

RenderWare Studio

Enterprise Author Help

Version 2.0.1

Published 07 September 2004

© 2002-2004 Criterion Software Limited. All rights reserved.

Contents

Introduction to Enterprise Author	5
Installing the application	6
Getting started with VBScript	7
Installing the Microsoft Forms 2.0 controls	8
Understanding the application	10
User interface basics	13
Splitter bars and splitter buttons	15
Stackers and tabbers	17
User interface reference	21
Design	22
Enterprise Modules	25
Main Menu	26
Object Browser	29
Objects	30
Property List	32
Script Editor	34
Toolbox	36
Using Enterprise Author	38
Creating and editing windows and dialogs	39
Changing the appearance of a window	41
Drawing controls on a page	42
Adding an Enterprise control	45
Configuring a splash window	47
Placing windows inside the application window	49
Creating and editing menus and toolbars	52
Index numbers of menu bar items	56
Showing and hiding windows and menu bars	58
Assigning objects to functional groups	60
Writing VBScript code	62
Debugging VBScript code	66
Creating and editing alternative layouts	69
Enterprise module reference	71
3D Object Controller	72
3D Object Pick Tool	73
Properties	74
Troubleshooting	76
3D Viewer	77
Properties	78
Camera Controller	80
Keyboard controls	81
Mouse controls	

Properties	84
Troubleshooting	87
Graph Tool	88
Keyboard Interface	89
Orthographic Viewer	91
Realibase File Processor	93
Properties	94
RenderWare Studio Utility Tool	96
Tooltip Viewer	97
Properties	98
Troubleshooting	100
Common properties	101
Color	102
Display driver	104
Capturing mouse events inside the Enterprise control	

Introduction

Enterprise Author is an integrated development environment for 3D visualization applications that allows you to edit and develop GUI applications quickly. It was used to develop Renderware Studio Workspace ("Workspace").

In use, Enterprise Author is a little like Visual Basic, in that it uses drag-and-drop techniques to define the way an application should look, and code to control how the drag-and-dropped components should communicate with each other.

However, Enterprise Author's palette is not as broad as Visual Basic's, and we can make some quite specific statements about any application it's been used to make:

- Applications created using Enterprise Author (including Workspace) consist of collections of interoperating ActiveX controls.
- Interaction between the user and the application, and communication between the ActiveX controls that make up the application, is handled through script code.

Note: The files that make up the Workspace application use VBScript for this purpose, but it's possible to use JScript instead if you prefer.

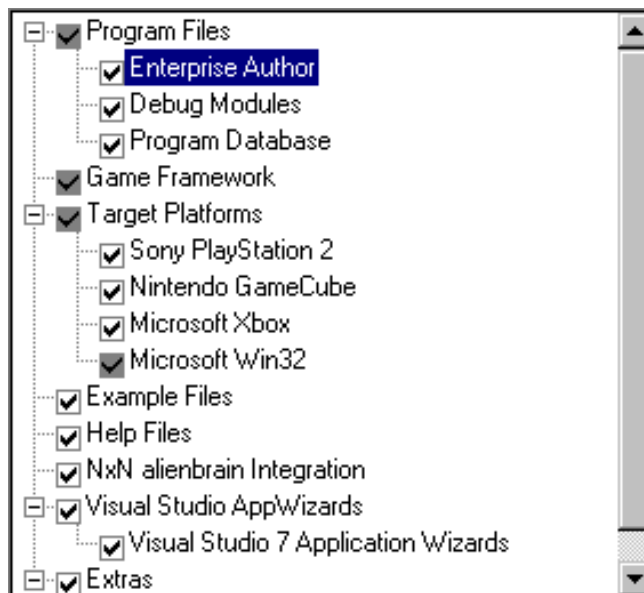
Any customization of Workspace that goes beyond shuffling windows' positions and creating new layouts involves writing or editing ActiveX controls or script code.

If you want to create ActiveX controls for Workspace's user interface, you'll need Visual C++ 7.1. For everything else—adding ActiveX controls to Workspace, and writing the script code that lets them interact with the controls that are already there—you need Enterprise Author.

Installing the application

Although Enterprise Author is *supplied* with RenderWare Studio, by default it is not *installed* at the same time as RenderWare Studio. If you didn't choose to include Enterprise Author at the time you installed RenderWare Studio, you can always add it later:

1. From Windows, select **Start ► Settings ► Control Panel ► Add/Remove Programs**.
2. In the dialog that's displayed, select **RenderWare Studio 2.0**, and then click **Change/Remove**.
3. Ensure that **Modify** is selected in the RenderWare Studio Setup dialog that appears, and then click **Next >**. The display will change to show this window:



4. Select the **Enterprise Author** check box, and complete the rest of the wizard as usual. Enterprise Author will be installed on your computer.

Other installation tasks

You don't *need* to do anything else to make Enterprise Author work, but there are some things that we *recommend* you do before using the application:

- Download the [Microsoft Script Debugger](#) (p.7), which links directly into Enterprise Author, and get hold of some good [VBScript documentation](#) (p.7).
- Install the [Microsoft Forms 2.0 ActiveX controls](#) (p.8).

With these pieces in place, you can start Enterprise Author, and begin to customize the Workspace user interface.

Getting started with VBScript

If you want to customize either the RenderWare Studio Workspace user interface or the Game Production Manager build rules, then you will need to know how to program in the Microsoft Visual Basic Scripting Edition (VBScript) language.

VBScript documentation

The best place to begin learning VBScript is the [Microsoft Scripting website](http://msdn.microsoft.com/scripting/) (msdn.microsoft.com/scripting/). At the time of writing, you could download VBScript documentation in an HTML Help file called *Microsoft Windows Script 5.6 Documentation*. If you have the Microsoft Developer Network (MSDN) Library, then you already have this documentation: look in **Web Development ► Scripting ► SDK Documentation ► Windows Script Technologies**.

There are also many good books on VBScript, such as *VBScript in a Nutshell* (ISBN 1565927206, published by [O'Reilly & Associates](http://www.ora.com/) (www.ora.com/)).

Microsoft Script Debugger

You can use Microsoft Visual Studio .NET 2003 to debug VBScript code, but if that isn't installed on your computer, we recommend that you download and install the Microsoft Script Debugger. This tool, which is freely available by following the links from the [Microsoft Scripting website](http://msdn.microsoft.com/scripting/) (msdn.microsoft.com/scripting/), enables you to set breakpoints, step through code, examine the call stack, and query or change values.

Tip: Make sure that you download the correct version of the debugger. There's one for Windows 98 and Me, and another for Windows NT 4.0, 2000, and XP.

Before using the debugger, you need to edit the Windows registry, as described in Microsoft Knowledge Base article [252895](http://support.microsoft.com/support/kb/articles/q252/8/95.asp) (support.microsoft.com/support/kb/articles/q252/8/95.asp):

1. Start the Registry Editor:
 - a. On the **Start** menu, click **Run...**
 - b. Type `regedit` and then click **OK**.
2. Find the following registry key:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows  
Script\Settings\JITDebug
```

3. Set its value to 1.

The debugging process

For information on starting, stopping, and controlling the debugging process, see the documentation supplied with the debugger, and the topic called [Debugging VBScript code](#) (p.66).

Installing the Microsoft Forms 2.0 ActiveX controls

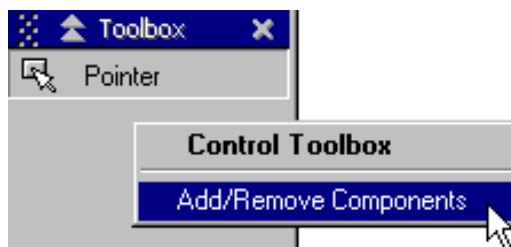
Customizing a user interface means working with edit controls, buttons, and list boxes, but Enterprise Author is not supplied with ActiveX controls for these basic user interface items. Before customizing an application with Enterprise Author, you need to install some ActiveX controls that implement these common features.

You can use *any* ActiveX controls with Enterprise Author, but for user interface elements we recommend the Microsoft Forms 2.0 ActiveX controls, for the following reasons:

- They are the controls that the developers of Enterprise Author used during product testing.
- They are supplied with Microsoft Office.
- They are available for free from Microsoft, as part of the Microsoft ActiveX Control Pad (see below).

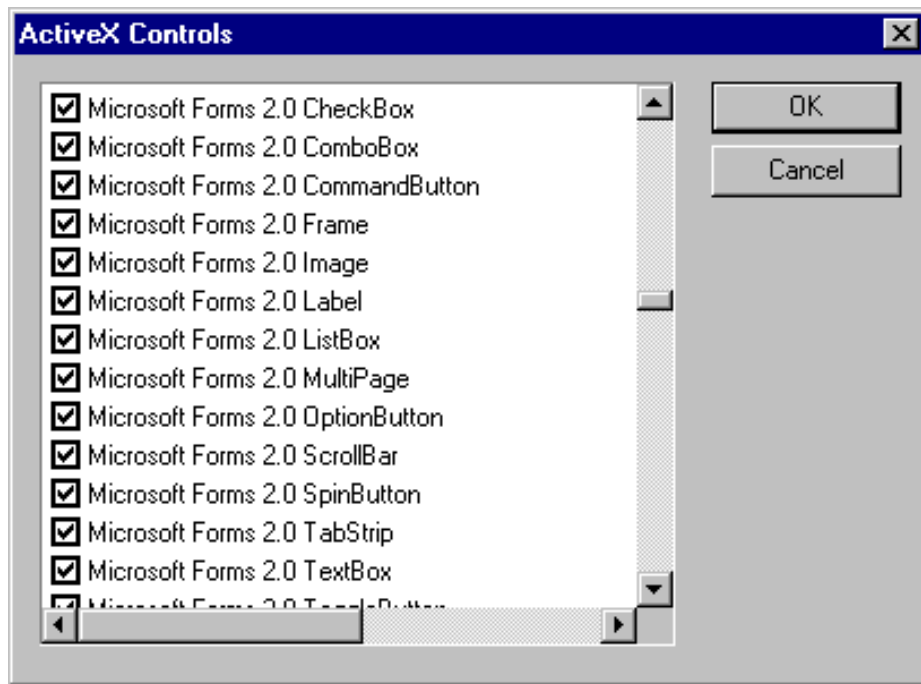
If Microsoft Office is installed on your computer, then you already have the Microsoft Forms 2.0 ActiveX controls. If you don't have Microsoft Office, then you need to download and install the Microsoft ActiveX Control Pad, and *then* add the controls to the Enterprise Author Toolbox:

1. Go to the [Microsoft Developer Network \(MSDN\) website](http://msdn.microsoft.com) (msdn.microsoft.com).
2. Search for Knowledge Base article 224305, which contains a link to the ActiveX Control Pad page.
3. Install the ActiveX Control Pad on your computer.
4. Start Enterprise Author:
Start ► Programs ► RenderWare Studio ► Enterprise Author
5. Right-click an empty area of the Toolbox window, and then click **Add/Remove Components**:



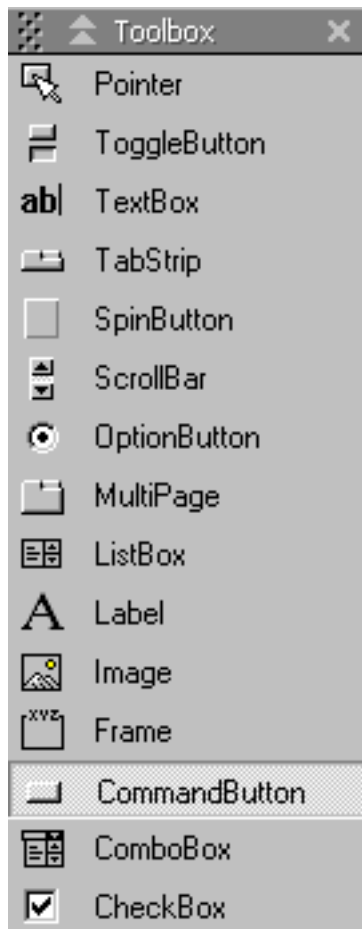
A dialog box appears, listing all of the ActiveX controls installed on your system.

6. Select the check boxes for each of the **Microsoft Forms 2.0** ActiveX controls:



7. Click **OK**.

The Toolbox now contains icons for the new controls:



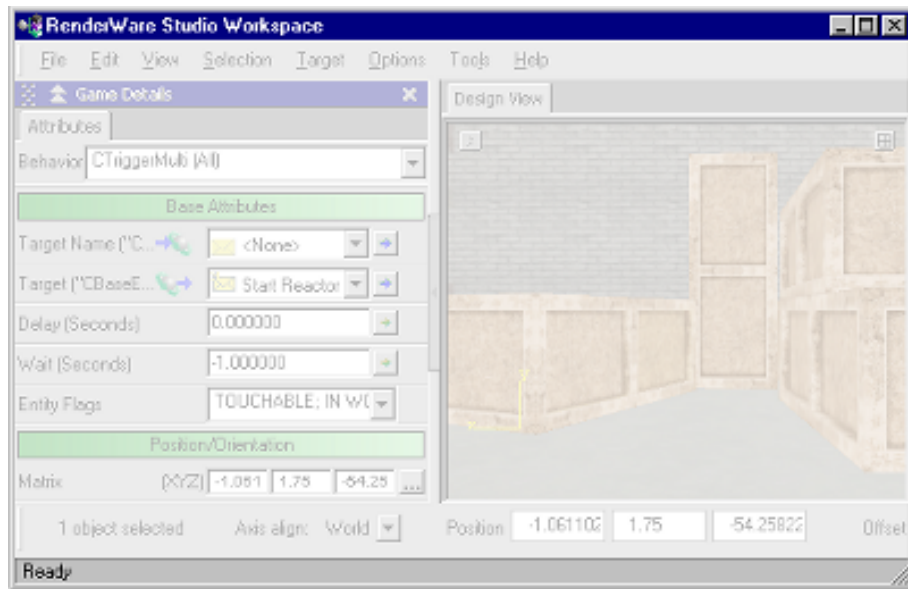
8. Close Enterprise Author.

Understanding the application

An application developed using Enterprise Author is a composite—that is, it's an assembly of separate *objects* that work together to provide the application's overall functionality. A given application can consist of a large number of these objects (60 or more is not uncommon), but the objects themselves have just seven different types:

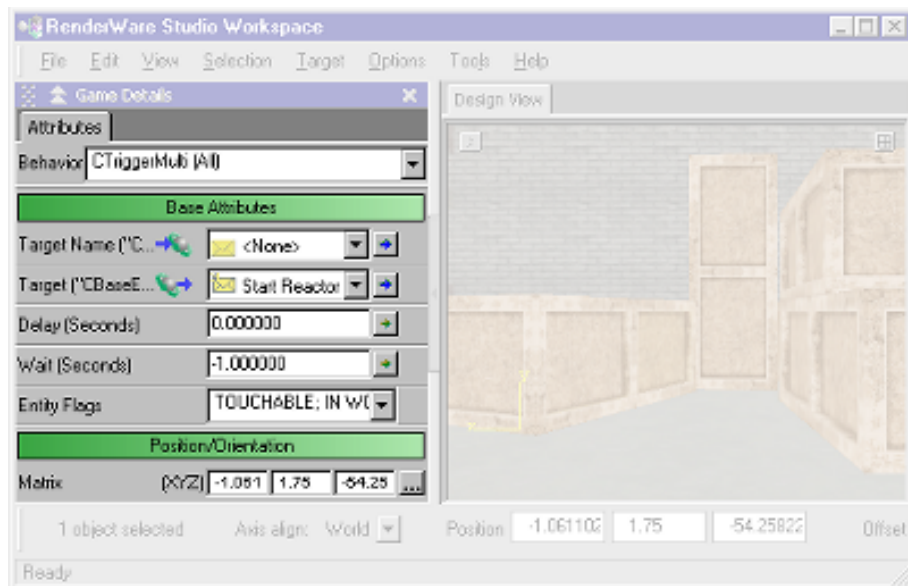
RENFrame

A frame that can play host to several other object windows. (Sometimes known as an *application window*.)



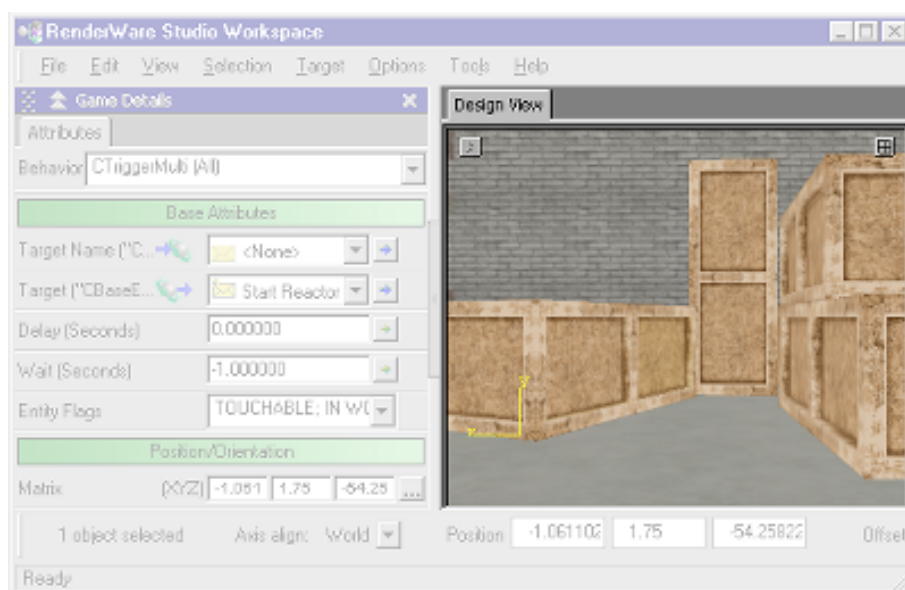
Control Host

A single ActiveX control that appears in a window that can be hosted by a RENFrame.



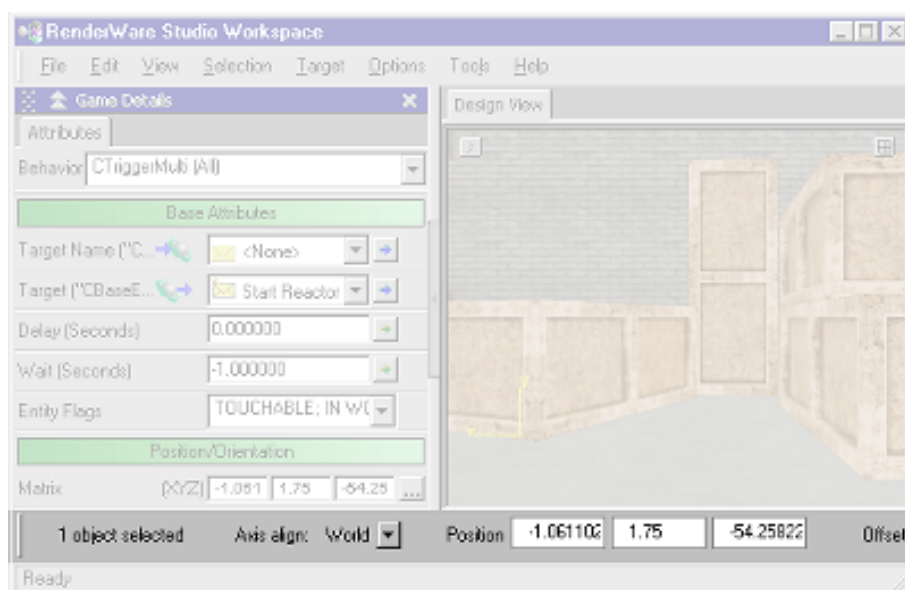
Enterprise Control

A single [Enterprise control](#) (p.71) that appears in a window that can be hosted by a RENFrame.



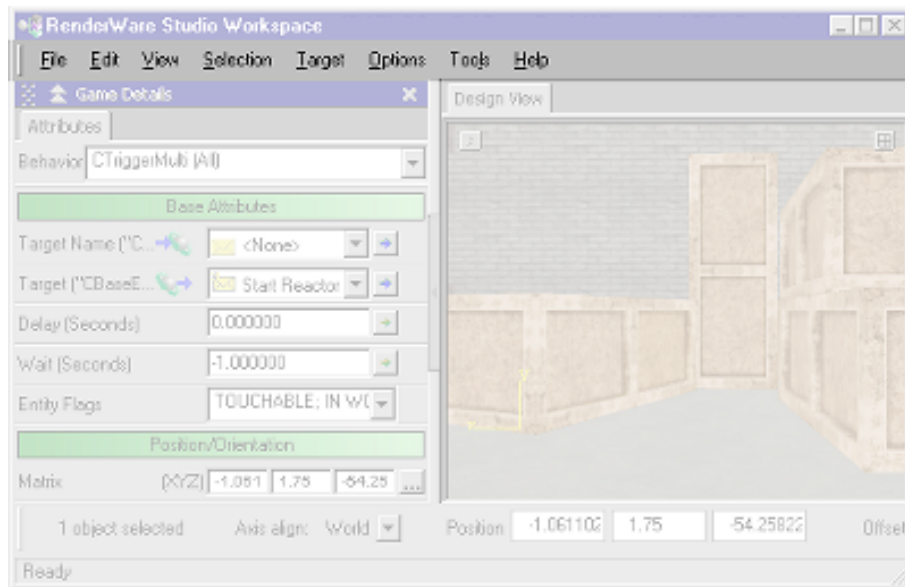
Page

A window that can be hosted by a RENFrame, and can itself contain several ActiveX controls.



Menu Bar

A menu bar, or a toolbar, or a context menu.



Script Module

An object that just contains script code, with no graphical representation.

Splash Window

A window containing an image (and, optionally, some text) that appears during application startup.

Enterprise Author provides ways of creating, configuring, and managing objects of these different types. As a result, it allows you to customize (and even to create from scratch) applications that are constructed from them.

What do you mean by “object”?

It's an unfortunate fact that the term “object” has become burdened with many different meanings. Depending on your background, you may have come across C++ objects, Visual Basic objects, COM objects, business objects, or several others.

In this part of the documentation, “object” means an Enterprise Author object—that is, something that's represented in the [Objects window](#) (p.30). When we need to refer to another kind of object, we'll spell it out.

User interface basics

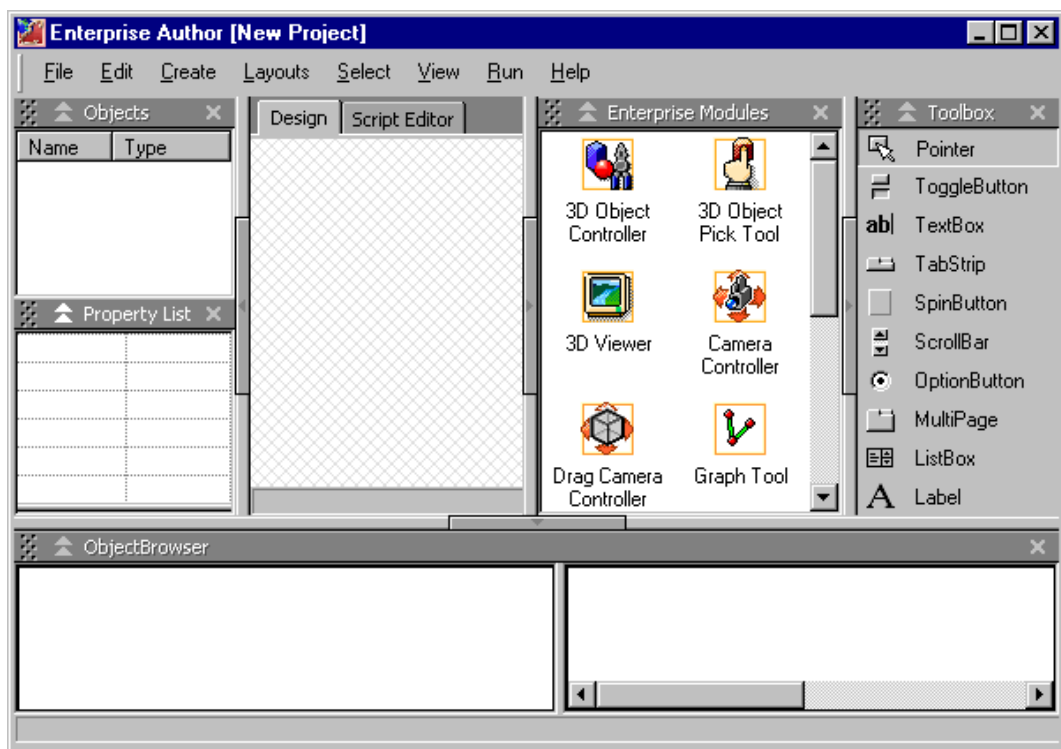
Enterprise Author's user interface is aimed at providing a logical environment in which to create Windows applications that are potentially quite complex. Consequently, Enterprise Author's look-and-feel is broadly similar to other modern Windows applications. However, its enhancements allow users greater-than-normal freedom to change its appearance.

Default appearance

Enterprise Author's user interface is completely configurable, and there are a couple of presets that you can select from the **View ► Layout** menu:

- *Standard*, which appears by default when you start Enterprise Author, and contains every window the application can display
- *Design View*, which is optimized for designing the layouts of the applications you create with Enterprise Author

When you start Enterprise Author for the first time, the application window looks something like this:



However, the versatility of the elements in the Enterprise Author interface means that something that appears as a tab in one layout can be a separate window in another. We'll use the screenshot above to introduce some useful terminology.

Element names

Beneath the menu bar in the screenshot, you can see four columns of user interface controls. From left to right, the first, third, and fourth of these (the ones

that have title bars) are *stacker windows* (“stackers”). The second column is called the *work area*.

Each stacker can hold one or more *stacker elements* in a column. (The leftmost stacker, for example, has two stacker elements, called Objects and Property List). All stackers are bordered by either the application window or a *splitter* that contains a *splitter button*.

A stacker element (and the work area) can contain a single window that completely fills it, or a *tabber* that can contain any number of tabbed windows. It's also worth noting that only the row containing the work area can be split into columns; all other rows occupy the full width of the application window.

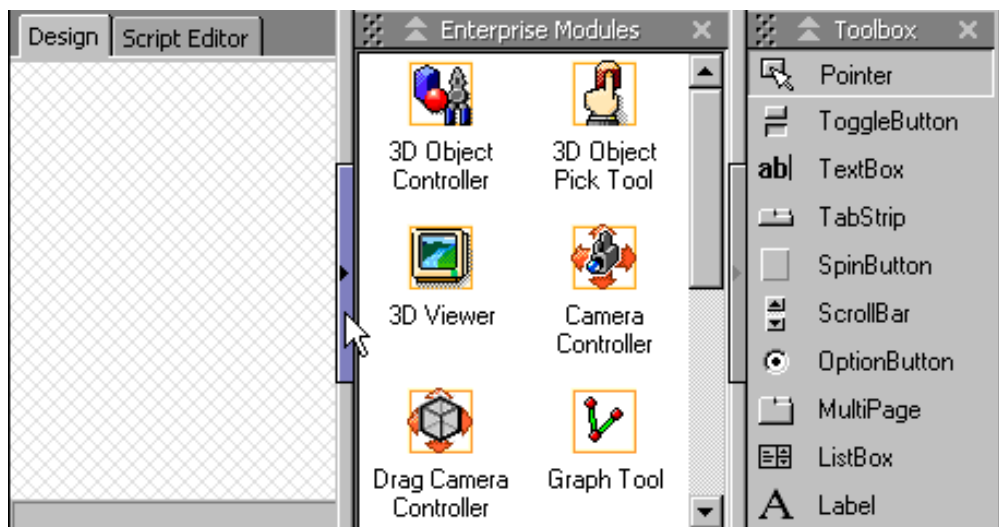
Note: Any changes you make to the Enterprise Author user interface are saved automatically when you quit the application. Windows' sizes, positions and types are all retained, so the project you open looks exactly the same as the one you closed.

Splitter bars and splitter buttons

One way of altering Enterprise Author's appearance is to drag a splitter to change the sizes of the areas it separates. Alternatively, you can click a splitter button to expand or collapse a stacker in the direction of the arrow drawn on it.

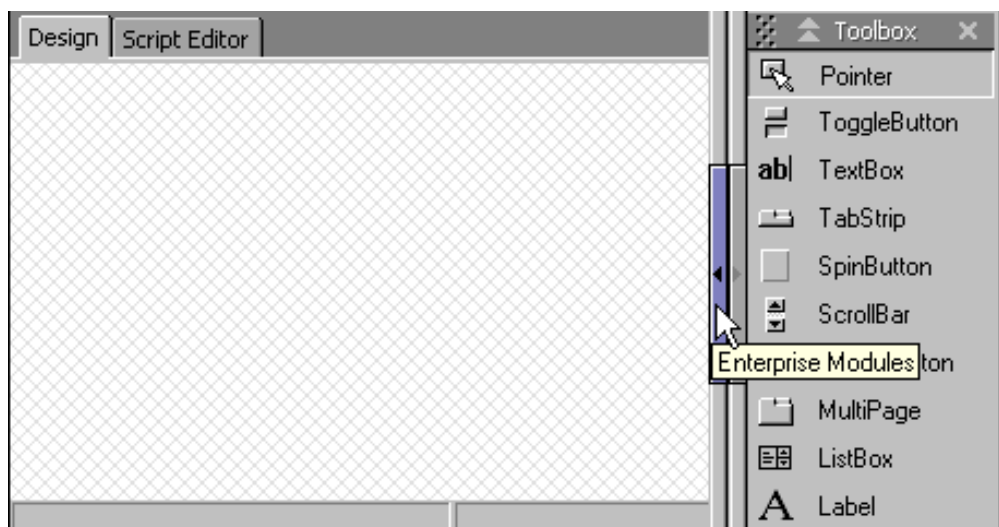
Stackers always collapse towards the edge of the application window, so a collapsed (hidden) stacker is indicated by a splitter button whose arrow points inwards, away from the edge. Try the following:

1. Click the splitter button between the work area and the stacker containing the Enterprise Modules element:



The splitter moves across to the next splitter in the row, the work area expands to fill the space available, and the stacker containing the Enterprise Modules element collapses.

2. Hover over the splitter button:



Not only has the arrow changed direction to indicate a hidden stacker, but also the tooltip reveals the name of the stacker elements that have been

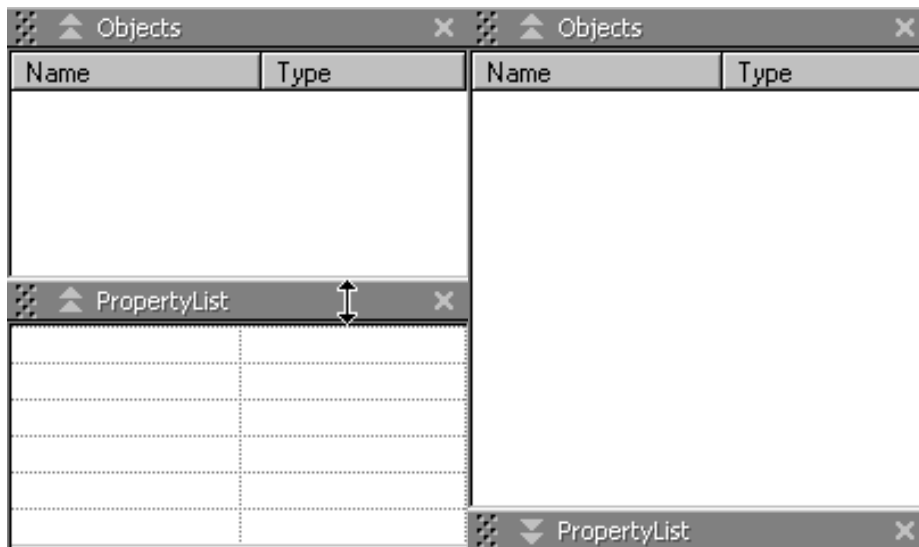
hidden.

Stackers and tabbers

The position, appearance, and content of stackers and tabbers can be managed through the title bars of individual stacker elements, and the tabs of tabbed windows.

Stackers

In simple use, the title bars of elements *within* stackers perform a similar function to the splitters *between* stackers:

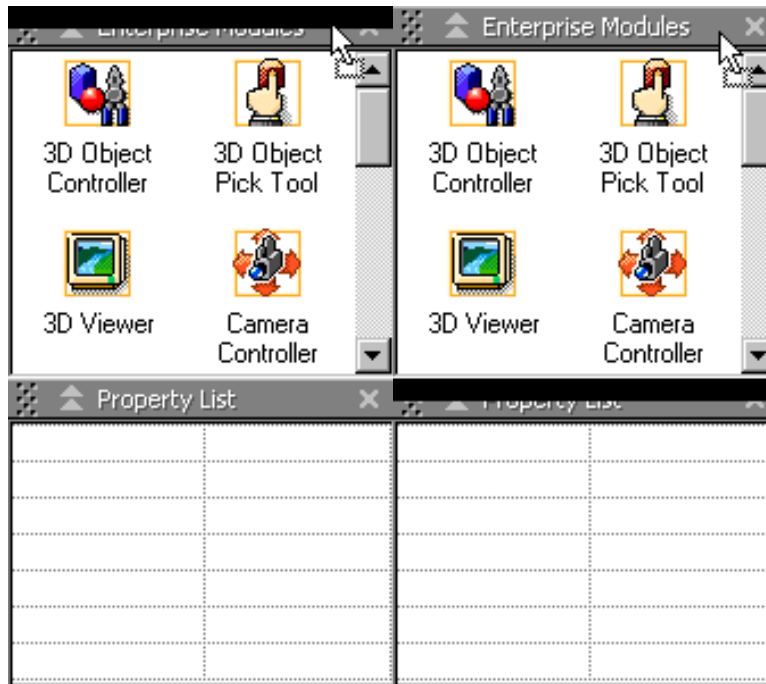


Dragging the title bar of the Property List element on the left, between the caption and the Close icon, changes the sizes of both elements within the stacker. Clicking the caption, on the other hand, has the result you can see on the right.

The double “up” arrow in the title bar represents that the stacker element will collapse when the title bar is clicked. The Objects element expands to fill the available space. Clicking the caption again restores both elements to their previous sizes and positions.

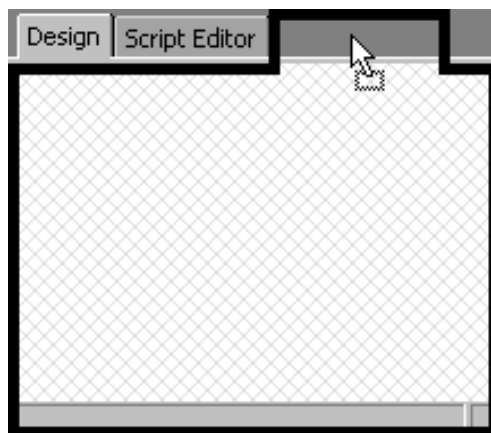
Dragging the *grabber* control (■) at the left-hand edge of a stacker element's title bar has a different effect: the element is relocated when you drop it. A black rectangle provides a cue to the location of the drop:

- If you drop the element on the top half of the title bar of another stacker element, it will take up position above that element, in the same stacker (left image, below).
- If you drop the element on the bottom half of the title bar of another stacker element, it will take up position below that element, in the same stacker (right image, below).



You can combine these techniques to reorder the elements in a stacker, or to move elements between stackers.

- If you drop the element over a tabber, it will be converted to a tabbed window and added to the set already present. If the element already contains a tabber, its tabbed windows will all be added to the destination set.



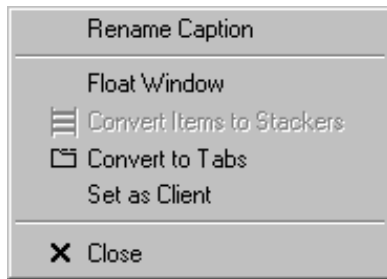
- If you drop the element on a vertical splitter, or one of the sides of the application window, a new stacker will be created in the area indicated by the cue rectangle.

Depending on the position of the drop point, this will either add a new column to the row containing the work area, or a new row above or beneath the work area.

Note: Pressing **Esc** during a drag cancels immediately the operation in progress.

Stacker element context menu

Right-clicking a stacker element's title bar displays this context menu:



Rename Caption

Allows you to edit the name of the stacker element, which appears in its title bar (and also in the tooltip that appears over a splitter button). Type the new name and press **Return**.

Float Window

Places the stacker element in a floating window that can be moved over and outside the application window. A floating element can be docked using its grabber control, in the same way as any other stacker element.

Convert to Tabs

Changes the content of the stacker element to a tabber containing a single tabbed window. After selection, this menu item becomes unavailable. It is replaced by **Convert Items to Stackers**, which performs the reverse operation.

Set as Client

Replaces the controls in the work area with the contents of the stacker element. The former contents of the work area are placed in a floating window that may be docked elsewhere.

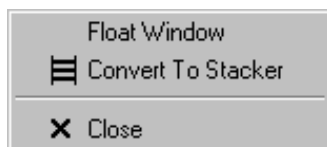
Tabbers

Tabbers and tabbed windows have a close relationship with stackers and stacker elements, and the two pairs of controls have broadly comparable functionality. In user interface operations, the tab of a tabbed window performs the same role as a stacker element's grabber control.

Dragging a tab from a stacker element, the work area, or a floating window allows you to reposition it in the Enterprise Author application window, following the same rules as for dragging a stacker element.

Tabbed window context menu

Right-clicking a tab within a tabber displays this context menu:



Float Window

Places this tab's window in a floating window that can be moved over and outside the application window. A floating window can be docked through its grabber control.

Convert To Stacker

Changes this tab's window to a separate stacker element that appears beneath (and in the same stacker as) the stacker element that spawned it.

Enterprise Author windows

Between the [splitters and stackers](#) (p.13), in a variety of combinations and positions, are the eight windows that make up the Enterprise Author user interface. At any given time during your work with Enterprise Author, you'll be dealing with some or all of the following:

Design (p.22)

The place in Enterprise Author where graphical objects appear when you create or edit them. To varying degrees, objects in the finished application look similar to how they appear in the Design window.

Enterprise Modules (p.25)

Lists the modules that you can add to an Enterprise control so that they're supported in the application you're building.

Main Menu (p.26)

A menu bar featuring generic items such as those for opening and saving projects, and application-specific items covering creation of and navigation between different interface layouts.

Object Browser (p.29)

Displays the Automation interfaces (their properties, methods, and events) that are exposed by the objects in your application, and therefore available for use in your script code.

Objects (p.30)

Lists all of the objects in the application you're creating, and allows you to select them for editing.

Property List (p.32)

Shows the properties of the object that's currently selected in the Objects window or the Design window, and makes them available for editing.

Script Editor (p.34)

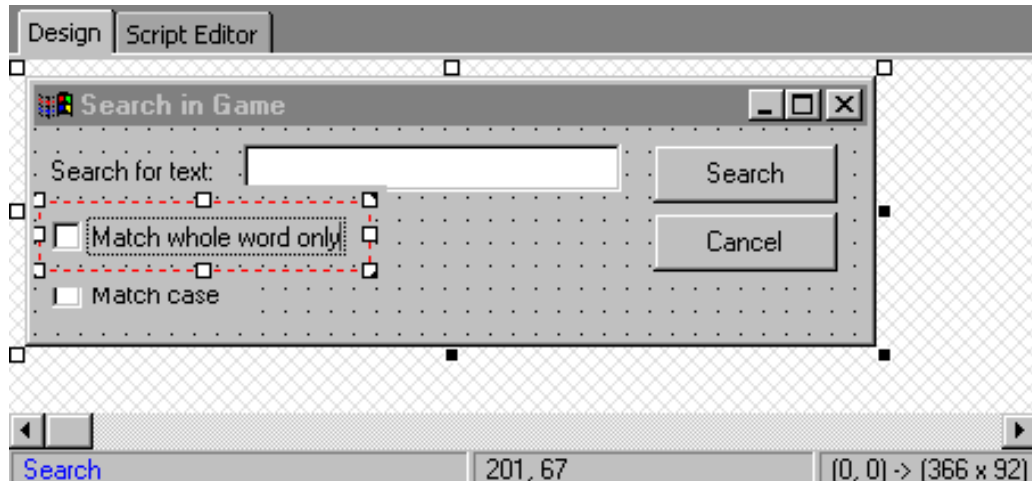
Shows the script module for the currently selected object, and provides editing functionality.

Toolbox (p.36)

Allows selection of the ActiveX controls that you can draw on a page object to form part of your application's user interface.

Design

The place in Enterprise Author where graphical objects appear when you create or edit them. To varying degrees, objects in the finished application look similar to how they appear in the Design window.



The user interface

Like many other parts of Enterprise Author, the appearance of the Design window depends on the object you've selected to edit from the [Objects window](#) (p.30). With a script module open, Design is empty, as a script has no graphical representation. Other objects provide different degrees of WYSIWYG functionality.

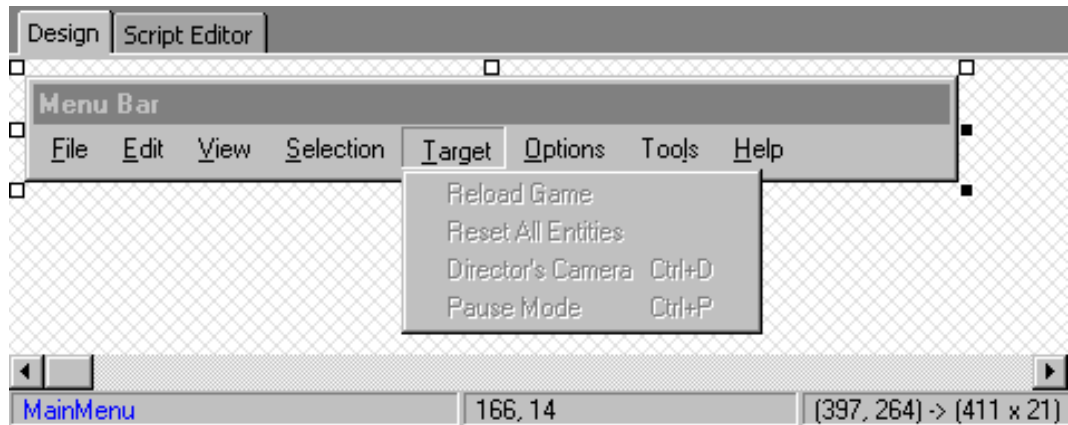
Pages

As illustrated above, Design displays page objects in fully editable form. You can draw controls from the [Toolbox window](#) (p.36) onto the page, and make changes to their window properties in an intuitive fashion.

Menu bars

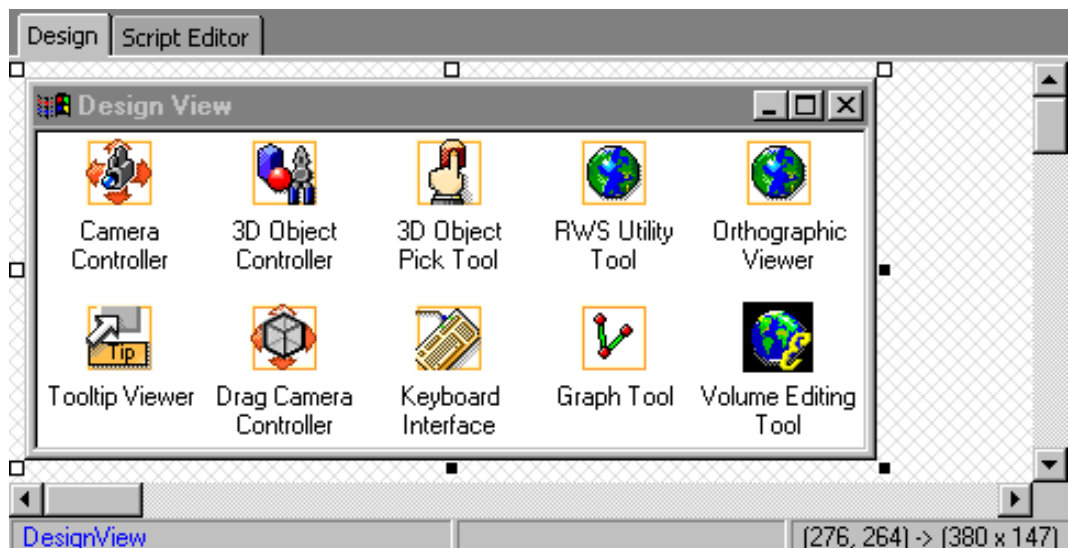
Menu bar objects are just as editable as page objects, but the procedure for doing so is different. There's no drag and drop mechanism, so you have to open the dialog from the [Property List window](#) (p.32) instead.

The Design window shows a representation of the menu bar as currently specified via Property List, but offers no editing features of its own.



Enterprise controls

Opening an [Enterprise control](#) (p.71) in the Design window displays a list of the Enterprise modules that the control will support in the running application. Modules can be added to the control by dragging and dropping from the [Enterprise Modules window](#) (p.25).



Control hosts

The appearance of a control host object in the Design window depends entirely on the ActiveX control involved. If it has a design-time interface, you'll be able to interact with it. If it doesn't, you'll just see an empty window.

Splash windows

In the Design window, splash window objects take on the appearance they will have in the finished application, including the bitmap image they contain and any strings they retrieve from the system registry.



RENFrames

Editing a RENFrame object in the Design window gives you the ability to specify the run-time layout of your application. Objects of any of the four preceding types can be dragged and dropped from the [Objects window](#) (p.30) into the RENFrame, and then manipulated according to exactly the same rules as those for manipulating the windows of Enterprise Author itself.

In addition, any objects that are editable when opened alone in Design are *also* editable when they're visible in the RENFrame interface.

Enterprise Modules

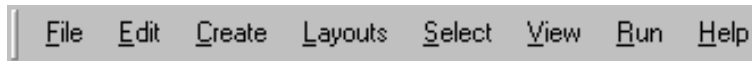
Lists the modules that you can add to an Enterprise control so that they're supported in the application you're building.



The Enterprise Modules window allows you to drag the Enterprise modules it contains into an Enterprise control that's visible in the [Design window](#) (p.22). It has no other functionality.

Main Menu

A menu bar featuring generic items such as those for opening and saving projects, and application-specific items covering creation of and navigation between different interface layouts.



The **File** menu provides standard new/open/save/close functionality, plus an item that allows you to compile your project into a single executable application.

File	
New...	Creates and opens an Enterprise Author project.
Open...	Opens an existing Enterprise Author project file (.ren).
Save	Saves the current project under its current name.
Save As...	Saves the current project under a different name.
<hr/>	
Make Executable...	Saves the current project as a standalone executable file (.exe).
<hr/>	
Recent Files ►	Lists projects that you have recently opened. To open one of these projects, click its file name in the list.
<hr/>	
Exit	Closes Enterprise Author.

The items of the **Edit** menu become active when you're working in the [Script Editor window](#) (p.34), and provide basic editing features.

Edit	
Undo	Undoes the previous edit.
Redo	Redoes an edit that was undone.
<hr/>	
Cut	Moves the selected text to the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes the contents of the clipboard at the current insertion point.

Delete

Deletes the selected text.

Find...

Finds the specified string in the current script module.

Find Previous

Searches backwards through the current script module for the find string.

Find Next

Searches forwards through the current script module for the find string.

Go To...

Scrolls to the specified line number in the current script module.

The **Create** menu provides, in a single location, the ability to create any of the [objects](#) (p.30) you may require in an Enterprise Author-generated application.

Create**Script Module**

Creates a script module object.

Forms Page

Creates a page object.

Menu/Toolbar

Creates a menu bar object.

Enterprise Control

Creates an Enterprise control object.

Splash Window

Creates a splash window object.

Control Window...

Creates a control host object.

Items in the **Layouts** menu are to do with the application you're creating or editing, rather than Enterprise Author itself. The menu allows you to create and edit the files that contain your different layout definitions.

Layouts**Create**

Displays a dialog asking for a name for your new layout, and then creates it. The name of the new layout is added to the Select menu.

Rename

Displays a dialog prompting you to give a new name for the layout that's currently being edited.

Delete

Deletes the current layout, provided that at least one other layout exists. It's impossible to delete your one-and-only layout.

The items in the **Select** menu are the names of all the layouts in your application. The layout that's currently open for editing has a check mark next to its name in the list.

Select**Layout 1**

Displays *Layout 1*, and opens it for editing.

Layout 2

Displays *Layout 2*, and opens it for editing.

Layout 3

Displays *Layout 3*, and opens it for editing.

The **View** menu allows you to choose between Enterprise Author's two built-in layouts, and to show or hide any of the windows listed [here](#) (p.21).

View**Layout ►****Standard**

Switches to Enterprise Author's *Standard* layout.

Design View

Switches to Enterprise Author's *Design View* layout.

The **Run** menu contains a single item that allows you to test your application as you edit it.

Run**Start**

Runs the current project.

The items in the **Help** menu provide standard help/about functionality.

Help**Enterprise Author Help**

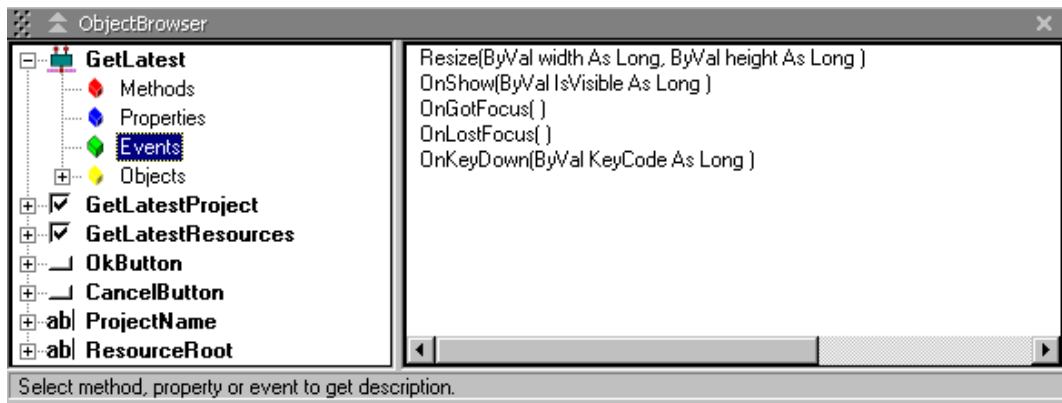
Displays this help file.

About Enterprise Author

Displays information about this version of Enterprise Author.

Object Browser

Displays the Automation interfaces (their properties, methods, and events) that are exposed by the objects in your application, and therefore available for use in your script code.



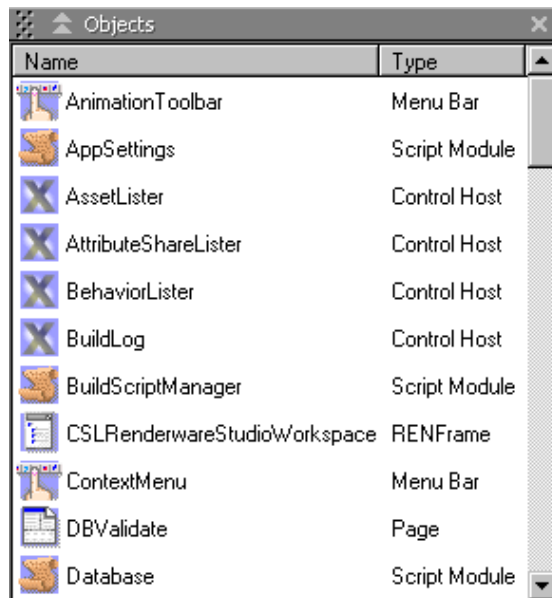
The Object Browser window helps with writing the script code that holds your application together.

In its left-hand pane, Object Browser lists the objects that are local to the current script module. For most object types, there will only be one item in this list: the object that's currently open for editing. For page objects, however, you'll also see the controls from the [Toolbox window](#) (p.36) that you've placed on the page. Expanding the entry for a given object lets you choose to view the methods, properties, and events supported by that object, or to “drill down” into its sub-objects.

On selecting to view an object's properties, methods, or events, a list appears in Object Browser's right-hand pane. Using VBScript syntax, method and event names are displayed along with the number and types of their parameters, while properties' types are also provided.

Objects

Lists all of the objects in the application you're creating, and allows you to select them for editing.



User interface

The Objects window's two columns list the names of all the objects in the application you're editing, and their Enterprise Author types. Objects appear in the window after being created through Enterprise Author's **Create** menu (apart from the single RENFrame object that's present by default in every project). For easier recognition, different object types have different icons:



Control Host: A window that contains a single ActiveX control in its client area. This can be any ActiveX control installed on your computer; you are not limited to the set in the Toolbox window.



Enterprise Control: A window that contains only an Enterprise control in its client area. An Enterprise control can contain any or all of the Enterprise modules. Without modules, an Enterprise control does nothing.



Menu Bar: A menu bar or a toolbar (the two are interchangeable in Enterprise Author) that provides for a high degree of customization. The script code you attach can invoke methods on other objects in your application.



Page: An empty page (a "blank" window) onto which you can draw any of the ActiveX controls in the Toolbox window. You can attach script code to handle the events fired by the controls it hosts.



RENFrame: A window that can hold any of the other Enterprise Author objects, alone or together, arranged in stackers and tabbers. Often referred to as the "application window".



Script Module: A store for global variables and procedures that may be used from any other object in your application. You can use a script module to store code that's common to several objects, avoiding the need for duplication.

Splash Window: A window containing an image that appears during application startup. It can also display short text strings containing registration or version information, for example.

Mouse controls

The object names in the left-hand column of the Objects window respond to clicks, double-clicks, and right-clicks:

- Clicking an object name (or its associated icon) selects it. Clicking the name of an object that's already been selected allows you to rename that object.
- Double-clicking an object name opens it for editing in the [Design window](#) (p.22) and/or the [Script Editor window](#) (p.34), and displays its properties in the [Property List window](#) (p.32).
- Right-clicking an object name displays the Objects window's context menu.

Context menu

The context menu for objects in the Objects window contains the following items:

Design

Opens the object for editing in the [Design window](#) (p.22) and/or the [Script Editor window](#) (p.34). (Equivalent to double-clicking.)

Rename

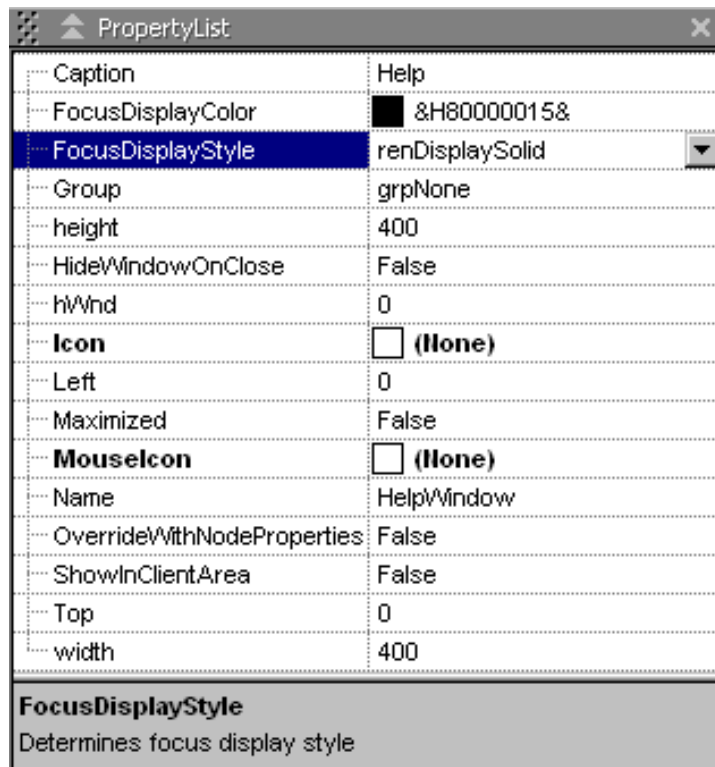
Opens the object's name for editing. (Equivalent to clicking a selected object.)

Delete

Displays a dialog prompting you to delete the object from the application you're building. (Equivalent to selecting an object and pressing **Delete**.)

Property List

Shows the properties of the object that's currently selected in the Objects window or the Design window, and makes them available for editing.

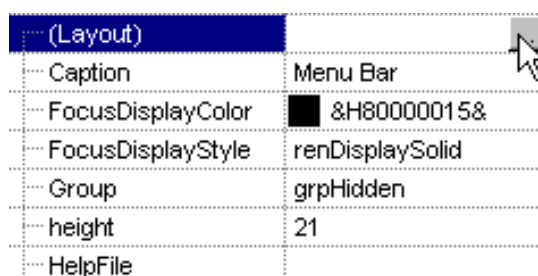


The user interface

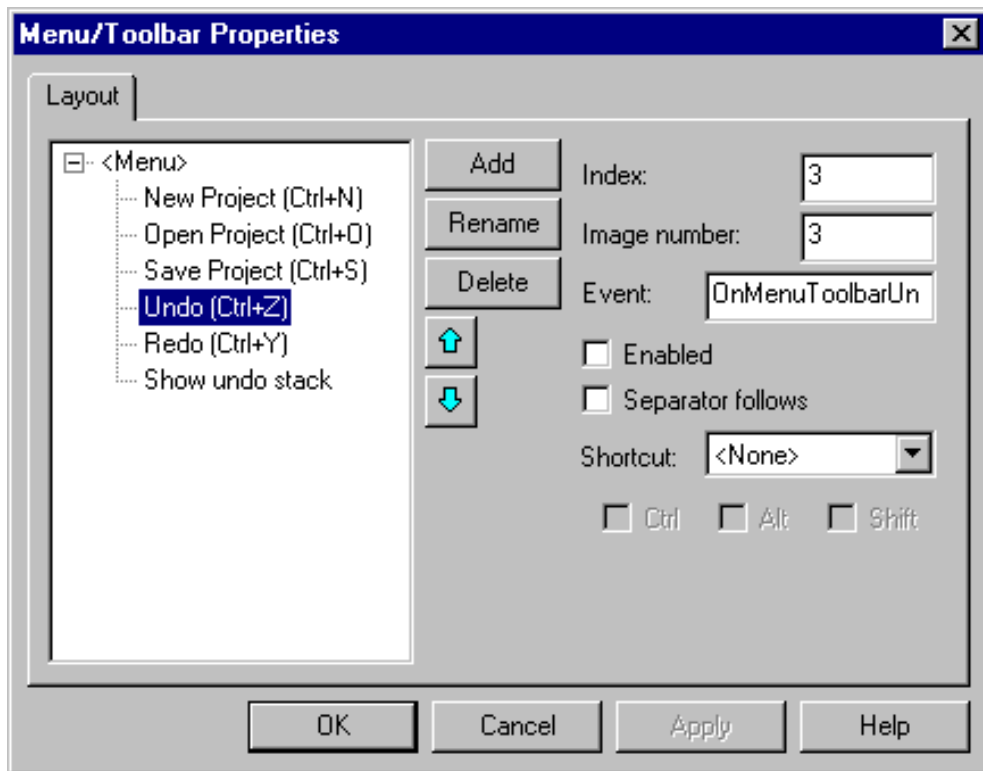
Enterprise Author's Property List is similar to the property windows found in many modern development environments. Its mouse-driven interface allows you to select and modify properties of the current object.

Property List's precise appearance varies according to the object you're editing. The screenshot above lists the properties of a control host object; the list for a menu bar is longer, as it contains (among other things) a property representing all the items in the menu.

Some object types (pages, menu bars, and Enterprise controls, for example) have complex properties that you set using a custom dialog. These are indicated through property names in parentheses at the top of the list:

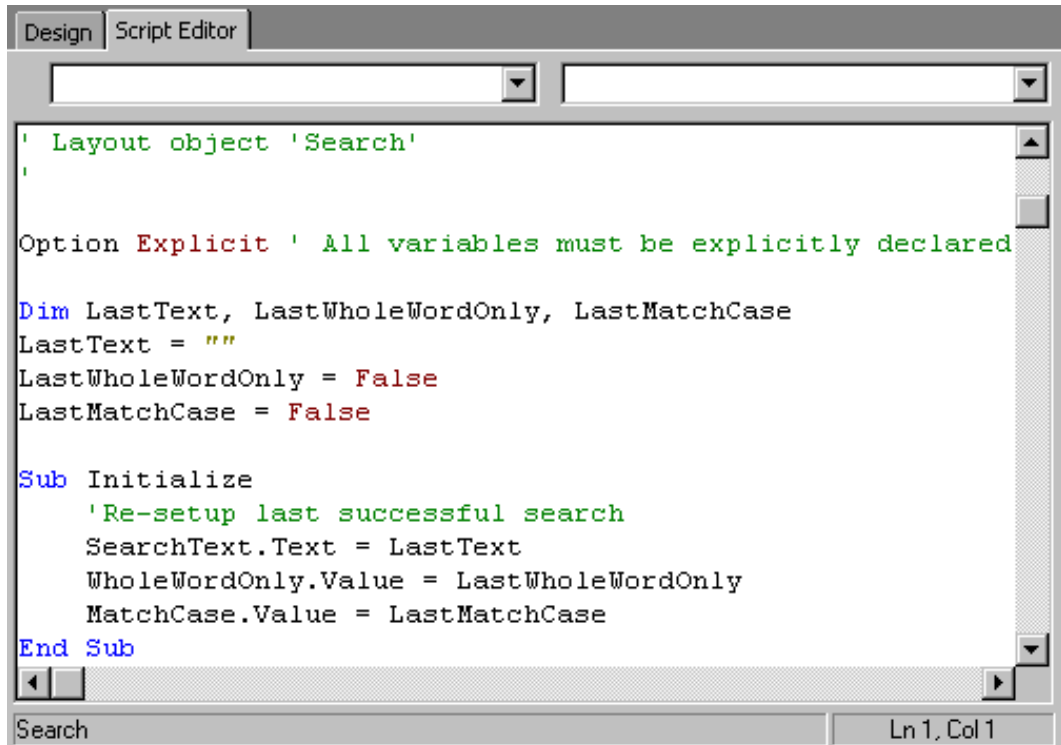


Clicking the button indicated here, which is in the Property List of a menu bar object, produces the following task-specific dialog:



Script Editor

Shows the script module for the currently selected object, and provides editing functionality.



Each object that you create in Enterprise Author has its own script module that can contain *event handlers* for the object and its controls. An [event handler](#) (p.62) is a sequence of script code statements that the application runs when a particular event occurs.

Text highlighting

The Script Editor window provides a code-editing environment that will be familiar if you've used Microsoft Visual Basic or a similar modern programming language. For example, most of what you type into the window is colored black, but some text is highlighted to help with reading and syntax:

- **Blue text** identifies a keyword
- **Maroon text** denotes a built-in value
- **Olive text** marks out string literals
- **Green text** indicates a comment

Code navigation

The two drop-down lists above the edit window in Script Editor provide a quick way of writing and navigating around your code. The one on the left contains a list of all the controls on the current object; the one on the right lets you choose from all the events supported by the control chosen in the left-hand list.

On making a selection from the right-hand list, Script Editor either creates a skeleton VBScript handler for the event you chose, or (if a handler already exists) displays that handler in the edit window.

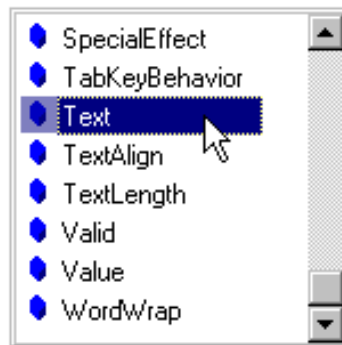
Context menu

Right-clicking anywhere in the edit window displays a context menu that's identical to [Main Menu's](#) (p.26) **Edit** menu.

Pop-up help

The Script Editor window's pop-up help feature makes writing code easier. When you get as far as typing the period in `SearchText.Text`, for example, a pop-up dialog appears containing the names of the entities that may legitimately come next:

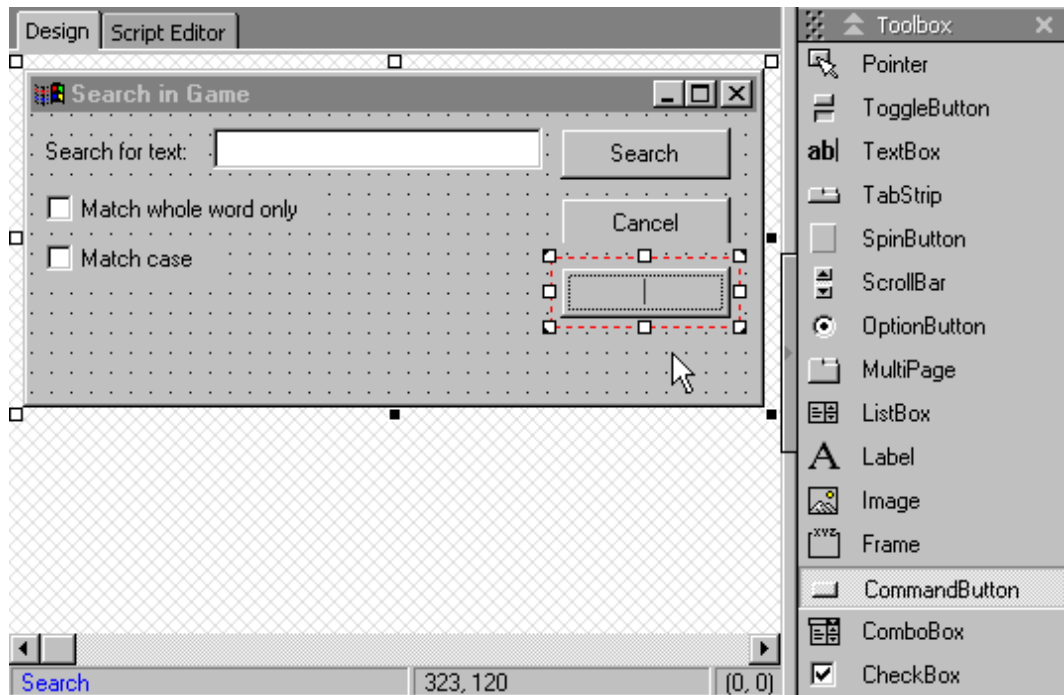
```
Sub Initialize
    'Re-setup last successful search
    SearchText.
```



Double-clicking **Text** then inserts it at the current position in the edit window, and displays a smaller prompt ("**(String)**") indicating the type of the property you just specified.

Toolbox

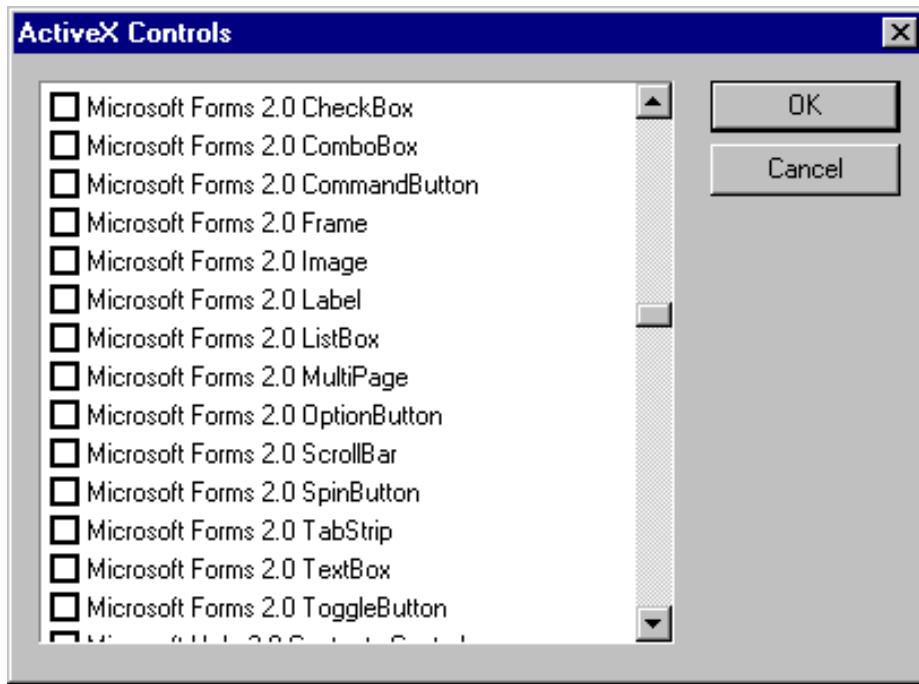
Allows selection of the ActiveX controls that you can draw on a page object to form part of your application's user interface.



You can use the Toolbox window whenever a page (or a RENFrame containing a page) is open in the [Design window](#) (p.22). With a control selected in Toolbox, dragging on a page object draws that control on the page.

Populating the Toolbox window

When you start Enterprise Author for the first time, Toolbox contains only the **Pointer**, which allows you to select the controls on a page. To add controls to the list, right-click in the window and select **Add/Remove Components**:



Select the names of the controls you want to use, and click **OK**. The *Microsoft Forms 2.0* controls have been added to the window in the screenshot at the start of this section, but in general it's possible to add any ActiveX control registered on your computer.

Enterprise Author techniques

The [user interface reference](#) (p.21) explains the *mechanics* of using Enterprise Author, detailing the capabilities of each window in its user interface. By contrast, this part of the documentation explores the *techniques* required to perform common user interface customization tasks with Enterprise Author:

- [Creating and editing windows and dialogs](#) (p.39) that are displayed during or throughout the life of an application
- [Creating and editing menu bars and toolbars](#) (p.52)
- [Assigning objects to functional groups](#) (p.60) in order to affect their behavior
- [Writing script code](#) (p.62) to handle the events that take place as the application runs
- [Debugging VBScript code](#) (p.66)
- [Creating and editing alternative layouts](#) (p.69) that reflect the application's different modes of operation

Creating and editing windows and dialogs

From Enterprise Author's point of view, the windows that appear *in* an application and the dialogs that pop up *during* an application are the same kind of object. The difference lies in whether you decide to place them in the user interface by default, or to display them in response to some script code.

You can choose to create three types of window in Enterprise Author:

- A window whose entire client area is occupied by a single ActiveX control
- A blank page (or “form”), onto which you can draw one or more ActiveX controls
- A “splash window” that displays an image to the user while the application starts up

A window occupied by a single control

1. From the main menu, select **Create ► Objects ► Control Window....**
The ActiveX Controls dialog appears, listing all of the ActiveX controls installed on your system.
2. Select the name of the control you want from the list, and click **OK**.
The new window appears in Design, and a new [control host object](#) (p.30) appears in the list in the Objects window. Its name is the control type, followed by a number (for example, *ListBox1* or *Animation1*).

Tip: As a shortcut for creating a window filled by an Enterprise control, select **Create ► Objects ► Enterprise Control**.

The extended control

When it creates a control host object, Enterprise Author “wraps” the ActiveX control in an *extended control*. This provides the wrapped control with some properties that are required by Enterprise Author-generated applications, but not necessarily provided by ActiveX control writers. (For example, caption and position properties.)

The extended control also deals with two mechanisms that are important from the application developer's point of view:

1. It allows the ActiveX control's events to be handled by methods in an [associated script code file](#) (p.62).
2. It enables *script joining*, in which methods in an associated script code file are treated as though they're methods of the ActiveX control itself.

A blank page to hold several controls

- From the main menu, select **Create ► Objects ► Forms Page**.
The blank form appears in Design, ready for [adding ActiveX controls](#) (p.42). The name of the new object in the Objects window is *RENPageN*, where *N* is an integer reflecting the number of pages in the application.

A splash window that's displayed at startup

- From the main menu, select **Create ► Objects ► Splash Window**.

An empty [splash window object](#) (p.47) appears in Design, and its properties fill the Property List window. The name of the new object is *RENSplashN*.

Choosing a sensible name

The name of an object that appears in the Objects window is the same as the name you'll use when referring to the object in script code, so you should always change the default name to something that's:

- Unique within the application
- Meaningful and memorable

Renaming an object *doesn't* update references in your code. If you rename an object, any code that refers to the old name will no longer work, and you'll need to edit the code manually to fix it.

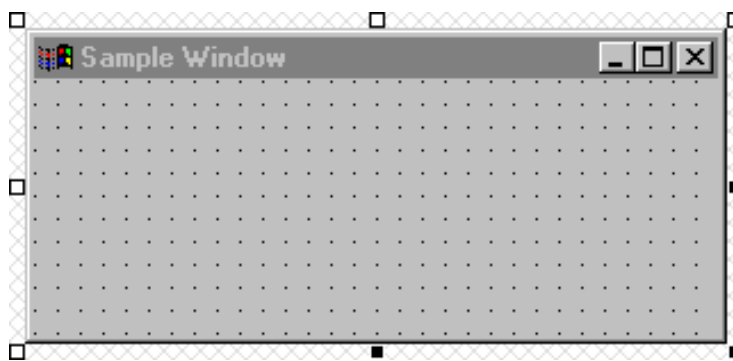
It's prudent, therefore, to change the names of your objects as soon as you create them.

Changing the appearance of a window

Graphical objects in Enterprise Author all support the same features for changing their appearance in your application's user interface. Bear in mind, though, that stackers and tabbers will dictate the size and appearance of the windows they contain. Some of the changes you make will therefore only be apparent when a window is displayed from code while your application is running.

Resizing a window

You can resize windows in Design by using the black-colored handles around their border:



If you display the window from code, it appears at the size it was shown in Enterprise Author. If you resize a window that's filled by a single ActiveX control, the ActiveX control expands or contracts accordingly.

Setting extended properties (“status flags”)

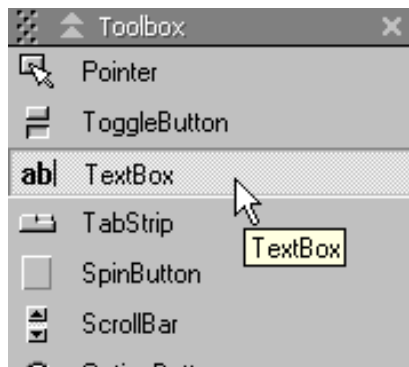
In Microsoft Windows, all windows have a set of “extended” properties that affect the way they appear on screen: whether they have a border, whether they can be resized, whether they should have a caption, etc.

In Enterprise Author, most of these properties are set on an object's behalf as a result of it being contained in the application window. Those that aren't, such as whether a window should appear maximized by default, can be set in the Property List window.

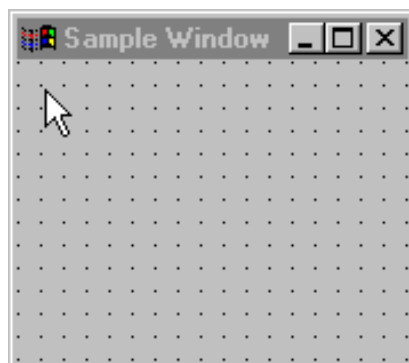
Drawing controls on a page

For page objects, Enterprise Author's Design window offers an editing interface similar to that provided in Microsoft Visual Studio. For example, to draw an ActiveX control from the Toolbox onto a page:

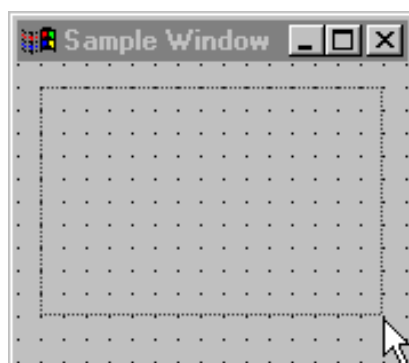
1. Point to the Toolbox and select the control you want to add.



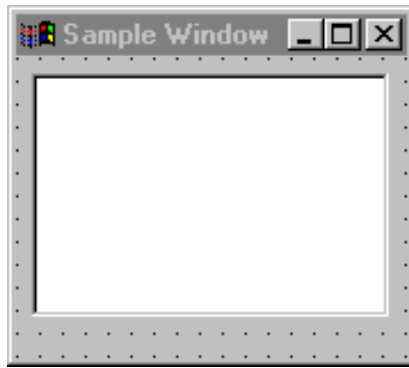
2. Point on the page to where you want the top-left corner of the control.



3. Drag to where you want the bottom-right corner of the control.



4. Release the mouse button. The control appears on the page.



Tip: To add an ActiveX control to the Toolbox (provided that it's already installed on your system), right-click an empty area of the Toolbox, and select **Add/Remove Components**.

Clicking on an added control selects it for editing and displays its properties in the Property List window. (For details on setting the properties of an Enterprise control, see [Adding an Enterprise control](#) (p.45).)

Naming and renaming controls

Adding an ActiveX control to a page is like creating a window occupied by a single control, in that the control you add is given a default name based on its type. Again, this is the name you'll use to refer to the control from script code.

Unlike control host objects, however, the names of ActiveX controls on pages have only to be unique among those on the same page. This means that you don't need to rename controls that you won't refer to from code (such as simple labels, for example). Their names will not clash with others in the application.

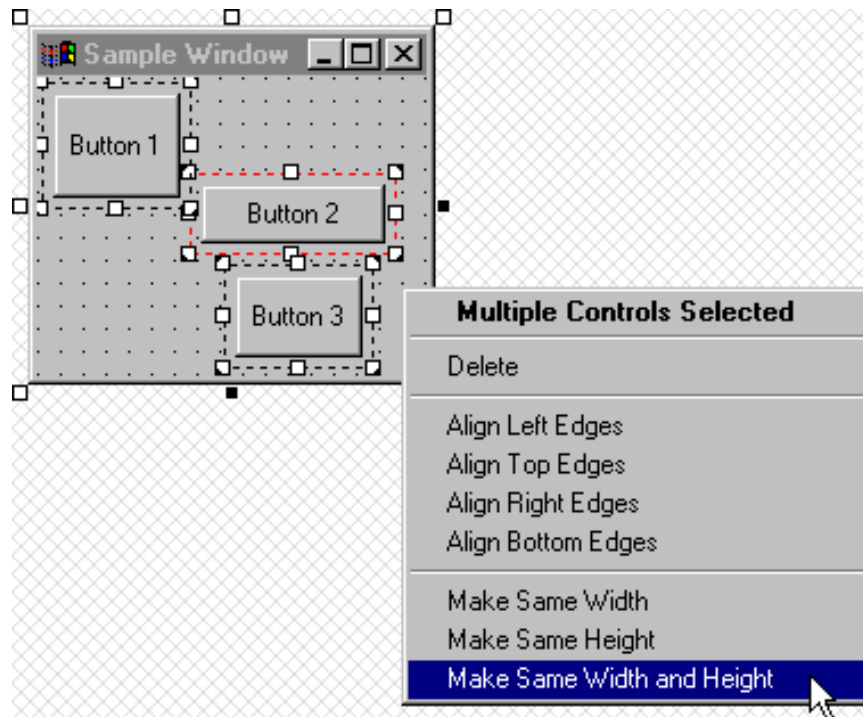
Aligning and resizing controls

To align the edges of several controls on your page, or to make controls the same width or height:

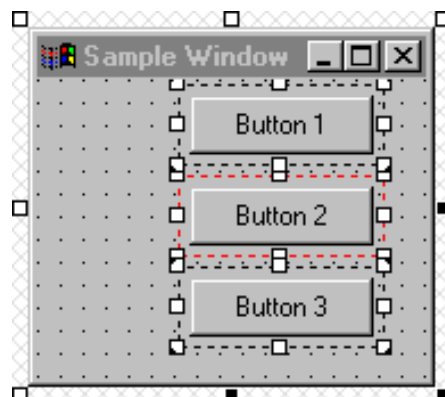
1. Hold down **Ctrl**, and click each of the controls that you want to align or resize.
2. Still holding down **Ctrl**, click the control to which you want to align the others, or which you'll use to set the others' width or height.

The marquee around the last control appears in red, indicating that the other selected controls will be aligned or resized according to this control.

3. Release **Ctrl**.
4. Right-click the page, or any of the selected controls. A context menu appears:



5. Click the alignment or sizing option that you want. Enterprise Author aligns or resizes the controls according to the last control you selected.



Adding an Enterprise control

An Enterprise control object is really just a special case of a control host object, and it will usually appear in your application in a window of its own. However, it's quite possible to place it on a page in the same way as any other ActiveX control. You just need to add the Enterprise control to your Toolbox—it's called **CSL RealIMation Enterprise Control** in the ActiveX Controls dialog.

Adding modules to an Enterprise control

An Enterprise control is a container for Enterprise modules. With no modules, an Enterprise control does nothing at all, but you're free to add one or more from the set available, like this:

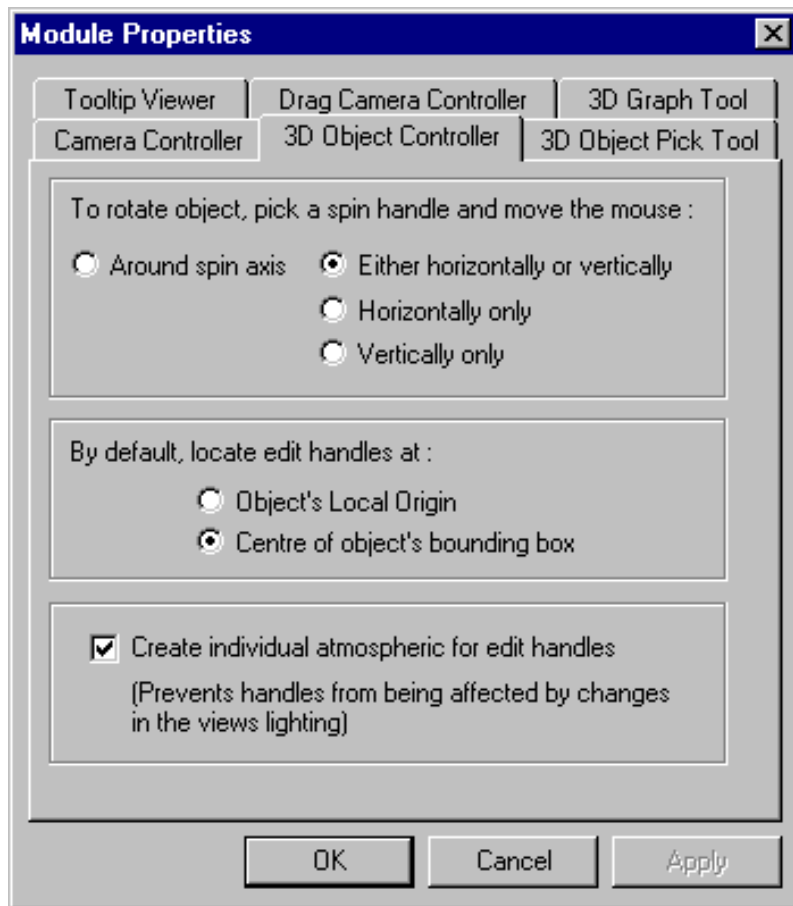
1. Open an Enterprise control for editing in the Design window.
2. Drag modules from the Enterprise Modules window into the open control.
For example:



Setting module properties

After adding a module to an Enterprise control, you need to set the properties of that module:

1. Select the the Enterprise control, and then go to Property List and click the ... button that appears opposite (**Enterprise Module Control**).
2. In the dialog that's displayed, click **Module Properties** to display the Module Properties dialog, which contains a tab for each module currently loaded in the Enterprise control.



3. Click the tab for the module whose properties you want to set.

For details on the properties of a module, see the individual topic for each module in the [Enterprise module reference](#) (p.71).

Removing modules

To remove a module from an Enterprise control:

1. Select the the Enterprise control, and then go to Property List and click the ... button that appears opposite (**Enterprise Module Control**).
2. In the **Loaded Modules** list, select one or more modules that you want to remove.
3. Click <<< **Unload Modules**.

Tip: Instead of adding new modules to an Enterprise control by dragging them, you can select modules in the **Available Modules** list, and then click **Load Modules >>>**.

Configuring a splash window

A splash window is another specialized control host object: it's an ActiveX control that's tailored to the purpose of showing an image to the user while an Enterprise Author-generated application initializes. Any splash window objects in a project are automatically displayed at startup, without the need for any additional coding. You just have to specify what information they contain.

Among the properties of a splash window object (most of which are standard ActiveX control properties) are five particularly important ones:

- *Picture* is a bitmap property that contains the image to be shown.
- *VersionStringRegKey* points to a location in the system registry where version information for the application can be found.
- *VersionSubText* stores a string that can also be displayed in the splash window.
- *Font* specifies the font in which any text appearing in the splash window should be displayed.
- *ForeColor* sets the color of any text in the splash window.

Choosing the image to be shown

There are no restrictions on the image you choose to employ in a splash window. To add a picture to a splash window object:

1. Open the splash window for editing in Design, and then click the ... button that appears opposite **Picture** when you select the latter in the Property List window.
2. In the Load Picture dialog that appears, browse to the image you want to use, and open it. The image will appear in the Design window.

Enterprise Author does not resize images, so make sure that you're happy with yours before you load it. A typical width for a splash window is 480 pixels.

Displaying version information

Splash windows in Enterprise Author-generated applications have the facility to display dynamic version information, as well as a fixed string that will prefix any such data on screen. The properties that enable this functionality are *VersionStringRegKey* and *VersionSubText*.

1. Set *VersionStringRegKey* to the path of a REG_SZ-type key in the system registry. When the application starts up, the contents of this key will be shown in the bottom-right corner of the splash window.
2. Set *VersionSubText* to a string that can sensibly precede the version number information. For example, if the version information is of the form "2.0 beta", then *VersionSubText* may simply be "Version: ".
3. Set the *Font* and *ForeColor* properties appropriately, so that the text is visible in front of the image you've chosen for the splash window.

Note: Like [toolbar images](#) (p.52), splash window images are stored in the project files, so you can't change an image just by saving a different picture with the same file name. You *must* reload the image into Enterprise Author.

Placing windows inside the application window

Applications that you create or edit with Enterprise Author always have one (and only one) RENFrame object. To make your objects appear as windows in the application window at run time, you need to drag them into the RENFrame object at design time.

1. Double-click the RENFrame object in the Objects window to display it in the Design window.
2. Drag any of your graphical objects from the Objects window to the Design window, and drop them into the RENFrame.

Depending on the type of object being dropped and the position of the cursor, Enterprise Author will offer to place it a variety of locations.

Placing control hosts, Enterprise controls, and page objects

An object of one of these types could “adhere” to one of the sides of the frame, like a toolbar would:



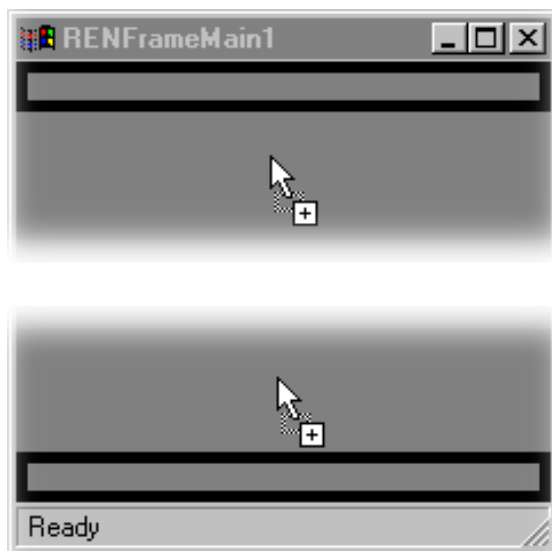
Or it could appear as an element in a stacker that occupies half of the available area:



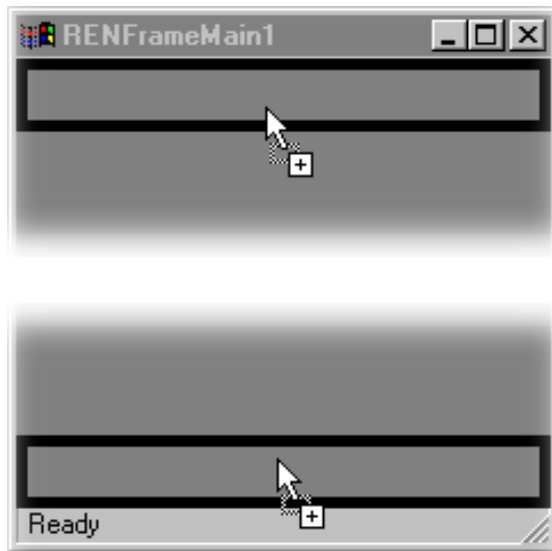
Note: By default, each time you create a new stacker in the RENFrame, that stacker will occupy half of the space that was previously the work area.

Placing menu bar objects

At the sides of the application window, menu bar objects behave in exactly the same way as the other object types. At the top and bottom, however, things are a little different. A narrow rectangle means that a minimised stacker element will be created:



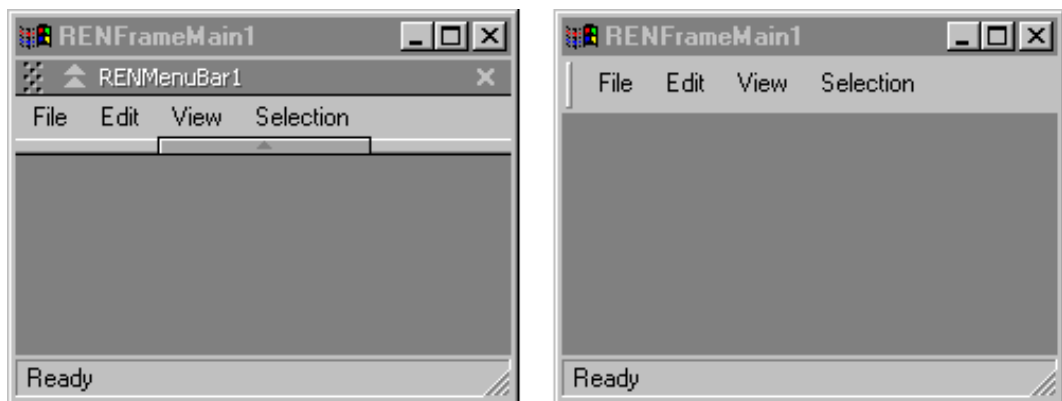
While a broader rectangle means that you'll get a more traditional menu bar or toolbar:



Note: Look closely at the position of the cursor here. To get “toolbar” behavior, you drop closer to the edge of the window, even though the cue rectangle is thicker.

Window-like behavior and toolbar-like behavior

Giving your objects window- or toolbar-like behavior in Enterprise Author is an important decision, because you can't switch between these behaviors in a running application. Toolbar-like objects can be moved to the top, left, right or bottom of the application window, but you can't add them to stackers or tabbers, and you can't make them float. You can see the difference here:



Note: Users can choose to make the menu bar on the left float, but the one on the right will always be anchored to the application window. Although we've illustrated this idea with toolbars here, these behavior options are available for all graphical objects in Enterprise Author.

When you've started adding objects to the interface of your application, you can customize its appearance in the same way as you can customize Enterprise Author itself. You can group your objects in stackers and tabbers, for example, and switch them in and out of the work area.

Creating and editing menu bars and toolbars

Menu bars and toolbars both allow you to present the actions that users can perform in your application.

Menu bar



Toolbar



Menu bars and toolbars have the following differences:

In a menu bar...	In a toolbar...
All items are displayed as text.	Top-level items are displayed as graphical buttons.
Clicking a top-level item (for example, File) typically displays a drop-down list of sub-items, and performs no other action.	Clicking a toolbar button typically performs an immediate action. Toolbar buttons often have a one-to-one mapping with menu sub-items.

In applications created with Enterprise Author, menu bars and toolbars are conceptually equivalent. They have the same object type (menu bar), and can be placed along any edge of the application window. The differences lie in the detail.

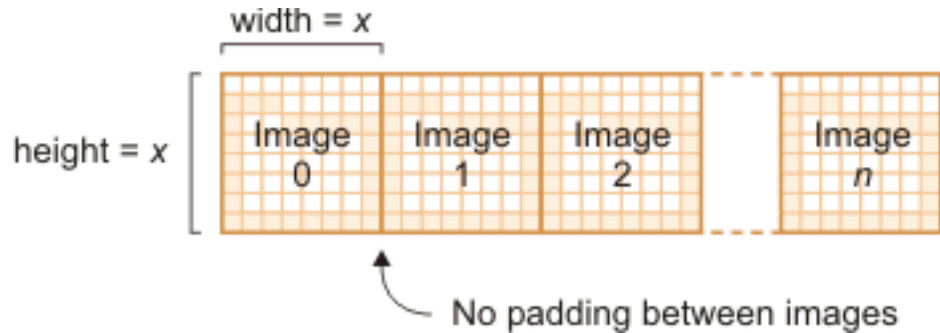
Creating a menu bar or a toolbar

The procedures for creating menu bars and toolbars in Enterprise Author are very similar, so we can treat them together and mention any differences as we go.

1. Select **Create ► Objects ► Menu/Toolbar**. This creates a new object named *RENMenuBarN* whose client area is occupied by a menu bar or a toolbar, depending on its *ViewType* property.

The default setting for *ViewType* is *rmbViewTypeMenu*, reflecting its initial menu-like appearance. The other possible setting is *rmbViewTypeToolbar*.

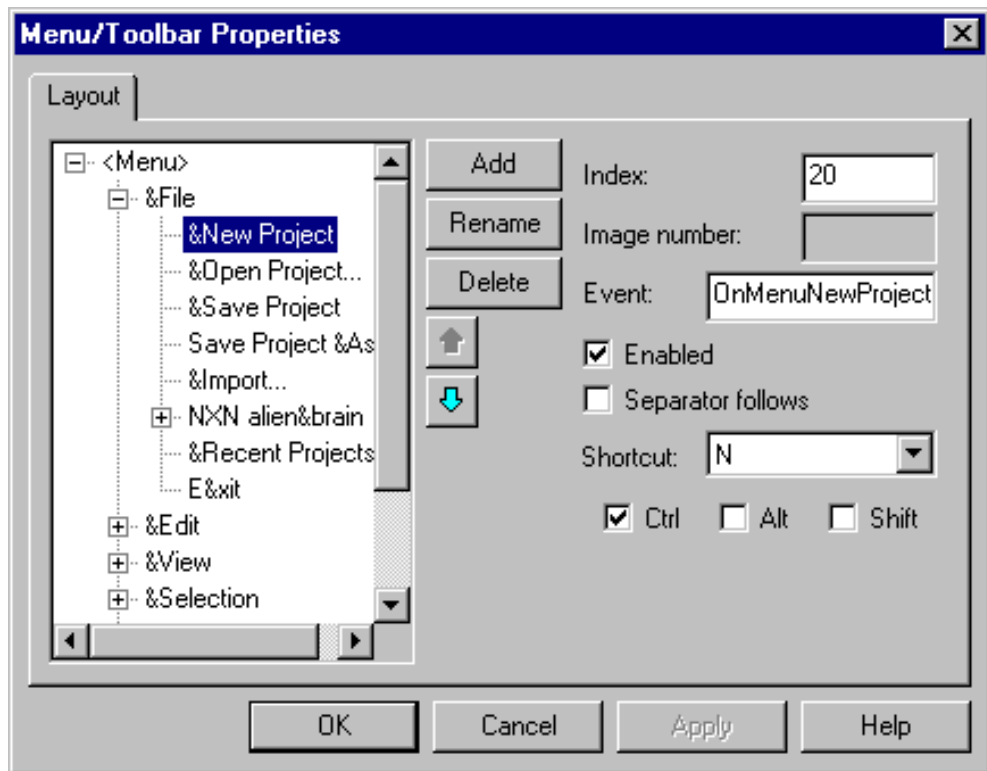
2. In the Property List window, set **ViewType** to *...Menu* or *...Toolbar* accordingly.
3. In the Objects window, rename *RENMenuBarN* to, say, *CustomMenu* or *CustomToolbar*.
4. If you're creating a toolbar, you need to import a bitmap file:
 - In the **Toolbar** box of the Property List window, click ... to select a Windows bitmap file (.bmp) containing a horizontal row of button images:



For example:



- The toolbar button images can be any size, but they must be square. Typical dimensions for toolbar buttons are 16 x 16 pixels.
 - To create the toolbar bitmap file, use a bitmap editor (such as Jasc® Paint Shop Pro® or Adobe® Photoshop®).
 - Only the top-level items in a toolbar can be represented by these graphical buttons; any subordinate items appear as text in a drop-down list, similar to a menu.
 - The bitmap file can contain more button images than are used by your application. Only the image numbers that you refer to (as described below) are used.
 - Enterprise Author stores the bitmap data in the project files. If you change the bitmap, you *must* re-import the .bmp file into Enterprise Author.
5. Click the ... button next to **(Layout)** in the Property List window. The Menu/Toolbar Properties dialog appears:



6. For each menu or toolbar item that you want to create:
 - a. Select the parent of the item you want to create (to create a top-level item, click **<Menu>**).
 - b. Click **Add**.
 - c. Type the item name, and then press **Enter**.

Tip: To create a menu separator line, select the **Separator follows** check box of the item before the line.
 - d. If you want the item to be shown as disabled ("grayed out") by default, then clear the **Enabled** check box.
 - e. If you want to associate a shortcut key with the menu or toolbar item, select the key in the **Shortcut** list, and choose which (if any) of the three modifiers (**Ctrl**, **Alt**, **Shift**) the key should be used with.

Note: Applications created by Enterprise Author all deal with key presses in the same way. The window in the application that currently has focus is offered the chance to deal with it first. If no appropriate handlers are found, the key press is passed on to the set of currently-visible menu bars and toolbars. The order of presentation within this set is undefined, so it's important not to try to handle the same key press in several places.
 - f. In the **Event** box, type a name for the event that this item will generate.

This is the name that you use in the script module (described below) to tell your application what to do when the item is selected. Typically, the event name of an item should be unique, unless you want to make the same action available from different items in the same menu or toolbar.
 - g. If you're creating a top-level toolbar item:

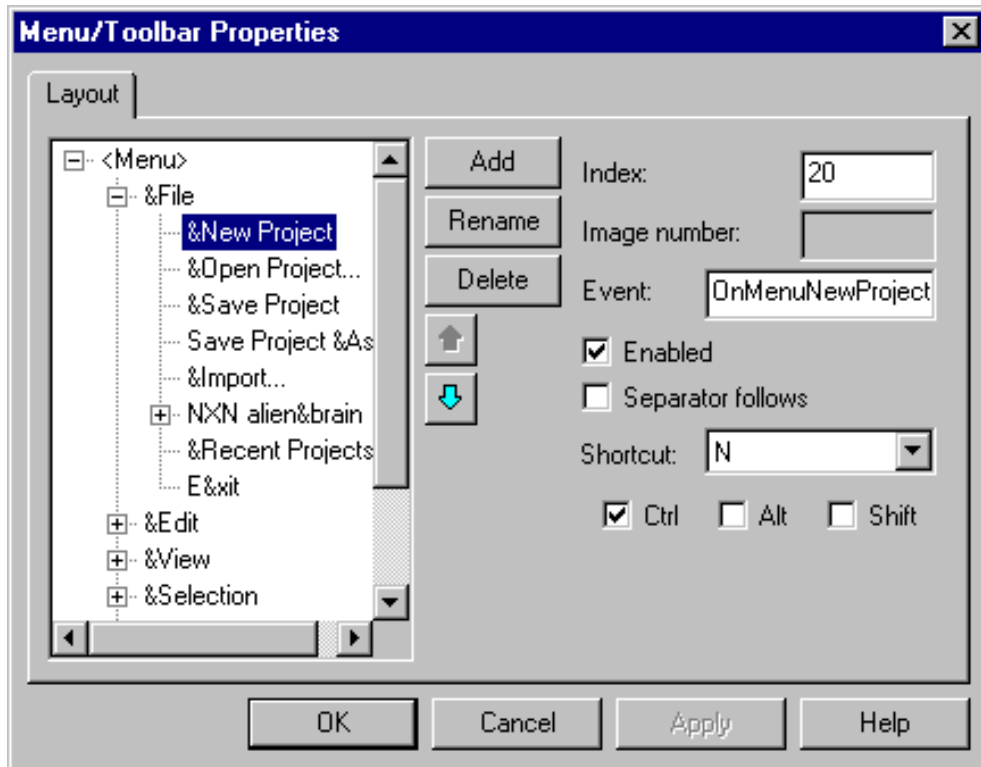
- In the **Image number** box, enter the number of the image that you want from the toolbar bitmap (counting from the left, starting from zero).
7. When you've finished adding items, click **OK** to close the dialog.
 8. In the script module for the menu bar object, create event handlers for the different items, like this:

```
Sub OnMenuNewProject()  
    ' Do this when File -> New selected  
End Sub  
  
Sub OnMenuOpenProject()  
    ' Do this when File -> Open selected  
End Sub
```

9. With your RENFrame object open for editing, drag your new toolbar or menu bar to your choice of location in the [application window](#) (p.49).

Index numbers of menu bar items

The **Index** setting at the top right of the Menu/Toolbar Properties dialog provides a way of uniquely identifying a menu bar item.



Enterprise Author assigns each item you create an index value that's one higher than any other index value at the time of creation. In this way, every item in a given menu bar object is guaranteed a different value. However, you're free to set your own index values if you wish, and there can be good reasons for doing so:

- To give related items index values in a narrower range, for organizational purposes
- To specify one of Enterprise Author's [reserved index values](#) (p.58), which activate some built-in functionality

Manipulating menu bar items from code

Several of the methods exposed by menu bar objects have an index value parameter that identifies the item to be operated upon. For example, the methods for enabling (that is, "graying" and "un-graying"), checking, and deleting menu bar items at run time are all called with index value arguments.

Calls like these are typical in an Enterprise Author-generated application's VBScript code:

```
MainMenu.EnableMenuItem 329, True
MainMenu.EnableMenuItem 348, False

SelectionToolbar.CheckMenuItem 14, False
```



```
For Each SubMenu In TexMenu.MenuItems
    VolumeToolbar.DeleteItem SubMenu.Index
Next
```

Legacy event-handling code

Index values also provide an alternative way of dealing with menu bar item selection that was standard practice in older versions of Enterprise Author. Rather than using the `OnXXXX()` syntax, you can use a `Click` event handler to do the work of reacting to selections:

```
Sub MainMenu_Click(index)
    Select Case index
        Case 20
            ' Do this when File -> New is selected
        Case 21
            ' Do this when File -> Open is selected
        .
        .
        .
    End Select
End Sub
```

Warning

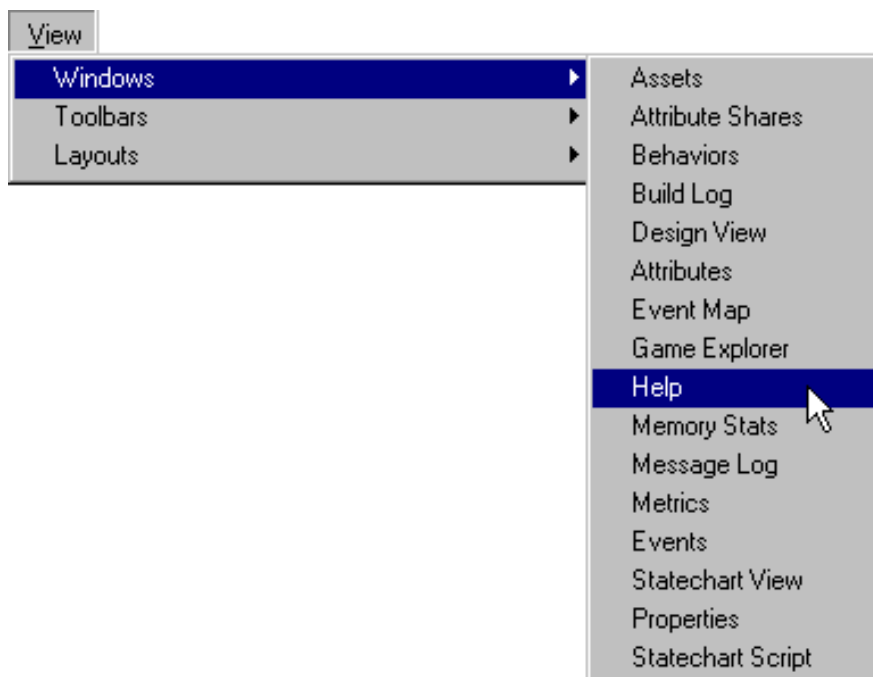
This technique is deprecated in new applications developed using Enterprise Author and is retained only for compatibility with old code. For all new menu bar objects, you should use the [event-based selection handling technique](#) (p.52).

Showing and hiding windows and menu bars

A common requirement in a Windows application is the ability to show and hide windows and menu bars on request. Enterprise Author supports these features without the need for you to write the code that deals with them.

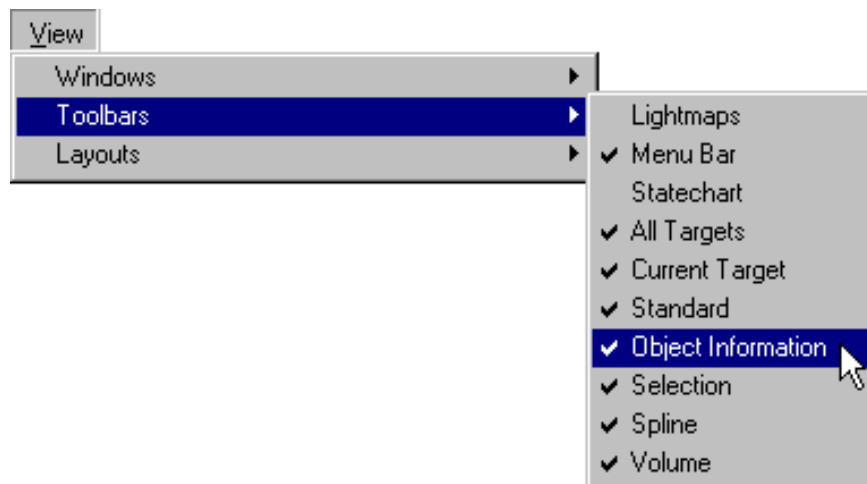
Enterprise Author reserves the index values **11000** and **11001** for the purpose of creating “view” menu items:

- If you give a menu bar or toolbar item an index value of 11000, that item will automatically contain sub-items for every object in the application whose *Group* property is set to *grpNone* (the default setting for control host objects).



In the running application, selecting one of these sub-items causes the object it represents to appear as a floating window (if it wasn't already present in the user interface) and/or to gain the focus.

- If you give a menu bar or toolbar item an index value of 11001, that item will automatically contain sub-items for every object in the application whose *Group* property is set to *grpToolbars* (the default setting for menu bar objects).



When a menu bar or toolbar is visible in the running application, its corresponding sub-item appears with a check mark . Selecting one of these sub-items toggles the display of the object it represents.

Assigning objects to functional groups

All objects in Enterprise Author-generated applications have a property named *Group* that can be set to one of five predefined values. The *Group* property is used internally by Enterprise Author to enable some kinds of built-in functionality. It can also be used from your custom script code, should you want to perform an operation on members of a particular group.

Default object group settings

All objects are assigned a default value for their *Group* property when they are created:

<i>Group</i> value	Object types	Purpose
<i>grpNone</i>	Control hosts, Enterprise controls, RENFrames, Script modules	The default setting for most objects. With the exceptions of RENFrames and script modules, any objects with this <i>Group</i> value appear in Enterprise Author's automatically generated menu for showing and hiding windows (p.58).
<i>grpDialogs</i>	Pages	The notional opposite of <i>grpNone</i> , objects with this setting are still “window” types, but do not appear in the menu mentioned above.
<i>grpToolbars</i>	Menu bars	Any object with this <i>Group</i> value appears in Enterprise Author's automatically generated menu for showing and hiding menu bars and toolbars (p.58).
<i>grpHidden</i>		Typically used for menu bar objects that you don't want to appear in the automatically-generated menu (such as an object that represents a context menu).
<i>grpSplashScreens</i>	Splash windows	Objects with this group setting behave as splash windows (p.47)—that is, they are displayed at application startup, but never again.

Changing objects' groups

As the table suggests, the default setting for the *Group* property is right for the majority of objects, most of the time. The alterations you're most likely to make are:

- Changing a control host from *grpNone* to *grpDialogs*, when you don't want its name to appear in a menu.
- Changing a page from *grpDialogs* to *grpNone*, for precisely the opposite reason.
- Changing a menu bar from *grpToolbars* to *grpHidden*, to prevent its name from appearing in a menu.
- Changing a control host or a page from *grpNone* or *grpDialogs* to *grpToolbars*, especially when you've given one of these objects [toolbar-like behavior](#) (p.49).

Writing VBScript code

Every object in an Enterprise Author-generated application can have a script module to which you can add script code. The most common use for such code is to handle the events that occur as a result of user interaction with your application, but you can also write code to run in response to events such as the application itself starting and ending.

VBScript event handlers

When an event occurs in a running application, the application looks for a procedure to handle the event. In VBScript code, the application looks in the relevant object's script module for a procedure whose name comprises:

- the name of the object or control that fired the event, followed by
- an underscore (_), followed by
- the name of the event.

For example, if you run your application and click a control named *CommandButton1*, the application calls a procedure called `CommandButton1_Click()`.

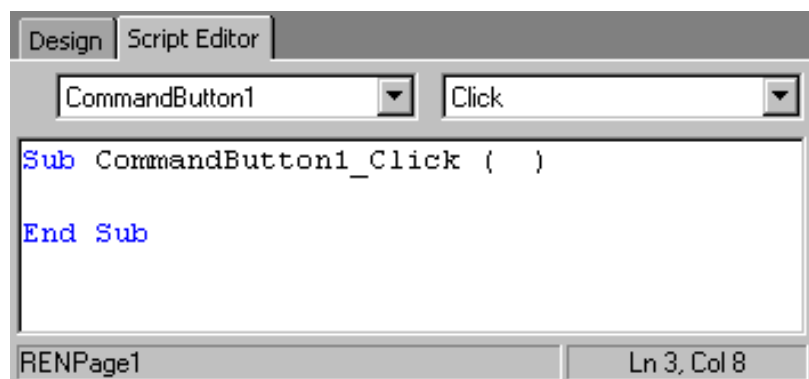
To write an event handler:

1. In the left-hand drop-down list at the top of the Script Editor window, select the name of the object (or one of its controls).

As in the Object Browser window, for anything but a page object, only the name of the object currently being edited appears in this list. For a page object, you see the name of the page, plus the names of all the ActiveX controls it's hosting.

2. In the right-hand list, select the name of an event.

The Script Editor inserts `Sub` and `End Sub` statements for the event procedure.



3. Between the `Sub` and `End Sub` statements, enter VBScript code to handle the event.

Enterprise module event handler names

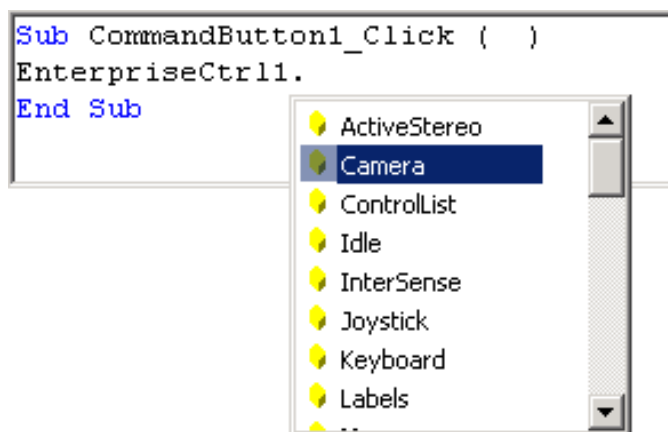
Procedure names for Enterprise module events are similar to other event handlers, except that the module name appears between the underscore and the event name (for example, `EnterpriseCtrl1_Pick3DPicked`). This is because, for scripting purposes, events fired by Enterprise modules are treated as belonging to the parent Enterprise control.

Automatically completing property, method, or object names

To complete the name of a property, method, or child object automatically:

1. Type the name of an object (for example, `EnterpriseCtrl1`).
2. Type a period (`.`).

A list of valid items appears below the period:



3. Highlight the item you want by using the cursor keys, or by typing the first few letters of the item.
4. Press **Enter** to insert the item.

Tip: To close the list without inserting an item, press **Esc**.

Text editing controls

The Script Editor window supports all the basic functionality you'd expect of a text editor:

- To copy selected text to the clipboard, select **Edit ► Copy**, or press **Ctrl+C**.
- To paste the contents of the clipboard at the current cursor position, select **Edit ► Paste**, or press **Ctrl+V**.
- To find text in the current script module (from the current cursor position onwards), select **Edit ► Find...** and then enter a search string.
- To scroll to a particular line in the Script Editor window, select **Edit ► Goto...** and then enter a line number.

Variable scope and lifetime

Every script module—even a global script module (see below)—is classed as a separate script *site*. As a result, the scope of public constants and variables is

limited to the script module in which they are declared.

Referring to objects inside another script module

In Enterprise Author, you cannot define “global” objects that are shared between script modules, but you *can* refer to objects (constants, variables, functions, or procedures) that are defined inside another script module. To do this, you prefix the name of the object with the name of its parent module, followed by a period.

For example, if the script module for *RENPage1* contained a procedure named `HelloWorld()`, then you could use the following code to call this procedure from another script module:

```
RENPage1>HelloWorld
```

Storing common procedures in a global script module

If you have procedures that are used by several script modules, then rather than storing them in the script module of a particular object, you can store them in a global script module. To create one, just select **Create ► Script Module**.

Calling a procedure in a global script module is exactly the same as calling a procedure in the script module of any other object. However, a global script module does not display a window; it is intended *only* as a repository for common script code.

For example, if a global script module called *RENGlobalScriptModule1* contained a procedure named `CommonTask()`, then you could use the following code to call this procedure from another script module:

```
RENGlobalScriptModule1.CommonTask
```

Note: Any script code outside a procedure definition (`Sub... End Sub` or `Function... End Function`) in *any* script module is executed immediately the application runs. Furthermore, any variables declared outside a procedure definition have “script-level” scope: they are visible from every procedure in the script module.

Automatic script module creation

An alternative way of creating a global script module is to place a script file (*ModuleName.vbs* or *ModuleName.js*) in the folder that contains your project's definition files.

Next time you open the project in Enterprise Author, the Objects window will contain a new script module object called *ModuleName*.

Ensuring objects are loaded before you refer to them

When an Enterprise Author-generated application starts, the script modules are loaded, and any script outside of a procedure definition is executed. When all script modules have been loaded, the application fires an `OnLoad()` event to each script module.

If you want to refer to objects in another script module when your application starts, then place those references inside an `OnLoad()` event handler. For example:

```
Sub OnLoad()
```



```
...  
    RENPage1.HelloWorld  
...  
End Sub
```

If you placed the call to `HelloWorld()` outside a procedure definition, then the call would be made as soon as the script module had loaded—which could be before *RENPage1* itself had loaded.

Debugging VBScript code

When you start an application from its **Run ► Start** menu, Enterprise Author attempts to trap script errors as they occur. When it comes across one, Enterprise Author stops the application, and then displays an error message in its own status bar, quoting the script line number.

To scroll to the line number in the Script Editor, select **Edit ► Goto...**, and then enter the line number.

To trap script errors when they occur in your application, use the VBScript `On Error` statement.

Tip: As a quick way to debug your application, temporarily insert `MsgBox` statements at key locations in your script.

Invoking and using an external debugger

To get more control over the debugging process, you can link an external script debugging application (such as the [Microsoft Script Debugger](#) (p.7), or Microsoft Visual Studio .NET 2003) to Enterprise Author or any application it's been used to create.

The easiest way to invoke either of these debuggers is to insert a `Stop` statement in your script, which acts like a breakpoint. When a running script reaches the `Stop` statement, the script halts execution and transfers control to the debugger.

Tip: If any “error” dialogs pop up as a result of a `Stop` statement, select the options that allow you to “debug”, rather than “close”, the application.

Attaching a debugger to a running application

Another way of debugging an Enterprise Author-generated application in a dedicated debugging environment, which does away with the need to insert calls to `Stop`, is to *attach* the debugger to your application while it's running. The procedure is similar, but not identical, for both the Script Debugger and Visual Studio .NET 2003:

1. Start up the application that contains the script you want to debug.
2. Attach your chosen debugger to the application you just started.

For the Microsoft Script Debugger:

- Start Microsoft Script Debugger, then click **View ► Running Documents**.

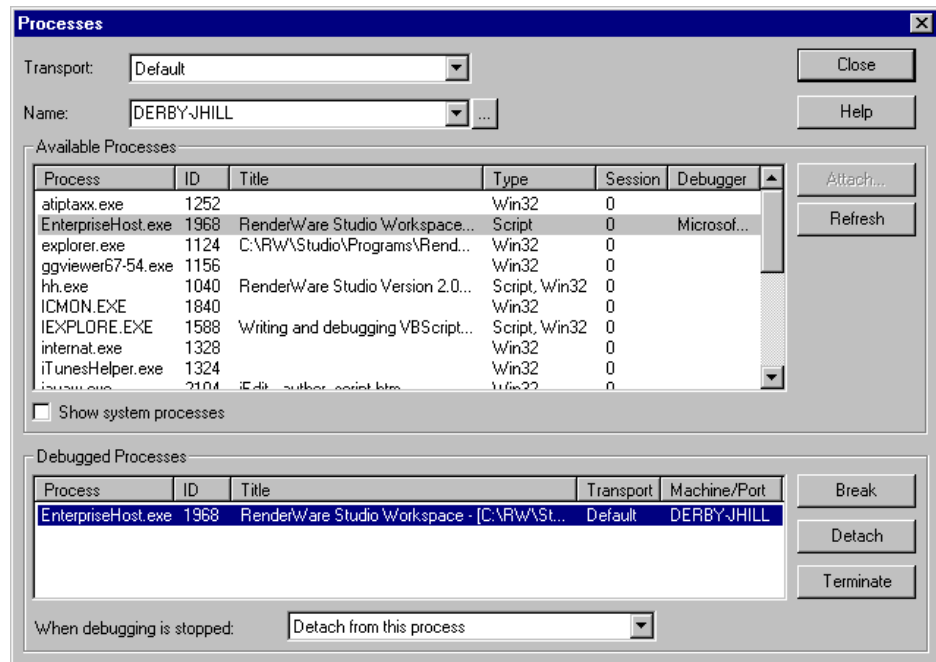
The Running Documents window appears, containing a hierarchical tree listing the script sites available in your application.

For Microsoft Visual Studio .NET 2003:

- Start Microsoft Visual Studio .NET 2003, and select **Debug ► Processes....**

The Processes window appears, containing a list of the processes to which you may attach the debugger.

- Select the process called `EnterpriseHost.exe`, and click **Attach....**
- Click **OK** in the next dialog to confirm that you're only interested in script code, and you'll be left with something like this:



In addition, the Running Documents window in the Visual Studio user interface now contains a hierarchical tree listing the script sites available in the application.

In both debugging applications, each script site corresponds to an object in the user interface, or a script module object.

3. To set a breakpoint in a script, click on the corresponding script site in the hierarchy tree.

Tip: To avoid crashing the debugger, remember to exit the debugger *before* exiting the application you are debugging.

Querying and setting VBScript variables

While the application is stalled at a breakpoint, you can query and set the value of variables in the VBScript code:

1. Display the command window:

For the Microsoft Script Debugger:

- Select **View ► Command Window** (or press **Ctrl+G**).

For Microsoft Visual Studio .NET 2003:

- Select **View ► Other Windows ► Command Window** (or press **Ctrl+Alt+A**).

2. To query the value of a variable, type the following in the command window:

`? variable`

and then press **Enter**.

The command window displays the value of the variable.

3. To set the value of a variable, type the following in the command window:

variable = 2

or:

variable = *variable* + 1

and then press **Enter**.

Creating and editing alternative layouts

If you want to display some pages only in certain situations, or you want to arrange windows inside different containers (stackers or tabbers) at different times, then you can create alternative layouts and switch between them while the application is running.

When you create a new application in Enterprise Author, a layout called *Default Layout* is created on your behalf. By default, it's this layout that you're editing when you add objects to the RENFrame object. To change it, you need to do the following:

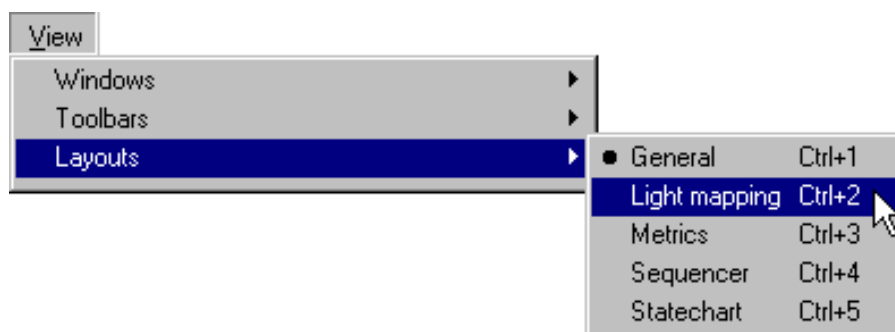
1. Select the **Layouts ► Create** menu item, and provide a name for your new layout.
2. Go to the **Select** menu and switch from Default Layout to the layout you just created. A newly empty version of your application window will appear in Design.
3. Drag objects into and around the application window, just as you would when editing *Default Layout* (or any other layout provided by the application you're customizing).
4. Optionally, make your new layout become the default layout by setting the *DefaultDockState* property of the RENFrame object to the name of your new layout.

Simply calling a layout *Default Layout* doesn't make it the default, and it's not compulsory to have a layout with that name in your application. You can use **Layouts ► Rename** to rename the currently selected layout whenever you wish.

Switching between layouts

In a running Enterprise Author-generated application, users need to be able to switch between the layouts you've created. Enterprise Author provides a mechanism for enabling this activity that's similar to the way it deals with [showing and hiding windows and menu bars](#) (p.58). It rests on another reserved index value for menu and toolbar items.

If you give a menu bar or toolbar item an index value of **10000**, that item will automatically contain sub-items for every layout defined in the application:



Here, the **Layouts** item has the index value 10000. When the application is

running, a bullet point indicates the current layout. Selecting a different layout immediately changes the appearance of the application.

Enterprise module reference

In an Enterprise Author-generated application, an Enterprise control object acts as the host for any number of *Enterprise modules*: tools that provide ways of displaying or navigating around the 3D data that the application processes. (For RenderWare Studio Workspace, “data” means the graphical assets and entities that comprise a game.)

For a detailed description of a particular Enterprise module, click its icon in one of the tables below.

Getting help for methods, properties, and events

This help file does not describe the complete Automation interface (the methods, properties, and events) for each module. Currently, it describes only the *design-time* properties for each module (the properties that you can set before running the application).

For complete information about the Automation interface of each module, use the [Object Browser](#) (p.29).

Note: Workspace uses only a few of the several dozen available Enterprise modules. For information about other Enterprise modules (not supplied with RenderWare Studio), contact [Realimation](http://www.realimation.com/) (www.realimation.com/).

Standard modules

Module	Description
3D Object Controller	Exposes an automation interface for controlling objects in a Realibase.
3D Object Pick Tool	Exposes an automation interface for determining when the user clicks ("picks") an object in a view, and the ID of the picked object.
3D Viewer	Displays a perspective projection of a view.
Camera Controller	Allows you to interactively control a camera using a variety of input devices (by default, a mouse).
Keyboard Interface	Exposes an automation interface for determining which key has been pressed, and defines actions for some keys.
Orthographic Viewer	Displays four projections of a view: one perspective and three orthographic (front, plan and side).
Realibase File Processor	Loads a Realibase, and exposes an automation interface that allows you to save the Realibase.
Tooltip Viewer	Displays a tooltip when the cursor hovers over an object in the view.

3D Object Controller

Exposes an automation interface for controlling objects in a RealiBase.

For information about this interface:

1. Add this module to an Enterprise control in your project.
2. Use the Object Browser to view the automation interface.

Note: This module has no properties that you can set in design-time. You can only access the properties of this module in run-time, via script.

3D Object Pick Tool

Exposes an automation interface for determining when the user clicks ("picks") an object in a view, and the ID of the picked object.

The ID of an object is defined in the RealiBase. You can use this ID as a key to retrieve information about the object from a separate database (such as a Microsoft® Access® database).

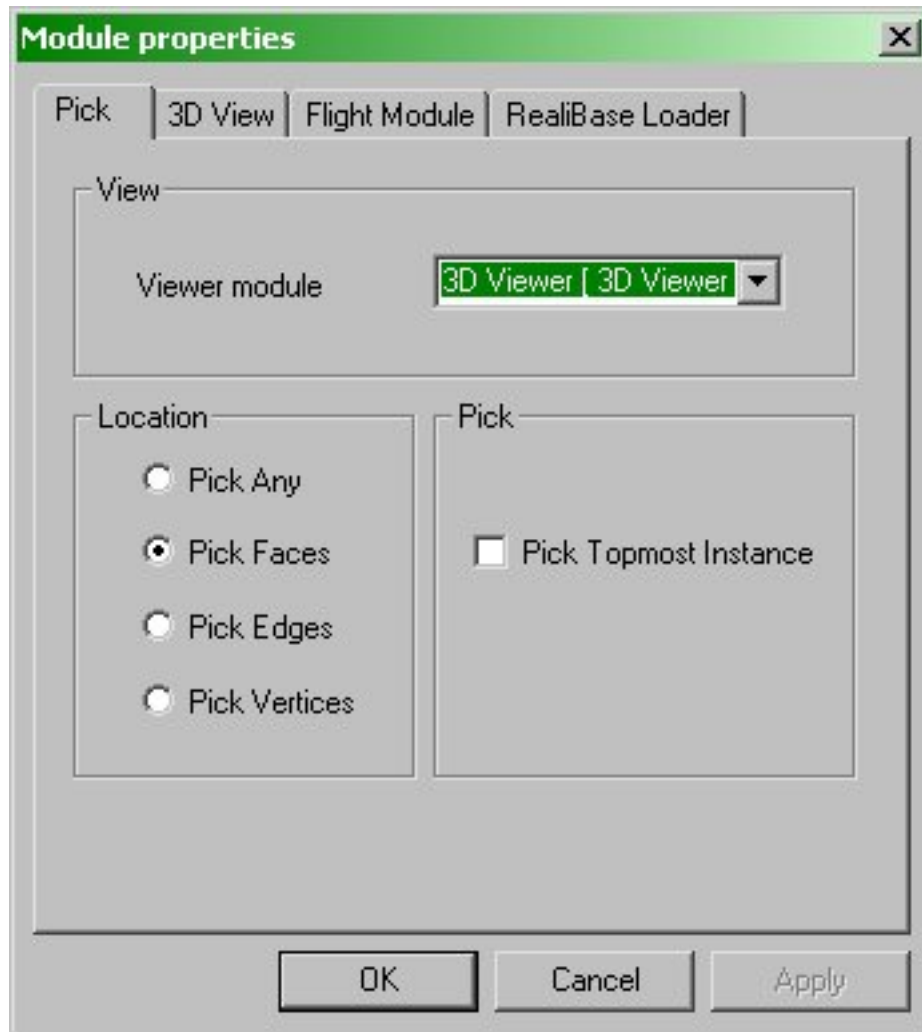
To enable a user to pick an object using a mouse, you need to set the Enterprise control module mouse focus property to this module. For details, see [Capturing mouse events inside the Enterprise control](#) (p.).

For information about this interface:

1. Add this module to an Enterprise control in your project.
2. Use the Object Browser to view the automation interface.

Note: This module has no properties that you can set in design-time. You can only access the properties of this module in run-time, via script.

3D Object Pick Tool properties



View

Viewer module

One of the following: Orthographic Viewer

Location

What is actually picked.

Pick any

Pick anything.

Pick faces

Pick the point on the face that is the intersection.

Pick edges

Pick the point on the edge closest to the intersection with the face.

Pick vertices

Pick the nearest vertex.

Pick

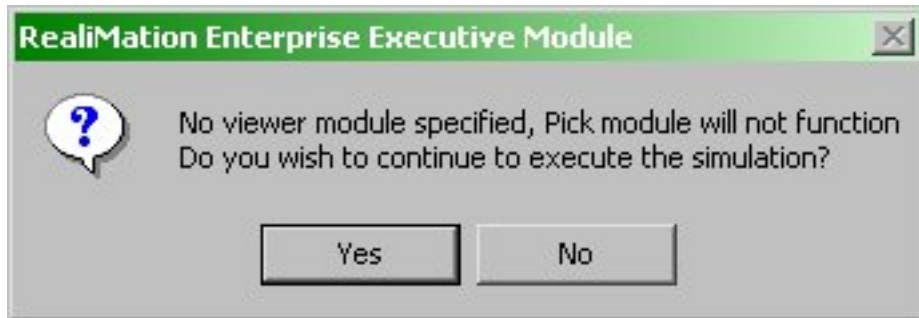
Which instance is picked.

Pick topmost instance

When selected, the instance directly in the view is returned. When deselected, the instance actually picked is returned.

3D Object Pick Tool troubleshooting

If you do not specify a viewer module, then you will get the following error message when you run your project:



Go back to the [properties](#) (p.74) page and select a viewer module.

3D Viewer

Displays a perspective projection of a view.

Related modules

Before you can display a view, you need to use the RealiBase File Processor to load the RealiBase that contains the view.

RealiBase File Processor

To navigate around a view, use the the Camera Controller module.

Camera Controller

3D Viewer properties



View

The view of the RealiBase to be displayed (a RealiBase can contain more than one view). You can select the view by either name or number:

Use view name

Selects the view by name.

Use view number

Selects the view according to its position in the RealiBase (for example, if you enter 3, then the Viewer displays the third view in the RealiBase).

Copy a view if not found

Uses the view with the lowest number in the RealiBase if the selected view is not found. If you do not select this check box, and the selected view is not found, then the Enterprise control will be blank.

Display mode

The level of rendering you want to display. Anti-aliasing might not be implemented on all display adapters with all drivers.

Display driver

The 3D display driver to use. To select a different driver, click the ... (Browse) button. This opens the [display driver](#) (p.104) properties page.

General

Show Frame Rate

Show the frame rate in the top left corner of the view.

Show Axes

Show the view axes in the bottom left corner of the view. If the GeoSpatial API Plugin is installed and the data has been set up then these axes will show the specified coordinate system.

Camera Controller

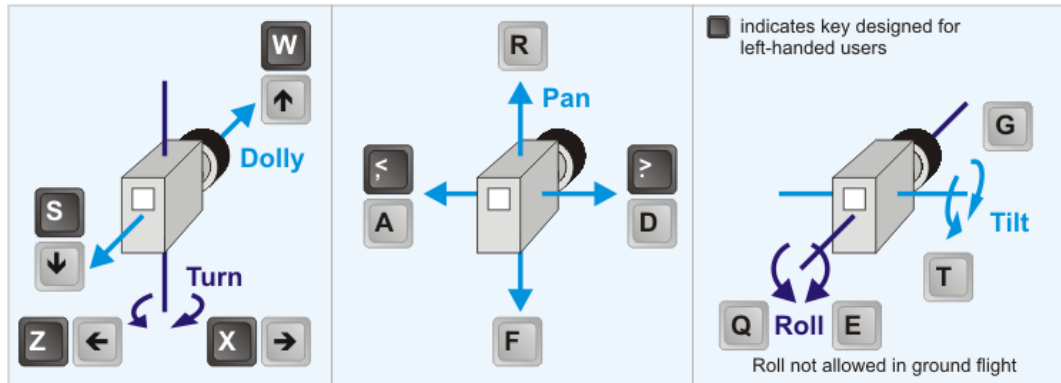
Allows you to interactively control a camera using a variety of input devices (by default, a mouse).

Using a mouse to control a camera

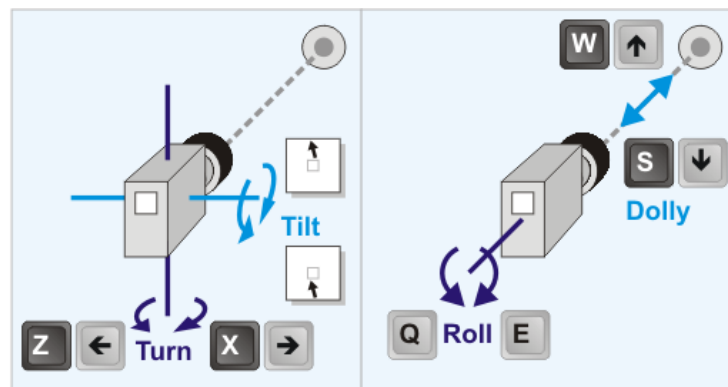
To enable this module to capture mouse events, you need to set the Enterprise control module mouse focus property. For details, see [Capturing mouse events inside the Enterprise control](#) (p.).

Camera Controller keyboard controls

Free flight and Ground flight



Orbit



Camera speed

There are two ways to control camera speed:

- Set the SensitivityValue property of the Camera Controller.
- Hold down the Ctrl or Shift key while pressing one of the keys shown above:

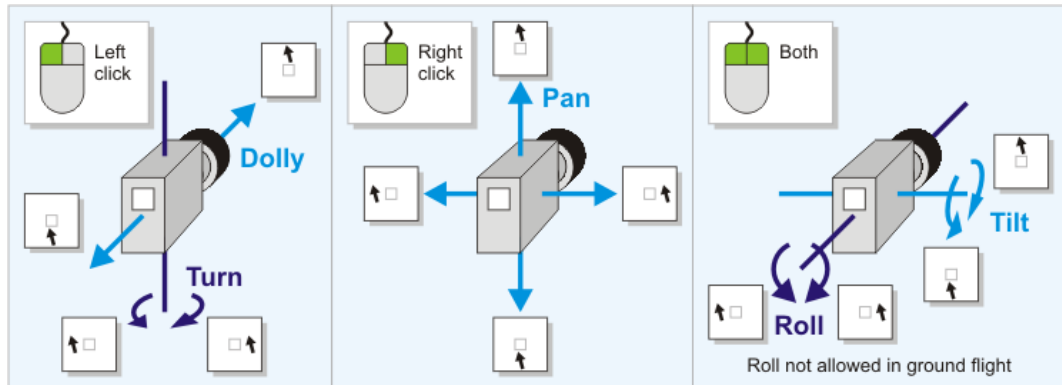


Related modules

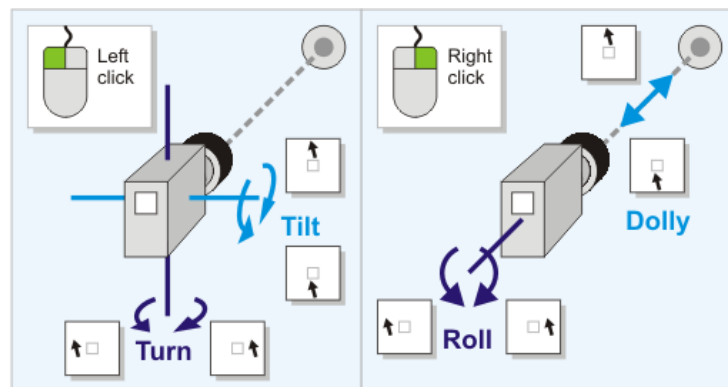
Requires the Keyboard Interface module.

Camera Controller mouse controls

Free flight and Ground flight



Orbit



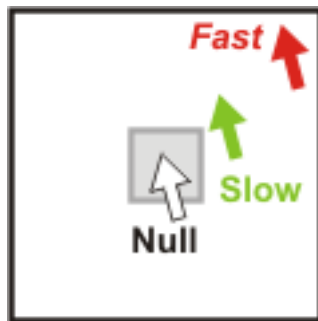
Camera speed

There are three ways to control camera speed:

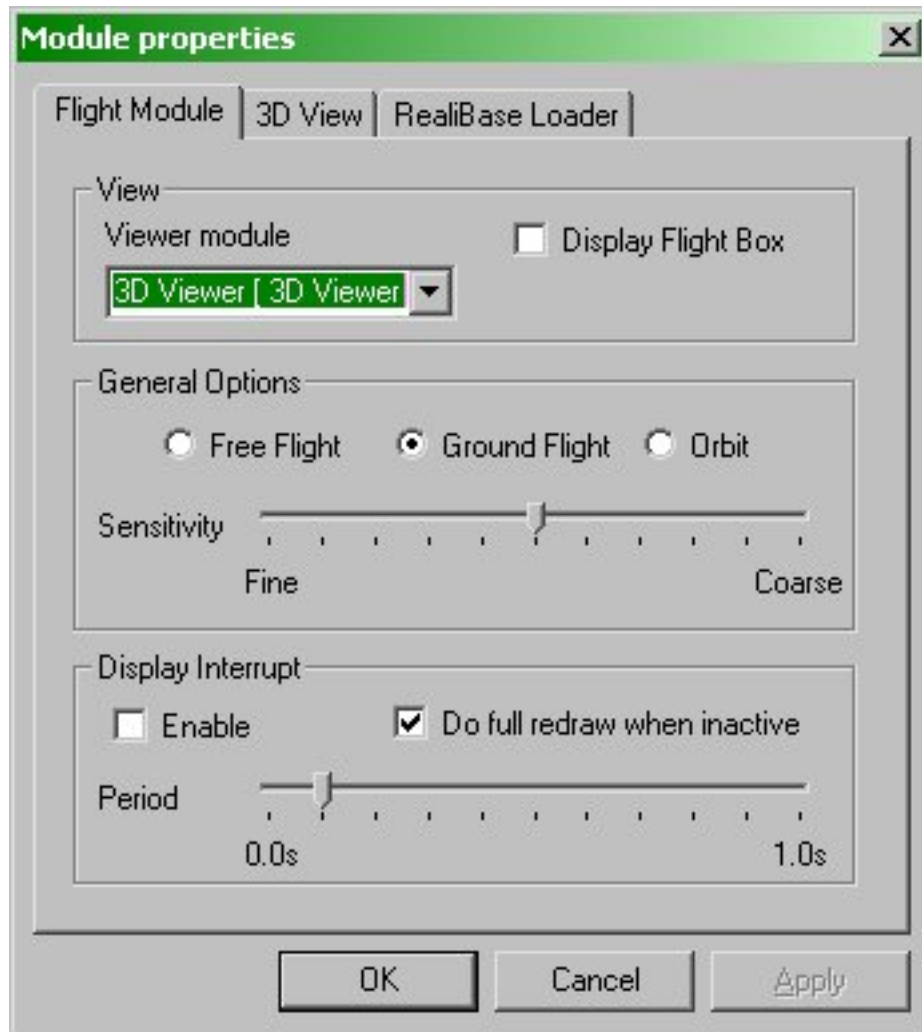
- Set the SensitivityValue property of the Camera Controller.
- Hold down the Ctrl or Shift key while pressing a mouse button:



- Click near the center of the view to move slowly, or near the edge of the view to move quickly:



Camera Controller properties



View

Viewer module

One of the following modules: Orthographic Viewer

Display flight box

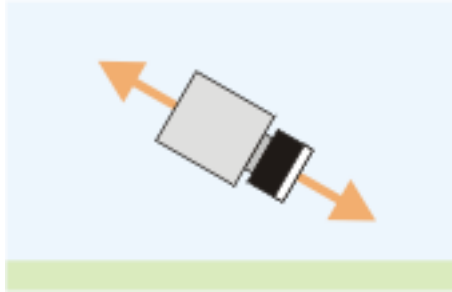
Displays a small square outline at the center of the Enterprise control, indicating a null zone.

Mouse actions inside the null zone do not affect the camera.

General options

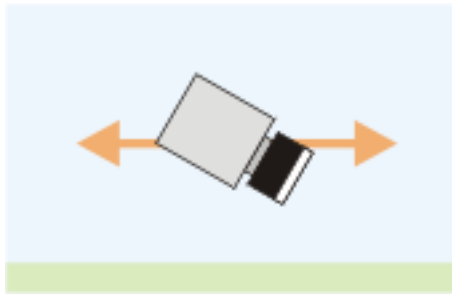
Free flight

Dolly (moving the camera in or out of a scene) follows the direction of the camera:



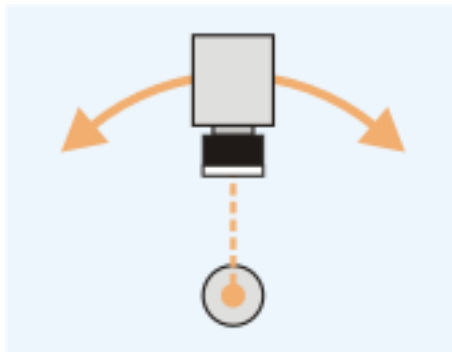
Ground flight

Dolly follows the ground, regardless of how the camera is oriented:



Orbit

The camera always faces the center of the view, or a particular object in the view:



Sensitivity

Controls how finely the camera responds to mouse and SpaceMouse movement.

Display interrupt

Use these options to interrupt the display of a frame if it is taking longer than the specified time. This puts a lower limit on the frame rate, at the expense of not fully drawing every frame.

Enable

If displaying the frame is taking longer than the specified period (below), then do not display any more of the frame.

Period

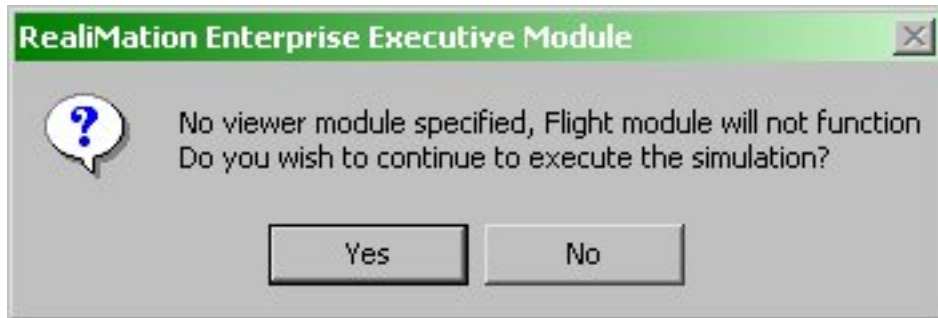
The maximum frame display time (in seconds).

Do full redraw when inactive

When the camera is stationary, redraw the entire frame.

Camera Controller troubleshooting

If you do not specify a viewer module for use with the Camera Controller, then you will get the following error message when you run the project:



To fix this problem, go back to the Camera Controller [properties dialog](#) (p.84) and select a viewer module.

Graph Tool

Displays custom “graph objects” (such as paths and AI nodes) in a 3D view.

Keyboard Interface

Exposes an automation interface for determining which key has been pressed, and defines actions for some keys.

Note: You can only access the properties of this module in run-time, via script. This module has no properties that you can set in design-time.

Predefined keys

The Keyboard Interface defines actions for the following keys:

Key	Action	Module
Page Up	Switch to next camera	Camera Controller
Page Down	Switch to previous camera	
+	Increase eye separation	
-	Decrease eye separation	
*	Increase view separation	
/	Decrease view separation	
Space Bar	Activate trigger	
F10	Start/stop animation	
F11	Rewind animation to time 0	

For example, if your application has a Camera Controller module and a Keyboard Interface module, then you can use the **Page Up** and **Page Down** keys to flip between cameras.

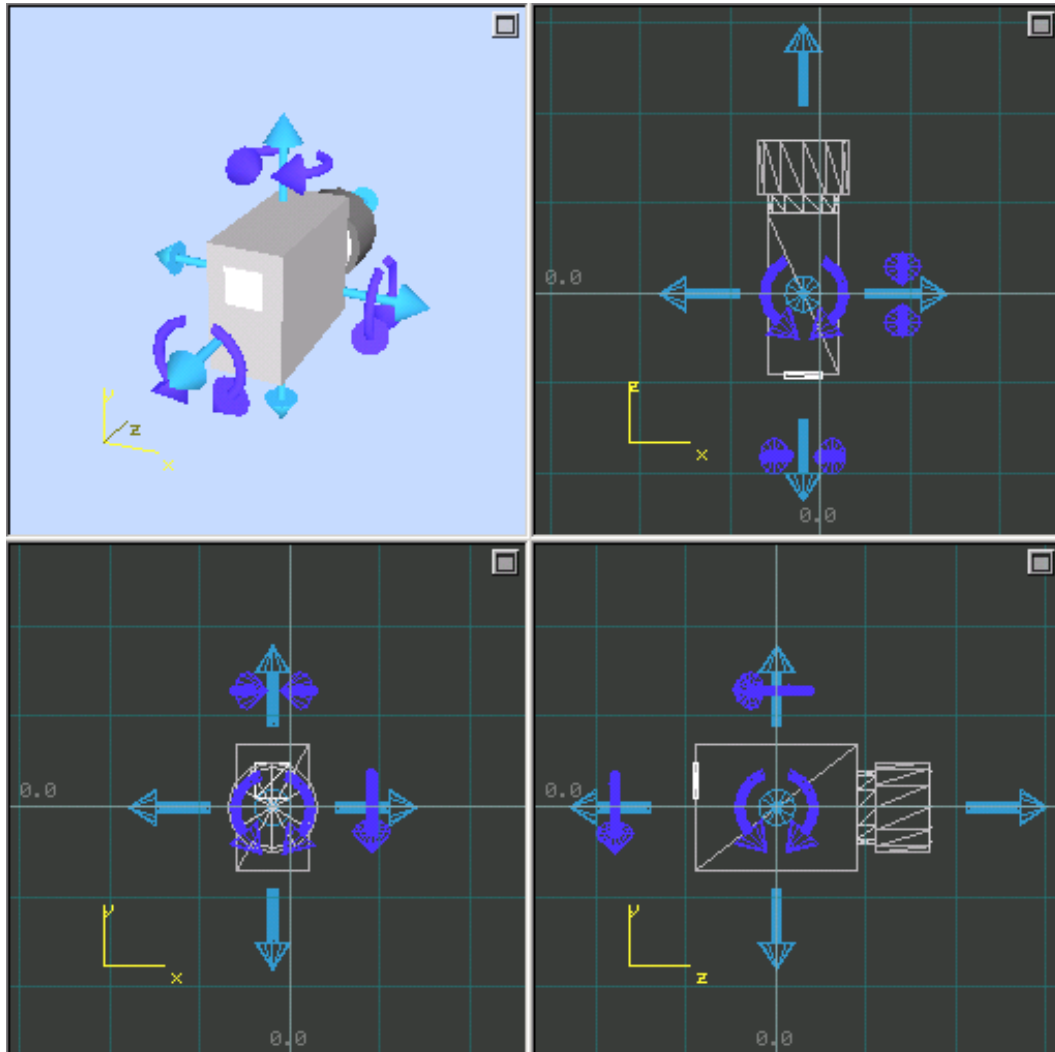
You can also trap key presses yourself, via script. The keys above will continue to perform their predefined actions, in addition to any actions that you define.

Controlling cameras via the keyboard

if your application has a Camera Controller module and a Keyboard Interface module, then you can use the keyboard to control camera movements. For details, see the Camera Controller module: Camera Controller


Orthographic Viewer

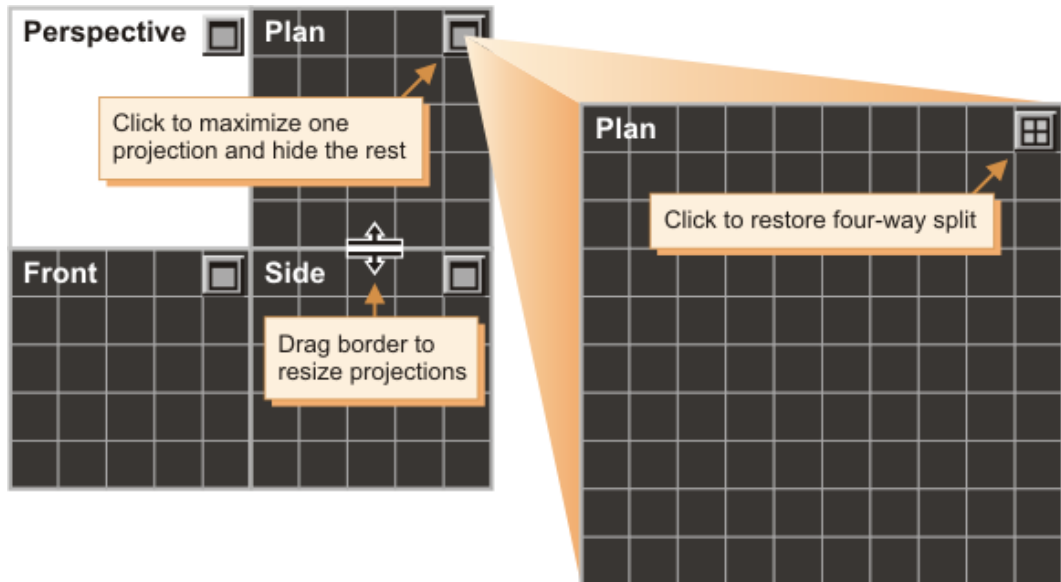
Displays four projections of a view: one perspective and three orthographic (front, plan and side).



Resizing the projections

To resize the projections, drag the border between them.

To maximize one of the four projections, click its  icon.

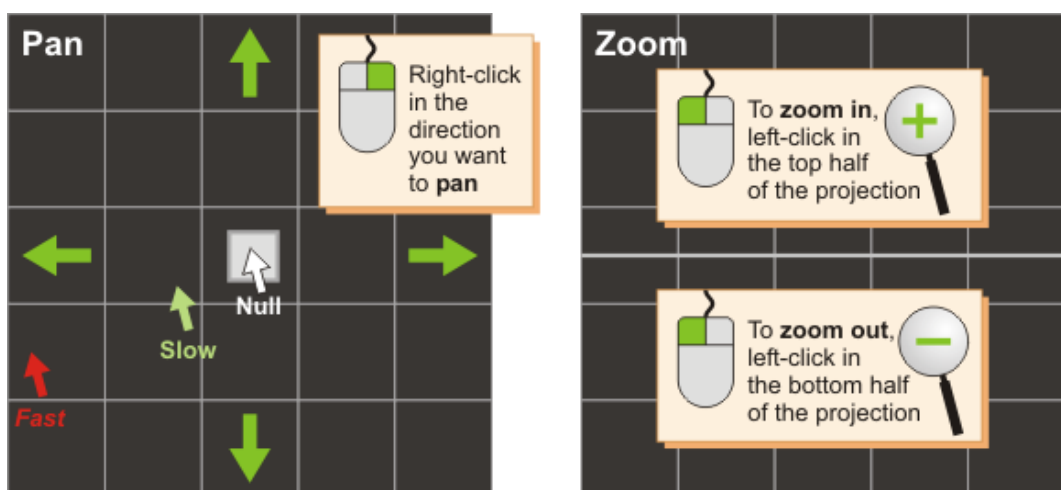


Moving around the perspective projection

The perspective projection is displayed by the 3D Viewer module. For information on controlling the camera in this projection, see the Camera Controller module: Camera Controller

Panning and zooming the orthographic projections

To pan and zoom each of the orthographic projections:



The closer to the edge of the projection you click, the faster you go. The center of the projection is a null zone; clicking here has no effect.

RealIBase File Processor

Loads a RealIBase, and exposes an automation interface that allows you to save the RealIBase.

Related modules

To improve performance for large RealIBase files, consider using the RealIBase Paging Processor:

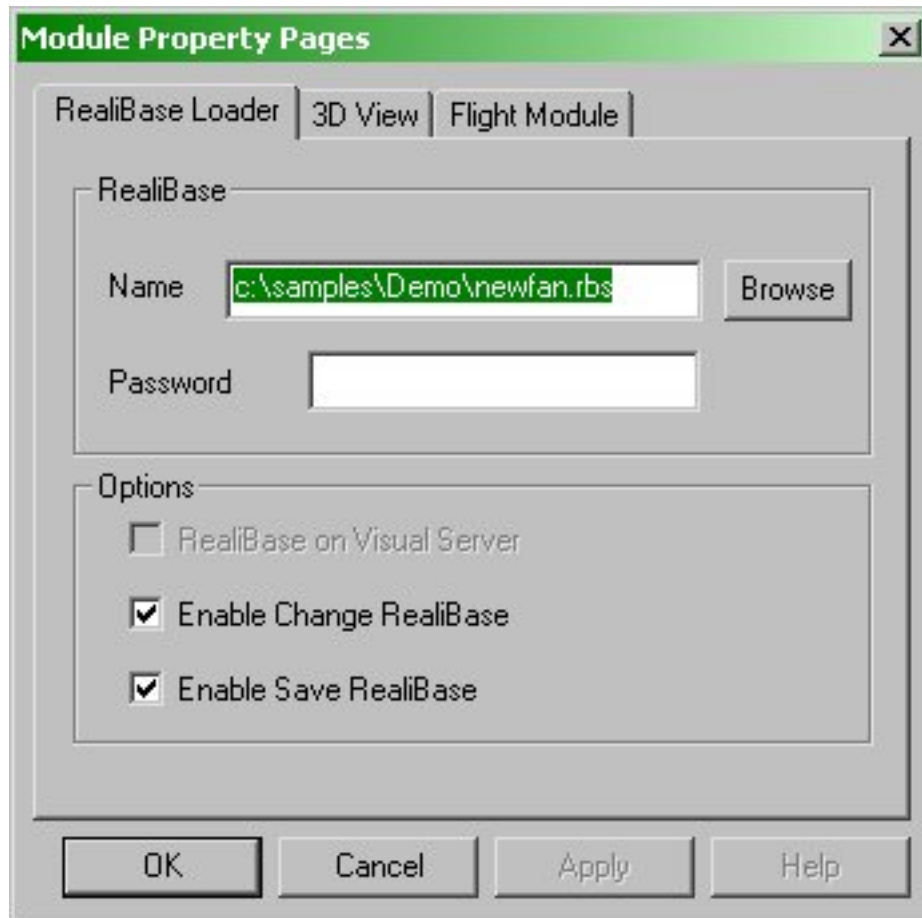
RealIBase File Processor

Currently, the only Enterprise module that allows you to change a RealIBase is the RealIMotion Script Interface:

RealIMotion Script Interface

You only need to save a RealIBase if you have used the RealIMotion Script Interface module to make changes.

RealiBase File Processor properties



RealiBase

RealiBase information

Name

The pathname of the RealBase (.rbs) that you want to load. To select a RealBase using the standard File Open dialog, click the **Browse** button.

Password

The password associated with this RealBase. If the RealBase has no password, then this field is ignored.

Options

RealBase on Visual Server

This check box is enabled only if your application uses one of the modules that sends views to remote computers:

- Multi Channel Viewer
- Passive Stereo Viewer
- Remote Viewer

By default, this check box is not selected: when one of the above modules

starts, it sends the entire RealiBase to each of the remote computers. For large RealiBases, this could be slow.

To improve performance, copy the RealiBase to each of the remote computers, and then select this check box.

Note:

- The pathname of the RealiBase must be exactly the same on all computers.
- For RealiBases that contain references, you need to copy the RealiBase, and all its references, to the remote computers.

Enable change RealiBase

Allow the user to change the RealiBase while the application is running. This is useful if your application is a general RealiBase viewer. If your application relies on a specific RealiBase, then uncheck this option.

This check box is only enabled if your application contains a Multi Channel Viewer or a Remote Viewer module.

Enable save RealiBase

Allow the user to save the current RealiBase. The methods to save the current RealiBase or save it to a new file are only available via the automation interface.

RenderWare Studio Utility Tool

Exposes an automation interface for various purposes, including loading and saving games, and detecting when objects have been selected.
For more information, see the RenderWare Studio Help.

Tooltip Viewer

Displays a tooltip when the cursor hovers over an object in the view.

To define tooltips for an object, use RealiMation Designer to add custom data to the object, of type RealiMation Tooltip.

Related modules

You can display tooltips for objects displayed by the following modules:

- 3D Viewer

- Active Stereo Viewer

- Multi Channel Viewer

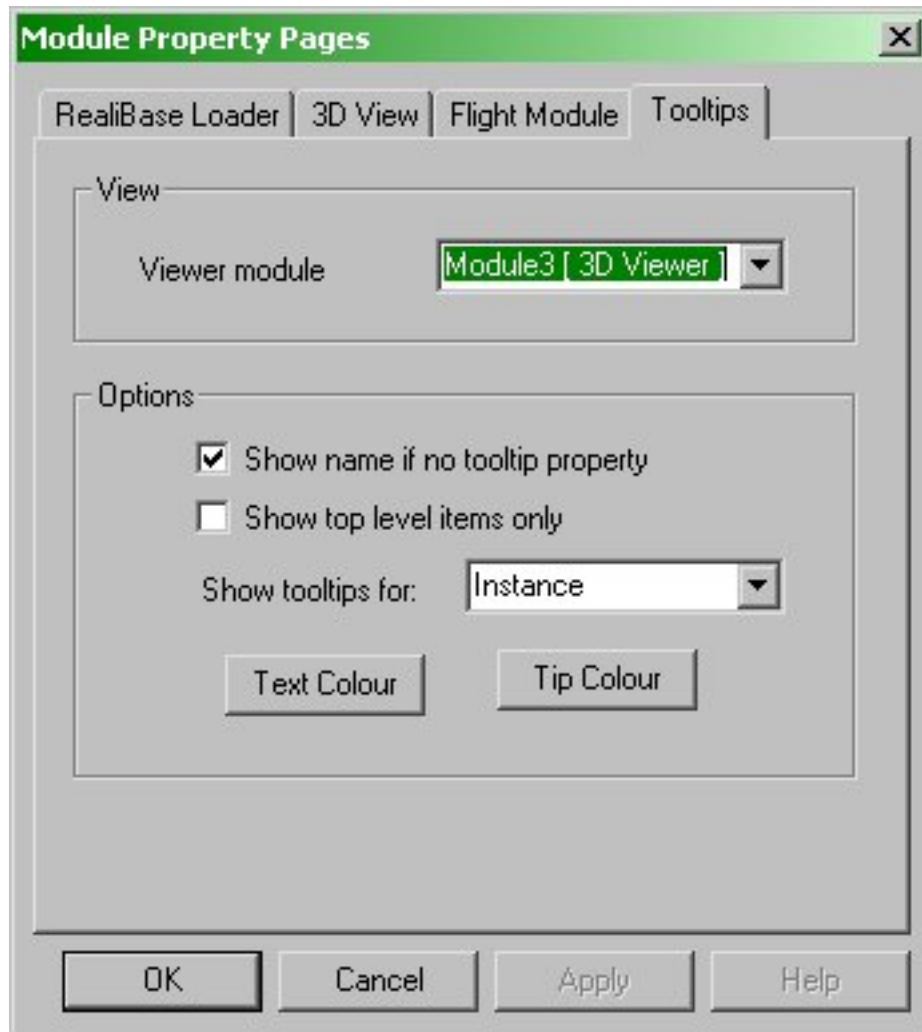
- Orthographic Viewer

- Plan Viewer

- Passive Stereo Viewer

- Remote Viewer

Tooltips Viewer properties



View

Viewer module

The viewer module can be one of the following:

- 3D Viewer
- Active Stereo Viewer
- Multi Channel Viewer
- Orthographic Viewer
- Plan Viewer
- Passive Stereo Viewer
- Remote Viewer

Options

Show name if no tooltip property

If there is no tooltip property attached to the instance or geometry object then display the object name instead.

Show top level items only

Only show tooltips for instances directly in the view.

Show tooltips for

The object types for which tooltips are displayed:

- Instances
- Shapes
- Materials
- Textures
- Images

Text color

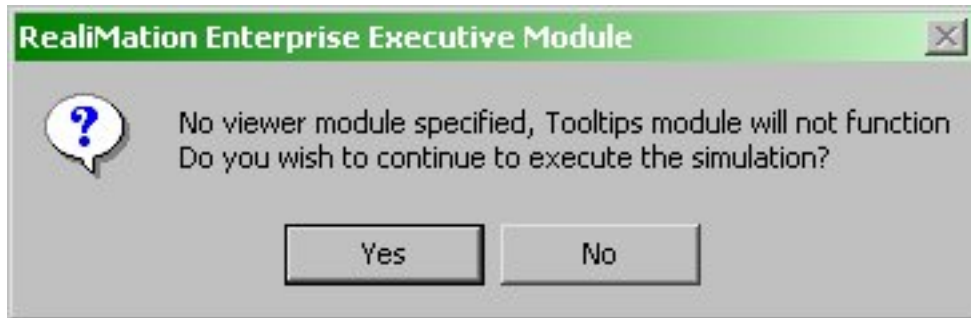
Select the color for the tooltip text using the [color properties](#) (p.102) dialog.

Tip color

Select the color for the tooltip text using the [color properties](#) (p.102) dialog.

Tooltips Viewer troubleshooting

If you do not specify a viewer module in which to display the labels, then you will get the following error message when you run the project:

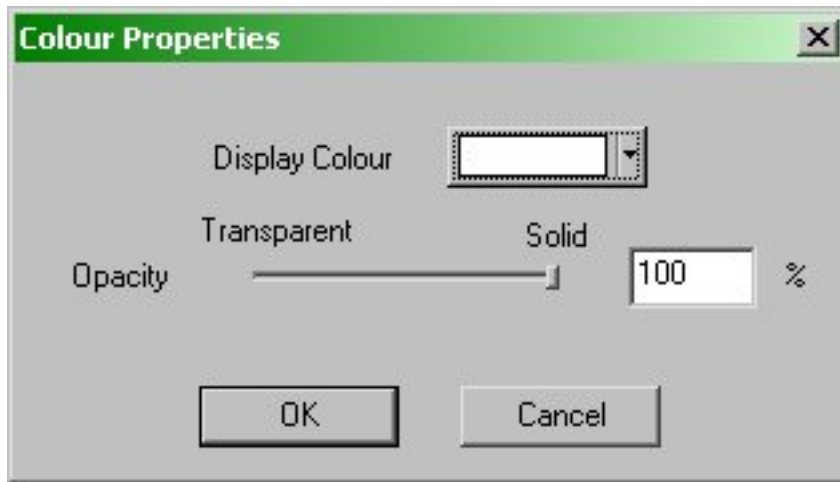


To fix this problem, return to the [properties](#) (p.98) dialog and select a viewer module.

Common properties

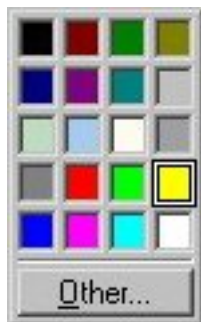
The dialogs for [color](#) (p.102) and [display driver](#) (p.104) properties are used by several module properties pages.

Color

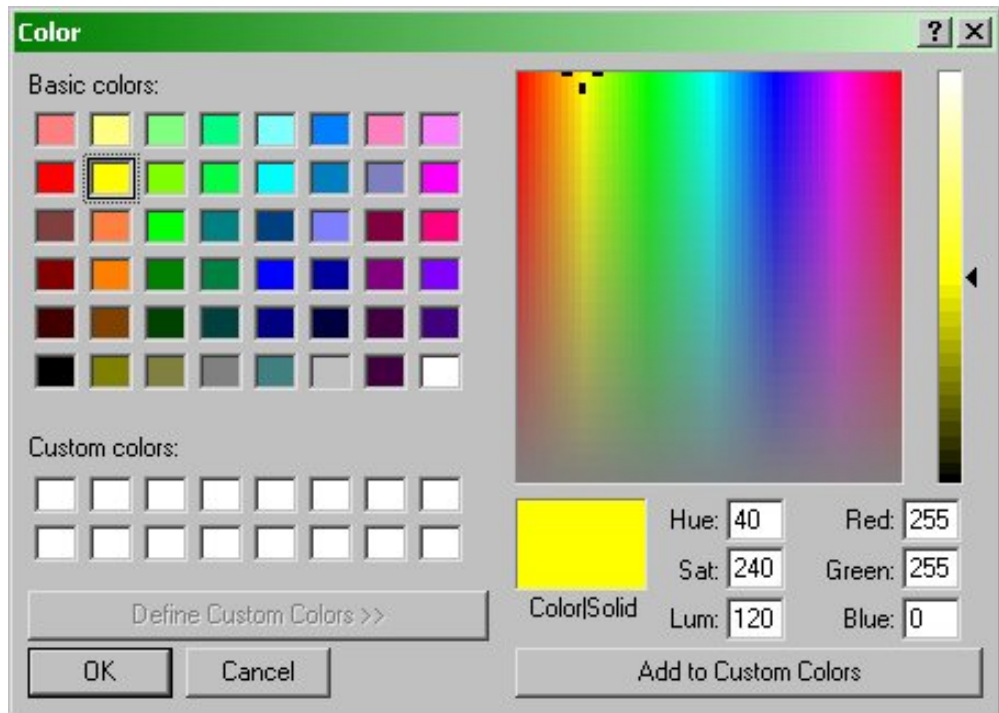


Display color

Click to display the color well control, allowing you to select the desired color:



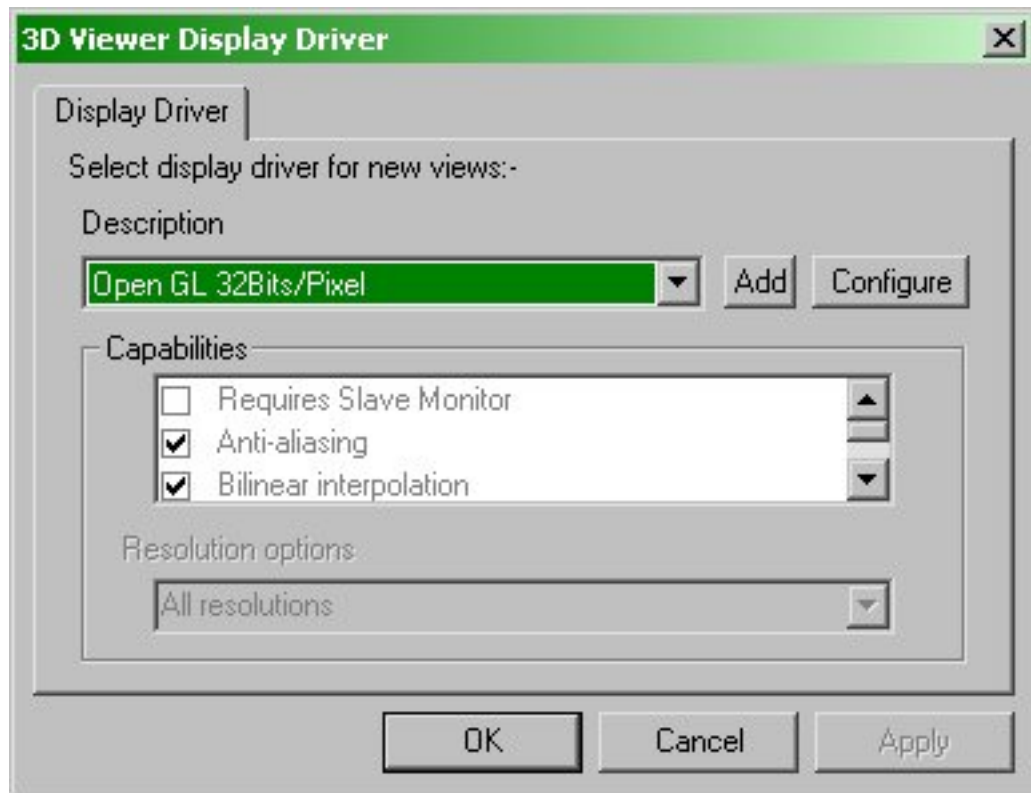
Select one of the standard colors or click **Other...** to display the standard color selector dialog.

**Opacity**

Specify a value between 0% (totally transparent, and hence invisible) and 100% (totally opaque, and hence fully visible).

Display driver

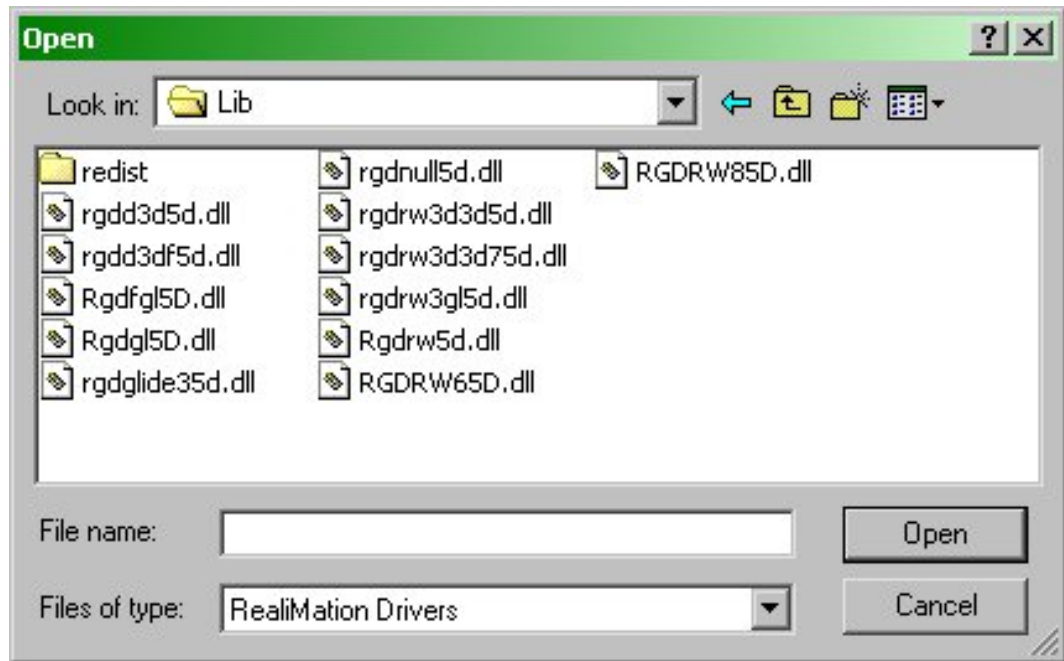
To select the driver that a viewer module uses to display a RealiBase, click the Browse (...) button on the viewer module properties page. This opens the display driver dialog:



Select a display driver from the **Description** list.

Adding a display driver

If the **Description** box does not list the display driver you want, but you know the driver is present on your system, then click **Add**. The following dialog appears:

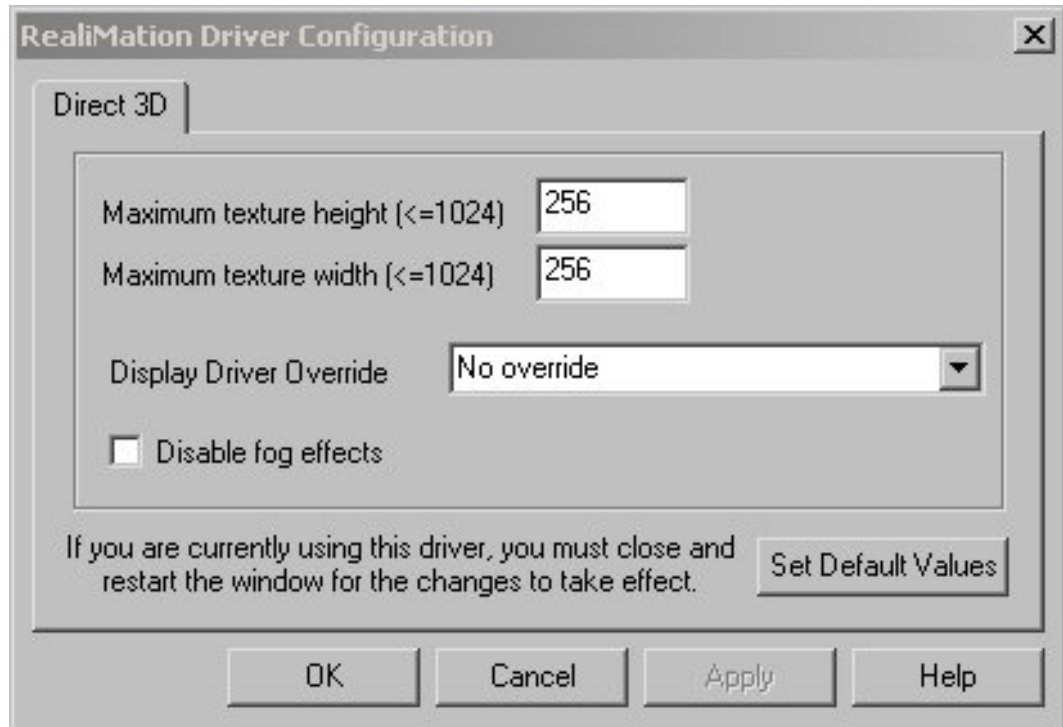


Browse to the folder containing the display driver, and then click **Open**.

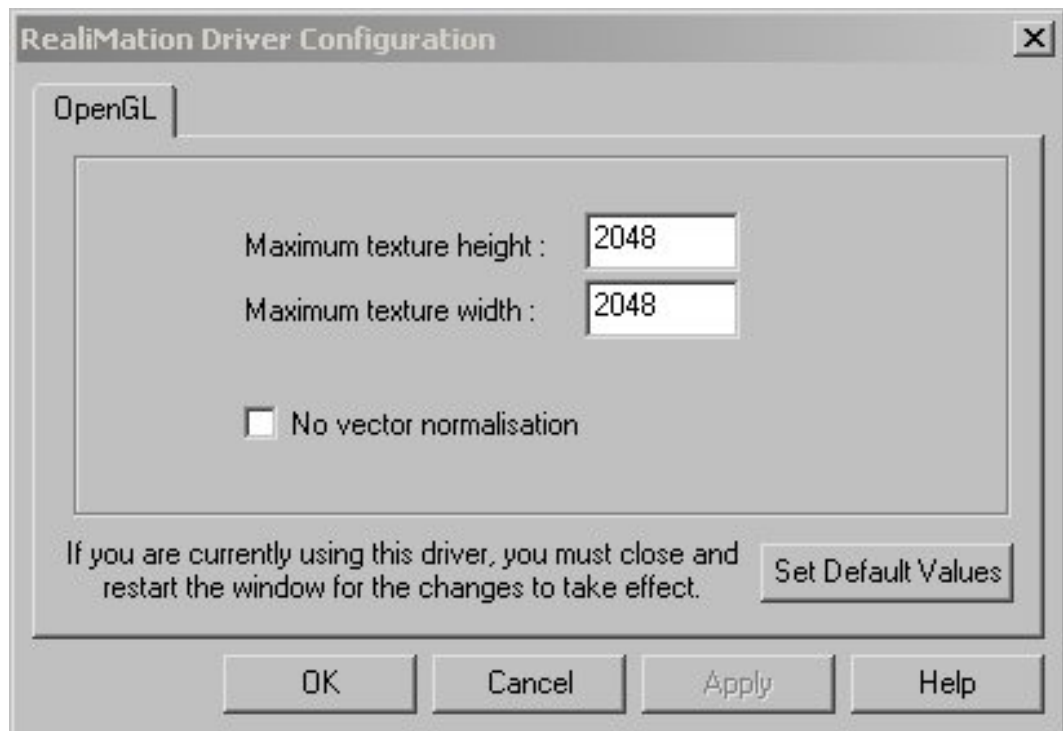
Configuring a display driver

To configure a display driver, select it from the **Description** box, and then click **Configure**.

For the Direct3D driver, the following dialog appears:



For the OpenGL driver, the following dialog appears:



Capturing mouse events inside the Enterprise control

The Enterprise control `ModuleMouseFocus` property determines which Enterprise module receives mouse events (for example, clicking) that occur inside the Enterprise control.

There are two Enterprise modules that can capture mouse events in the Enterprise control:

3D Object Pick Tool

Camera Controller

To enable one of these modules to capture mouse events, you need to set the `ModuleMouseFocus` property to refer to that module:

- In design-time, use the **Module mouse focus** drop-down list box on the Enterprise control properties page.
- In run-time, set the `ModuleMouseFocus` property to the `Pick3D` or `Camera` object:

```
object.ModuleMouseFocus = object.Pick3D
```

or

```
object.ModuleMouseFocus = object.Camera
```

where *object* is an Enterprise control object (for example, `EnterpriseCtrl1`).

Switching between navigating and picking

An Enterprise control can contain both a Camera Controller module and a 3D Object Pick Tool module, but only one of these modules at a time can have the mouse focus. If you want your application to include both navigating around a view and picking objects in the view, then you need to switch the mouse focus between these two modules.

Example: using the Alt key to change mouse focus

The following script sets the mouse focus to the 3D Object Pick Tool when you hold down the **Alt** key. When you release the **Alt** key, the script sets the mouse focus to the Camera Controller.

```
Sub EnterpriseCtrl1_KeyboardKeyState ( key, value )
    With EnterpriseCtrl1
        ' If Alt key down
        If key = 164 and value < 0 Then
            ' then set mouse focus to 3D Object Pick
Tool
                                .ModuleMouseFocus = .Pick3D
        Else
            ' else set mouse focus to Camera
Controller
                                .ModuleMouseFocus = .Camera
        End If
    End With
End Sub
```

This script requires the Keyboard Interface module:
Keyboard Interface