DOWNLOAD PDF CUSTOMIZING THE FONT DIALOG

Chapter 1 : c# - Customizing dynamic font style and size in a custom dialog box - Stack Overflow

To provide a custom template for the Font dialog box Create the custom template by modifying the default template specified in the www.nxgvision.com file. The control identifiers used in the default Font dialog template are defined in the Dlgs.h file.

If you need to control the dialog window-closing behavior or if you do not want the dialog to be modal, then you should directly instantiate JOptionPane and add it to a JDialog instance. Then invoke setVisible true on the JDialog to make it appear. The showMessageDialog method displays a simple, one-button dialog. The other two showXxxDialog methods are used less often. A fourth method, showInputDialog, is designed to display a modal dialog that gets a string from the user, using either a text field, an uneditable combo box or a list. Here are some examples, taken from DialogDemo. For more example code, see DialogDemo. You can easily specify the message, icon, and title that the dialog displays. Here are some examples of using showMessageDialog: With this method, you can change the text that appears on the buttons of standard dialogs. You can also perform many other kinds of customization. You must then add the option pane to a JDialog, register a property-change listener on the option pane, and show the dialog. See Stopping Automatic Dialog Closing for details. The following list describes each argument. Component parentComponent The first argument to each showXxxDialog method is always the parent component, which must be a Frame, a component inside a Frame, or null. If you specify a Frame or Dialog, then the Dialog will appear over the center of the Frame and follow the focus behavior of that Frame. The JOptionPane constructors do not include this argument. Object message This required argument specifies what the dialog should display in its main area. Generally, you specify a string, which results in the dialog displaying a label with the specified text. Choose from one of the following standard sets: Choose from one of the following values: Icon icon The icon to display in the dialog. Object[] options Generally used to specify the string displayed by each button at the bottom of the dialog. Can also be used to specify icons to be displayed by the buttons or non-button components to be added to the button row. Object initialValue Specifies the default value to be selected. You can either let the option pane display its default icon or specify the icon using the message type or icon argument. By default, an option pane created with showMessageDialog displays the information icon, one created with showConfirmDialog or showInputDialog displays the question icon, and one created with a JOptionPane constructor displays no icon. To specify that the dialog display a standard icon or no icon, specify the message type corresponding to the icon you desire. To specify a custom icon, use the icon argument. The icon argument takes precedence over the message type; as long as the icon argument has a non-null value, the dialog displays the specified icon. Customizing Button Text When you use JOptionPane to create a dialog, you can either use the standard button text which might vary by look and feel and locale or specify different text. By default, the option pane type determines how many buttons appear. The following code, taken from DialogDemo. The first dialog is implemented with showConfirmDialog, which uses the look-and-feel wording for the two buttons. The second dialog uses showOptionDialog so it can customize the wording. With the exception of wording changes, the dialogs are identical. Even if you change the strings that the standard dialog buttons display, the return value is still one of the pre-defined integers. Here is an example of using showInputDialog to create a dialog that lets the user choose one of three strings: In the Java look and feel, substituting null for possibilities results in a dialog that has a text field and looks like this: Because the user can type anything into the text field, you might want to check the returned value and ask the user to try again if it is invalid. Another approach is to create a custom dialog that validates the user-entered data before it returns. For example, CustomDialog has a getValidatedText method that returns the text the user entered. In this case, you must implement your own property change listener so that when the user clicks a button, the dialog does not automatically close. DialogDemo contains two dialogs that implement a property change listener. One of these dialogs is a custom modal dialog, implemented in CustomDialog , that uses JOptionPane both to get the standard icon and to get layout assistance. Though this dialog is rather useless as written, its code is simple enough that you can use it as a template for more complex dialogs. If you do not care to be

notified when the user closes the window explicitly, then ignore the bold code. The API is listed as follows:

Chapter 2 : Customizing the font dialog

To customize Font Dialog, handle the www.nxgvision.comrmShowing Event. In its event handler, you can substitute the standard dialog with a custom one. In its event handler, you can substitute the standard dialog with a custom one.

Download demo project - 45 Kb Introduction In one project I was involved in, we had to display font selection dialog. He is very focused on reducing support costs by simplifying user interface, to eliminate anything that might be confusing or misleading. OK, it is hard to argue against this goal. Here is what the standard CFontDialog looks like: I was starting to feel a little apprehensive since I had never once won an argument with him , so I pointed out that we could easily eliminate some of the unnecessary controls. I asked him to come back tomorrow for a demonstration. I was only partially successful, because CFontDialog provides very little in its API to facilitate customizing user interface. When the Product Manager came by next day, this is what I showed him: He thought it looked really lame, and I had to agree. The Font Style combo was still there, and getting rid of Effects controls only left a big hole in the dialog - the Sample control had not even been resized to take advantage of free space on left. Finally, the best I could do was to disable Script combo. But I had found out two things in my research into CFontDialog: Dlg; and second, there was a hook proc you could supply in your own CFontDialog-derived class. After a few days playing with CFontDialog , I knew it would be possible to customize the font dialog to look the way we wanted. Mostly this involved editing the template used for the standard font dialog, and moving the unwanted controls outside the dialog: I also had to create new controls for sample text, because CFontDialog insists on reverting to text "AaBbYyZz" when you select new font in fonts combo. To jump ahead a bit, here is final dialog: Plus, it gave us an opportunity to add some features that permitted better integration with our apps. This struct includes the lpfnHook member that allows you to set your hook proc, and the lpTemplateName and hInstance members, which must be set in order to load the customized template. After I found it, next step was to set up a separate XFontDialog. To set up XFontDialog. The advantage of this is that there is no possibility of conflicts with any other dialog resource ids in your project. Now I can use Visual Studio Resource Editor to configure controls like I wanted - moving unwanted controls to the side and marking them not visible. And finally I am able to use Class Wizard for the new class: You will notice that all symbolic names in Font. Dlg have been converted to numeric IDs. I have added two handler functions to CXFontDialog: I also add DoModal function, which is where dialog template and hook function are specified. This is where font filters if any are applied. You also need to add XFontDialog. Next, include header file XFontDialog. Now you are ready to start using CXFontDialog. Revision History Version 1. Previously I was using this technique: Added API to allow setting the sample text. Enlarged the dialog to make more fonts visible in the fonts combo. Usage This software is released into the public domain. You are free to use it in any way you like, except that you may not sell this source code. If you modify it or extend it, please to consider posting new code here for everyone to share. This software is provided "as is" with no expressed or implied warranty. I accept no liability for any damage or loss of business that this software may cause.

Chapter 3 : www.nxgvision.com FontDialog Control

This is an implementation of my class and the dialog template. The motivation for doing that: Our program stores font info in the drawing.

Download source code - 22 KB Introduction It is a common problem that we want to use Common Dialogs like the Font dialog , but we do not want to allow users to select all the properties offered by the dialogs. This article shows how we can do that, with the font dialog as an example. Background I definitely needed a font dialog without the font size property, for a drawing application which manages font size automatically, so we do not want users to be able to alter it. There are two options to select from: Rewriting the whole Font dialog. Since it offers a nice Font preview, I decided to take the second approach. Hacking If we have the handle of the window to be hacked, the whole problem is very easy. A few native function calls should be added, like: In the case of Font Size, these are and ; so, we can hide the unwanted components by calling: The sample shows three possible solutions, two of which are popular but have their own problems. Before showing the dialog, a timer is started, which finishes if the dialog is found. This approach has the following problems: What is the right interval of the timer? If a large amount of time is selected, users can see the unwanted UI elements which later disappear. FindWindow takes the title of the window as an input parameter. If it cannot find the right dialog, it will continue running till eternity. Let us suppose that the active window is our Font dialog; again, start a timer before showing it, and hope that the font dialog will be the active window when the timer first ticks. Again, the right timer interval is unknown, and if by any chance our Font dialog is not the topmost, we send commands to another application. I removed all the unneeded code from it to get a simple solution to our problem. Its ShowDialog method creates a dummy, invisible form named DummyForm. DummyForm waits until it gets activated, this is to be found in its overwritten WndProc ref Message m function. FontDialogNative is created after the right handle is found, so our unneeded UI elements can be hidden in the constructor of FontDialogNative. History This is the first version of Font Dialog hacking.

Chapter 4 : Font Dialog Box | Microsoft Docs

The hook procedure for a Font dialog box can send any of the WM_CHOOSEFONT_* messages to the Font dialog box. For more information, see Font Dialog Box. The Page Setup dialog box sends the WM_PSD_* messages if you have enabled a PagePaintHook hook procedure.

That way, you can apply this new theme to your other presentations. On the View tab, select Slide Master. Then on the Slide Master tab, select Themes. Click Save Current Theme. In the File name box, type an appropriate name for the theme, and click Save. The revised theme is saved as a. Change theme colors When you click Colors in the Themes group, the colors that you see next to the theme name represent the accent and hyperlink colors for that theme. If you change any of these colors to create your own set of theme colors, the colors that are shown on the Colors button and next to the Theme name will be updated accordingly. The Theme Colors gallery displays all the color sets from the built-in themes. As shown below, theme colors contain four text and background colors, six accent colors, and two hyperlink colors. Under Sample, you can see how the text font styles and colors look before you settle on your color combination. The colors inside the Colors button represent the theme applied to your presentation. Under Theme colors, click the button next to the name of the theme color element that you want to change. Under Theme Colors, do one of the following: Click the down arrow of the color that you want to change, and then choose a color from the main list. Click More Colors, and do one of the following: On the Standard tab, select a color. On the Custom tab, enter a recipe for a color that you want. Repeat steps 2and 3 for all of the theme color elements that you want to change. Under Sample, you can see the effect of the changes that you make. In the Name box, type an appropriate name for the new theme colors, and then click Save. If you want to return all theme color elements to their original theme colors, click Reset before you click Save. Change theme fonts Every Office theme defines two fonts â€” one for headings and one for body text. They can be the same font used everywhere or two different fonts. PowerPoint uses these fonts to construct automatic text styles. Changing the theme fonts updates all of the title and bullet text in your presentation. When you click Fonts in the Themes group, the names of the heading font and body text font that are used for each theme font appear below the theme name. As shown below, you can change the heading and body text fonts of an existing theme to meet the style of your presentation. In the Heading font and Body font boxes, select the fonts that you want to use. In the Name box, type an appropriate name for the new theme fonts, and then click Save. Choose a set of theme effects Theme effects are sets of lines and fill effects. As shown below, you can choose from different groupings of effects to meet the style of your presentation. Although you cannot create your own set of theme effects, you can choose the effect that you want to use in your own document or presentation theme. On the Design tab, in the Themes group, click Effects.

Chapter 5 : MFC, NewbieQ: Customizing a Font Dialog

Since the inbuilt Font Dialog returns a 'Not a True Type Font' Exception on selecting a Non True Type Font, I'm trying to create a Custom Font Dialog using Font-families which filter out non true type fonts.

You can customize the fonts used in the SAS windowing environment with the following resources: The font must be a monospace font. The default is dynamic, which means that the default value is determined at run time. If you change the value for the SAS. DMSboldFont resource to one that has the same style, set width, font size, point size, spacing, and number of pixels or dots per inch. DMSFont resource to one that has the same style, set width, font size, point size, spacing, and number of pixels or dots per inch. Most fonts in the X Window System are associated with an XLFD, which contains a number of different fields delimited by a dash - character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. For example, the following pattern which is the default matches any font that has a regular slant, is not bold, is monospaced, and is an ISO font: SAS excludes all fonts that have X and Y resolution values different from the current X display, all fonts that have variable character cell sizing such as proportional fonts , and all fonts that have point sizes smaller than 8 points or larger than 15 points. If this step results in an empty list, SAS chooses a generic and usually fixed font. The font with the largest point size is chosen from the remaining list. If the font resources are explicitly specified, then the auto-selection processing is not invoked. Explicitly specified values for the SAS. DMSboldFont resources take precedence over automatic selection. If you have not specified a value for the SAS. In many cases font names are aliased so that a shorter name can be used to refer to a font that has an XLFD name associated with it. The name used in determining a SAS. This allows the user to optimize or control the use of X fonts within the context of various SAS graphics applications. Cases where it may be appropriate to restrict the font search include X servers with an excessive number of fonts or X servers operating on performance-limited environments. The system font is used in most dialog boxes and menus. If this resource is not defined, SAS uses a Helvetica font. If you have not saved a font using the Fonts dialog box, but you have set the SAS. If you have not set the SAS. If no resources have been set, SAS chooses a font from the fonts that are available on your server. If the normal SAS. If SAS cannot automatically select or load a bold font, the normal font is also used for the bold font. In many cases, font names are given aliases so that a shorter name can be used to refer to a font that has an XLFD name associated with it. Specifying Font Aliases If your server does not provide fonts to match all of those that the SAS System supplies, you can use font-alias resources to substitute the fonts that are available on your system. Use the following syntax to specify font aliases in your resource file: For example, suppose that your system lacks a Palatino font, but has the following Lucinda font: Do not specify a SAS font as a font alias. A conflict can occur if you specify a font that is supplied by the SAS System as a font alias, as in the following example:

DOWNLOAD PDF CUSTOMIZING THE FONT DIALOG

Chapter 6 : Dialogs | Android Developers

Customizing the Font Dialog Box on earlier versions of Windows You can provide a custom template for the Font dialog box, for example, if you want to include additional controls that are unique to your application.

Please do not create new links to this page. Customizing Textbox Appearance Before we can explain how to customize your dialogue box, you need to design it first. The easiest way to do this is to render a mock of a dialogue box in your photo editor. Get it to look exactly how you want, text included. You can either create a box that will be chopped up by the Frame code, or you can plug in the dialogue box image directly by cropping it to be the width of your game engine. Be sure to uncomment the code by removing the mark before it. Straight out of a new game, your new framed dialogue box will need a lot of tweaking. Most everything you need is already there for you, but it is commented out. Go to your options. Find these lines of code: Margin is space surrounding the window, where the background is not drawn. The Margin of the box is for how far away the dialogue box is from the edges of the screen. This is what our PinkBox looks like with large margins padding untouched: Here is what is looks like with large padding margin back to default: Use it to define how tall the box must be at all times but it can get taller if it needs to be. This is an important property for the pre-fitted image dialogue boxes next. Trial and error is sometimes the only path here. Using pre-fitted images The other type is easier to implement because you have already placed it how you want it to look in your photo editor. To make this easy on you, do NOT crop your image to just the dialogue box in your editor. This means that your resulting image should be as wide as your game default: Just insert your file name into the style. Set all your margins to 0. When you start your game, you should see your box! The problem with the latter is that you will need to make two versions of the same image: Because the narrator when no character is speaking will look weird with an empty name box lying around. You can use the Frame function with this, too. Just copy whatever you did with the window. Styling Text To edit how your text looks, look a little further down in your options. The file containing the default font. Even though it is not included in your file, you can also change just about any aspect of the styles that you want. An easy way to find styles is to launch your game and then hover your mouse over the text you want to change. The one that you want is the last style listed. If you see this: What you want to change about that style. Again, you can find all the kinds of properties here. Numbers for sizes, strings things in quotes for filenames, Hex values for colors, etc. Like how to implement your images, there are two ways to show your character names: By default, the name is embedded into one window. Set it to True, and do this for every character you want to have a separate window. By default, it will inherit the same background as your dialogue box. The styles for the name and window are: Use positional properties like xalign or xpos and xanchor to place the name within the box you have made. Click to Continue icon A fun feature is to show an icon to the player when they need to click in order to advance the story. If you want to animate it, build the animation first, and then link to the animations call name. Also, you can have the CTC indicator embedded into the text, placed right after the last word on the screen, or you can place it in a fixed location on the screen. Here is an example of an animated CTC icon that is placed in the corner of the screen fixed: Just plug in the name in the quotes. Make sure to put position properties into the image if you want it to be fixed on the screen somewhere. See this forum thread for an example of what it looks like all put together.

Chapter 7 : XFontDialog - Customizing CFontDialog Part I: Adding Font Filters - CodeProject

The Select Font dialog box allows you to customize the font of certain kinds of text; for example, you can use this dialog box to customize the font that appears in viewport annotations. A similar dialog box is used to customize the font of the Visualization module labels and titles.

The Places Bar on the Save dialog box displays the custom folders and chosen system folders. Download the tool using the link at the end of this article and follow the instructions to install it. If the User Account Control dialog box displays during installation, click Yes to continue. Use the shortcut created on the desktop to run PlacesBar Editor. You may see the User Account Control dialog box again. The first time you run PlacesBar Editor, the following dialog box displays, encouraging you to donate. Your default web browser also opens to a PayPal page, suggesting a donation. The main interface displays showing a toolbar and two tabs. To define a custom target folder for one of the places, select the Custom check box for that place. Click the folder button to the right of the User Folders edit box to select a folder, or type in the full path to a folder. To select a system folder for a place, select a folder from the drop-down list under System Folders for that place. As mentioned at the beginning of this article, some programs use the Windows Explorer-like File Open and File Save dialog box. Microsoft Office programs and some other Microsoft programs, like Notepad and Paint use this style of dialog box. Click the Office tab. For each custom folder you want to add, enter a name for the folder in the edit box under Folder Name. Use the folder button to select the desired folder or enter the full path to the desired folder in the edit box under User Folders. You can add up to five additional custom folders. These folders are added to the Favorites, and also display as Favorites in Windows Explorer. To apply changes for each tab, you must click Save when that tab is active. To apply changes for Office programs, make sure the Office tab is active and click Save. To save changes you made on the Windows tab, you must click the Windows tab and click Save again. A dialog box displays telling you that the changes were made successfully. Once you apply, or save, your changes, on a tab, you can test those changes by clicking the Test button. The appropriate dialog box for the currently selected tab displays. If the Office tab is active, an Office program like Word or Excel opens and the Open dialog box displays. Click Cancel on the Open dialog box to close the dialog box and the program. You can revert back to the default settings for both types of File Open and File Save dialog boxes by clicking the Defaults button. You must click the Defaults button for each tab separately. A Confirm dialog box displays. To close PlacesBar Editor, click Exit. Customizing the File Open and File Save dialog boxes with custom folders can improve your productivity, especially if the folders you use most often are several layers deep in your folder structure. Download PlacesBar Editor from http:

Chapter 8 : Customize or create new styles in Word - Office Support

Tip: User-customized settings are marked in the Change Fonts dialog with a small User Set icon to the left of the listing. To return a customized setting to the default font, single-click that item, then click Undo at the bottom of the dialog.

There are multiple preset views you can choose, like one that displays the most recent emails only. Also on the View tab, you can select Message Previews and choose Off to turn the preview off. Or, choose either 1, 2 or 3 to view that number of lines of the message text beneath its header. You can configure any of these options either for the current folder or for all mailboxes. Click View Settings for options for customizing a view, such as adding columns or rearranging their order. The Arrangement options let you sort emails by Date, Subject and so on. Then type a name for the view, and specify which folders it can be used on and by whom. In the future, return to this view by selecting Change View and selecting your saved view. Maybe you want to just change the size to make the font bigger or smaller, or change the font to one you like better. To change the font settings for your message list, open Outlook and click on the View tab. On the "Conditional Formatting" dialog box, click Add button to add a new rule. Change the Font, Font style, and Size, and select other settings such as Effects and Color, as desired. Click OK when you have made your changes. A dialog box displays, warning you of that fact. The font for all parts of each message in the message list except for the excerpt of the message text is changed to the font and size and other font settings you selected. You can delete the rule you created to go back to the default settings, or you can deselect it in the list of Rules for this view on the Conditional Formatting dialog box. Enlarge font size in the Reading Pane Using a high resolution has the benefit of a sharper display which is more relaxed for your eyes. The downside of this is that your font might become really small and hard to read which is anything but relaxed for your eyes. In Outlook and Outlook , you can also zoom via the zoom slider in the bottom right corner. The zooming factor set via this slider or by scrolling is not persistent. To set a default zooming level, you can use VBA to set this setting. Using this feature is not supported by Office support team. Together with your modified Plain Text font settings, the larger font size will always apply. Via the Infobar, which will display on top of a converted message, you can easily change it back to HTML format when needed like for some special layout or newsletters. To do this; Windows 7 and Windows 8 Right click on an empty spot on your Desktop and choose Screen resolution. Click on the blue text link: Make text and other items larger or smaller. To set a custom DPI value: Windows 8 and Windows 8. In the new dialog that pops up you can directly choose a predefined value to make the fonts bigger. You can also type any percentage you like. Check the "Mark items as read when viewed in the Reading Pane" checkbox. Adjust the number of seconds you want the message to appear in the Reading Pane before Outlook marks it as read. The default is 5 seconds. After that, while you preview messages in the Reading Pane, only the emails that appear there longer than 30 seconds will be marked automatically as read. Write your own rules for how messages appear Outlook has a new way of indicating unread messages. This blue is applied using conditional formatting. However, you can change both the color and font. Better still, you can write your own rules and format your emails using colors of your choice. You can assign a certain color to emails based upon who sent them, or upon which words appear in the subject line. You will see the Unread Messages rule and the blue color. Here you can change the font or color as you wish. Start by typing a name for the rule in the Name box. When the Filter dialog opens, select the options that describe which types of emails will be formatted with your new settings. The order in which you select these options determines the order they appear in the To-Do Bar. For example, Task, selected first, is at the top. The second item, People, appears below that and the third, Calendar, appears at the bottom of the bar. However, the To-Do bar no longer functions as it did in earlier versions of Outlook. Regardless of how wide the bar is, you only see a one calendar month.

DOWNLOAD PDF CUSTOMIZING THE FONT DIALOG

Chapter 9 : Customizing Common Dialog Boxes | Microsoft Docs

Customizing dynamic font style and size in a custom dialog box. Ask Question. up vote 0 down vote favorite. I created my own custom message/dialog box already in xaml.

If you want to create a custom Excel format with some other currency, follow these steps: Open the Format Cells dialog, select Currency under Category, and choose the desired currency from the Symbol drop-down list, e. Switch to Custom category, and modify the built-in Excel format the way you want. Or, copy the currency code from the Type field, and include it in your own number format: How to display leading zeros with Excel custom format If you try entering numbers or in a cell with the default General format, you would notice that Microsoft Excel removes leading zeros because the number is same as 5. But sometimes, we do want , not 5! The simplest solution is to apply the Text format to such cells. Either way, Excel will understand that you want any cell value to be treated as a text string. As the result, when you type , all leading zeros will be preserved, and the number will show up as If you want all numbers in a column to contain a certain number of digits, with leading zeros if needed, then create a custom format that includes only zeros. As you remember, in Excel number format, 0 is the placeholder that displays insignificant zeros. So, if you need numbers consisting of 6 digits, use the following format code: If you are entering phone numbers, zip codes, or social security numbers that contain leading zeros, the easiest way is to apply one of the predefined Special formats. Or, you can create the desired custom number format. For example, to properly display international seven-digit postal codes, use this format: For credit card numbers with leading zeros, apply this format: For example, to display percentages as integers, use this format: As the result, the number 0. To display percentages with 2 decimal places, use this format: Exactly which way Excel displays the fraction is determined by the format codes that you use. To round fractions to a specific denominator, supply it in your number format code after the slash. For example, to display decimal numbers as eighths, use the following fixed base fraction format: To implement this alignment in your custom format, use the question mark placeholders? To enter a fraction in a cell formatted as General, preface the fraction with a zero and a space. Create a custom Scientific Notation format To display numbers in Scientific Notation format Exponential format , include the capital letter E in your number format code. To show negative values in parenthesis, simply include them in the second section of your format code, for example: To line up positive and negative numbers at the decimal point, add an indent to the positive values section, e. This can also be done in your custom Excel number format. As you remember, the zero layout is determined by the 3rd section of the format code. So, to force zeros to appear as dashes, type "-" in that section. General; -General; "-" To turn zeroes into blanks, skip the third section in the format code, and only type the ending semicolon: The commonly used indent codes are as follows: To indent from the left border: For example, to indent positive numbers and zeros from the right and text from the left, you can use the following format code: To move values from the cell edges by more than one character width, include 2 or more consecutive indent codes in your number format. The following screenshot demonstrates indenting cell contents by 1 and 2 characters: Change font color with custom number format Changing the font color for a certain value type is one of the simplest things you can do with a custom number format in Excel, which supports 8 main colors. To specify the color, just type one of the following color names in an appropriate section of your number format code.