CoolEdit

Windows edit box history and filename completion

Introduction

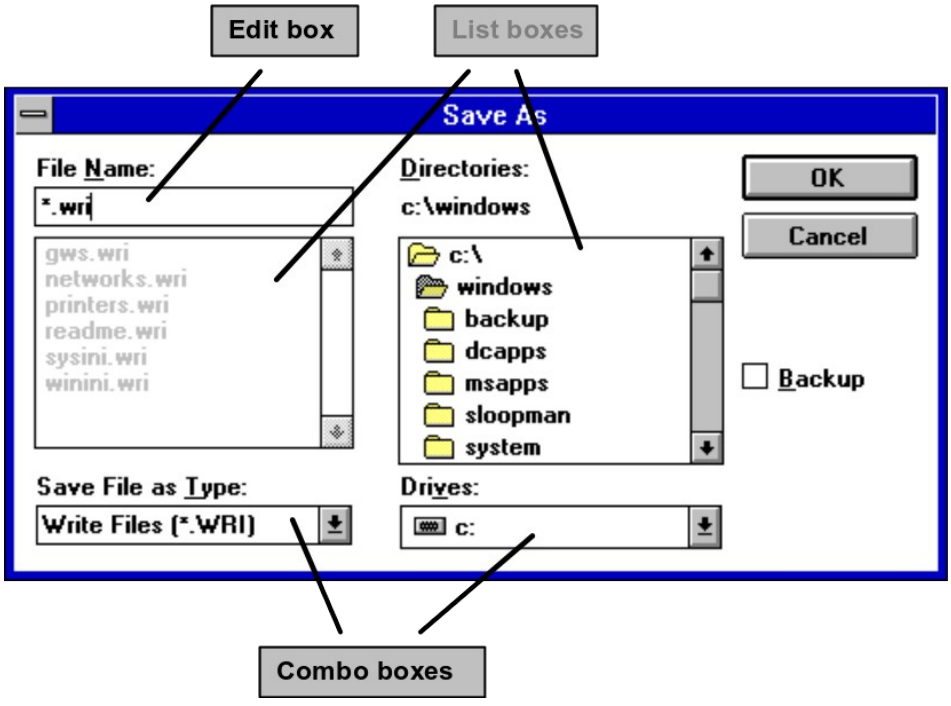
CoolEdit is a Windows application that provides a history list and filename completion features for **all** Windows edit boxes. Separate history lists are maintained for each edit box. The history lists are stored in a file when you exit Windows so that they will still be available the next time you run Windows.

You can use CoolEdit 'straight out of the box', just read the Quick Start topic in the help file, the default settings will work well in most cases. You may wish to read this document after you have been using CoolEdit for a few days and you wish to know more about it.

Terminology

The diagram below shows an example combo box, the 'Save As' box from Write, with some of the controls labelled.

Figure 1-Example Dialog Box



Edit Boxes
All edit boxes are enhanced with the history and filename completion features.

List Boxes
CoolEdit does not operate on list boxes.

Combo Boxes
Combo boxes are formed by combining an edit box and a list box. CoolEdit will provide the edit box part of a combo box with the history and filename completion features.

History Lists
History lists, or command buffers, remember the text that is typed into an edit control and make previous text strings available to you when you open the Combo Box again.

Filename completion
Filename completion evaluates all the existing files that match the text in the edit box.

Using CoolEdit

Using the history list

The up and down arrow keys replace the text in an edit box with items from the history list, in a combo box <Control> up-arrow or <Control> down-arrow keys achieve the same effect.

Using text and filename completion

The text and filename completion features of an edit control are activated by pressing the space bar twice; the first time you press the space bar a space is inserted in the text, the second time it is pressed CoolEdit starts attempting to complete the text.

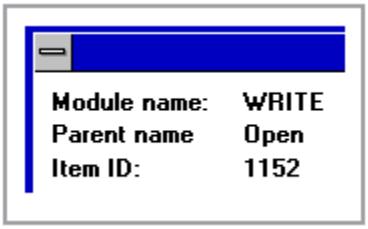
To complete the text CoolEdit first looks for lines in the history buffer that match the text up to the cursor position. Each time you press the space bar the text in the edit box is replaced with next possible completion. After all the matching lines from the history list have been shown CoolEdit starts looking for directory names, and then filenames, that match the text up to the cursor position.

At any time during completion you can press <Control><Spacebar> to undo the last completion.

Configuring CoolEdit

Uniquely identifying edit boxes (information only)

Figure 2-The control identifiers in the properties box



In order to maintain a seperate history list for all the different controls in Windows CoolEdit uses three features of a control that usually uniquely define a particular control.

The module name
The module name is a string identifier given to an application by the developers of the application, it will usually be similar to the application name. The module name for Word for Windows, for example, is MSWORD.

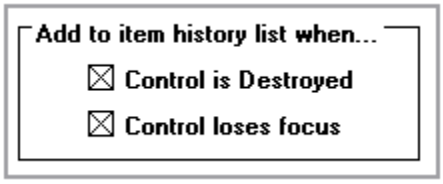
The parent name
The parent name will *usually* be the title of the dialog box that encloses the edit control. In some cases it may be the title of the main application window.

The item ID
The item ID is a numeric identifier given to the edit control by the developers of the application. In most cases the ID of each control in a combo box will be different. In some cases the ID will be the same for a group of controls, under these circumstances CoolEdit can not differentiate between the controls with the same ID

These three identifiers are listed, for information only, in the properties box for a control.

Adding items to the history list

Figure 3-Defining when items are added to a history list



CoolEdit obviously needs to know when the text in the edit box should be added to the history list for the box. Fortunately Windows notifies CoolEdit when the edit box either loses the focus or when the edit control is destroyed. The edit box will lose the focus when you either tab out of the box or when you click the mouse on a different control. The edit box will usually be destroyed when the enclosing dialog box is closed.

The *properties* dialog allows you decide when CoolEdit should add items to the history list

You will probably find it convenient to leave both these boxes checked, so that the text is added when either focus is lost or the box is destroyed, ***this is the default setting for these properties****.*

Resolving filenames and using case sensitive searching

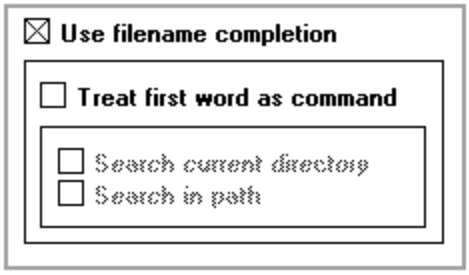
Figure 4-Resolving filename and using case sensitive searching

If the 'Resolve filename' property is enabled then CoolEdit will convert a simple filename (cooledit.ini for example) into the full path of the file (c:\windows\cooledit.ini) when it adds the item to the history list. ***The default for this option is enabled.***

If the 'Case sensitive searching' option is enabled CoolEdit will treat items as different if they contain the same text but are in different cases. ***The default for this option is disabled.***

Configuring filename completion

Figure 5-Configuring filename completion



You can configure CoolEdit to attempt filename completion using different methods to suit a particular edit control.

In File|Open, File|Save As etc dialog boxes you will want to have filename completion enabled. In other cases, Edit|Find dialogs for example, you may not want CoolEdit to attempt filename completion. ***The default setting is to enable filename completion.***

You can also specify that the first word in a edit control should be treated as a command, in this case CoolEdit will attempt to complete the word with a .COM, .EXE, .BAT or .PIF file. You will probably want to enable this option in the Program Manager and File Manager File|Run dialogs. ***The default setting is to disable this option.***

If the 'Treat first word as command' option is enabled you can also specify where cooledit should search for command files, in either, or both, the current directory or the MSDOS path. You would normally want to check both these controls.

Global configuration

The Options dialog allows you specify how many items can be added to a history list before earlier items are deleted. You can also specify how many characters in lengh an item must be before it is added to the history list; you'll probably want to ignore items of less than three characters.

Figure 6-The Options dialog

These are a few of my favourite things...

Here are a few of the places I find CoolEdit to be most helpful, I hope you'll find others.

File|Run
Neither the File|Run in Program Manager or File Manager includes a history mechanism so CoolEdit is great for remembering those. I also included the 'Treat first word as command' option to enhance filename completion in these these even more.

File|Copy, File|Search
Both of these File Manager dialogs are much improved with CoolEdit running.

File|Open
Most applications have a number (usually 4) of recently opened files. CoolEdit can remember far more than that. I also find the Directory and Files boxes cumbersome to work with, that's why I put the filename completion feature in. I included the 'Resolve filename' option for these boxes, that little change gave an enormous improvement to the usefulness of CoolEdit.

Edit|Find
A lot of applications still don't have histories for search and replace dialogs, they do if you're running CoolEdit.

Parameters for Dos Programs
If you include a ? in the Optional Parameters field in the PIF file for a DOS application Windows pops up a box asking you for the parameters for the program. CoolEdit maintains a separate history list for each PIF file, great for those Dos programs with impossible to remember switches that you always need!

Revision History

.99a	7/4/94	Fixed some problems with international COMMDLG.DLLs
.99	27/3/94	First release of CoolEdit outside of Cheltenham, UK

Technical issues

Memory usage

History lists are stored in a file until a control becomes active and written to the file when the control is destroyed in order to conserve memory. The memory overhead per control is about 16 bytes, although this is difficult to estimate exactly.

Non-Windows Controls

Some applications do not use the standard Windows edit controls, the non-standard controls that they use cannot be subclassed and so CoolEdit will not operate on these controls.