BBj 2.0: More Than One Way To Skin An App

By Nick Decker

With the introduction of BBj® 2.0, your GUI BBj programs are now skinnable! Skinning is all the rage in a great deal of today's computer software - from the early days of skinnable media players to today's skinnable operating systems. If you haven't heard of skinning before, the burning question is, "What the heck is skinning, and how does it affect my applications?"

Put simply, skinning allows you to change the appearance of an application or even your operating system itself. Automobiles are a good analogy. Most cars are basically similar in that they all usually have tires, doors, windows, headlights, bumpers, brake lights, etc. However, most cars on the road today look dramatically different from one another because all of these attributes and components vary from car to car. They can vary in color, size, shape, location and even availability.

This shows four different skins used to achieve different looks for the same BBj application. The process of skinning allows you to quickly design and brand your application's interface.

The same concept can be applied to your computer applications. Their individual components, such as buttons, checkboxes, scrollbars, menu bars and even colors, can be changed in these same ways via the skinning mechanism. Thus, skinning can have a dramatic effect on what the application looks like, and more importantly, even on how it functions. When designing a skin for your application, you can opt to make subtle changes or go all-out and radically affect its appearance - it's up to you!

Through skinning, you can effectively brand your product and give it a customized look and feel that separates it from the rest of the crowd. You can create a skin that uses your company colors and logos, for example.

You could even create several different skins that brand the same application for different customers! For those that may doubt the psychological effect this can have,

think back to what Microsoft Windows 3.1 looked like compared to Microsoft's latest offerings. The Windows 3.1 style looks very dated and second-rate with its use of large fonts, lack of 3-D controls and elementary color schemes. Their current operating systems look much more advanced, robust and capable. But in addition to determining the application's appearance, skinning allows you to determine an application's interface. You can opt for a simple, minimalist approach or present the user with an advanced, complex interface. In these ways, a skin can dramatically affect an application's ease of use.

How does BBj fit into the skinning paradigm? Using a product called Skin Look and Feel (SkinLF) that skins any Java Swing-based application, we've made it possible to skin BBj 2.0 itself. This means that you can skin the various components of BBj, from the IDE to the interpreter. BBj 1.0 allowed you to change the look and feel of the BBjIDE, but you were limited to that component and to the Windows, Metal or Motif look and feel that were part of the Java Runtime Environment. But with BBj 2.0, once the interpreter is skinned, all of the GUI programs that run in that instance of the interpreter will also be skinned! This allows you to quickly and easily change the look and feel for any and all of your GUI applications. You can also choose different skins for separate BBj programs if you'd like. And using SkinLF, you have hundreds of look and feel choices, or themes from which to choose. You can even design your own.

Instead of requiring a proprietary theme format, SkinLF was written to take advantage of existing GTK+ (the GIMP Toolkit) and KDE (K Desktop Environment) themes. These have been popular for years with UNIX users who have been using them to skin, or customize, the look and feel of the X Window System's graphical interface. Today, they can be used to skin BBj too, regardless of platform.

A complete skin for BBj consists of a combination of GTK+ and KDE themes. The KDE theme provides information regarding the basic colors, the window frame for child windows and the sounds associated with various events. The GTK+ theme provides all of the widgets that are used in GUI programs, such as buttons, checkboxes, listboxes, etc. The theme paradigms for GTK+ and KDE are very similar: they consist of a resource file describing the theme and a collection of several graphic images to be used in the skin. The resource file is an ASCII file and can range from a couple of dozen lines for a KDE theme to almost 1,200 lines for a typical GTK+ theme. The resource file is responsible for describing every skinnable element to the graphics engine so that it can determine the end result of the graphical interface. For example, the following is an excerpt from a gtkrc resource file that describes a typical button control:

```
style "button"
{
  engine "pixmap"
    {
    image
      {
      function        = BOX
        recolorable     = TRUE
        state           = PRELIGHT
        shadow          = OUT
        file            = "button_hovered.png"
        border          = { 5, 5, 4, 4 }
        stretch         = TRUE
      }

        ... More button definitions ...
    }
}
class "GtkButton" style "button"
```

This section of the gtkrc file instructs the theme engine to use a pixmap (external graphic file), called button_hovered.png, when rendering a button control. It contains other critical information, such as the state of the button (PRELIGHT, meaning when

the mouse cursor is hovering over the button), whether to stretch or tile the pixmap, and what the fixed left, right, top and bottom borders are for the pixmap, as defined in pixels. Because you can specify the pixmap and all of the button's attributes, you can make your application's buttons round, oval, square or any other shape. You can determine what color they are, how they react and change on a mouse-over event, and what they look like when pressed. The theme engine will take all of this into account and render the button, just as you described in the theme file, when your application is run.

Between these two resource files, you can customize the appearance and behavior of more than 100 different graphical elements that together determine the look and feel of your application. With the freedom provided by the theme engine comes a potential for complexity, however. The basic technology is easily learned, but it is possible to get bogged down in the intricate details that comprise a skin. Thus, the skinning process is sometimes referred to as an art, as it requires someone (or a team, in many cases) skilled in design, user interfaces, illustration and programming to create an appealing and functional skin. The rewards can be numerous though, whether you're giving an older application a face-lift or creating a brand- new user interface for your next-generation application.

For further information on skinning, visit the following sites.

Skin Look and Feel Web page:
www.L2FProd.com/software/skinlf/index.html

Gateway to skinning in the X Window environment:
www.themes.org

All about skinning the GTK+ toolkit:
http://gtk.themes.org

All about skinning the KDE Window Manager:
http://kde.themes.org