowserSearch

Purpose: Display the system's default search site in a Web Browser object.

Category: Objects

Syntax: BrowserSearch "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserSearch "WebBrowser1"`

BrowserPrint

Purpose: Print the page currently displayed in a Web Browser object.

Category: Objects

Syntax: BrowserPrint "object name" "options"

object name
The name of an existing Web Browser object.

options
Use "PrintPreview" to preview the page prior to printing or leave this parameter blank to skip the preview.

Example: `BrowserPrint "WebBrowser1"`

Note: *If the Web Browser's Silent Mode is disabled, the BrowserPrint Action will display the Windows Print screen before printing. When Silent Mode is on, the document will print immediately without showing the Print screen. See The Tool Palette for information on configuring Web Browser properties.*

BrowserStop

Purpose: Cancel a Web Browser object's pending navigation or download.

Category: Objects

Syntax: BrowserStop "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserStop "WebBrowser1"`

BrowserExport

Purpose: Copy the contents of a Web Browser object to a variable.

Category: Objects

Syntax: BrowserExport "object name" "variable"

object name
The name of an existing Web Browser object.

variable
The name of the variable to store the HTML code.

Example: `BrowserExport "WebBrowser1" "[Code]"`

BrowserFind

Purpose: Display a Find dialog box and allow the reader to search for text within a Web Browser object.

Category: Objects

Syntax: BrowserFind "object name"

object name
The name of an existing Web Browser object.

Example: `BrowserFind "WebBrowser1"`

BrowserLoadFromStr

Purpose: Manually load HTML source code into a Web Browser object.

Category: Objects

Syntax: BrowserLoadFromStr "object name" "code"

object name
The name of an existing Web Browser object.

code
The HTML code to display in the Web Browser object.

Example:
```
BrowserLoadFromStr "WebBrowser1" "<html>....</html>"
```

BrowserExecScript

Purpose: Execute Java or VB scripts inside a Web Browser object. This action requires that the Internet Security/Active Scripting option be enabled. See also: Passing Information Between the Browser and VisualNEO for Windows.

Category: Objects

Syntax: BrowserExecScript "object name" "script code" "script language"

object name
The name of an existing Web Browser object.

script code
This is the actual written in the scripting language specified below.

script language
One of the following:

JavaScript
JScript
VBScript

Example: The following action will replace the contents of the object WebBowser1 with "Hello from VisualNEO for Windows!":

```
BrowserExecScript "WebBrowser1" "text=[#34]Hello from VisualNEO for Windows!
[#34];|document.write(text);" "JavaScript"
```

BrowserGetElement

Purpose: Copy the contents of an HTML-based element to a VisualNEO for Windows variable. This action can be used to retrieve information from web-based forms. See also: Passing Information Between the Browser and VisualNEO for Windows.

Category: Objects

Syntax: BrowserGetElement "object name" "element name" "variable"

object name
The name of an existing Web Browser object.

element name
This is the name of the HTML element. Optionally, the element name may also include the name of the frame and/or form the element belongs to. If the element name includes a frame, separate the frame and the element name with a period. If the element name includes a form, separate the form and the element name with a colon. For example:

HTML element only:

```
"Element"
```

frame+element:

```
"Frame.Element"
```

form+element:

```
"Form:Element"
```

frame+form+element:

```
"Frame.Form:Element"
```

variable
The name of the VisualNEO for Windows variable to store the returned data.

Example: The following will copy the contents of the HTML Text Box *FirstName* from WebBrowser1 to the VisualNEO for Windows variable [Name]:

```
BrowserGetElement "WebBrowser1" "FirstName" "[Name]"
```

BrowserSetElement

Purpose: Set the contents of an HTML-based element. This action can be used to fill in web-based forms. See also: Passing Information Between the Browser and VisualNEO for Windows.

Category: Objects

Syntax: BrowserSetElement "object name" "element name" "data"

object name
The name of an existing Web Browser object.

element name
This is the name of the HTML element to set. Optionally, the element name may also include the name of the frame and/or form the element belongs to. If the element name includes a frame, separate the frame and the element name with a period. If the element name includes a form, separate the form and the element name with a colon. For example:

HTML element only:

```
"Element"
```

frame+element:

```
"Frame.Element"
```

form+element:

```
"Form:Element"
```

frame+form+element:

```
"Frame.Form:Element"
```

data
The data to be stored in the HTML element.

Example: The following will set the contents of the HTML Text Box *FirstName* to "Joe":

```
BrowserSetElement "WebBrowser1" "FirstName" "Joe"
```

Article and Linked-Article

ArticleJumpTo

Purpose: Jump to a bookmark in an Article or Linked-Article object.

Category: Objects

Syntax: ArticleJumpTo "object name" "bookmark name"

object name
The name of an Article or Linked-Article object.

bookmark name
The bookmark name as created in the Text Editor.

Example:
```
ArticleJumpTo "Article1" "Top"
```

Picture

PictureMagnify

Purpose: Change the magnification/zoom level of a Picture object.

Category: Objects

Syntax: PictureMagnify "object name" "magnification percentage"

object name
The name of an existing Picture object.

magnification percentage
To view a picture in its normal resolution use “100” To increase the magnification, enter a number greater than 100. Numbers smaller than 100 will reduce the picture’s size. The magnification level only affects how the picture is displayed on screen. The original image file is not altered.

Example: The following example displays an image twice its normal size:

```
PictureMagnify "Picture1" "200"
```

Flash Player

FlashPlay

Purpose: Begin playing a Flash object.

Category: Objects

Syntax: FlashPlay "object name" "mode"

object name
The name of an existing Flash Player object.

mode
Leave this parameter blank to play normally or use “Loop” to play the file repeatedly.

Example:
```
FlashPlay "Flash1" "Loop"
```

FlashPause

Purpose: Pause a playing Flash object. If object is already paused, then the Flash file will start playing from the current frame.

Category: Objects

Syntax: FlashPause "object name"

object name
The name of an existing Flash Player object.

Example: `FlashPause "Flash1"`

FlashStop

Purpose: Stop a playing Flash object.

Category: Objects

Syntax: FlashStop "object name"

object name
The name of an existing Flash Player object.

Example: `FlashStop "Flash1"`

FlashRewind

Purpose: Rewind a Flash object.

Category: Objects

Syntax: FlashRewind "object name"

object name
The name of an existing Flash Player object.

Example: `FlashRewind "Flash1"`

FlashGotoFrame

Purpose: Jump to a specific frame in a Flash object.

Category: Objects

Syntax: FlashGotoFrame "object name" "frame"

object name
The name of an existing Flash Player object.

frame
The number of the frame to display.

Example: `FlashGotoFrame "Flash1" "10"`

FlashForward

Purpose: Move a Flash object ahead one frame.

Category: Objects

Syntax: FlashForward "object name"

object name
The name of an existing Flash Player object.

Example: `FlashForward "Flash1"`

FlashBack

Purpose: Move a Flash object back one frame.

Category: Objects

Syntax: FlashBack "object name"

object name
The name of an existing Flash Player object.

Example: `FlashBack "Flash1"`

FlashGetVar

Purpose: Copy a Flash variable to a VisualNEO for Windows variable.

Category: Objects

Syntax: FlashGetVar "object name" "flash variable" "VisualNEO for Windows variable"

object name
The name of an existing Flash Player object.

flash variable
The name of the Flash variable to copy.

VisualNEO for Windows variable
The name of the VisualNEO for Windows variable to store the Flash data.

Example:
```
FlashGetVar "Flash1" "Counter" "[Result]"
```

FlashSetVar

Purpose: Set a Flash object variable.

Category: Objects

Syntax: FlashSetVar "object name" "flash variable" "value"

object name
The name of an existing Flash Player object.

flash variable
The name of the Flash variable to copy.

value
The value to assign to the Flash variable.

Example:
```
FlashSetVar "Flash1" "Counter" "50"
```


Pages

Pages

SetPageBackground

Purpose: Change a page's background color or wallpaper.

Category: Pages

Syntax: ShowPageBackground "page title" "style" "color1" "color2" "wallpaper file" "options"

page title
The title of the page to modify.

style
One of the following:

Solid	The page background will be a single solid color.
Gradient	The page background will be a gradient - a blending of two colors.
Wallpaper	Use an image file as the page background.

color1
If style is Solid, this is the page background color. If style is Gradient this is the top/left color in the gradient. (See Defining Colors.) If style is Wallpaper, this parameter is ignored.

color2
If style is Gradient this is the bottom/right color in the gradient. (See Defining Colors.) If style is Solid or Wallpaper, this parameter is ignored.

wallpaper file
If style is Wallpaper, this is the name of the image file for the page background. If style is Solid or Gradient, this parameter is ignored.

options
If style is Gradient, this is the direction of the gradient - either **Horzontal** or **Vertical**. If style is Wallpaper, this is the image display mode and may be one of the following:

Tile	If the image is smaller than the page, it will be duplicated (tiled) as many times as needed to fill the page.
Center	The image will be centered within the page's boundaries.
Stretch	The image will be stretched to fit the dimensions of the page.

If style is Solid, this parameter is ignored.

Example: `SetPageBackground "New Page" "Solid" "248,208,168" "" "" ""`

SetPageEffect

Purpose: Change a page's transition effect.

Category: Pages

Syntax: SetPageEffect "page title" "effect" "speed"

page title
The title of the page to modify.

effect
One of the following:

None, Dissolve, Slide Left, Slide Right, Slide Up, Slide Down, Explode, Implode, Weave Horizontal, Weave Vertical, Split Horizontal, Split Vertical, Wipe Left, Wipe Right, Wipe Up, Wipe Down, Circle, Grow, Blocks, Checkerboard, Block Dissolve, Fade or Page Turn.

speed
The effect's speed (0 = fastest, 10 = slowest).

Example: `SetPageEffect "Main" "Fade" "2"`

ShowMasterPage

Purpose: Display objects from the Master Page.

Category: Pages

Syntax: ShowMasterPage "page title"

page title
The title of the page to modify.

Example: `ShowMasterPage "Contents"`

HideMasterPage

Purpose: Hide objects from the Master Page.

Category: Pages

Syntax: HideMasterPage "page title"

page title
The title of the page to modify.

Example: `HideMasterPage "Contents"`

Menus

ShowMenuItem

Purpose: Show a hidden menu item. See App Properties > Main Menu or Tray Menu for more information about menus.

Category: Menus

Syntax: ShowMenuItem "menu item"

menu item
The Item ID assigned to the menu heading or item to be shown.

Example:
```
ShowMenuItem "MenuItem1"
```

HideMenuItem

Purpose: Hide a visible menu item. Use in conjunction with ShowMenuItem. See App Properties > Main Menu or Tray Menu for more information about menus.

Category: Menus

Syntax: HideMenuItem "menu item"

menu item
The Item ID assigned to the menu heading or item to be hidden.

Example:
```
HideMenuItem "MenuItem1"
```

EnableMenuItem

Purpose: Enable a menu item. An enabled menu item can respond to mouse and keyboard events. Use in conjunction with DisableMenuItem. See App Properties > Main Menu or Tray Menu for more information about menus.

Category: Menus

Syntax: EnableMenuItem "menu item"

menu item
The Item ID assigned to the menu heading or item to be Enabled.

Example:
```
EnableMenuItem "MenuItem1"
```

DisableMenuItem

Purpose: Disable a menu item. A disabled menu item ignores mouse and keyboard events. See App Properties > Main Menu or Tray Menu for more information about menus.

Category: Menus

Syntax: DisableMenuItem "menu item"

menu item
The Item ID assigned to the menu heading or item to be disabled.

Example:
```
DisableMenuItem "MenuItem1"
```


Internet

Basic Internet

CheckInternetConnection

Purpose: Determine if this computer is connected to the Internet.

Category: Internet

Syntax: CheckInternetConnection "variable"

variable
The name of a variable to store connection status. If connected, the variable will be set to "True", otherwise, it will be set to "False".

Example:
```
CheckInternetConnection "[Connected]"
If "[Connected]" "=" "True"
  AlertBox "Result" "Connection found!"
Else
  AlertBox "Result" "Not connected."
EndIf
```

Note: *You cannot rely solely on CheckInternetConnection to determine if you have a valid active Internet connection. It is impossible for CheckInternetConnection to determine if the entire connection to the Internet is functioning without sending a request to a server. This is why you may also need to send some type of simple request (see InternetGet) to determine if you are really connected or not.*

ConnectToInternet

Purpose: Open an Internet connection. (Requires proper hardware and a valid Internet account.)

Category: Internet

Syntax: ConnectToInternet "variable"

variable
The name of a variable to store connection status. If successful, the variable will be set to "True", otherwise, it will be set to "False".

Example:
```
ConnectToInternet "[Result]"
```

DisconnectFromInternet

Purpose: Disconnect from the Internet. (This may not work for always-on broadband type connections.)

Category: Internet

Syntax: DisconnectFromInternet

Example:
```
DisconnectFromInternet
```

Internet Explorer

InternetLink

Purpose: Launch the computer's default Internet browser, and navigate to the specified web site URL. Requires a valid Internet account and browser software.

Category: Internet

Syntax: InternetLink "url address"

url address
The address of a web site or local file to pass to the browser.

Example:
```
InternetLink "http://www.visualneo.com"
```

This Action can also be used to launch the computer's default email application. For example:

```
InternetLink "mailto:info@visualneo.com"
```

eMail

SendMail

Purpose: Send an email message with an optional file attachments over the Internet. A valid Internet account and connection are required.

Category: Internet

Syntax: SendMail "to" "from" "subject" "message" "attachments"

to
The email address of the person who will receive the message. Multiple addresses can be separated with the pipe character "|".

from
The address of the person sending the message. Some Internet services will not allow messages to be sent if the sender's address does not match the account.

subject
The subject of the message.

message
The message text. Can be either plain text or HTML. Line breaks can be entered into the message body using the pipe character "|".

attachments (optional)
The name of an external file to be attached to the message. Multiple files can be separated with the pipe character "|".

options
Enter "HideProgress" if you do not want to give the user any feedback while sending email. Leave options blank to display a graph showing the email's progress.

Example:
```
SendMail "info@visualneo.com" "xyz@hotmail.com" "Request for product information" "Dear SinLios,||Please send me some information about VisualNEO for Windows.||Thank you.||Sincerely,||John Q. Smith" "" ""
```

Note: *The behavior of the SendMail Action can be modified by using the global [MailServer], [MailUserID] and [MailUserPassword] variables. See Understanding Actions and Variables for details.*

HTTP

DownloadFile

Purpose: Download a file from the Internet.

Category: Internet

Syntax: DownloadFile "source url" "destination" "options"

source url
The name and location of the file to download.

destination
The path and name where the downloaded file is to be saved.

options
One or both of the following:

HideProgress By default a graph showing the download progress is

displayed. Include "HideProgress" here if you do not want the graph to be visible.

Async Allow VisualNEO for Windows to respond to other input during the download. Without this option, VisualNEO for Windows will not be able to accept user input until the download is complete.

Example: `DownloadFile "http://www.yoursite.com/sample.txt" "[PubDir]sample.txt" ""`

Note: *You can download files from a secure/password protected server by using the global [HTTPUserID] and [HTTPUserPassword] variables. See Understanding Actions and Variables for details.*

InternetFileExists

Purpose: Determine if a remote file exists on an Internet server.

Category: Internet

Syntax: InternetFileExists "source url" "variable" "options"

source url
The name and location of the remote file to download.

variable
The name of the variable to store the result of the search. If the file exists, the variable will be set to "True", otherwise, it will be set to "False".

options
Enter "Async" here to allow VisualNEO for Windows to respond to other input while waiting for the server to reply. Leave this parameter blank to suspend user input until the action is complete.

Example: The example below displays an alert box if the file sample.txt exists:

```
InternetFileExists "www.yoursite.com/sample.txt" "[Result]" ""
If "[Result]" "=" "True"
  AlertBox "Status" "The file was found!"
EndIf
```

InternetFileSize

Purpose: Obtain the size (in bytes) of a remote file located on an Internet server.

Category: Internet

Syntax: InternetFileSize "source url" "variable" "options"

source url
The name and location of the remote file to examine.

variable
The name of the variable to store the file's size.

options
Enter "Async" here to allow VisualNEO for Windows to respond to other input while waiting for the server to reply. Leave this parameter blank to suspend user input until the action is complete.

Example: `InternetFileSize "www.yoursite.com/sample.txt" "[Size]" ""`

InternetGet

Purpose: Request information from an Internet server. Similar to DownloadFile except that the returned content will be stored in a variable. This should only be used for content that is compatible with VisualNEO for Windows variables.

Category: Internet

Syntax: InternetGet "source url" "variable" "options"

source url
The name and location of the file, content, etc. to download.

variable
The name of the variable to store the data returned from the server.

options
One or both of the following:

HideProgress	By default a graph showing the download progress is displayed. Include "HideProgress" here if you do not want the graph to be visible.
Async	Allow VisualNEO for Windows to respond to other input during the download. Without this option, VisualNEO for Windows will not be able to accept user input until the download is complete.

Example:
```
InternetGet "www.yoursite.com/index.html" "[Result]" "HideProgress+Async"
```

InternetPost

Purpose: Send an HTTP command and data to an Internet server. The response from the server will be stored in the supplied variable.

Category: Internet

Syntax: InternetPost "url" "data" "variable" "options"

url
The url that will receive the post command.

data
Additional data (if any) to send to the server. The formatting requirements for this data is highly server and application specific.

variable
The name of the variable to store the server's response.

options
Enter "Async" here to allow VisualNEO for Windows to respond to other input while waiting for the server to reply. Leave this parameter blank to suspend user input until the action is complete.

Example:
```
SetVar "[Host]" "http://www.yoursite.com/test.php"
SetVar "[Data]" "fname=Larry&lname=Morton"
InternetPost "[Host]" "[Data]" "[Result]" "Async"
```


Applications

Application Launching

Run

Purpose: Execute an external Windows application or DOS command.

Category: Applications

Syntax: Run "command" "parameters" "options" "subroutine" "variable"

command

The file name and location of the application to be launched.

parameters

Optional items to be passed to the application (a file name, command switches, etc.)

options

One of the following:

Normal	Run the application normally.
Wait	Suspend VisualNEO for Windows until the application closes.
LoadComplete	Suspend VisualNEO for Windows until the application has finished loading.

Additionally, you may combine one of the choices above with any of the following:

RunOnce	Prevent more than one copy of the application from running at a time.
Minimized	Launch the application in minimized mode as an icon on the Windows Task Bar.
Hidden	Hide the application's main window. Some types of applications will not allow their main window to be hidden and will ignore this option. Hidden works well when running most DOS command line utilities, batch files, etc. that do not require interaction from the user. You cannot use both the "Minimized" and "Hidden" options at the same time.

subroutine (optional)

The name of a subroutine block to execute when the application closes. Leave this blank i you do not wish to use a subroutine. (See App Properties > Actions for more information about subroutines.)

variable (optional)

The name of the variable store the application's unique identification number. This ID can be used to identify the application to other Actions.

Example:

```
Run "c:\windows\notepad.exe" "" "Normal+RunOnce" "" "[AppId]"
```

When compiling a publication, VisualNEO for Windows will localize the command by removing the drive and path from the application's file name. In the example above, "c:\windows\notepad.exe" would be changed to "notepad.exe". This behavior is useful for publications that are designed to be used on a variety of computer systems where drives an directories may be organized differently. In some situations, however, you may prefer to leav the drive and path information intact. This can be accomplished by simply placing an exclamation point character (!) at the beginning of the file name. For example:

```
Run "!c:\windows\notepad.exe" "" "Normal+RunOnce" "" "[AppId]"
```

More often than not you will probably want to distribute programs launched with Run along with your compiled VisualNEO for Windows publication. In that situation, you will want to use the [PubDir] variable instead of a literal path. This will insure that your publication will always be able to locate the application you're attempting to launch. For example:

```
Run "![PubDir]My App.exe" "" "Normal" "" "[AppId]"
```

At runtime, VisualNEO for Windows will replace the [PubDir] variable with the location of your compiled VisualNEO for Windows publication exe. In the above example, both "My App.exe" and the compiled pub exe are located in same folder. This will work regardless of whether the

files are run from the hard drive, CD, etc.

Before distributing any compiled exe files or utilities that were created by someone else, you should obtain permission from the person who owns the rights to that program.

Plug-in developers and others may find the global [AppID.ProcessID], [AppID.ProcessHandle] [AppID.WinHandle] and [AppID.ExitCode] variables useful.

RunInRectangle

Purpose: Execute an external Windows application inside a Rectangle object. The application's window will be sized and/or clipped to fit within the bounds of the specified Rectangle object. This may not work with some types of applications.

Category: Applications

Syntax: RunInRectangle "object name" "command" "parameters" "options" "subroutine" "variable"

object name
The name of an existing object.

command
The file name and location of the application to be launched.

parameters
Optional items to be passed to the application (a file name, command switches, etc.)

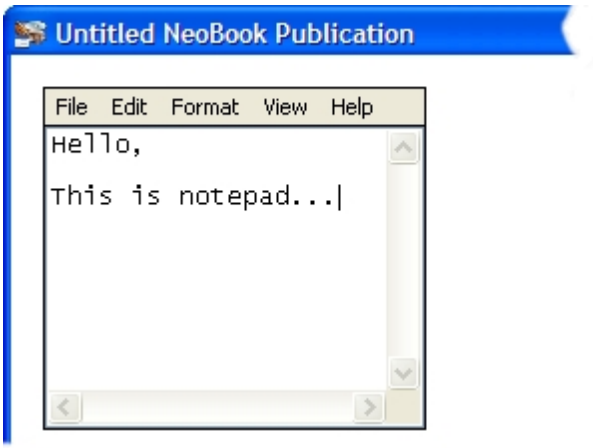
options
One or both of the following:

HideMenu	Attempt to hide the application's menu bar. May not work with some applications.
HideSizeBox	Attempt to hide the application's size box, which usually appears in the lower right corner of the main window.

subroutine (optional)
The name of a subroutine block to execute when the application closes. Leave this blank if you do not wish to use a subroutine. (See App Properties > Actions for more information about subroutines.)

variable (optional)
The name of the variable store the application's unique identification number. This ID can be used to identify the application to other Actions.

Example:
```
RunInRectangle "Rectangle1" "notepad.exe" "" "" "" "[AppId]"
```



Plug-in developers and others may find the global [AppID.ProcessID],

[AppID.ProcessHandle], [AppID.WinHandle] and [AppID.ExitCode] variables useful.

CloseApp

Purpose: Close a running application program.

Category: Applications

Syntax: CloseApp "application" "mode"

application
The file name of a running program or an application ID variable returned by Run or RunInRectangle.

mode
One of the following:

RequestClose	Send a message to the application asking it to close. The application may ignore this request.
ForceClose	Force the application to close. The application will not have an opportunity to follow its normal shutdown routine which may result in lost data.

Example:
```
CloseApp "[AppID]" "RequestClose"
```

IsAppRunning

Purpose: Determine if a specific application is running.

Category: Applications

Syntax: IsAppRunning "application" "variable"

application
The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

variable
The name of the variable to store the results. If the application is running, the variable will be set to “True”, otherwise, it will be set to “False”.

Example:
```
IsAppRunning "notepad.exe" "[Result]"
```

SendKeys

Purpose: Send keystrokes to another Windows application. This Action allows you to control another application by simulating keys being typed on the keyboard. If the application is not already loaded, then SendKeys will launch it prior to sending any keystrokes.

Category: Applications

Syntax: SendKeys "application" "keystrokes"

application
The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

keystrokes
Keystrokes may include text and any of the special key codes listed below:

{BS}	{F2}	{F7}	{F12}	{Left}	{Del}
{Tab}	{F3}	{F8}	{Home}	{Right}	
{Enter}	{F4}	{F9}	{End}	{PgUp}	
{Esc}	{F5}	{F10}	{Up}	{PgDn}	

{F1} {F6} {F11} {Down} {Ins}

You can also specify the state (up/down) of the Shift, Control and Alt keys to access virtually the entire range of key combinations. Key codes for these keys are:

{ShiftDn} {ShiftUp} {CtrlDn} {CtrlUp} {AltDn} {AltUp}

These keys must always be used in pairs. For example, if you use {ShiftDn} you must remember to follow it with a {ShiftUp} code, otherwise your keyboard will behave as if the Shift key is stuck in the down position.

Example: The following example opens the Windows Notepad utility types "Hello world" and executes the File/Print command:

```
SendKeys "notepad.exe" "Hello world{Enter}{AltDn}FP{AltUp}"
```

SendMenuCommand

Purpose: Send a menu command to an application.

Category: Applications

Syntax: SendMenuCommand "application" "commands"

application

The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

commands

The menu commands to execute separated by commas. Can be formatted as text "File,Open", or a numeric index "1,2" indicating the position of the items in the menu tree.

Example: `SendMenuCommand "[AppID]" "File,Open"`

DropFile

Purpose: Pass a file name to an application by simulating a drag and drop event.

Category: Applications

Syntax: DropFile "application" "file name"

application

The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

file name

The name of an external file to be dropped on the application.

Example: `DropFile "notepad.exe" "[PubDir]notes.txt"`

SetWindowPos

Purpose: Set the size and position of an application's main window.

Category: Applications

Syntax: SetWindowPos "application" "left" "top" "width" "height"

application

The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

left, top

The window's new left, top position. Use "-1" instead to center the window on the screen.

width, height

The window's new width and height. Use "-1" to keep the window's current dimensions.

Example: `SetWindowPos "[AppID]" "100" "100" "350" "450"`

GetWindowPos

Purpose: Get the current size and position of an application's main window.

Category: Applications

Syntax: GetWindowPos "application" "x var" "y var" "width var" "height var"

application
The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

x var
The name of the variable to store the window's left position.

y var
The name of the variable to store the window's top position.

width var
The name of the variable to store the window's width.

height var
The name of the variable to store the window's height.

Example: `GetWindowPos "[AppId]" "[X]" "[Y]" "[W]" "[H]"`

BringAppToFront

Purpose: Bring an external application's main window to the foreground.

Category: Applications

Syntax: BringAppToFront "application"

application
The file name of a running program or an application ID variable returned by Run or RunInRectangle.

Example: `BringAppToFront "[AppID]"`

SendAppToBack

Purpose: Send an application's main window to the background.

Category: Applications

Syntax: SendAppToBack "application"

application
The file name of an executable program or an application ID variable returned by Run or RunInRectangle.

Example: `SendAppToBack "[AppID]"`

DisableApp

Purpose: Disable mouse and keyboard input to an application's main window.

Category: Applications

Syntax: DisableApp "application"

application
The file name of a running program or an application ID variable returned by Run or RunInRectangle.

Example: `DisableApp "[AppID]"`

EnableApp

Purpose: Enable mouse and keyboard input to an application's main window. Use in conjunction with DisableApp.

Category: Applications

Syntax: EnableApp "application"

application

The file name of a running program or an application ID variable returned by Run or RunInRectangle.

Example: `EnableApp "[AppID]"`

RunVisualNEO for Windows

Purpose: Run another compiled VisualNEO for Windows publication. The current publication will be closed and the new one opened in its place. RunVisualNEO for Windows can be used to link several smaller publications together to gain the same functionality as a single larger publication.

When using this option to link multiple publications, the first publication must be compiled as a fully executable exe or runtime package (with NBRun5.exe). Other publications in the group can be compiled as runtime packages or web browser plug-ins to conserve space.

Category: Applications

Syntax: RunVisualNEO for Windows "file name" "options"

file name

The name of a compiled VisualNEO for Windows publication (*.exe, *.pkg).

options

Can be none, one or both of the following:

ClearVars	Clear all variables before loading the other publication. Omit this option to share variables between the two publications. Variables assigned to objects that are auto-initializing, like Radio Buttons, Check Boxes and Text Entry Fields, will be reset to their default state when the new publication loads. Other variables, not assigned to auto-initializing objects, will remain when the ClearVars option is omitted.
CloseWindows	Close any open windows created with ImageWindow, TextWindow or CustomWindow. Omit this option to leave windows open after loading the other publication.

Example: `RunVisualNEO for Windows "[PubDir]MyOtherPub.exe" "ClearVars+CloseWindows"`

Note: *Sharing custom windows between publications, can lead to a situation where you have more than one object with the same name. VisualNEO for Windows prevents this from happening within a single publication, but there's no way to do that across multiple publications. This only a problem if you need to affect such an object using an Action that requires an object name such as ShowObject. When two objects with the same name exist at the same time, the object in the custom window will have precedence over the publication-based object. When this situation exists, it's not possible to affect the publication-based object because the windowed object will intercept the command first. This shouldn't present a problem as long as you take care when designing your publications.*

Add-On Utilties

ExecuteAddon

Purpose: Activate a third party Add-On function designed for VisualNEO for Windows 3.x.

Category: Applications

Syntax: ExecuteAddon "add-on file name" "commands"

add-on file name
The name of the Add-On's exe file.

commands
The required commands will vary depending on the Add-On used. Please consult the individual Add-On's documentation for instructions.

Example:
```
ExecuteAddOn "c:\add-ons\NeoSock.exe" "nsOpen 21"
```

Note: *Add-ons created for previous versions of VisualNEO for Windows can still be used, but should be replaced with the newer Plug-In versions of those tools when available.*

Obsolete

DOSCommand

Purpose: Execute an external Windows application or DOS command. This Action is obsolete, please use **Run** instead.

Category: Applications

Syntax: DOSCommand "command" "params" "options"

Example:
```
DOSCommand "notepad.exe" "c:\test.bat" "Normal"
```


Windows

Basic Windows

ImageWindow

Purpose: Display an image file in a movable window.

Category: Windows

Syntax: ImageWindow "title" "left" "top" "file name"

title
The text that will appear in the window's title bar.

left, top
The window's left, top corner relative to the upper left corner of the publication's work area. Enter "-1" for both to center the window on the screen.

file name
The name of the image file to display.

Example:
```
ImageWindow "Map" "-1" "-1" "c:\samples\downtown.bmp"
```

TextWindow

Purpose: Display a plain text (ASCII/ANSI) or formatted Rich Text (RTF) file in a floating window.

Category: Windows

Syntax: TextWindow "title" "left" "top" "width" "height" "file name" "options"

title
The text that will appear in the window's title bar.

left, top
The window's left, top corner relative to the upper left corner of the publication's work area. Enter "-1" for both to center the window on the screen.

width, height
The desired width and height for the window. Enter "-1" for both to detect the file's optimal width and height.

file name
The name of the text file to display.

options
Use "Wordwrap" to automatically format the text to fit within the window's margins.

Example: `TextWindow "Help" "-1" "-1" "350" "400" "c:\documents\help.txt" "Wordwrap"`

CloseWindow

Purpose: Close a window previously opened with ImageWindow or TextWindow.

Category: Windows

Syntax: CloseWindow "file name"

file name
The file name of the window to close, or leave blank to close all open windows.

Example: `CloseWindow "C:\Samples\Chart1.bmp"`

Custom Windows

CustomWindow

Purpose: Display an object or group of objects in a window. This Action can be used to create custom dialog boxes, tool windows, etc.

Category: Windows

Syntax: CustomWindow "title" "left" "top" "object name" "options"

title
The text that will appear in the window's title bar (if it has one).

left, top
The coordinates for the window's left, top corner relative to the upper left of the publication window. Enter "-1" for both to center the window on the screen.

Object name
The name of an existing object, Container or group that will comprise the contents of the window.

options
One of the following:

NoBorder	The window will be displayed without a title bar or border.
DialogBox	The window has a title bar and dialog box style border.
ToolWindow	The window has a small tool palette style title bar and border. You can use this option to create special windows (like navigation bars, etc.) that float above your publication!
Sizeable	The window has a title bar and a border. The window can be

resized.

FixedPos The window is attached to the publication at a fixed position. The window will retain its relative position even when the publication window is moved. The window will not contain a title bar or border.

Additionally, you may combine one of the choices above with one or both of the following

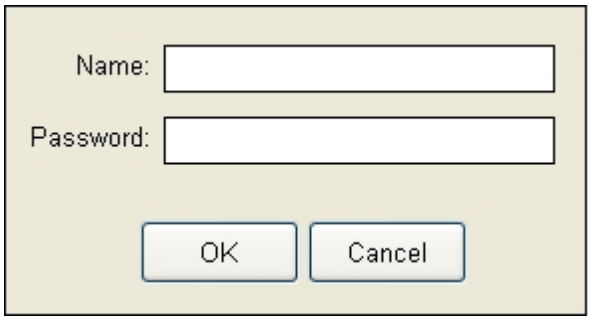
Exclusive Other parts of the publication are suspended until the window is closed. This allows you to build modal dialog boxes that require that the reader input information or select options before continuing.

NoCloseBtn Hide the window's close button.

CAUTION: If you use both the Exclusive and NoCloseBtn options, you must provide your own method for closing the window. Otherwise, the reader will not be able to close the window and return to your publication. You can programmatically close a custom window b creating a button that executes the CloseCustomWindow Action. Your close button should be added to the Container or group that makes up the contents of the window.

Example: Custom windows can be constructed from a single object, a group of objects or a Container Just place the object(s) you want to use outside the visible portion of a page or on a page that you do not intend to display. (Actually, it doesn't really matter where you place these objects, but while they are displayed inside the custom window, they disappear from the page since they can't be in both places at the same time.) The objects also do not need to be on the page that's visible when the CustomWindow Action is executed. To use more than one object, you can combine them using the Arrange menu's Group command, but the preferred method is to simply place the object's you wish to use on a Container.

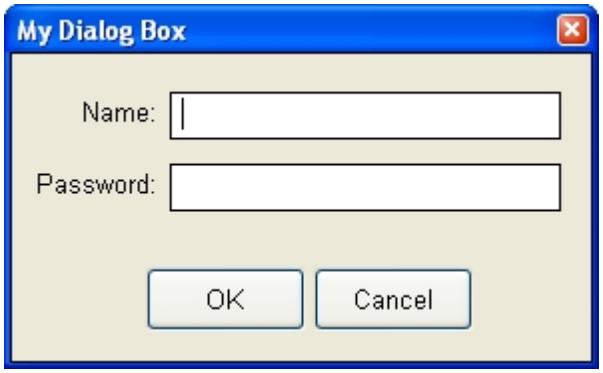
The interior of the custom window will match the size of the object, Container or group you select. When using a Container, the Fill Color you select for the Container will become the background of the custom window. You can add a Picture object to the Container if you wa a really fancy background.



A Container with several attached objects ready for use with VisualNEO for Windows's CustomWindow Actio

Once you have arranged the objects for your window, pass the Container's name to the CustomWindow Action. The following example uses the "Exclusive" option to display "Container1" in a dialog box:

```
CustomWindow "My Dialog Box" "-1" "-1" "Container1"  "DialogBox+Exclusive"
```



The same Container displayed in a CustomWindow with a dialog box style border.

The reader can close the dialog box by clicking on the window's close button, or you can provide a Push Button that executes the CloseCustomWindow Action. For example, the OK and Cancel buttons above do this:

```
CloseCustomWindow "Container1"
```

You can also use CustomWindow to display a single object (like a Media Player or Web Browser) instead of a group or Container:

```
CustomWindow "Video" "-1" "-1" "MediaPlayer1" "ToolWindow"
```

Plug-in developers may find the global [Object.WinHandle] variable useful.

Special Options:

You can define special subroutines that will be executed when a specific custom window is opened or closed. To do this, simply create two subroutines - one called ObjectName_OnOpe and one called ObjectName_OnClose. Replace ObjectName with the name of object used when opening the window. For example, the subroutines for our Container1 sample above might look like this:

```
:Container1_OnOpen
  AlertBox "Hello" "The window is opening."
Return
:Container1_OnClose
  AlertBox "Hello" "The window is closing."
Return
```

See App Properties > Actions for more information on creating subroutines.

If you're creating a sizeable custom window, you can restrict the window's size by setting th following variables at some point before the window is created:

```
[ObjectName_MinWidth]
[ObjectName_MinHeight]
[ObjectName_MaxWidth]
[ObjectName_MaxHeight]
```

Replace ObjectName with the name of the object used when opening the window.
You can obtain the current position of an open custom window by passing the name of the object used to open the window to the GetObjectInfo Action. For example:

```
GetObjectInfo "Container1" "Left" "[WinX]"
GetObjectInfo "Container1" "Top" "[WinY]"
```

This will return the upper left and top coordinates relative to the publication's upper left corner.

CloseCustomWindow

Purpose: Close a window previously opened with the CustomWindow Action.

Category: Windows

Syntax: CloseCustomWindow "object name"

object name
The name of the object passed to CustomWindow when the window was opened.

Example:
```
CloseCustomWindow "Container1"
```


Control

Conditional

If

Purpose: Change the flow of script execution, based on the results of a simple comparison of two items. When the statement is True, execution continues until an "EndIf" or "Else" Action is encountered. Otherwise, execution begins with the first line following the next "Else" (optional) or "EndIf" command.

Category: Control

Syntax: If "first item" "operator" "second item"

first item
The first item to compare. Can contain text, numbers, math expression, variables, etc.

operator
One of the following:

=	First item is equal to second item.
<	First item is less than second item.
>	First item is greater than second item.
<>	First item is not equal to second item.
<=	First item is less than or equal to second item.
>=	First item is greater than or equal to second item.

second item
The second item to compare. Can contain text, numbers, math expression, variables, etc.

Example: The following example examines the contents of the variable [Name]. If [Name] is empty an error message appears, otherwise, the next page is displayed:

```
If "[Name]" "=" ""
  AlertBox "Error!" "I don't know your name."
Else
  GotoNextPage
EndIf
```

You can also use If to detect the correct entry of a password:

```
If "[Password]" "=" "Bravo172"
  GotoNextPage
Else
  AlertBox "Error!" "The password is incorrect."
EndIf
```

IfEx

Purpose: Change the flow of script execution based on the result of a complex expression. This is an advanced version of the standard If Action. When the statement is True, execution continues until an "EndIf" or "Else" Action is encountered. Otherwise, execution begins with the first line following the next "Else" (optional) or "EndIf" command.

Category: Control

Syntax: IfEx "expression"

expression

A basic expression consists of three elements - two items to be compared separated by a special operator. For example:

```
"item operator item"
```

The two items can be text, numbers, math expressions, variables, etc. The operator must be one of the following:

= First item is equal to second item.
< First item is less than second item.
> First item is greater than second item.
<> First item is not equal to second item.
<= First item is less than or equal to second item.
>= First item is greater than or equal to second item.

For example, the following expression compares the variable [City] to "Pittsburgh":

```
"[City] = Pittsburgh"
```

If a comparison item contains spaces, then it must be surrounded with quotes, which in VisualNEO for Windows are specified using the special code: [#34]. For example:

```
"[City] = [#34]St. Louis[#34]"
```

You can also construct more complex statements by combining multiple expressions with "and" and "or". For example, to find out if [City] equals either "Pittsburgh" or "Washington":

```
"[City] = Pittsburgh OR [City] = Washington"
```

To find out if [City] equals "Pittsburgh" and another variable [Name] also equals "Jones":

```
"[City] = Pittsburgh AND [Name] = Jones"
```

For extremely complex statements you may want to use "(" and ")" to make sure expressions are evaluated in the correct order.

Example:

```
IfEx "[Account] = Guest OR [Account] = Admin"
  GotoPage "Welcome"
Else
  AlertBox "Error!" "The account is incorrect."
EndIf
```

While

Purpose: Repeat a series of Actions until a specified condition is no longer valid. Any Actions between the While and its matching EndWhile statement will continue to execute until the specified condition is no longer true.

Category: Control

Syntax: While "first item" "operator" "second item"

first item

The first item to compare. Can contain text, numbers, math expression, variables, etc.

operator

One of the following:

=	First item is equal to second item.
<	First item is less than second item.
>	First item is greater than second item.
<>	First item is not equal to second item.
<=	First item is less than or equal to second item.
>=	First item is greater than or equal to second item.

second item

The second item to compare. Can contain text, numbers, math expression, variables, etc.

Example: In the following example, 100 tones are played using the PlayTone Action. The sound begins at a frequency of 0 Hertz, and is incremented by 25 Hertz, until 100 tones have been played in sequence:

```
SetVar "[Count]" "0"
SetVar "[Freq]" "0"
While "[Count]" "<" "100"
  Math "[Count]+1" "0" "[Count]"
  Math "[Freq]+25" "0" "[Freq]"
  PlayTone "[Freq]"
EndWhile
```

WhileEx

Purpose: Repeat a series of Actions until a specified condition is met. This is an advanced version of the standard While Action.

Category: Control

Syntax: WhileEx "expression"

expression

The expression to be evaluated. See IfEx for information about constructing expressions.

Example:

```
WhileEx "[X] > 0 AND [Y] > 0"
  SetVar "[X]" "[X]-1"
  SetVar "[Y]" "[Y]-1"
EndWhile
```

ExitWhile

Purpose: Exit the current While/WhileEx/EndWhile block. Execution continues with the Action following the next EndWhile statement.

Category: Control

Syntax: ExitWhile

Example:

```
SetVar "[Name]" ""
While "[Name]" "=" ""
  InputBox "Welcome" "Enter your name:" "[Name]"
  If "[Name]" "=" "Administrator"
    GotoPage "Setup"
```

```
    ExitWhile
  EndIf
EndWhile
```

Loop

Purpose: Repeat a group of Actions a specified number of times.

Category: Control

Syntax: Loop "start value" "stop value" "variable counter"

start value
The starting value for the loop.

stop value
The ending value for the loop.

variable counter
The name of the variable to use as a counter. (Required)

Actions between the Loop and EndLoop statements will execute the number of times it takes to increment the counter variable from start to stop.

Example: The following example reads and displays the first five lines of a file:

```
Loop "1" "5" "[Counter]"
  FileRead "Products.dat" "[Counter]" "[Data]"
  AlertBox "Products" "[Data]"
EndLoop
```

ExitLoop

Purpose: Exit the current Loop/EndLoop block. Execution continues with the Action following the next EndLoop statement.

Category: Control

Syntax: ExitLoop

Example:
```
Loop "1" "100" "[Count]"
  Random "100" "[R]"
  If "[R]" ">" "75"
    ExitLoop
  EndIf
EndLoop
```

GotoLine

Purpose: Jump to a specific line number or label in the current Action script and continue execution from that point.

Category: Control

Syntax: GotoLine "label or line number"

label or line number
A line number or label name. A label is a descriptive word preceded by a colon ":" used to mark the beginning of a block of code.

Example:
```
:StartHere
InputBox "Stop" "Please enter your password:" "[PWord]"
If "[PWord]" "<>" "Charlie227"
  GotoLine "StartHere"
EndIf
```

Function library

Call

Purpose: Execute a predefined script from the Function Library.

Category: Control

Syntax: Call "function name" "function parameters (optional)"

function name
The name of the function to execute. If the desired function is located in a sub folder ther the folder's name must also be included. For example: "Math\InterestRate"

function parameters
If the function you select requires parameters those must be added here surrounded by quotes. For functions without parameters, only the function name is required.

Example:
```
Call "Save Bookmark"
```

Subroutine Access

GoSub

Purpose: Execute a section of the publication's Subroutine Action.

Category: Control

Syntax: GoSub "subroutine name"

subroutine name
The name of the subroutine section to execute.

Example:
```
GoSub "SaveCustomerInfo"
```

Return

Purpose: Exit the current Action script or subroutine and return control to the previous one.

Category: Control

Syntax: Return

Example:
```
:MySubroutine
  AlertBox "Hello" "This is a subroutine"
Return
```

Note: *The Return Action is most often used with Subroutines.*

Mouse Functions

SetMousePos

Purpose: Move the mouse pointer.

Category: Control

Syntax: SetMousePos "x pos" "y pos"

x pos, y pos
The mouse pointer's new left, top position relative to the upper left corner of the publication window.

Example:
```
SetMousePos "100" "150"
```

GetMousePos

Purpose: Get the position of the mouse pointer relative to the upper left corner of the publication window.

Category: Control

Syntax: GetMousePos "x var" "y var"

min var

The name of the variable to store the cursor's X position.

max var

The name of the variable to store the cursor's Y position.

Example:
```
GetMousePos "[X]" "[Y]"
```

ClickMouse

Purpose: Simulate a mouse button click.

Category: Control

Syntax: ClickMouse

Example: The example below moves the mouse pointer and simulates a press of the left mouse button:

```
SetMousePos "100" "150"
ClickMouse
```

Mathematical

Math

Purpose: Perform a mathematical calculation.

Category: Control

Syntax: Math "formula" "decimal places" "variable"

formula

A mathematical formula. The formula can include the following operators and functions: +, -, *, /, ^, Abs, Sin, Cos, Atan, Sqr, Sqrt, Round, Trunc. Parentheses and variables may also be used in formulas.

decimal places

The number of decimal places to include in the result. Use 0 (zero) to round the result to the nearest whole number or -1 to have VisualNEO for Windows automatically determine the optimal number of decimal places

variable

The name of the variable to store the result.

Example: In the example below, a variable [MonthlyRent] is used to calculate the amount of rent paid per week. [MonthlyRent] could be set using the SetVar Action, or associated with a Text Entry object to allow the user to enter a number. The result contains two decimal places (the second parameter), and is stored in the variable [WeeklyRent].

```
Math "([MonthlyRent]*12)/52" "2" "[WeeklyRent]"
```

When using Math functions, such as Sin and Cos, you must enclose the portion of the formula to be evaluated by the function within parentheses. For example:

```
Math "Cos(180)" "2" "[Result]"
```

Note: *Actions that expect numeric values can also accept formulas as parameters like those used by the Math Action.*

Random

Purpose: Generate a random number.

Category: Control

Syntax: Random "maximum value" "variable"

maximum value
The maximum possible value for the generated number. The number returned will be somewhere between 0 (zero) and maximum value.

variable
The name of the variable to store the random number.

Example: The example below, uses the publication's screen width and height to generate a random set of coordinates to display an image:

```
Loop "1" "5" "[Counter]"
  Random "[PubWidth]" "[X]"
  Random "[PubHeight]" "[Y]"
  PopUpImage "[X]" "[Y]" "c:\flower.bmp" "2000" "Dissolve" "3"
EndLoop
```

DateToNum

Purpose: Convert a formatted date to a number that can be used in mathematical calculations. The result, a numeric date, represents the number of days since December 30, 1899. A numeric date can be converted back into a formatted date with the NumToDate Action.

Category: Control

Syntax: DateToNum "date" "format" "variable"

date
A formatted date consisting of two or three numbers, separated by "\", "/" or "-". Year values between 0 and 99 are assumed to be in the current century. The date must match the format parameter below.

format
A string that identifies the format of the date. For example:

m/d/y	Month/Day/Year
d/m/y	Day/Month/Year
y/m/d	Year/Month/Day
Default	Use the system's default date format.

variable
The name of the variable to store the converted date number.

Example: The following example calculates the number of days since December 15th, 2004:

```
DateToNum "12/15/2004" "m/d/y" "[StartDate]"
DateToNum "[DateShort]" "Default" "[Today]"
Math "[Today]-[StartDate]" "0" "[Result]"
AlertBox "Answer" "It has been [Result] days since 12/15/2004"
```

NumToDate

Purpose: Convert a numeric date to a formatted date. Use in conjunction with DateToNum.

Category: Control

Syntax: NumToDate "number" "format" "variable"

number
An numeric date. (A numeric date represents the number of days since December 30,

1899, and can be created using the DateToNum Action.)

format

The format for the converted date. You can compose your own custom date display formats using codes from the table below:

d	Show the day as a number without a leading zero (1-31).
dd	Show the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
ddddd	Display the date using the system short date format as defined in the Windows Control Panel.
dddddd	Display the date using the system long date format as defined in the Windows Control Panel.
m	Show the month as a number without a leading zero (1-12).
mm	Show the month as a number with a leading zero (01-12).
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full name (January-December).
yy	Show the year as a two-digit number (00-99).
yyyy	Show the year as a four-digit number (0000-9999).
/	Display the date separator character as defined in the Windows Control Panel.
Default	Use the system's default short date format. (Same as ddddd above.)

For example, entering "d/m/yyy" will result in a date formatted as Day/Month/Year or "3/15/2008". Entering "dddd, mmmm m, yyyy" will result in a date formatted as "Saturday, March 15, 2008".

variable

The name of the variable to store the formatted date.

Example: `NumToDate "38426" "m/d/yyyy" "[FormattedDate]"`

Miscellaneous

Delay

Purpose: Pause for a specified number of milliseconds.

Category: Control

Syntax: Delay "milliseconds"

milliseconds

Number of milliseconds to pause. A millisecond is one thousandth of a second. For a one second interval enter 1000 milliseconds. For one minute interval enter 60000 milliseconds.

Example: `Delay "2000"`

SystemInfo

Purpose: Obtain information about the publication and the reader's computer, including environment variables.

Category: Control

Syntax: SystemInfo "information type" "variable"

Information type

One of the following:

CDRomDrive, CommandLine, CurrentDir, NetworkDrive, PubAuthor, PubColors, PubDir, PubDrive, PubHeight, PubTitle, PubWidth, ScreenBits, ScreenColors, ScreenHeight, ScreenWidth, SystemDir, TempDir, UserName, WinHandle, WindowsDir, WindowsPlatform, WindowsVer or a system environment variable surround by % characters.

variable

The name of the variable to store the information.

Example: The following example compares the color resolution of the reader's computer with the publication's color resolution. If the computer's resolution is lower that the publication's a message is displayed.

```
SystemInfo "ScreenColors" "[ScreenMode]"
SystemInfo "PubColors" "[PubMode]"
If "[PubMode]" ">" "[ScreenMode]"
  AlertBox "Notice!" "This pub requires [PubMode] colors."
EndIf
```

To obtain the contents of a system environment variable, replace information type with the name of the variable surrounded by % characters. For example:

```
SystemInfo "%PATH%" "[Result]"
```

The SystemInfo Action is somewhat obsolete since the information types are also Global Variables an can be referenced directly. For example, the script above could be rewritten like this:

```
If "[PubColors]" ">" "[ScreenColors]"
  AlertBox "Notice!" "This pub requires [PubColors] colors."
EndIf
```

The SystemInfo Action is still available for use with environment variables and to provide backward compatibility with publications created with earlier versions of VisualNEO for Windows.

Suspend

Purpose: Temporarily suspend processing of mouse and keyboard messages. This can be useful when you want a series of commands to finish executing before allowing the reader to click on another button. Also very useful for complex mouse enter/exit Actions.

Category: Control

Syntax: Suspend "state"

state

Use "True" to disable message processing or "False" to re-enable message processing. If you forget to re-enable the suspend state, it will be done automatically when the current script is completed.

By default, VisualNEO for Windows will automatically remove mouse and keyboard events from the Windows buffer when Suspend is turned off. If you prefer to leave these events in the buffer for processing by VisualNEO for Windows, you can add the keyword NoBuffer when state is False. For example:

```
"False+NoBuffer"
```

Example: The following example disables message processing until a series of Actions have completed:

```
Suspend "True"
ShowObject "Picture1" "Dissolve" "5"
```

```
ShowObject "Picture2" "Dissolve" "5"
ShowObject "PushButton1" "Dissolve" "5"
Suspend "False"
```

Note: *Be careful when using Suspend with Loop or While Actions. If message processing is suspended prior to entering the Loop or While statement, readers may not be able to interact with your publication until message processing is re-enabled or the loop is complete.*

ShowErrors

Purpose: Turn the display of error messages on or off.

Category: Control

Syntax: ShowErrors "state"

state
Use "True" to turn the display of error messages on, or "False" to handle errors yourself. The most recent error will be stored in the [LastError] variable.

Example:
```
ShowErrors "False"
FileRead "config.dat" "1" "[regkey]"
If "[LastError]" ">" ""
  AlertBox "Error" "Configuration invalid."
EndIf
ShowErrors "True"
```

LoadIcon

Purpose: Load a new application or tray icon. Leave the icon file field blank to restore the publication's original icon (specified in App Properties).

Category: Control

Syntax: LoadIcon "file name" "which icon"

file name
A Windows format icon (ico) file.

which icon
One of the following:

MainIcon	Replace the publication's main icon.
TrayIcon	Replace the publication's system tray icon. This is only valid for System Tray Applications.

Example:
```
LoadIcon "C:\Samples\Busy.ico" "MainIcon"
```

Debug

DebugBreakPoint

Purpose: Pause execution of the current script and display an optional message. This Action only works in test mode and is ignored by compiled publications.

Category: Control

Syntax: DebugBreakPoint "message"

message
The dialog box message. Line breaks can be entered using the pipe character "|".

Example:
```
DebugBreakPoint "Execution paused."
```

Variables

Variable Utilities

SetVar

Purpose: Assign a value to a variable.

Category: Variables

Syntax: SetVar "variable" "value"

variable
The name of a variable.

value
The value (text, number, etc.) to assign to the variable. To delete an existing variable, leave the value parameter empty.

Example: In the example below, the variable [Price] is set to "49.95" and the variable [Name] is set to "Ralph Jones":

```
SetVar "[Price]" "49.95"
SetVar "[Name]" "Ralph Jones"
```

You may also use SetVar to modify or combine existing variables. For example:

```
SetVar "[FullName]" "[LastName], [FirstName] [MiddleName]"
SetVar "[DisplayPrice]" "$[Price]"
```

Normally, if value contains a valid mathematical formula, VisualNEO for Windows will compute the formula and store the results in the variable. If for some reason, you want the variable to contain the actual formula instead of formula's computed result, insert an exclamation point "!" at the beginning of value. For example, the following statement:

```
SetVar "[Test]" "1+1"
```

will set the variable [Test] to "2". This is because VisualNEO for Windows knows that "1+1" is a mathematical equation and adds the two numbers together. By adding an exclamation point you can instruct VisualNEO for Windows to ignore the formula. For example:

```
SetVar "[Text]" "!1+1"
```

will set the variable [Test] to "1+1". (Note: The exclamation point is removed by VisualNEO for Windows.) You can use this technique in any VisualNEO for Windows Action where numeric parameters are allowed.

DefineVar

Purpose: Define a typed variable.

Category: Variables

Syntax: DefineVar "variable" "type" "format" "scope" "value"

variable
The name of the variable to be defined.

type
One of the following:

Undefined The variable's content is not limited. This is the same as a

generic variable created with SetVar.

String	The variable can contain any characters, either alpha or numeric.
Integer	The variable is limited to whole numbers.
Currency	The variable is limited to numbers and will be formatted as currency according to the regional settings in the Windows Control Panel.
Decimal	The variable is limited to numbers and will be displayed with the number of decimals specified by format.
Boolean	The variable is limited to "True" or "False" values.
Date	The variable is a date and will be displayed using the specified date format (m/d/y, etc.) below.

format

For decimals this parameter is the number of decimal places. For dates, format is composed from the items below separated by the "/" character:

d	Show the day as a number without a leading zero (1-31).
dd	Show the day as a number with a leading zero (01-31).
m	Show the month as a number without a leading zero (1-12).
mm	Show the month as a number with a leading zero (01-12).
yy	Show the year as a two-digit number (00-99).
yyyy	Show the year as a four-digit number (0000-9999).
Default	Use the system's default date format.

For example, entering "d/m/yyy" will result in a date formatted as Day/Month/Year or "3/15/2007" The format parameter is ignored for all other variable types.

scope

One of the following:

Global	The variable exists until it is explicitly deleted or the publication closes.
Local	The variable exists only while the current script is executing. When the current script ends, the variable is automatically deleted.

value (optional)

An initial value to assign to the variable. Must be compatible with the selected type.

Example: Example:

```
DefineVar "[Birthdate]" "Date" "m/d/yyyy" "Global" "1/1/1980"
```

In order to delete a variable created with DefineVar, you must first re-declare the variable as "Undefined", then clear it with SetVar. For example:

```
DefineVar "[Birthdate]" "Undefined" "" "Global" ""
SetVar "[Birthdate]" ""
```

SaveVariables

Purpose: Save the contents of all existing variables to a file. Use this Action to save a publication's entire complement of variables at once. Since variables reside in memory and are cleared when a publication closes, you can use SaveVariables and LoadVariables to remember

publication settings between sessions.

Category: Variables

Syntax: SaveVariables "file name"

file name
The name of the file to store the variable data.

Example:
```
SaveVariables "mypub.var"
```

LoadVariables

Purpose: Load variables from a file previously created with SaveVariables. Use this Action to load an entire group of variables at once. Since variables reside in memory and are cleared when a publication closes, you can use SaveVariables and LoadVariables to remember variables between sessions.

Category: Variables

Syntax: LoadVariables "file name"

file name
The name of an external file containing the variables to be loaded.

Example:
```
LoadVariables "[PubDir]MyPub.var"
```

ClearVariables

Purpose: Clear variables from memory.

Category: Variables

Syntax: ClearVariables "variable list"

variable list
A list of a list of variables to be cleared. Separate multiple variables with comma or pipe "|" characters. Leave this parameter blank to clear all variables.

Example: The following example will clear all variables in the publication:

```
ClearVariables ""
```

The following example will clear only those variables listed:

```
ClearVariables "[FirstName],[LastName],[Street],[City],[State],[Zip]"
```

DeleteArray

Purpose: Delete an array of variables.

Category: Variables

Syntax: DeleteArray "array name" "array size"

array name
The name of the array variable to delete.

array size
The number of elements to delete or "All" to delete the entire array.

Example:
```
DeleteArray "Clients" "All"
```

GetArrayInfo

Purpose: Count the number of items in a variable array. Returns the array's upper and lower bounds plus total number of items in array.

Category: Variables

Syntax: GetArrayInfo "array name" "min var" "max var" "size var"

array name
The name of the array variable to examine.

min var
The variable to store the index of the lowest element in the array.

max var
The variable to store the index of the highest element in the array.

size var
The variable to store the total number of elements in the array.

Example:
```
GetArrayInfo "[Names]" "[Low]" "[High]" "[Size]"
```


Using the Text Editor

VisualNEO for Windows's built-in Text Editor provides you with the tools necessary to create formatted text documents for display in your publications. The Text Editor provides basic word processing functions, plus the ability to add Hyperlinks and Bookmarks to your text. Documents created with the Text Editor are stored as Rich Text Format (RTF) files. VisualNEO for Windows's Article and Linked-Article objects are used to display files created with the Text Editor.

Starting the Text Editor

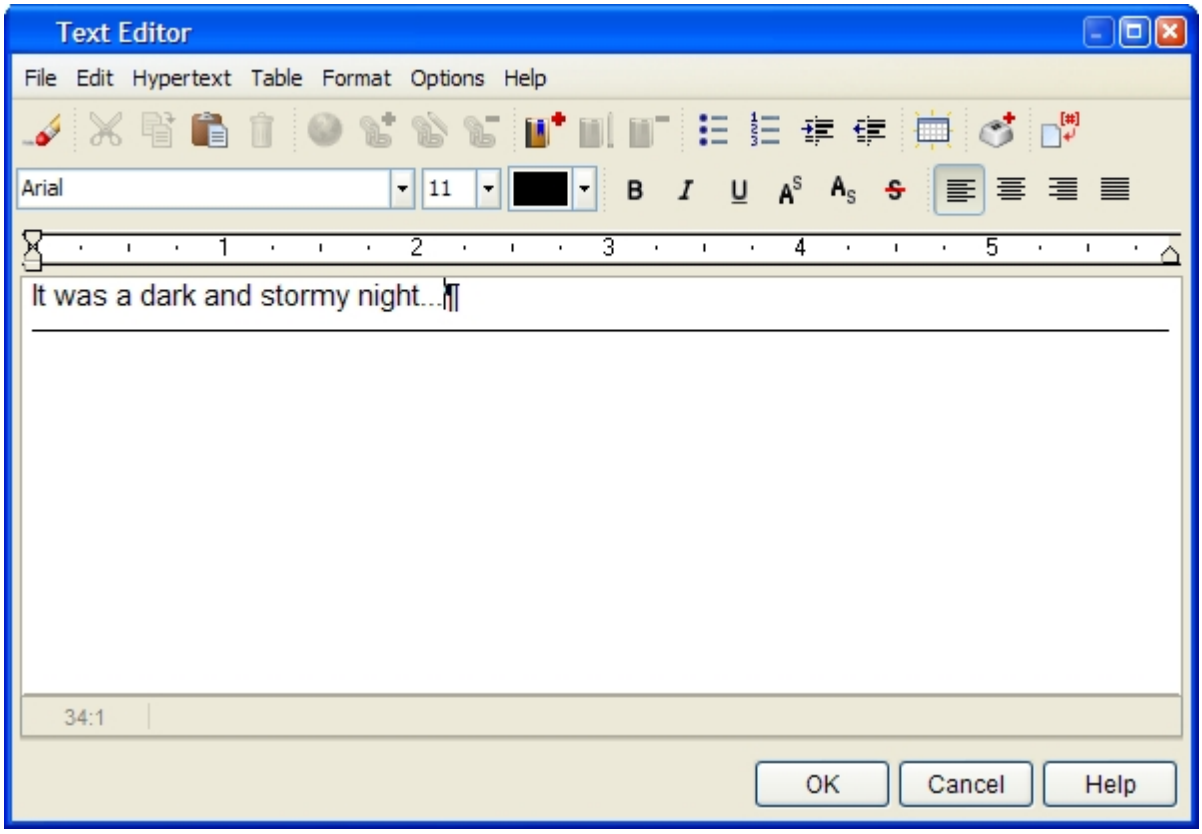
The Text Editor can be opened using any of the following methods:

- Select the **Create/Edit** > **New Text File** command from VisualNEO for Windows's **Edit** menu.
- Select either the Article or Linked-Article Tool from VisualNEO for Windows's Tool Palette. Use the mouse to draw a rectangle where you would like the Article object to appear. You will be prompted to select an existing file, create a new document, or convert a document created with another program. Choosing to create a new document will open the Text Editor.
- Right click on an existing Article or Linked-Article object to display its Properties screen. Clicking the button to the right of the **Text File** field will open the Text Editor and display the file associated with the Article.
- Use the Tool Palette's Selection Tool to select an existing Article object. The file associated with the Article should then appear in the **Edit** Menu's **Create/Edit** command. Click the file name to open the file in the Text Editor.

If the above methods don't display the Text Editor, select **Set Preferences** from VisualNEO for Windows's **Options** menu. Click the **Tools** icon on the left side of the Preferences screen. Under Text Editor, make sure the **Use VisualNEO for Windows's Built-in Editor** option is selected.

The Text Editor's Screen

The large space in the middle of the screen is the editor where you will enter and format your text. The editor functions very much like a miniature word processor – just type the desired text into the editor window.



Above the editor is a Speed Bar containing several buttons and controls that will assist you in formatting and editing your text. These items are described below:

- Undo the last edit.
- Cut the selected text to the Clipboard.
- Copy the selected text to the Clipboard.
- Paste text from the Clipboard into the editor.
- Delete the selected text.
- Create a Hyperlink. (See the Add Link menu command.)
- Edit the Hyperlink under the cursor.
- Delete the Hyperlink under the cursor.
- Create a Bookmark. (See the Add Bookmark menu command.)
- Edit the Bookmark under the cursor.
- Delete the Bookmark under the cursor.
- Format the current paragraph or selection as a bulleted list.

  Hint: *Additional bullet styles are available by selecting the Bullets command from the Text Editor's Format menu.*
- Format the current paragraph or selection as a numbered list.

Increase the left margin of the current paragraph or selection.

Decrease the left margin of the current paragraph or selection.

Insert a simple table into the document at the current cursor position. A small screen will appear allowing you to define the number of rows and columns for the table. Once created, text may be entered into the table's cells. The attributes of the Table containing the cursor may be modified using the commands in the Text Editor's Table menu.

Insert an extended ASCII character into the text. (The characters available depend on the font selected.)

Insert a VisualNEO for Windows Variable into the text.

A variable is simply an area of the computer's memory that you can use to temporarily store information (text or numbers) relevant to your publication. For example, you might request that your reader enter his or her name at the start of a publication. That variable could then be used to personalize the publication. For example, you might enter something like this in the Text Editor:

```
Suddenly, [Name] saw a bright flash in the sky. "Could it be a spaceship," [Name]
asked? No, it's probably just an airplane.
```

Notice the word Name surrounded by square brackets – that's a variable. In VisualNEO for Windows parlance, the names of variables are always surrounded by [brackets]. That's how VisualNEO for Windows knows that you're referring to a variable called [Name] and not the word "Name". At runtime VisualNEO for Windows will automatically update your document with the contents of any variables you specify. For example, the same document as the reader might see it:

```
Suddenly, Billy Johnson saw a bright flash in the sky. "Could it be a spaceship,"
Billy Johnson asked? No, it's probably just an airplane.
```

Use these controls to select a font, size and color for the currently selected text.

B Bold the selected text.

I Italicize the selected text.

U Underline the selected text.

A^S Display the selected text as Superscript.

A_S Display the selected text as Subscript.

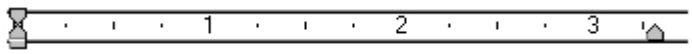
~~S~~ Display the selected text as Strikethrough.

Align the current paragraph to the left.

Align the current paragraph to the center.

Align the current paragraph to the right.

Justify the current paragraph.



The ruler can be used to set margins and tabs for the current paragraph. You can set tab stops by clicking the ruler where you want the tab to appear. Tabs can be deleted by dragging them off the ruler. You can indent the current paragraph by dragging the little hourglass symbols located at each end of the ruler. You can also use commands found in the Format menu to set tabs and modify the appearance of paragraphs.

The Text Editor's Menu

The File Menu

New

This command empties the contents of the editor and opens a new blank document. If the current contents of the editor have been modified, you will be prompted to save your changes.

Open

This command allows you to load an existing Rich Text (RTF) or plain text (ASCII/ANSI) document for editing. A standard Windows file selector will appear allowing you to select a file from your hard drive. If the current contents of the editor have been modified, you will be prompted to save your changes. Only one document at a time may be opened in the Text Editor.

Save

Choose this command to save changes made to the current document. If this is a new document that has not previously been saved, you will be prompted to specify a file name.

Save As

Use this command to save the current document using a different file name. If you select a file name which already exists, the Text Editor will ask for confirmation before replacing the file.

Note: *Text Editor document file names normally end with the file extension ".RTF". It is recommended that you use this extension to avoid confusion with other types of files.*

Page Setup

You can use this command to define the default margins and paper orientation (landscape/portrait) for the current document. This allows more precise formatting of printed documents. This information is stored inside the document and will be used as the defaults for the PrintTextFile Action. The paper size cannot be specified because this is limited by the media available on the printer attached to the reader's PC.

Print

This command prints the current document or selection.

OK

This command saves the current document and closes the Text Editor.

Cancel

This command closes the Text Editor without saving the current document.

The Edit Menu

Undo

This command removes the last change you made to the current document.

Cut

Use this command to remove the selected text and place it onto the Windows Clipboard.

Copy

Use this command to place a copy of the selected text onto the Windows Clipboard.

Paste

This command inserts the contents of the Windows Clipboard into the document at the cursor position.

Note: *The Paste command will disabled if the contents of the Clipboard are not compatible with the Text Editor.*

Delete

Use this command to remove the selected text from the document. Deleted text is not placed onto the Windows Clipboard and cannot be pasted back into the document. If you delete text by mistake, you can use the Undo command to recover it.

Select All

Use this command to select all the text in the document.

Insert Text File

Use this command to insert an external Rich Text or plain text file into the current document at the cursor position.

Insert Image File

This command allows you to insert a small graphic file into your document. The file must be in Windows Bitmap format (BMP).

Find

Search for text within the current document.

Find Next

Repeat the last search command.

Replace

Search for and replace existing text within the current document.

The Hypertext Menu

Add Link

After selecting a block of text, use this command to convert the selection into a Hyperlink.

Hyperlinks containing hidden Action commands can be inserted into your text. At runtime, Hyperlinks appear as underlined text, which readers can click to execute the associated Actions. Hyperlinks are often used to navigate between pages or display information related to the underlined text.

To define a Hyperlink, highlight a word or phrase in your text and choose **Add Link** from the **Hypertext** menu (or click the **Add Hyperlink** button  in the Speed Bar). VisualNEO for Windows's Action Editor will appear allowing you to specify what will happen when the reader clicks this Hyperlink. The Hyperlink can be instructed to jump to a Bookmark (see below) within this document or execute a series of Action commands.

Edit Link

Use this command to edit the Actions associated with the Hyperlink under the cursor.

Remove Link

Use this command to delete the Hyperlink under the cursor.

Add Bookmark

This command Inserts a Bookmark into the text at the cursor position.

Bookmarks are hidden codes embedded within your text that can be used in conjunction with Hyperlinks to quickly jump to another section of the same document. To create a Bookmark, position the cursor near the text that you want to jump to and click the button. A small screen will appear requesting that you enter a name that describes the Bookmark. The name will be referenced by the Hyperlink that you'll use to jump to this point in the text. Neither the name or the Bookmark code will be visible to the reader.

Edit Bookmark

Select this command to change the name of the Bookmark under the cursor.

Remove Bookmark

This command deletes the Bookmark under the cursor.

The Table Menu

New Table

Select this command to insert a simple table into the document at the cursor position. A small screen will appear allowing you to define margins and the number of rows and columns for the table. Once created, text may be entered into the table's cells.

Insert Row

This command adds a new row (at the cursor position) to the current table.

Insert Column

This command adds a new column (at the cursor position) to the current table.

Delete Row

Selecting this command will delete the row containing the cursor in the current table.

Delete Column

Use this command to delete the column containing the cursor in the current table.

Combine Cells

This command will merge selected cells horizontally in the current table. To select cells, drag the mouse cursor across the cells to be selected.

Split Cells

This command will divide the current cell into two cells.

Cell Properties

This command will display a screen allowing you to define the text alignment, color, spacing and border for the current cell or selected cells.

Table Properties

This command displays a screen allowing you to define the margins for the current table. The margins define amount of space between the table and the left and right sides of the document.

The Format Menu

Font

This command displays a list of fonts currently installed on your computer. You may select a font, size and style to apply to the current text selection. To change the font for the entire document, use the Select All command.

Hint: *You can also use the Speed Bar's Font Style controls to change the font, size and color of selected text.*

Paragraph

This command will display a screen allowing you to define the text alignment, color, spacing and border for the current paragraph.

Tabs

This command allows you to set or adjust tab stops for the current paragraph. To create a new tab stop, enter the position of the new tab, select an alignment method and click the Set button.

Hint: *You can also click the ruler to set tab stops. Tabs can be deleted by dragging them off the ruler.*

Bullets

Selecting this command allows you to format the current paragraph or selection as a bulleted or numbered list.

The Options Menu

Check Spelling

This command checks your text allowing you to identify and correct misspelled words.

Show Formatting Codes

When this option is enabled, special symbols will appear onscreen showing tab characters and the end of each paragraph. The symbols appear only in the Text Editor and will not be saved as part of your document.

Unit of Measurement

Use this command to set the unit of measurement used for setting tabs and margins. You may choose between inches and centimeters.

The Help Menu

Using the Text Editor

Choose this option to display the section of VisualNEO for Windows's Help file containing instructions for using the Text Editor.

Using NeoToon

NeoToon is a simple utility included with VisualNEO for Windows that allows you to create your own animated cartoons. A cartoon is a series of images displayed sequentially to produce the illusion of animation - similar to a classic cartoon flip book. Viewers of the cartoon will perceive subtle changes in the sequence of images as movement. Cartoon files

are generally small in size, which is an advantage when creating publications that will be downloaded or stored on diskette. Cartoons may also be moved across the VisualNEO for Windows screen along a path which you specify.

NeoToon's Screen
NeoToon's Menu
Incorporating Cartoons into Publications

Cartoons are assembled by importing a series of images created with a paint program, screen capture utility, etc. Many animation and video editing programs also allow you to export animations as a series of images, which you may then import into NeoToon. NeoToon itself is not an image editor, so you'll need to use a separate paint or image editing application to create images for your cartoons.

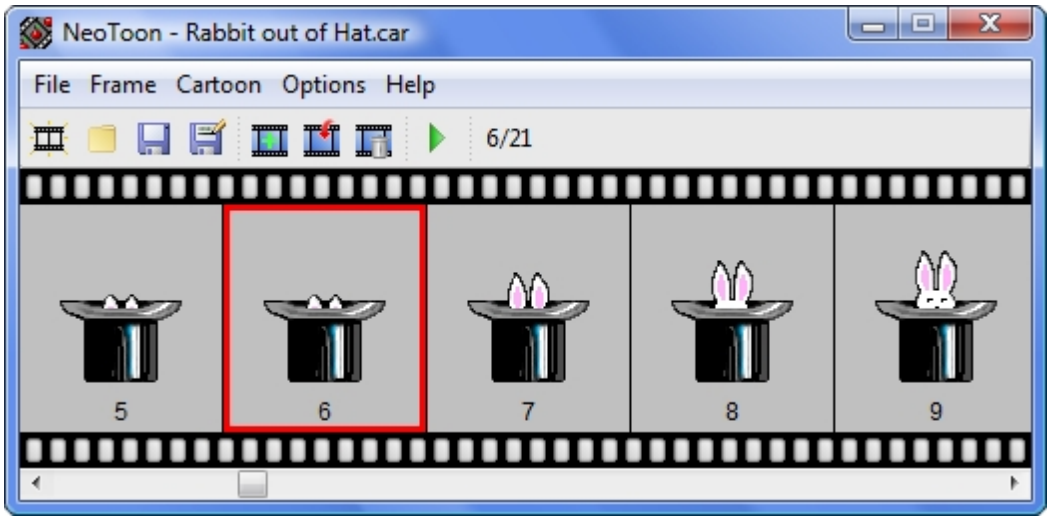
Note: *PixelNEO® from the makers of VisualNEO for Windows can be used for creating cartoon images.*

Once imported into NeoToon, images (or frames) may dragged using the mouse to change their position within the animation sequence. Use NeoToon's Play option to preview your cartoon. When you are satisfied with the animation, save the cartoon and return to VisualNEO for Windows to add it into your publication.

Note: *NeoToon will use the color of the pixel in first frame's lower left corner as the cartoon's transparent color. The screen background will show through any pixels in the cartoon's frames that match this color. Be sure and keep this in mind when creating images for a cartoon.*

NeoToon's Screen

NeoToon's screen consists of a work space, a menu bar and a speed bar. The parts of the screen are described below:



Menu Bar

The Menu Bar is a standard Windows component which appears at the top of most applications. NeoToon's Menu Bar includes commands for opening, saving and modifying cartoons. The commands found in the menu are described in the next section .

Speed Bar

The Speed Bar contains buttons that provide single click access to several often used

NeoToon commands. At the right side of the speed bar is a counter which displays the position of the currently selected (highlighted) frame, and the total number of frames in the animation.

Work Space

The Work Space occupies the largest portion of NeoToon's screen. This is the area where you will create and edit your cartoons. You may click on an frame to select it. Use your mouse drag frames to change their position within the animation sequence. Below the work space is a scroll bar which can be used view additional frames when the cartoon is too large to fit in the space available.

NeoToon's Menu

The File Menu

New

Use this function to create a new cartoon file. Enter a width and height (in pixels) to use for each of the cartoon's frames or choose the Get Width and Height from First Frame option to let NeoToon determine the size based on the first image you import.

Open

This command allows you to load an existing cartoon file (CAR) for editing.

Reopen

This command displays a list of recently opened cartoons. To load one of these cartoons, click on its title.

Save

Choose this command to save changes made to your cartoon. (Closing NeoToon while a modified cartoon is open will automatically prompt you to save your work.) If you choose not to Save your work, it will be lost once NeoToon is closed.

If this is a new cartoon that has not previously been saved, you will be prompted to specify a file name.

Save As

Use this command to save an existing cartoon using a different file name. If you select a file name which already exists, NeoToon will ask for confirmation before replacing the existing file.

Note: *Cartoon file names should end with the file extension ".CAR".*

Exit

This command closes NeoToon.

The Frame Menu

Add

This command allows you to add a new frame image to the cartoon. A standard Windows file selector will appear allowing you to locate and select an image file to import. NeoToon supports BMP, PCX, GIF, TIF, PNG and JPEG format image file. The image will be added to the end of the animation. To relocate the image, click on the frame, and drag it to a new position.

Insert

This command works just like the Add command above, but instead of inserting the image at

the end of the cartoon, the image will be inserted before the currently selected frame.

Duplicate

Use this command to create a copy of the currently selected frame. The copy will be insert immediately following the selected frame. To relocate the copy, click on the frame, and drag it to a new position. Since many animations have frames which repeat at some point, using this command will save time over using Add to repeatedly import the same image from your disk.

Delete

This command removes the currently selected frame from the cartoon.

Export

Use this command to export the currently selected frame as a Windows Bitmap file (BMP). This is handy if you've lost one of the original images used to create a cartoon.

The Cartoon Menu

Resize

Use this function to enlarge or shrink all of the frame images within the cartoon.

Note: *Stretching frames can sometimes distort images and reduce the quality of the cartoon.*

Play

Use this command to preview the cartoon as it will appear when played.

VCR style buttons across the top of the preview screen allow you to control the playback of the cartoon. To the right of the VCR controls is a counter which displays the number of the current frame, and the total number of frames in the animation. The playback speed can be adjusted using the track bar control at the bottom of the preview screen. Drag the track bar button to left to reduce the speed or to the right to increase the speed.

When you have finished viewing the preview, click the OK button to return to NeoToon's editing screen.

The Options Menu

Screen Color

You can use this command to change the background color used in NeoToon's Work Space and the Preview screen. The background color will show through transparent portions in the cartoon's frames. This color merely serves as the background for viewing frames in NeoToon

and will not be saved as part of the cartoon file.

The Help Menu

NeoToon Help

This command displays NeoToon's help file.

About

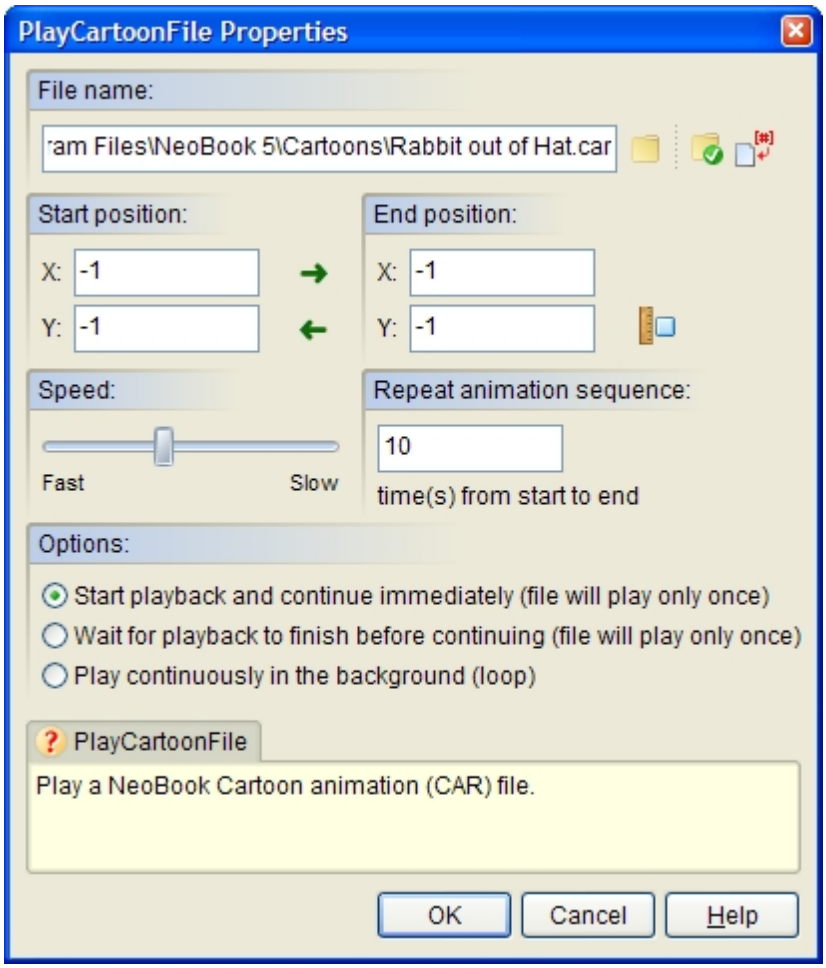
Displays information about your version of NeoToon and contact information for SinLios Soluciones Digitales.

Incorporating Cartoons into Publications

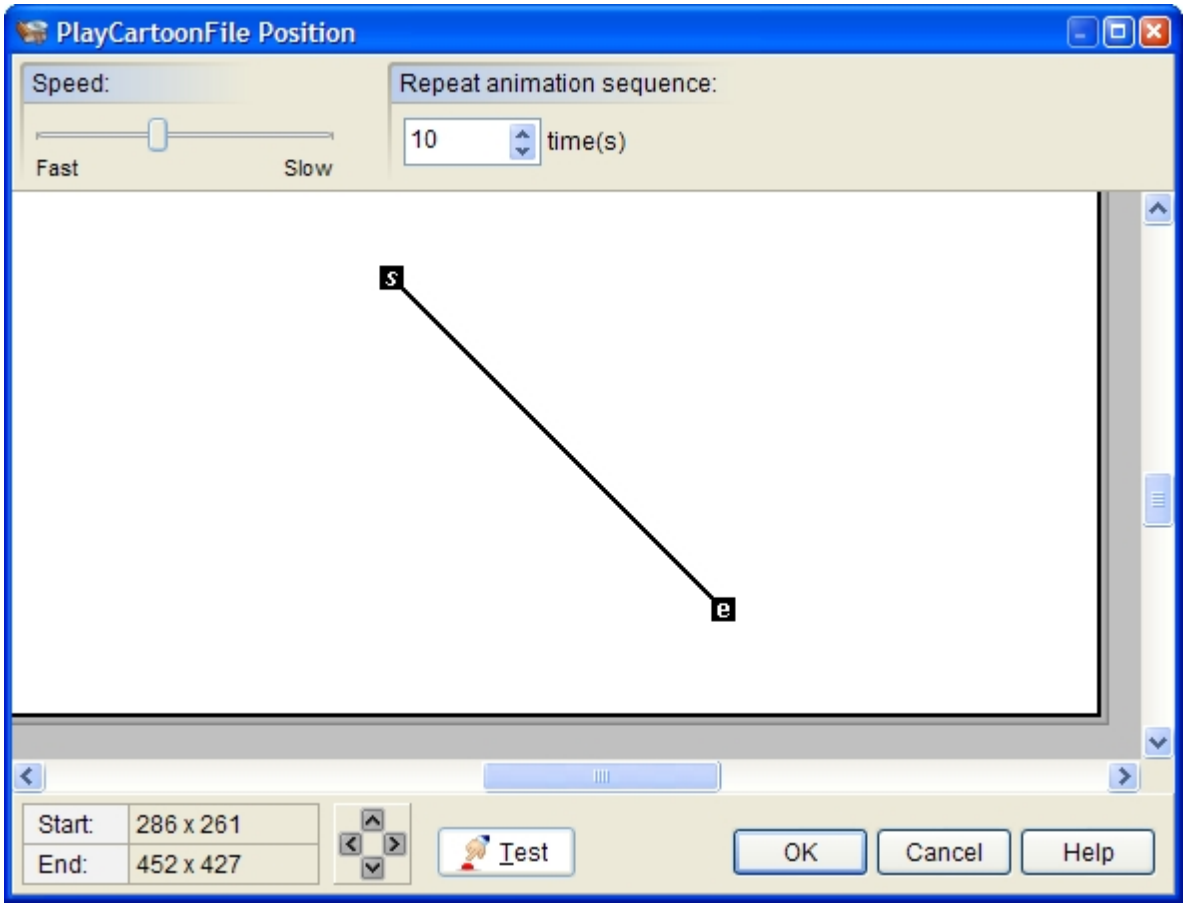
Once you have created a cartoon, you'll need to use VisualNEO for Windows's PlayCartoonFile Action to add the animation to your publications. PlayCartoonFile can be triggered by page change or an object event, such as clicking a Push Button . See Understanding Actions and Variables for instructions on adding Actions to pages and objects.

Once you have decided where your animation will appear, display the Action Editor for the object or page you will use to activate the cartoon. Use the editor's Insert Action button and select PlayCartoonFile from the list. A Windows file selector will appear allowing you to locate and select a cartoon file (*.CAR).

After you've selected a cartoon file, the PlayCartoonFile Properties screen will appear. Here you will specify where on the page the cartoon will be displayed, how fast the animation plays, and how long the animation will play.



Unlike static animations, you can specify a trajectory that the cartoon will travel across the screen. You can define a trajectory by manually entering starting and ending coordinates into the properties screen. However, most authors find it easier to click the preview button and interactively position the start and end points for the trajectory.



Note: *If you want to have the cartoon image remain stationary, enter identical starting and ending coordinates.*

Use the Speed control to determine how fast the animation travels between the end points. The Repeat Animation Sequence field also influences the animation's speed by controlling how many times each frame will be displayed as the cartoon travels between the start and end points.

Under Options indicate whether Actions continues executing ("Normal" mode) once animation begins, if the animation will run continuously between the start point and the end point using "Loop" mode, or if the animation must complete ("Wait" mode) before the next action is executed.

Use the StopMedia Action to halt an animation in progress.

License Agreement

License Agreement

The following conditions govern your use of the VisualNEO for Windows software and other materials:

You are granted a limited license to use this software. This software may be used or copied only in accordance with the terms of this license, which is described in the following paragraphs.

Copyright

Trademarks
VisualNEO for Windows™, VisualNEO for WebApps & Mobile™ and PixelNEO™ are trademarks of SinLios Soluciones Digitales. All other brand or product names are trademarks of their respective holders.

License Agreement
This software, manual and any accompanying documentation are protected by both United States and international copyright laws. Except as noted below, duplication by any means is strictly forbidden and a violation of copyright laws. This license permits you to use the accompanying software on one single-user computer system. You may not duplicate or transmit any portion of this manual, labels, packaging, serial number/registration code, Product Registration Card, or related printed information included with this product. You are welcome to share the unregistered, evaluation version of this software with friends and associates. You may NOT give anyone your serial number/registration code or transfer your registration to another person or company. You may NOT decompile, alter, tamper with or modify in any way this software or accompanying files for any purpose.

Disclaimer / Limitation of Liability
You acknowledge that this software may not be free from defects and may not satisfy all of your needs. SinLios Soluciones Digitales. warrants all media on which the software is distributed for 30 days to be free from defects in materials and workmanship under normal use. The software and any accompanying written materials are licensed "as is". SinLios Soluciones Digitales. makes no warranty concerning the function or fitness of any programs reproduced on the media included in this package. Your exclusive remedy during the warranty period shall consist of replacement of distribution media if determined to be faulty. In no event will SinLios Soluciones Digitales. be liable for direct, indirect, incidental or consequential damage or damages resulting from loss of use, or loss of anticipated profits resulting from any defect in the program, even if it has been advised of the possibility of such damage. some laws do not allow the exclusion or limitation of implied warranties or liabilities for incidental or consequential damages, so the above limitations or exclusion may not apply.

This license agreement shall be governed by the laws of the Spain and the European Union, and shall inure to the benefit of SinLios Soluciones Digitales. or its assigns.

Specific Restrictions
In accordance with the computer software rental act of 1990, this software may not be rented, lent or leased without the express written permission of SinLios Soluciones Digitales.

VisualNEO for Windows Runtime License
Licensed users may use VisualNEO for Windows to produce stand-alone applications and web content containing original material. These may be distributed in stand-alone form, provided your agreement with anyone who will use your stand-alone application and content which you

distribute acknowledges the following items:

1. VisualNEO for Windows is owned by SinLios Soluciones Digitales;
2. SinLios Soluciones Digitales will not be held responsible for any damages caused by the applications and content you create and distribute; and
3. SinLios Soluciones Digitales is under no obligation to provide product or technical support to users of the products which you create using VisualNEO for Windows.

You may only distribute the stand-alone applications and content produced by VisualNEO for Windows. You may not distribute any portion of this documentation, files, executables, packaging, serial numbers, or other related materials and printed information which are included with VisualNEO for Windows.

Technical Support

Technical Support

If you should encounter a technical problem or question, you may use one of the following avenues to obtain technical assistance:

Mail Postal Correspondence to:

SinLios Soluciones Digitales.
C/ Antonio Machado Nº7
28490. Becerril de la Sierra (Madrid)
SPAIN.

eMail Service:
info@visualneo.com

Web Site:
www.visualneo.com

Online Support Forum:
community.visualneo.com

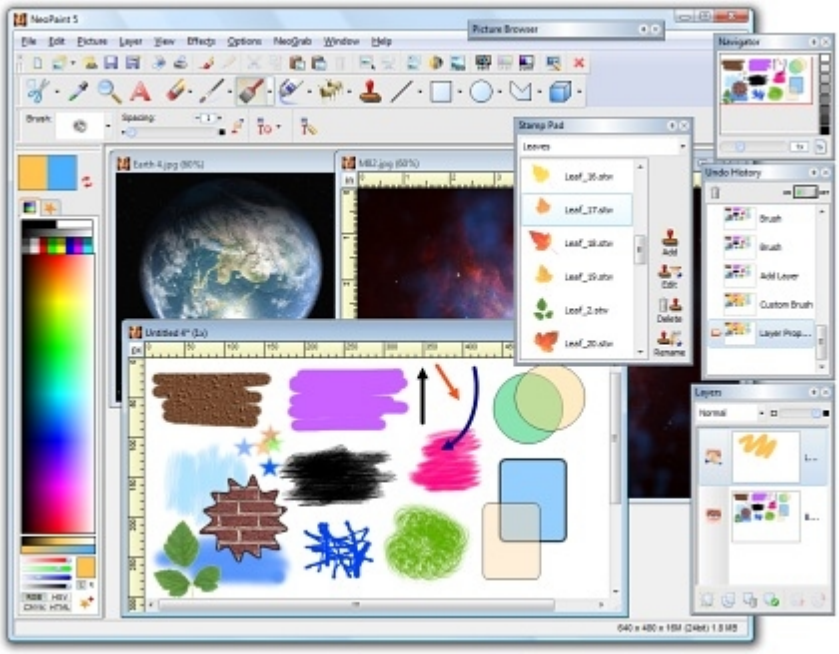
When contacting technical support, please include your VisualNEO for Windows version number, Windows version and a detailed description of the problem and how to reproduce it.

PixelNEO

Image Editor Software for Windows

The Perfect Tool for Artists, Photographers, Website & Graphic Designers, Desktop Publishers, PowerPoint™ Users, Bloggers, App Developers & Programmers, Experienced and Novice Users Alike!

PixelNEO is a full featured image editor and graphics studio that's easy-to-learn, powerful and affordable! It's simple enough for beginners, yet packed with powerful features and advanced tools that power users will love. PixelNEO makes it easy for business and home users to touch up photographs and create great looking graphics for desktop publishing, presentations, and the Internet!



Some of PixelNEO's many powerful features include: support for popular image formats like JPEG, JPEG 2000, GIF, PCX, TIFF, BMP, ICO, CUR, PSD, PNG and numerous RAW formats; integrated support for scanners and digital cameras; professional quality output to any Windows compatible printer; multiple undo levels; layers; alpha channel transparency; dozens of special effects and textures; gradients; custom paint brushes; photo retouch and color correction tools; high quality color and format conversion; levels/histogram adjustment; masking; red-eye removal; tooth whitening; cloning; Stamp Pad; 3D objects; color separation; Picture Browser with file management tools; built-in Screen Capture; and dozens of other tools and options. PixelNEO also includes some fun natural media tools like charcoal, crayon, chalk, watercolor, fountain pen and more. You can even create your own custom paint brushes.

Photographer's will appreciate PixelNEO's many photo-retouch features including red-eye removal, tooth whitening, cloning, interactive cropping, color balance, gamma correction, brightness/contrast, hue/saturation/luminosity, levels/histogram and curves.

Creating transparent 32-bit images can often be a complicated process, but PixelNEO's excellent alpha channel handling makes the process straightforward and painless. In case you're wondering, an alpha channel is a special mask used to identify which portions of a picture should be transparent (see through) and which portions should be opaque (solid).

PixelNEO's integrated screen capture utility, NeoGrab, is an extremely useful tool for creating manuals, training materials, newsletters, websites and more. Captured images are automatically transferred to PixelNEO for editing, cropping, saving, etc.

In addition, PixelNEO includes built-in tools for creating and editing Animated GIFs, and a built-in Icon/Cursor Editor for creating and editing multi-resolution icons and cursors. Unlike some basic stand-alone GIF animation utilities and icon editors, PixelNEO's integrated editor provides you with complete access to a full range of powerful painting and editing tools.

PixelNEO can also be used to create and edit VisualNEO for Windows Cartoon (CAR) files.

A trial copy of PixelNEO can be downloaded from visualneo.com.

Acknowledgments

Acknowledgements

We will be forever greatful to Dave and NeoSoftware.