07/04/2011 Updates

buzztouch v1.5 Cutter

Because we love boats, this release of buzztouch v1.5 is code-named “Cutter” and replaces the previous “Schooner” release on 05/11/2011.

Dynamic Menus: List and Button menus can now be created dynamically using a backend text file or script to drive the menu. Previously, menu items and buttons were connected to screens and features that existed in the main BT_config.txt file. Now, it's possible to load a screen that does not exist in the BT_config.txt file. This approach allows advanced menu controls that are 100% dynamic. Flexibility was a core reason for dynamic menu support but not the only reason. Another reason is related to performance. If an application contains many hundreds (or thousands) of screens it's main configuration file gets very large in file size. The large file size of the BT_config.txt file can be problematic when the application refreshes it's data. For this reason, large application can now be created without having to list all the screens and features in the BT_config.txt file. It's now possible to use a BT_config.txt file that includes only a small amount of data (fast loading) because each menu loads it's screens dynamically.

Published Documentation: Official v1.5 documentation has been published. The documents are designed to help novice and advanced users. We hope to improve on the documentation over time. The docs are here: http://www.buzztouch.com/docs/v1.5/

Android: We have been working on buzztouch v1.5 for Android for many months. We are closer to releasing it but first needed to release “Cutter” for iOS. This release was necessary before the Android release for a few technical reasons. It's important that the app-creation process, and BT_config.txt file, and the way the application processes it's data be consistent for both platforms. Because of this, some changes were necessary in the iOS version to support this consistent approach.

Memory Leaks: Several memory leads were discovered and fixed. The leads were related to how iOS was releasing (or not releasing) objects. Discovering and fixing random memory leads is an ongoing process and any help in this area is greatly appreciated. We are admittedly not too skilled at using Apple's Instruments package to isolate and fix memory leaks and do encourage Early Adopters to help us with this tedious process. If you're comfortable with Instruments, and can find any memory leaks, let us know.