T Minus 0 – BASIS LaunchDock Blasts Off

By Nick Decker

Each time another BASIS conference rolls around, we start wracking our brains to come up with a new way of organizing the numerous demonstration programs into the workbench application. One essential design consideration is that the workbench program should incorporate some of the latest BBj® features, which is why some of the past workbench applications took advantage of constructs like MDI (multiple document interface), expandable palettes, versatile toolbars, and more. Given that BBj now supports Java GUI objects on the client's machine, we decided to make BBj's new ClientObject capability the centerpiece of the workbench application.

ClientObjects Break New Ground

In addition to a radically different interface, we wanted to include plenty of 'eye candy' to the workbench program. After all, form is often just as important as function especially when you are showing off. Just ask all those Olympic gymnastic judges.

In order to accomplish these goals we did something that has never been possible in BBj up until now – we created the top level window as a semi-transparent, alpha-blended window replete with a drop shadow, rendered based upon the contents of a graphics file (**Figure 1**).

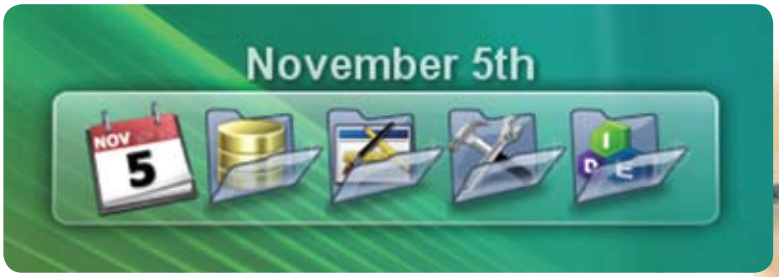


Figure 1. The BASIS LaunchDock menus for the November 5th session of TechCon2007

To decrease our development time dramatically, we took advantage of third party freeware Java libraries that use native code to create a transparent JWindow. This is significant for a couple of reasons. First, the resultant window is a BBj ClientObject, meaning that it is a Java-based GUI component that exists in the Client's JVM and displays on the Client's machine. Second, since the window isn't a BBjWindow and it exists in a completely different JVM than the BBj Interpreter is running in (which is on the server machine), we have to do a little extra work to connect the Java window to the BBj program so that we can interact with it successfully.

A Closer Look at the Technologies Involved

The LaunchDock is comprised of several nested constructs and technologies. At the core of the LaunchDock is a Java program that handles all of the user interaction. This includes the GUI components, such as the Java-based transparent window, as well as the events - including mouse clicks and drags. One of the first things the Java program does is create the transparent window. To do this, it instantiates a TransparentWindow object. The TransparentWindow object extends a typical Java JWindow, but employs code from the Java Native Access (JNA) libraries to make the resultant window transparent. The transparent window serves as a blank canvas on which the Java program paints the background image and icons. Because the window is transparent, you can see other windows and the wallpaper underneath the window.

Nick Decker
Engineering Supervisor

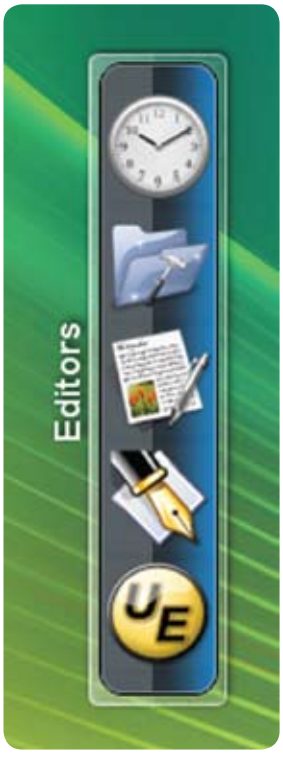
The LaunchDock Java program takes advantage of other free Java libraries such as the SwingX collection of components. The TimingFramework library facilitates the timing and animations that the LaunchDock uses for icon hover effects. In addition to adding some pizzazz to the LaunchDock, the animations enhance the user's experience. They provide visual feedback to the user in response to events such as zooming in on an icon when the mouse hovers over it or bouncing an icon after the user clicks on it. The SwingX libraries also enable the LaunchDock to provide additional effects for the user-supplied icons. This means that the LaunchDock can take a stock icon and enhance it on the fly by adding a drop shadow or a reflection as shown in **Figure 2**.

continued...

Figure 2. The LaunchDock displaying real-time reflections of the icons

Passing User-Generated Events Back to the BBj Program

Because the LaunchDock's window is a Java-based JWindow instead of a BBjWindow, we have to create a way for LaunchDock to notify the BBj program that instantiates it whenever the user interacts with the dock. If we had used a standard BBjWindow, we could easily register for callback events on the window to find out when the user clicked on it or one of the icons. But in this case, the LaunchDock's window is a JWindow that exists in a different JVM on the client's machine, so the LaunchDock utilizes Java's Listener mechanism to accomplish this communications link. The LaunchDock Java program exposes methods that allow other objects to add themselves to a collection of listeners, and then relays important user-generated events such as mouse clicks to those registered objects. The result is that the BBj program receives custom events from the LaunchDock and acts upon those events in the appropriate manner, whether it is launching the desired program or showing the About Box.



Customizing the LaunchDock

While we primarily programmed the LaunchDock to serve as the TechCon2007 workbench application, we knew that if we designed it in a flexible and extensible fashion, it could be a powerful tool, extending its usefulness well beyond the conference. With that in mind, we came up with the concept of an Archive object to define the contents of the LaunchDock. In simple terms, an Archive is a hierarchical collection of folders, programs, and icons that describes all of the programs that the LaunchDock should display. This makes it easy to use the LaunchDock for several applications - just pass it one Archive for the BASIS demos and another to launch a completely different set of programs. Users can even create and manage their own Archives with our custom Archive editor. We have exploited this capability at BASIS and use the LaunchDock for our in-house program menu. BASIS employees are also free to customize their copy of the Java Web start deployed LaunchDock by adding their Windows, Linux, or Apple Mac office productivity application programs to their individual copy of the Archive.

In addition to using the LaunchDock for various purposes, programmers can customize its appearance via several built-in methods. Dozens of aspects are configurable such as the graphics used for the background as well as the sizing and placement of the icons. Developers are able to tweak each parameter to alter the LaunchDock's appearance and behavior.

Users have their preferences as well, so the LaunchDock's popup menus enable the end user to set the location and orientation of the LaunchDock. Users can then decide what works best with their desktop – whether they have the LaunchDock docked to the top, bottom, left, or right sides of the screen as shown in **Figure 3**. If that still does not fit their needs, they have the option of undocking it and moving it anywhere on the desktop by dragging it with their mouse. Moreover, if the dock needs to be accessible all of the time, selecting the On Top option prevents other windows from obscuring it, thus causing it to be visible all of the time.

Figure 3. The LaunchDock in a vertical orientation

Summary

Without regard to the technology employed, the primary goal for any conference workbench application is to come up with an intuitive way to arrange and separate the demo programs and still make it easy to navigate to a particular demo. The BASIS Demo LaunchDock takes advantage of BBj's new ClientObject capability to do exactly that – in a single compact user interface. Do not let the simplistic nature of the interface fool you though, as there are countless advanced technologies utilized under-the-hood to make this the most sophisticated workbench program ever.

To see the LaunchDock in action, download the demos with BBj from www.basis.com/products/bbj/download.html and select "Demos" in the Optional File section After completing the installation, select **BBj > Demos > LaunchDock** from the BASIS folder to run the demo.