Let BUI Put Your App in Touch With Your Users

There are a great number of reasons to write your next BUI web app for a smartphone, but probably the most compelling one is that you and your users can have immediate access to the web app anytime, anywhere. Today's smartphones have fast browsers with optimized JavaScript and hardware graphics rendering that are more than capable of running BUI apps with ease.

Creating a BUI version of your desktop app is a definite advantage as users can continue to work and be productive outside of the office – in line at the grocery store, walking on the treadmill at the gym – you name it! A full-blown BUI app is not limited to a simple static report, but is capable of being a fully immersive, and more importantly, interactive experience where the user can make payments, record receipts, adjust sales priorities, and even reroute shipments with just a few taps of a finger.

This article is a simple guide to help developers fearlessly take the plunge and put their apps in touch with their users!

Targeting the End User's Device

Now that we are sufficiently motivated <grin>, our first inclination would be to fire up the BASIS IDE and get our fingers dirty by starting to write some code. But before we open the SYSGUI device, we should stop for a minute and think about how the smartphone differs as a client compared to the traditional desktop computer. There are numerous differences, but the prominent ones are:

- Smaller screen
- Slower CPU
- Touch interface instead of a mouse
- Higher network latency and lower bandwidth
- Variable screen real estate and orientation
- Tiny simulated or physical keyboard
- No cursor
- No hover effects
- No right clicks

By Nick Decker
Engineering Supervisor

Reading through the list may have dampened your enthusiasm a bit, but none of these are deal breakers. Instead, they are simply challenges to keep in mind while designing our web app. Some of them may even lead to relatively simple code solutions with dramatic and satisfying results. More importantly, they underscore how the differences in the target device encourage us to thoughtfully plan and structure our app so that end users can navigate the app in a simple and effective manner.

Designing for Smaller Screens

Web designers have been dealing with many of these same issues over the years, so we should take a cue from them and write a web app that is well suited for a smartphone. To illustrate the point, take a look at the comparison of the Amazon website when viewed on a desktop with it viewed on an iPhone. **Figure 1** shows the difference between viewing the desktop site on the small screen versus the site that Amazon optimized for a mobile device.

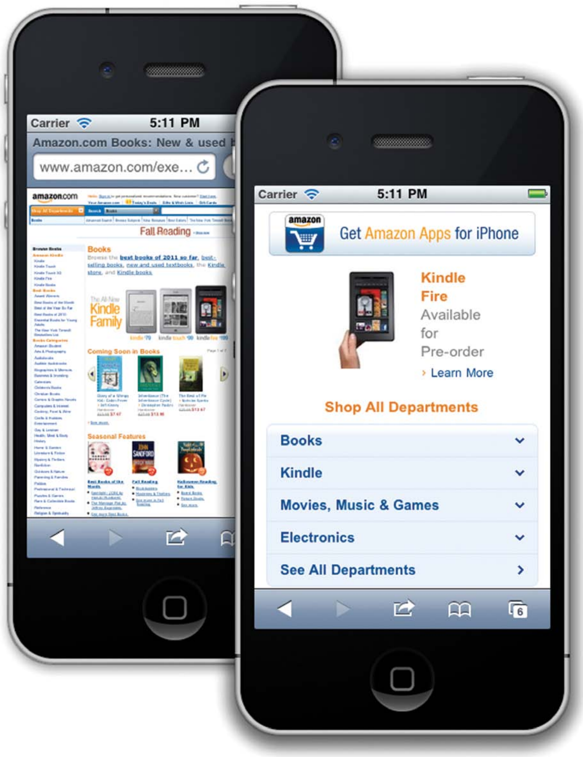

Figure 1. Desktop (left) and mobile (right) versions of the Amazon website on iPhone

This comparison illustrates some noteworthy differences from which we can learn some things of value.

- **The mobile version of the website optimizes the limited screen space by telling mobile Safari to hide its Address Bar.** This is a smart move, and thankfully it is one that BBj® takes care of automatically for the BUI developer. BBj does this without any effort on the developer's part in order to maximize viewing area of the application.
- **The amount of information displayed on each page is strikingly different.** The traditional desktop page has dozens of tiny links that are impossible to accurately tap with your finger, regardless of how hard you may have concentrated and aimed. This highlights a couple of the most important guidelines:
  - Limit screen content to only that which is critical
  - Increase the size and spacing of controls

Because a smartphone's screen is both physically smaller and has fewer pixels, we should restrict screen content to information that is absolutely critical. Attempting to cram a full-size application onto a smartphone will almost assuredly lead to a frustrated user that spends more time trying to interface with the app instead of actually using it and getting things done.

Putting the User in Control

The good news is that we do not need to conduct usability studies and put our app in front of multiple focus groups to determine how best to construct an efficient user interface – all of that work has already been done for us. The takehome points are few and pretty easy to implement. In a nutshell, we should make tap-able elements such as buttons large enough and allow for enough room between > >

screen elements to avoid finger mis-taps. Fingers are fast, but interestingly enough, they are not usually as accurate as a mouse. So a user that has no problem clicking on the typical button that is 23 pixels high in a desktop application will have difficulty tapping on the same control on a small screen. Therefore, we should make buttons significantly larger for a smartphone app; 44 pixels high turns out to be a pretty good average. While that height is almost double the typical desktop button height as shown in **Figure 2**, it makes sense.

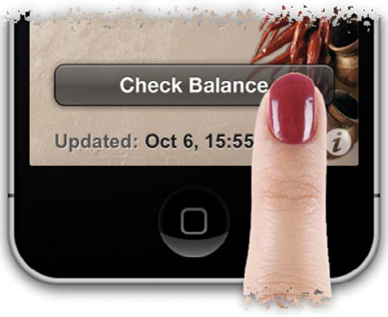

Figure 2. Big buttons provide easy targets

Many smartphone users operate their devices with one hand and oftentimes tap on buttons exclusively with their thumbs. Although tapping on a button may seem easy while waiting in line at the grocery store, accurately hitting the intended target may be quite another matter while jostling about on the metro. That fact leads nicely to the second point: arrange controls with enough space between them to make them easily touchable and to prevent the user from accidentally tapping a neighboring control.

The Tiny Keyboard

As fast as teenagers may be at texting their "BFFs", not all smartphone users enjoy the same level of keyboard competence and some are downright snails when it comes to how fast they input text. To combat this, the next goal is to allow users to accomplish as much as is feasible without ever having to bring up the keyboard. This means that traditional text-based controls are not necessarily the first choice when it comes to input. Instead, buttons, spinners, and graphical custom controls like the rating system and guest count controls employed by the "BUI Tip Calculator" (see **Figure 3** and accompanying article Adding Style to BBx Web Apps With Custom CSS on page 26) can dramatically reduce the number of taps, eliminate the popup keyboard, and result in a more enjoyable and productive user experience. Directly manipulating screen content via graphics is no longer considered extravagant, it is now the mark of a well-designed and efficacious app.

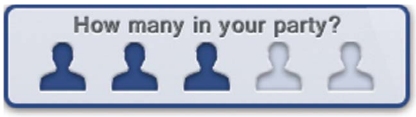

Figure 3. Specifying the number of guests in the dinner party with a custom graphical control

Single or Multiple Windows?

Typical desktop applications have a title bar and may show multiple windows at once. These windows may be modal, or in some cases the user is actively encouraged to interact with many windows in order to facilitate copying information from one screen to another. When designing our BUI app for the mobile Web, one of the more fundamental design decisions is whether to allow multiple windows or constrain the app to a single window to more accurately emulate a native smartphone application. The best choice, given the particulars of our app, may be to display multiple windows for increased flexibility. However, the tabbed window bar does consume more screen space and, if overused, could possibly confuse or frustrate mobile device users that are unaccustomed to it. When choosing the single-window paradigm, we need to be aware that the tabbed window bar will display automatically if more than one window is visible at once so we must be sure to write extra code to prevent this. **Figure 4** shows a comparison of the BUI Menu program running as a single-window app and as a multi-window app.

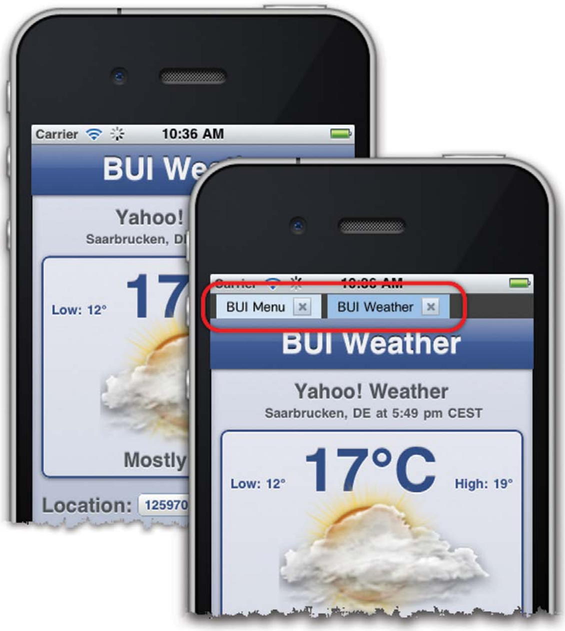

Figure 4. The BUI Menu demo program with and without the tabbed window bar

Screen Size and Orientation

Because screen space is at a premium on a smartphone, making the best use of every pixel is imperative. That means that we'll have to figure out what the target device's screen resolution is and decide whether to support both portrait and landscape orientations. BBj gives us all of the necessary information to gather the info and react accordingly via the following:

`INFO(3,6)` - Use to determine if the app is running in BUI

`INFO(3,8)` - Use to determine the web browser

`INFO(3,8)` - Retrieves the navigator.userAgent string from the browser when running in BUI

`FIN(sysgui_chan,IND=0)`- Use to determine the screen size

Armed with this information, it is possible to determine, for example, that our app is running in BUI on an iPhone in portrait orientation. It's important to realize that the available screen size can vary – not only between devices (such as comparing an Android device to an Apple iOS device), but also when the device orientation changes. The Android platform, for example, runs on a wide variety of hardware and supports at least four different screen resolutions. The screen size also differs depending on how the user launches the app. Mobile browsers typically reserve portions of the screen for common UI elements such as an upper URL address/search bar and a bottom button bar that handles forward/backward navigation, bookmarks, sharing, and more. In order to maximize space for a BUI app, BBj automatically directs Mobile Safari to hide the top URL bar resulting in an extra 320x60 pixel area that is now available for use by our app. The bottom button bar is still shown, as there is no way currently for BBj or any web page >>

to force Mobile Safari to hide it. However, there is a way for the user to accomplish this manually – simply save the BUI app's link to iPhone's home page, subsequent launches from that home screen icon will run the app in full screen.

Running the app in full screen results in a couple of benefits. First, the BUI app now has access to another 320x44 pixels of precious screen space as shown in **Figure 5**.

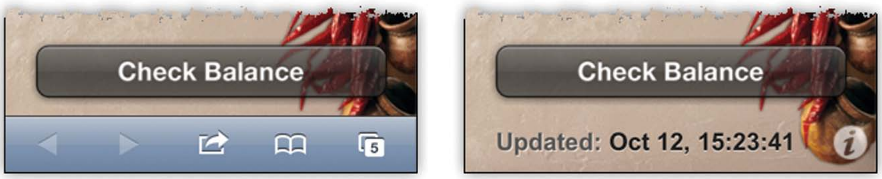

Figure 5. Maximizing screen space previously consumed by the browser's bottom button bar

The second benefit is that users now have an icon on their home screen that they can use to gain quick access to our app. Not only will it be easier to launch our app from the home screen, but our app also shows up in the phone's search results. By default, iOS will create an icon that is essentially a small version of a screenshot of the BUI app. This may suffice for some pages and apps, but the image is so small that it is usually quite difficult to see much of the detail. BBj helps out in this area as well, by allowing you to specify a custom icon when defining a BUI app in Enterprise Manager. **Figure 6** shows two copies each of the BUI Menu app and the ChileCustomer Balance app. The first icon for each app is the default screenshot version, and the second icon is a custom image that we associated with the BUI app in Enterprise Manager Customizing the icon for our app allows us to use an icon that has visual impact and is immediately recognizable, helping it to stand out amongst the other icons on the home screen.

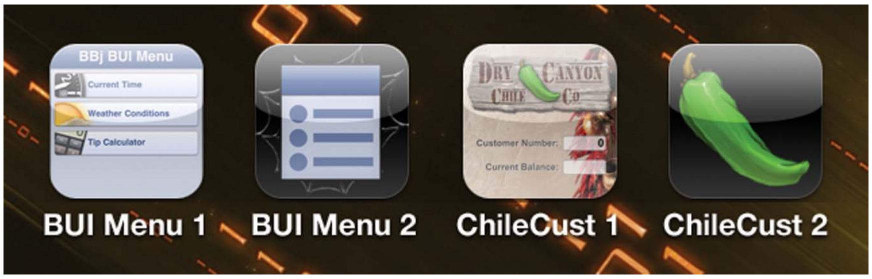

Figure 6. Comparing default and custom BUI App icons on an iPhone

Handling Orientation Changes

Many smartphone apps are aware of the device's orientation and adjust their layout automatically. A BBj BUI program can do the very same thing by registering for the ON_SCREEN_RESIZE event that fires any time the screen resolution, or the browser's client area in the case of BUI, changes. On a desktop, this typically occurs when users resize their browser window. On a mobile device, this event typically occurs when the orientation changes, as the available screen size is different in portrait mode versus landscape. After registering a callback for that event, a BUI program can query the screen width and height via the aforementioned FIN() function to determine the size and orientation. Once the app determines that the orientation has changed, it can modify the control layout on the screen or, in the case of resources, load a new resource file that was specially built for the new orientation. Many end users have come to expect different layout options, and while supporting multiple layouts means added work and development time, it certainly is thrilling to see our app change its UI in response to rotating the phone!

Use Your Finger, Not Your Mouse

Regardless of whether a user launches a BUI app from their phone, tablet, or desktop browser, the app executes in much the same way. However, how users interact with the program can change dramatically depending on whether they use a mouse on their desktop or their finger on a touch screen. As mentioned earlier, a touch screen on a smartphone or tablet does not have the concept of a "hover" state and since you navigate without a mouse but with your fingers, the only interaction with the device occurs when you contact the screen.

Hovering your finger close to the screen doesn't register any events, at least not with today's hardware. It is something to look forward to in the future, though, as designers experiment with proximity sensors. In the meantime, carefully review any existing applications that you plan on deploying in BUI. During this review, look for areas to address that are incompatible with touchscreen devices such as concepts and events that are mouse-specific. In addition to hover events, BUI programs should not rely on mouse events such as the Mouse Move/Enter/Exit events. Right click events are not generated either, so if your programs rely on users selecting commands from popup menus launched from a right click event, you will have to make those commands accessible from the program menu or some other means.

Summary

Designing a BBj BUI application for a smartphone device brings with it several new concepts and challenges that are common to touchscreen devices. Although these may appear to be daunting at first, the challenges will gradually morph into a learning and ultimately rewarding experience. As a developer, there is nothing quite like seeing your code running in the palm of your hand. Putting that power and capability in the hands of your users is even more rewarding, and reaping the benefits from a "constantly connected" workforce ought to be motivating enough to "the powers that be" to get you started right away! ■