Super-Charged SYSGUI

By Adam Hawthorne

With the release of BBj® 5.0, BASIS introduces a new and improved version of the SYSGUI subsystem. The new SYSGUI is a rewrite of the SYSGUI code by which BASIS engineers address a number of issues in the original SYSGUI implementation. Some of these problems include *latency* and bandwidth performance issues with BBj ThinClient, the size of the client jar files, the speed of calling object methods, and overall stability. The key to the resolution of many of these issues is the new SYSGUI.

This article explains some of the design elements of the new SYSGUI and how they affect BBj developers and their applications.

Latency – The ping time between the server and the client. Across the Internet, ping times are typically in the range of 1/10th to 1/5th of a second – 100 to 200 milliseconds – or about 1000 times slower than in a LAN environment.

Maintenance

Maintenance is one of the most overlooked aspects of any application development, especially under tight deadlines. Ongoing maintainability was one of BASIS' major design goals. BBj developers benefit from this design decision through improved stability, faster turnaround time on reported problems and fewer recurring defects. This means a faster time-to-market for BBj developers, and a more stable platform on which to develop GUI applications. As one example, this rewrite eliminated the need to use the same character encoding between the client and the server; character-encoding conversions now occur automatically.

One significant factor that affects maintainability is excessive dependencies between various modules. A change in one module should not affect other modules. Each control in the new SYSGUI is an independent object that has limited interactions with other controls. Because of this, BASIS can test improvements and new features in each control more easily, with more certainty of correctness and with less possibility of adversely affecting other controls. As a result, BBj developers will experience an increase in the stability in subsequent versions of the product, thereby giving them the ability to use new features more quickly.

Server Orientation

The second major design criterion was to perform as much processing as possible on the server. In the original implementation of SYSGUI, most queries on the SYSGUI channel or GUI objects required the server to query the client and wait for the client to respond with the answer. This round trip process slowed down execution speed significantly, particularly in high-latency environments.

In the new SYSGUI, if an application program requests GUI information that the user interface cannot affect, it handles the request entirely on the server side. This allows BBj GUI programs to run more quickly and efficiently. For example, a BBj program might query the SYSGUI device, and then perform a database query. If the server can provide the answer to the SYSGUI query without communicating with the client over the network, it can proceed with the database query immediately. If the server must query the client for some piece of information, the application must wait for the client before it can continue with the database query. For a moderate latency connection, this delay can be 100 milliseconds or more.

The server-oriented approach also allows BASIS to reduce the amount of code in the thin client jar file. Smaller jar files lead to faster application deployment.

Another benefit of the server-oriented environment is greater application responsiveness. While BBj programs must still wait for some elements of the user interface, the overall reduction in the client/server network traffic means that the application is much more responsive to the user.

continued...

Adam Hawthorne
Software Engineer

Automatic Batching

BASIS also implemented automatic batching as an integral part of client/server communications in the new SYSGUI. In previous versions of BBj, the developer could optionally implement batching in order to optimize network traffic between the server and the client. Now, due to the server-oriented architecture with more knowledge on the server about which operations require a client response, the SYSGUI is able to manage all of this automatically. It buffers multiple commands and sends them to the client in mini-batches, several times per second. This reduces the number of network packets sent to the client and decreases the amount of redundant information in each network packet. The SYSGUI manages this optimization automatically, eliminating the need for the application developer to write special code.

This approach also leads to an increase in parallelism between the server and the client. Because the server does not have to wait for as many client responses, it spends more time doing real work. It was possible to achieve a similar effect in earlier versions of BBj through the performance setting to disable "UI Ack Backs," but a side effect was that the application would not receive certain errors. BBj 5.0 now reports all errors, so developers no longer have to choose between "development" and "production" settings.

For optimized performance, consider structuring an application program's initialization section like this:

1. Create all windows and controls.

2. Perform any non-GUI-related setup (open files, create data).

3. Start processing events to begin working with the client-side GUI.

The benefit of this approach is that the client-side GUI continues to instantiate the windows and controls while the server-side continues opening files and creating server-side data.

Performance Enhancements

In some cases, the original implementation of SYSGUI accepted commands even when it was possible to determine that they would fail or generate incorrect results at runtime (for example, invalid accelerators in the 'SETMENU' command). The new SYSGUI produces more readable error messages and enforces stricter syntax rules. This enables developers to identify programming errors immediately, and eliminate them before releasing the application to customers.

Developers will notice that the new SYSGUI repaints GUI screens less frequently, loads forms faster, and performs grid processing more rapidly. The new grid generates fewer notify events because it combines multiple notifications whenever possible. BASIS implemented these enhancements to ensure that GUI programs run more efficiently in BBj 5.0.

Feature Enhancements

The SYSGUI subsystem in BBj 5.0 also introduces a number of new features. Control and form validation (see page 8) enables the BBj developer to verify that fields contain valid data before allowing the user to interact further with the application. A new BBjImage object gives developers more control over image reuse and reduces network traffic for reused images. Additions to the BBjAPI provide object methods for features previously available only through SENDMSG(), CTRL(), or mnemonics. BBjGrid, for example, implements more than two dozen new object methods.

Summary

BBj 5.0 marks the first release of the implementation of the new SYSGUI. GUI developers can look forward to a myriad of benefits, including improved performance and new features, with the release of BBj 5.0 and the introduction of the new SYSGUI.