Color Selector Custom Dialog Control

Help Index

Color Selector Custom Dialog Control

Product Information

Copyright Information

CLRCTRL

Color Selector Custom Dialog Control
for Microsoft Windows (Tm) Applications

Version 1.2
8/25/1991

Scott Gourley
Compuserve ID 72311,613
105 Union Street, Watertown, MA 02172
(617)-924-5761

Color Selector Custom Dialog Control

Registration

This software product is SHAREWARE. You are permitted to evaluate this software product for a period of 30 days. If, after that period, you find the software product useful, you must register the software product and send $15 with your name and address to the address shown below. You will then receive a registered copy of the kit , including the files necessary to use the OBJ interface to the control. You will also be entitled to receive a registered copy of the next major revision of the product plus technical support at no charge.

Scott Gourley
Clickon Software
105 Union Street
Watertown, MA 02172

You may also make additional copies of the evaluation kit for the purpose of allowing others to evaluate the software product, as long as no modifications or additions are made to the software, its documentation, or any associated files, and this kit is not bundled in a distribution of any other software except that which is distributed as Shareware or Public Domain.

Since this product is a programmer's kit, the RUN-TIME version of this product, which consists of the .DLL file alone plus code compiled against the CLRCTRL.H file, may be distributed as part of a RUN-TIME ONLY distribution of a commercial, shareware, or public domain application.

In the case of the OBJ interface to the code, the run-time portion of the product consists of the appropriate .OBJ file for the control, plus code compiled against the .H file.

Color Selector Custom Dialog Control

Feedback

If you find this software product useful and have any interesting comments or ideas on how it might be improved, please let me know! I will attempt to incorporate the best of these suggestions in future versions of this software product. And, if you happen to provide particularly valuable feedback, I will, at my discretion, register you free of charge.

Also, watch for other custom controls to be available as shareware soon. If I get a positive response from this product, I have many more that I will upload in the future!

Thanks again for evaluating this product!

Scott Gourley
Clickon Software
Compuserve ID 72311,613
(617)-924-5761

Color Selector Custom Dialog Control

Disclaimer

This software product is made available on an "as is" basis, and carries no warranties, express or implied, including, but not limited to, merchantability or fitness for a particular purpose. The author shall in no way be held liable for any damages resulting from the use of this software product or the media on which it is distributed, including, without limitation, loss of business profits, interruption of business, loss of information, damage to equipment, or any other incidental or consequential damages.

Color Selector Custom Dialog Control

Overview

Thank you for trying this product!

The CLRCTRL kit makes it easy for Win 3.0 programmers to include color selection controls in their application dialogs. The core of the kit is a "dynamic link library," CLRCTRL.DLL, which is a self-contained package that can easily be integrated into any program. The DLL allows the custom control to be manipulated the same way a built-in control is manipulated, using the SDK Dialog Editor.

The color selector control is a combobox that contains rectangles of color selectable by a user using the normal input actions for comboboxes. The advantage of this method of color selection is that it allows an easy, familiar way for the user to make a color selection, while requiring minimal space for the control on the dialog box.

The default color selector uses the standard 16 "pure" colors as its selection palette, but this can be changed using normal combobox messages. The following table gives the "pure" colors and their positions in the default color selector control.

Color	RGB value	Index in combobox list
Black	RGB (0x00, 0x00, 0x00)	0
Dark Red	RGB (0x80, 0x00, 0x00)	1
Dark Green	RGB (0x00, 0x80, 0x00)	2
Dark Yellow	RGB (0x80, 0x80, 0x00)	3
Dark Blue	RGB (0x00, 0x00, 0x80)	4
Dark Magenta	RGB (0x80, 0x00, 0x80)	5
Dark Cyan	RGB (0x00, 0x80, 0x80)	6
Dark Grey	RGB (0x80, 0x80, 0x80)	7
Bright Grey	RGB (0xC0, 0xC0, 0xC0)	8
Bright Red	RGB (0xFF, 0x00, 0x00)	9
Bright Green	RGB (0x00, 0xFF, 0x00)	10
Bright Yellow	RGB(0xFF, 0xFF, 0x00)	11
Bright Blue	RGB (0x00, 0x00, 0xFF)	12
Bright Magenta	RGB (0xFF, 0x00, 0xFF)	13
Bright Cyan	RGB (0x00, 0xFF, 0xFF)	14
White	RGB (0xFF, 0xFF, 0xFF)	15

Uses for this product

Because this control is flexible enough to allow its behavior to be modified by the programmer, it is useful in any situation where a selection of color must be provided. These situations range from simple text and background color selection in a text-based application to palette definition in a paint program. It can be up to the programmer what color choices are available and what the color choice means to the application program.

Color Selector Custom Dialog Control

Files Included In This Kit

The following files are included in this kit:

Custom Control Files

CLRCTRL.DLL	This is the dynamic link library containing the code that defines and maintains the color selector control. The library includes code to interface with the user program as well as code to interface with the SDK Dialog Editor.
CLRCTRL.H	This is the header file that defines the source-code interface to the control. It contains information that the user program can use to access the control and its DLL library.

Sample Program Files

CLRTEST	This is the makefile for the CLRTEST.EXE program. It should be generic enough to build the program in your environment. If not, it can be easily modified.
CLRTEST.C	This is the source code for the CLRTEST.EXE program. This program provides a simple test of the functioning of the color selector, and serves as an example of using the custom control kit.
CLRTEST.H	This is the main header file for the CLRTEST.EXE test program. It contains menu IDs, prototypes, variable defaults, and other information needed by the program.
CLRTESTD.H	This is the header that contains the dialog IDs used for controls defined in the dialog in CLRTEST.DLG.
WINSTD.H	This is a general header file of information to configure Windows applications.
CLRTEST.DEF	This is the module definition file for the CLRTEST.EXE program. All Windows applications require a module definition file.
CLRTEST.RC	This is the resource script file for the CLRTEST.EXE test program. It contains a definition of the application menu structure.
CLRTEST.DLG	This is the dialog definition for CLRTEST.EXE test program. The dialog allows the user to select a text color and a background color, and uses this data to paint the program's main window.
CLRTEST.ICO	This is the program icon for the CLRTEST.EXE test program.
CLRTEST.EXE	This is a pre-built copy of the color selector test program.

Documentation Files

CLRCTRL.HLP	This help file.
README.TXT	This file is an ASCII file that explains this software kit and how to use it. It contains largely the same information as this text.
RELEASE.120	This file is the release notes for the Version 1.2 upgrade of the kit.

OBJ Interface Files (registered kit only)

CLRCTRLS.OBJ	Small model object code for control's OBJ interface
CLRCTRLM.OBJ	Medium model object code for control's OBJ interface
CLRCTRLL.OBJ	Large model object code for control's OBJ interface

Color Selector Custom Dialog Control

Color Selector Behavior

Because the color selector control is defined as a combobox, all of the behavior associated with a normal combobox is supported in the color selector. The only exception to this is that direct text entry in the edit field is not implemented. The following paragraphs briefly describe the normal functioning of a color selector control.

Keyboard Interface

If the programmer defines the control to have the WS_TAB style, the user can give the control input focus by moving to the control with the TAB key.

If the programmer defines the control to be in a group using WS_GROUP, the left and right arrow keys also can be used to give the control the input focus. Moving the input focus out of the color selector works in the same way.

Once the control has the input focus, the up and down arrows cause the currently selected color, as displayed in the edit box, to change, moving through the defined set of color choices.

Pressing Alt-Up arrow or Alt-Down arrow will both cause the listbox to be alternately dropped down and removed. While dropped down, the list box will display up to six of the colors defined in the list. The up and down arrows then still work in the normal way, moving the "item selected" highlighting through the list box as appropriate.

Mouse Interface

With a mouse, the interface is also straightforward. Clicking the mouse on the control will give the control the input focus if it does not already have it. When it has the input focus, clicking on another control will cause the control to lose the input focus.

Clicking on the drop-down button of the control will cause the listbox to be displayed (or removed if it is already displayed) as described above.

When the list box is dropped down, a new color can be selected by using the mouse to scroll through the list and click on another color.

Color Selector Custom Dialog Control

Implementing Color Selectors

The following sections describe the steps necessary to add color selector controls to an application:

Color Selector Custom Dialog Control

Accessing the Color Selector DLL

To use the color selector control, the application must access the dynamic link library (DLL) file for the control. To accomplish this, the following steps are necessary:

1. Load the DLL Library
2. Free the DLL Library
3. Distribute the DLL with the Application

Color Selector Custom Dialog Control

Load the DLL Library

During program initialization, the DLL library must be loaded and initialized by the program. Add the following code to the *WinMain* function somewhere before the main message loop:

```
HANDLE hClrLib;
.
.
.
if ((hClrLib = LoadLibrary ("CLRCTRL.DLL")) < 32) return 0;
```

This code loads the library for the color selector control. If it cannot be loaded, returning a zero value from *WinMain* will cause the program to end. (If program clean-up is necessary, do it before the return statement.)

Note that the name of the DLL file is defined in the CLRCTRL.H header file under the symbol "CLRCTRL_DLLNAME." This symbol can be used in the *LoadLibrary* call, provided the CLRCTRL.H file is included by the .C file that contains *WinMain*.

Free the DLL Library

During program shutdown, the DLL library must be released by the application. Add the following code to the *WinMain* function somewhere after the main message loop:

```
FreeLibrary (hClrLib);
```

This code releases the program's access to the DLL library. The parts of the library that have been loaded into memory can be discarded by Windows once no applications are still accessing the library, so it is important that any application that uses the DLL frees it during shutdown.

Note that the hClrLib parameter needs to be the same value as that returned from the call to *LoadLibrary*. If the calls are both made directly from *WinMain*, hClrLib can simply be a local variable used in both calls. If the calls are instead made from subordinate functions defined in the application, programmer needs to provide a way of keeping the value around during the life of the program's execution.

Distribute the DLL library

Since the DLL becomes a separate but integral part of the application, it must be distributed with the application. The rules for where Windows looks for the DLL file are documented in the Windows SDK Guide to Programming, among other places. Normally, however, it is easiest to keep the DLL in the same directory as the application's .EXE file.

Please refer to the Registration section of this help document for information about distributing the files in this kit.

Accessing the Alternative OBJ Interface

An optional OBJ interface to the control is available to registered users of this kit, which allows an application to access the control's code without the need for including a separate .DLL file with the application.

The advantages of using the OBJ interface are:

- fewer files to add to the application's distribution kit and copy during installation, especially when using many custom controls
- less chance that the user will delete, misplace, or overwrite the .DLL file, causing the application to fail

The disadvantages of using the OBJ interface are:

- the application's .EXE file is larger
- the control's .OBJ file must be linked into the application, in contrast to the pre-linked .DLL file (which means that implementing new versions of the control with the application requires a relink)
- the .DLL file may still need to be kept in the development environment in order to allow using the Dialog Editor to modify dialog boxes that use the control
- the .OBJ file used must match the memory model that the application uses

The following steps are necessary to implement access to the control using the OBJ interface; these steps replace the section, "Accessing the Color Selector DLL":

1. Register the Control Class
2. Export the Control Window Procedure
3. Link the .OBJ file to the Application

Register the Control Window Class

When using the OBJ interface to the control, the *ClrCtrlRegisterClass* function should be called from *WinMain* sometime during the program initialization process. This function registers with Windows the special window class that is needed by the color selector control. The function takes an argument that is the program's instance handle, as in the following example:

```
ClrCtrlRegisterClass (hInstance);
```

Export the Control Window Procedure

Since the window procedure (*ClrCtrlWndProc*) for the color selector control will be called by Windows' Dialog Manager code, this function must be exported when using the OBJ interface to the control. Add the function to the list of exported functions in the application's .DEF file, as in the following example:

```
EXPORTS
        ...various function names ...
            ClrCtrlWndProc
        ...various function names ...
```

Link the .OBJ with the Application

When using the OBJ interface to the control, the appropriate .OBJ file must be linked into the application. Depending on the memory model (small, medium, or large) used by the application, link the proper .OBJ file (CLRCTRLS.OBJ, CLRCTRLM.OBJ, or CLRCTRLL.OBJ) into the executable file for the application. Any one of these files replaces all of the run-time functionality of the control that is defined in the .DLL.

Color Selector Custom Dialog Control

Color Selector Access Using the Dialog Editor

The easiest way to add color selector controls to an application's dialog is to edit the dialog using the Dialog Editor found in the Microsoft SDK. The following sections describe the steps necessary to add color selector controls to a dialog using the Dialog Editor.

Color Selector Custom Dialog Control

Installing the DLL Library

To access the color selector custom control from within the Dialog editor, the CLRCTRL.DLL file that defines the control must be installed" in the Dialog Editor. To do this, execute the Add Custom Control menu option from the File menu of the Dialog Editor, and give the full pathname of the control's .DLL file. This pathname will point to wherever this custom control kit is installed.

If the .DLL file ever needs to be de-installed, use the Remove Control option from the Dialog Editor's File menu, and choose the control library to be removed from the list presented.

Color Selector Custom Dialog Control

Creating a Color Selector in a Dialog

To use the color selector in a dialog, choose the Custom menu option from the Control menu. Then choose the CLRCTRL control from the list presented. The control also can be chosen from the Toolbox, if it is displayed. Once the control has been selected, position the plus sign cursor where the upper left corner of the control should be on the dialog, and click the left mouse button to add the control.

Modifying the Color Selector

After adding a color selector to a dialog, it can be moved and resized in the same way as a standard control. Keep in mind that the size of the control is really larger than the visible portion of the control, because of the drop-down area. To make a color selector the current object in the Dialog Editor, click the mouse in the drop-down area, instead of in the visible area, because the latter mouse click will be interpreted by the control and not the Dialog Editor.

Also, it is important to note how the vertical size of the control affects the control. The default vertical size of a color selector control is sixty dialog units. At this size, the height of the edit box and drop down button are the same as the height of their standard Windows counterparts. When dropped down, six color rectangles are displayed (or fewer if there are less than six color choices in the list.) If the size of the control is changed the size of the color rectangles and the size of the edit box and drop down button also change. There will still be six colors displayed in the dropped down list.

Within the Dialog Editor, the behavior is different. If the size of the control is changed and the dialog is then tested within the Dialog Editor, the edit box and drop down button do not change size. In addition, the number of color rectangles displayed when the list box is dropped down changes, instead of the size. Keep this difference in mind when sizing the color selector controls within a dialog.

Color Selector Custom Dialog Control

Color Selector Control Styles

A color selector's ID value is the only "style" associated with this type of control. To modify this value, double-click the mouse on the control or make the control the current object and press Control-C. Choosing the Styles menu option in the Edit menu also works. These actions cause the control's styles dialog box to be presented, which has an edit field for the control's ID value. This ID value field can be used in the same way as with a standard control; a number can be entered or a string value can be used that equates to a number using a #define in the header file associated with the dialog. See the SDK's Tools manual for information on how to maintain a header file of ID values for the dialog.

It is also possible to modify the dialog file without using the Dialog Editor using a standard text editor. A color selector control in a dialog uses the CONTROL statement in the dialog file and its format is the same as the CONTROL statement for a standard control. The class string for color selector's CONTROL statement is "ClrCtrl" -- see the SDK tools manual for information on the full format of the CONTROL statement.

Color Selector Custom Dialog Control

Color Selector Dialog Procedure Handling

To access a dialog's color selector control from the application, code must be added to the dialog procedure to initialize the state of the color selector and retrieve its current selected color at the end of dialog processing. To implement this access, perform the following steps:

1. Include the Color Selector Include File
2. Modify the Control's Color Choices
3. Set the Current Color Choice
4. Get the Current Color Choice

Color Selector Custom Dialog Control

Include the Color Selector Header File

The header file for color selector control access, CLRCTRL.H, should be included in any .C modules that define dialogs using the color selector control. This header file defines message codes specific to the color selector and other information useful to access the control.

Color Selector Custom Dialog Control

Modify the Color Choices

During WM_INITDIALOG message processing for the dialog, it is possible to modify the color choices available in the control. To do this, the standard Windows messages for modifying items in a combobox can be used.

For the following examples, hClrCtrl is assumed to be an HWND value, initialized to be a color selector's window handle. This value can be obtained in several ways, as explained in any Windows programming reference.

To add a color selection to the end of the control's list, use the CB_ADDSTRING message. For example,

```
SendMessage (hClrCtrl, CB_ADDSTRING, 0, RGB (0xC0, 0x40, 0x00));
```

will add an orange color to the end of the color selector's list. (Keep in mind that the color capability of the video hardware that the application is being run on will determine whether a particular RGB color is rendered as a pure color.)

To remove a color choice from the list, determine the index of the color in the list (starting at 0) and send the CB_DELETESTRING message to the control. For example,

```
SendMessage (hClrCtrl, CB_DELETESTRING, 3, 0L);
```

will remove the fourth color selection in the list. Note that removing an item will cause the indices assigned to all colors below the removed color to be decremented by one, so if more than one color selection is to be removed, it is best to remove them from the bottom up.

To insert a color choice in the middle of the list, determine the index of the position at which to insert the item and send the CB_INSERTSTRING message to the control. For example,

```
SendMessage (hClrCtrl, CB_INSERTSTRING, 7, RGB (0x80,0x00,0xFF));
```

will insert a lavender color after the first seven colors in the list. Note that inserting an item will cause the indices assigned to all colors below the inserted color to be incremented by one, so if more than one color selection is to be inserted, it is best to insert them from the bottom up.

For special situations, it may be desirable to remove all color selections and then add back a complete set. To do this, send the CB_RESETCONTENT message to the control to remove all current color selections in the list. For example,

```
SendMessage (hClrCtrl, CB_RESETCONTENT, 0, 0L);
```

will remove all color selections. (The last two parameters are ignored.)

Color Selector Custom Dialog Control

Set the Current Color Choice

During WM_INITDIALOG processing, it is possible to select the default color choice for a color selector. This can either be a hardcoded default choice, or it can be the saved value of the choice that was selected during the last time the dialog was processed. If the index of the desired default color choice is known, the CB_SETCURSEL message can be sent to the control. For example,

```
SendMessage (hClrCtrl, CB_SETCURSEL, 6, 0L);
```

sets the seventh color in the list as the default.

If the RGB color value of the desired default color is known, but the index of the color is not known, a special color selector message, CLRM_SETCURCOLOR can be used. For example,

```
SendMessage (hClrCtrl, CLRM_SETCURCOLOR, 0, RGB (0xFF,0x00,0x00));
```

sets the current color selection to be red. Note that if the exact RGB color specified does not exist in the control's list, the current color selection will not be changed, and a CB_ERR value will be returned. (When a combobox control is created, its initial current selection is index 0, until changed by a message such as those above.)

Color Selector Custom Dialog Control

Get the Current Color Choice

When a user action indicates that the current dialog control values should be retrieved and used (such as when the user presses an "OK" or "Apply" button), the current color value for a color selector can be retrieved as an RGB value by using the special color selector message, CLRM_GETCURCOLOR. For example,

```
COLORREF rgbColor;
.
.
.
rgbColor = SendMessage (hClrCtrl, CLRM_GETCURCOLOR, 0, 0L);
```

will store in rgbColor the current RGB color selected in the control. (The last two parameters are ignored.)

Color Selector Custom Dialog Control

Windows Message Interface

To make the color selector control as flexible as possible, most of the standard Windows messages and notification codes that are supported by a combobox control also are supported by the color selector control. The following sections contain further information about this support.

Color Selector Custom Dialog Control

Color Selector Messages

The following messages are defined as part of the interface to color selector controls:

CLRM_GETCURCOLOR	retrieve the RGB color of the current selected item in the control. wParam and lParam are not used. The return value of the *SendMessage* call is the current selected RGB value. See the Color Selector Dialog Procedure Handling section for information on using this message.
CLRM_SETCURCOLOR	set the current selected item of the control to the specified RGB color. wParam is not used for this message. lParam is used to pass the desired RGB color value. The return value of the *SendMessage* call is CB_ERR if the specified RGB color is not in the control's list. See the Color Selector Dialog Procedure Handling section for information on using this message.

Color Selector Custom Dialog Control

Windows Messages

The following Windows messages are supported in the color selector control, either by special processing or by default processing handled within Windows.

WM_CREATE	create the control on the dialog
WM_DESTROY	remove the control from the dialog
WM_SIZE	resize the control
WM_PAINT	repaint the control
WM_COMMAND	process commands from the user
WM_ACTIVATE	activate or inactivate the control
WM_CHAR	process a keyboard character sent to the control
WM_ENABLE	enable or disable the control
WM_KEYDOWN	process a key press for a non-system key
WM_KEYUP	process a key release for a non-system key
WM_KILLFOCUS	remove the input focus from the control
WM_MOVE	move the control on the dialog box
WM_SETFOCUS	give the input focus to the control
WM_SYSCHAR	process a system keystroke sent to the control
WM_SYSKEYDOWN	process a key press for a system key
WM_SYSKEYUP	process a key release for a system key

See the SDK Reference manual (volume 2) for more information on these messages.

The control sends the following messages to its dialog parent:

WM_CTLCOLOR	ask the dialog to change the drawing attributes used to paint the control (note that these attributes are used to draw the structural aspects of the control, and do not affect the color choices in the color selector's list)
WM_DELETEITEM	tell the dialog that a color choice has been removed from the color selector's list

These messages control the comobox-specific aspects of the color control:

CB_ADDSTRING	add an item to the end of a combobox's list
CB_DELETESTRING	delete an item from a combobox's list
CB_GETCOUNT	determine the number of items in a combobox's list
CB_GETCURSEL	determine the index of the currently selected item in a combobox
CB_GETITEMDATA	retrieve the data associated with an item in a combobox (for color selectors, this data is the stored RGB color value)
CB_INSERTSTRING	insert an item in the middle of a combobox's list
CB_RESETCONTENT	remove all items from a combobox's list
CB_SETITEMDATA	store a data value in a combobox item (for color selectors, this data is the RGB color value)
CB_SETCURSEL	change the currently selected item in a combobox

Color Selector Custom Dialog Control

Windows Notification Codes

The color selector control returns the following combobox notification codes to its parent window, in WM_COMMAND messages:

CBN_DROPDOWN	notify the dialog that the color selector listbox has been dropped down
CBN_KILLFOCUS	notify the dialog that the color selector control has lost the input focus
CBN_SELCHANGE	notify the dialog that the color selector current color has changed
CBN_SETFOCUS	notify the dialog that the color selector control has gained the input focus
CBN_EDITCHANGE	notify the dialog that the color selector control's edit box may have changed
CBN_EDITUPDATE	notify the dialog that the color selector control's edit box will be changed

See the SDK Reference manual (volume 2) for more information on these codes.

Color Selector Custom Dialog Control

Sample Program

This kit comes with a sample Windows program, CLRTEST.EXE. The following sections describe the design and use of the program.

Color Selector Custom Dialog Control

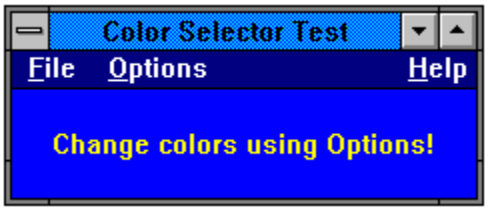
Sample Program Purpose

CLRTEST.EXE serves two purposes: first, it provides a good test of the control and its application interface; second, it represents a clean example of the use of the control in a program.

In addition, it may come in handy as a starting point for testing special ways of interfacing with the control, whenever changes to the control's standard behavior are desired.

Color Selector Custom Dialog Control

Using the Sample Program



The CLRTEST.EXE program consists of a normal application window and a short application menu. The window contains a line of sample text that is colored according to a default color value defined in the test program, displayed on a background that is colored by another default color value.

Color Selector Custom Dialog Control

Sample Program Commands

The sample program has an application menu with the following options:

File	Controls the exit options of the program
Options	Controls the testing options of the program
Help	Provides access to online help for the color selector kit

File Menu

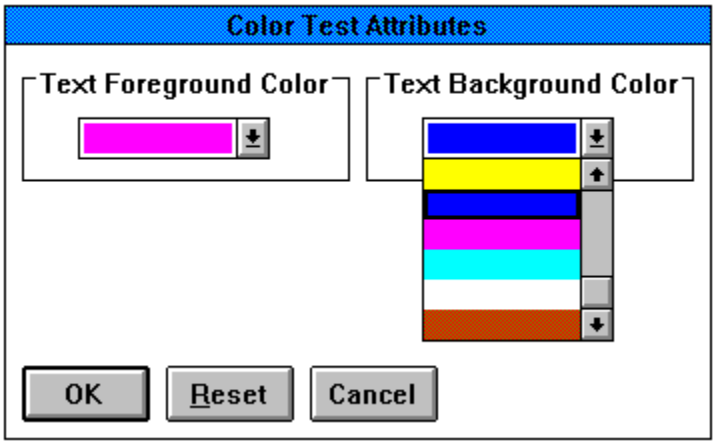
The File menu has a standard meaning on most Windows applications, but in this program, only one standard File menu option is defined: Exit. When Exit is chosen, the program simply shuts down.

Color Selector Custom Dialog Control

Options Menu

Under the Options menu, there is one option: Test Color Selector. This option displays the "Color Test Attributes" dialog.

Color Test Attributes Dialog



This dialog is used to test the functionality of the color selector control. It contains the following controls:

Text color selector

Changing the currently selected text color using this control changes the corresponding current text color used when repainting the program's main window.

Background color selector

Changing the currently selected background color using this control changes the corresponding current background color used when repainting the program's main window.

OK pushbutton

Clicking on the OK pushbutton causes the colors currently selected in the color selectors to be stored as the program's current color settings, and the main application window is repainted with those color settings.

Cancel pushbutton

Clicking on the Cancel pushbutton causes the color settings to be left as they were before the dialog box was displayed.

Reset pushbutton

Clicking on the Reset pushbutton resets the current colors in the color selectors to the default values defined in the program code.

To test and demonstrate the ability to modify the default 16 "pure" color choices in a color selector, the following changes were made to the controls in this dialog:

Text color selector

Dark Magenta was removed, and Sky Blue was added in the fourth position in the list.

Background color selector

An orange color was added at the end of the list.

These changes are made with standard Windows messages defined for comboboxes.

Color Selector Custom Dialog Control

Help Menu

The Help menu provides access to this online help text. Besides help information for the test program, this text contains information about using the color selector control in other applications.

In addition, an "About Color Test..." option is defined, which provides general information about the kit.