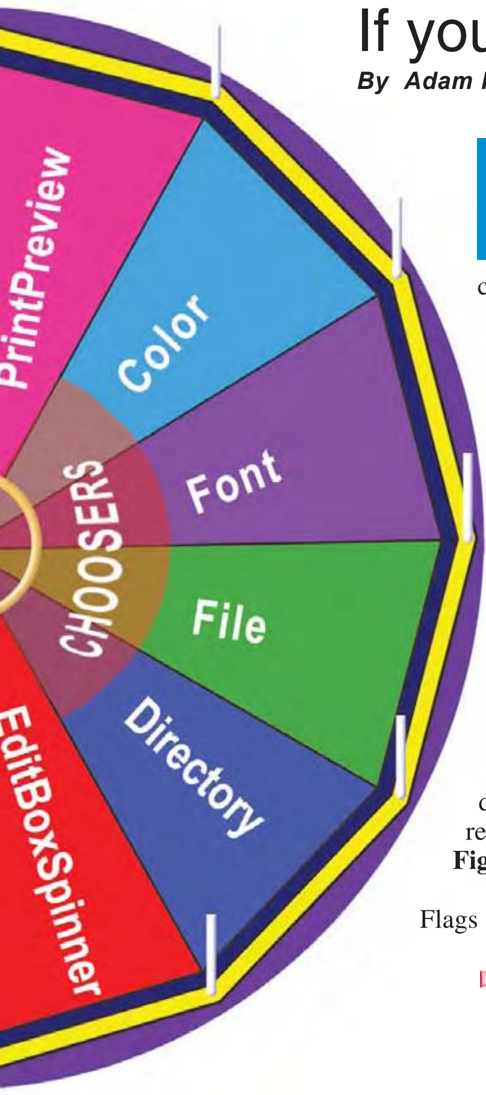


If you Have Choices, we Have Choosers

By Adam Hawthorne

User interfaces continue to evolve, providing more customization and allowing users to tailor their environments to suit their own needs. In many applications, selecting a font style or color or even a file is as easy as clicking on the selection from a list of options. BBj® 7.0 introduces new controls, called choosers, with which developers can offer similar choices to their users who expect real-time interaction with their application. This article introduces the various new BBj chooser controls and shows how they can enhance any GUI business application.

File/Directory Chooser

BBj developers are no strangers to using a file chooser dialog. The new BBjFileChooser control is a highly configurable SYSGUI-based complement to the existing FILEOPEN() and FILESAVE() functions and the 'FILEOPEN' and 'FILESAVE' mnemonics of the SYSWINDOW. For developers who only need a simple blocking file choice, these solutions will suffice, but to add a custom preview panel to a file chooser dialog or provide additional configuration options on the dialog window, it is only possible with the BBjFileChooser.

For example, a developer could put a file chooser control on a window with pre-existing controls and behavior. Then, allow the BBjFileChooser to interact with standard READ RECORD or PROCESS_EVENTS event loop to avoid interrupting program flow or to remain responsive while the dialog is visible. The developer could also add the file chooser to an existing form to provide real-time responses to file selections, or augment the UI with an Explorer-like directory tree. **Figure 1** shows a sample of the file chooser in action.

Flags on the BBjFileChooser gives developers a wide variety of options, such as:

- Allow the previously mentioned directory selection in an Explorer-like fashion by simply adding the flag `($0008$)` to the addFileChooser method, turning a standard file chooser into a much more efficient directory chooser dialog as shown in **Figure 2** on the next page.

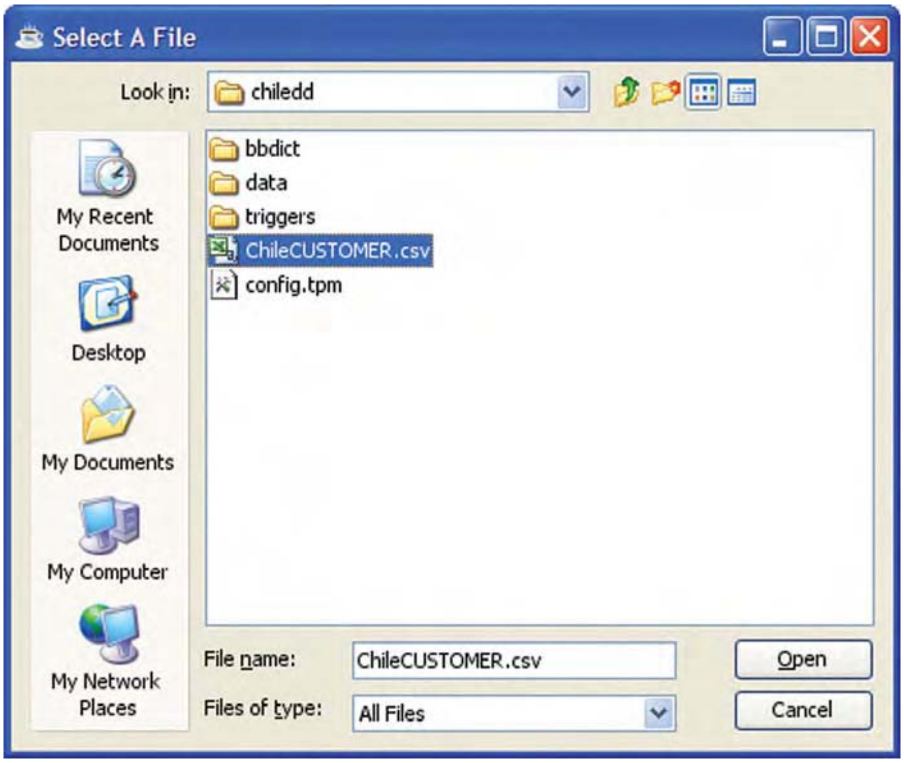


Figure 1. Sample file chooser in action

- Use the default behavior of an OPEN dialog, or specify the SAVE flag `($0100$)` to achieve standard internationalized labels for the file chooser and its buttons.

- Browse on the server's filesystem by default, or allow users to browse on their own computer's filesystem using the client-browsing flag `($0004$)`.

The BBjFileChooser also provides methods to interact with the user's view of the filesystem, flexible glob-based (pattern matching) file filter configuration, and full single and multiple file selection support.

continued...

Adam Hawthorne
Software Engineer

BBjFontChooser

The BBjFontChooser provides a fully functional chooser for fonts. Rather than building a dialog and performing the countless steps needed to determine the list of fonts; populating list controls with the fonts, sizes, styles, and other tedious tasks; developers can use the ready-made control that will correctly follow the locale settings of the client machine. It also automatically produces a list of fonts, sizes, and styles, and shows a preview of the font without any programmer interaction, although the programmer can set the text to preview the font (see **Figure 3**).

The BBjFontChooser seamlessly integrates with BBj's Visual PRO/5®–compatible treatment of fonts, specifically respecting the settings of the 'SCALE' mnemonic. For those who do not have a dependency on compatibility or simply do not want this level of compatibility, the `$0004$` flag turns off the compatibility mode.

BBjColorChooser

The BBjColorChooser delivers a thorough treatment of the color spectrum. With three ways to make color choices, even the most detail-oriented graphics expert will have no cause to complain. The color chooser allows users to select colors based on standard swatches as well as RGB and HSB color spaces as shown in **Figure 4**. While including all the standard chooser features, the BBjColorChooser also adds the ability to hide the preview panel and implements the new drag and drop architecture to drag colors from one application to another.

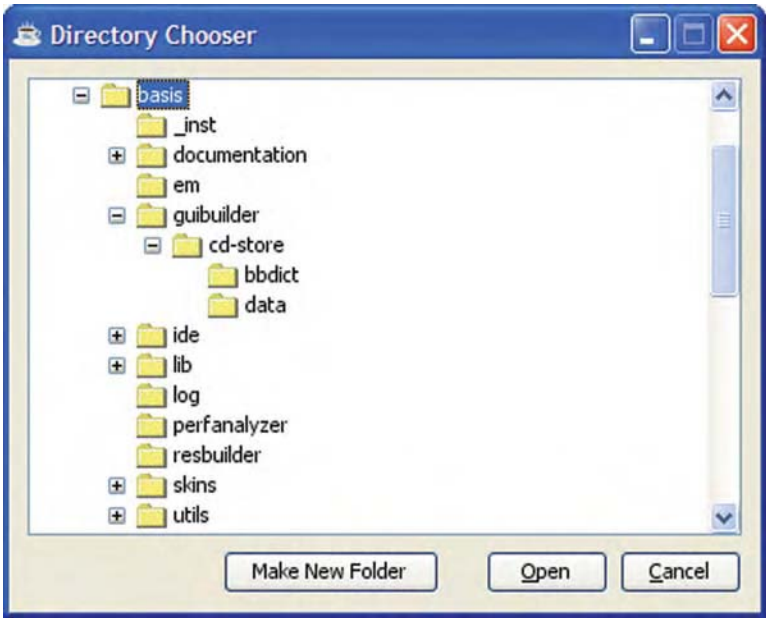


Figure 2. Sample directory chooser dialog

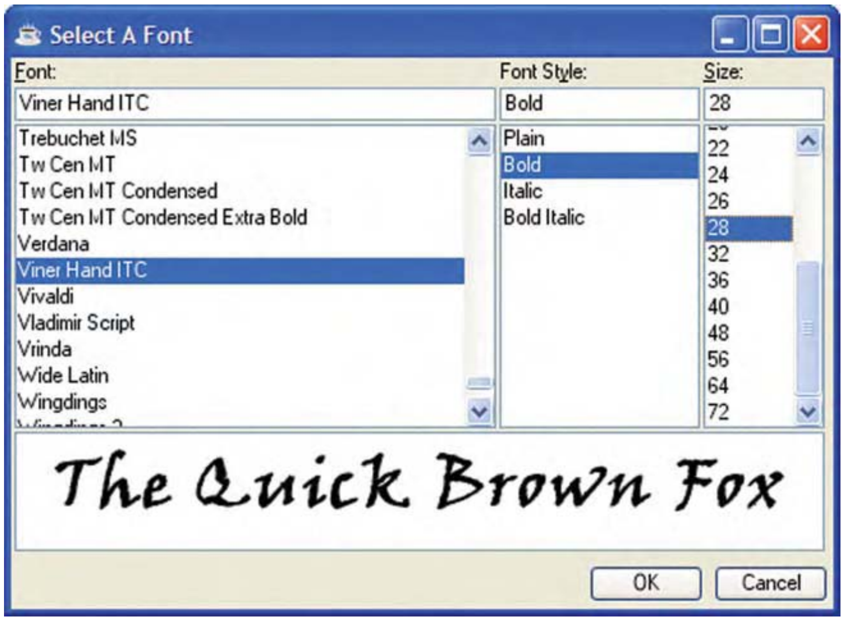


Figure 3. A dialog displaying the BBjFontChooser

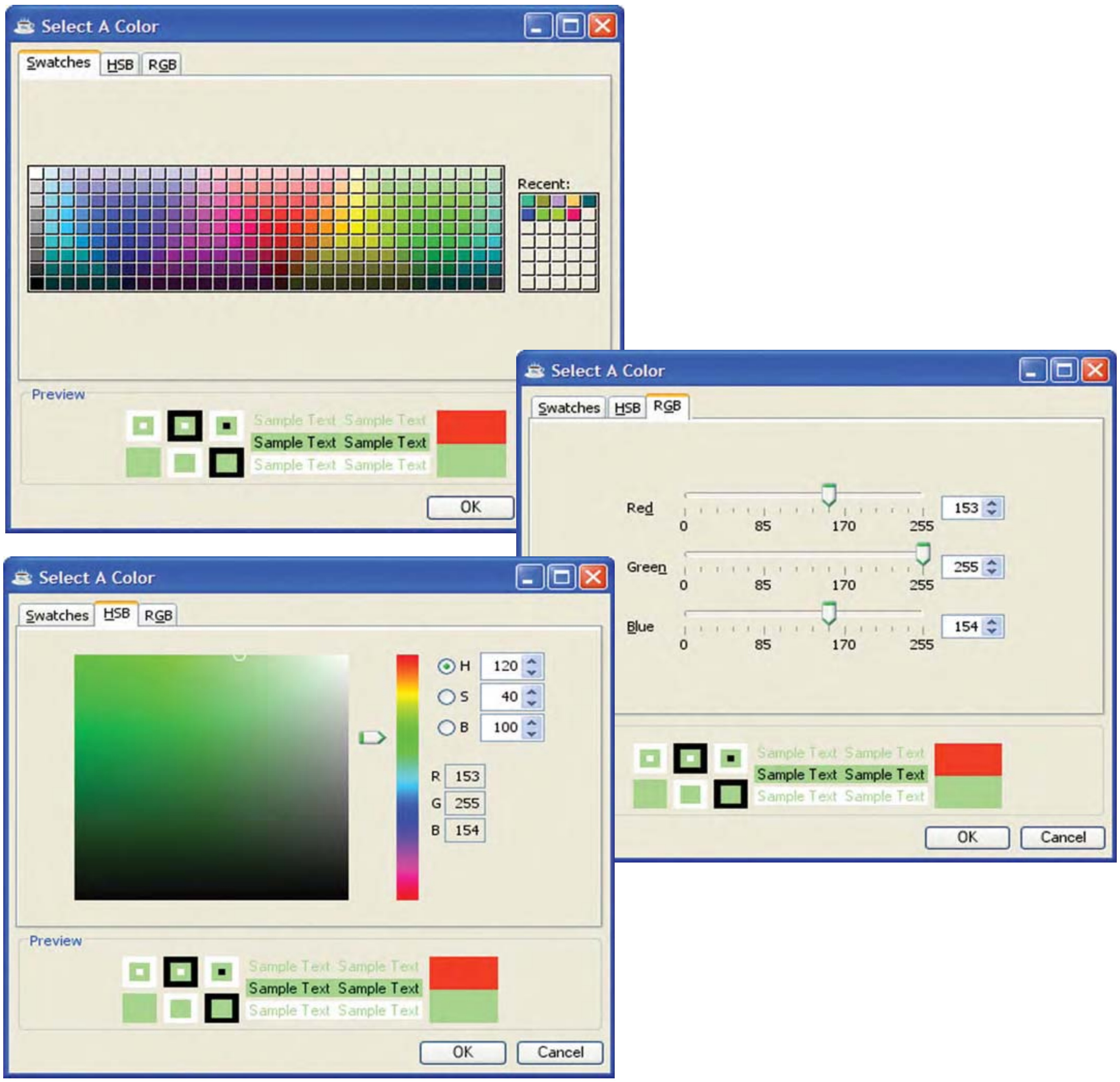
Working with Control Buttons

The new chooser controls provide shared functionality regarding the standard control buttons. For example, when the user selects a control button like [OK] or [Cancel], each chooser fires an Approve or Cancel event. To mimic a traditional chooser dialog, add a chooser control to a window that has a `$00080000$` dialog behavior flag, then set the chooser's control id to `1` like an [OK] button, and register for the Approve and Cancel callbacks. The chooser will correctly map the [Enter] and [Esc] keys to the included buttons and fire the events when appropriate. When the user completes the selection, the program can respond to the event by hiding or destroying the dialog window.

Multiple Choices

All of the chooser controls offer extended functionality and customization capabilities. For example, developers can customize the control button text if desired. The BBjFileChooser allows setting the text of the [Open] or [Save] button and the BBjFontChooser and BBjColorChooser allow setting the text of both the [OK] and [Cancel] buttons. It is even possible to hide the control buttons completely, making the choosers a perfect choice for inclusion in a palette.

continued...



Red 153
Green 255
Blue 154

H 120
S 40
B 100
R 153
G 255
B 154

Figure 4. Color selection options

Chooser Interaction

Each chooser has an approveSelection and cancelSelection method to allow programmatic activation of the associated buttons. Use these methods to facilitate combining event-handling code and the response to in-program actions or use them to decouple the event from the response. Lastly, all of the chooser controls provide Change events that respond to real-time selection changes within the chooser. This enables the program to react instantly to any choice the user is considering, and is often used to provide previewing capabilities.

Summary

With BBj chooser controls, BBj developers can now provide behaviors and customizations easily and effortlessly, looking back on the bygone days of recreating standardized dialogs from scratch, and implementing and testing complicated controls and their interaction. Today, they can toss a control on a window and respond to the events, resting easy knowing that these controls conform to system standards with correct localizations. Customizing the user interface is just a matter of extending the concept of a 'standard dialog' by creating a form with a chooser and various other controls. Now that BBj offers so many choices, developers can start choosing!