techBASIC App Builder™ 1.0

Reference Manual

A product of

Byte Works®, Inc.

http://www.byteworks.us

Credits

Programming

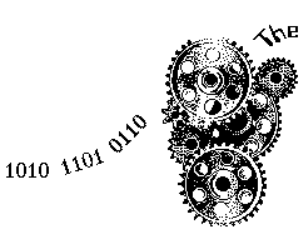
Mike Westerfield

Art

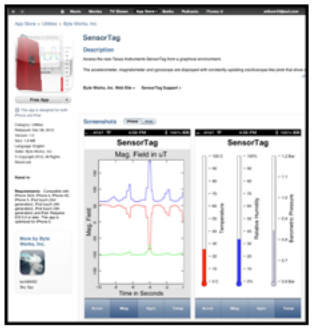
Karen Bennett

Documentation

Mike Westerfield



What is techBASIC App Builder?



Write Compile Sell

techBASIC™ is the technical programming app for iOS that makes it easy to collect information from internal and external sensors, process the information with a powerful scientific computing language, and display the results using stunning interactive graphics. Until now, these apps had to be executed from within a copy of techBASIC on each iOS device.

techBASIC App Builder™ takes techBASIC programs written on iOS and converts them to fully functional stand-alone apps. These apps look, work, and indeed are, just like any other app written for the App Store—they just happen to be written in techBASIC instead of Objective C. Once created, these apps can be submitted to the App Store just like any other app. They can be distributed free, or you may charge for the apps. They can also be submitted using Apple's B2B program or distributed as ad hoc apps. Or, of course, you can simply compile your own apps for your own use, installing them on your own iOS devices, without submitting them to the App Store at all.

Prerequisites

This guide makes some assumptions about your skills and resources.

1. You should already have, and know how to use, techBASIC. While it is possible to create techBASIC programs directly from techBASIC App Builder and test them from the iOS Simulator that comes with Xcode®, it is generally easier to write and debug techBASIC programs directly from the iOS device using its debugger and built-in help systems.
2. Your techBASIC app must run from the graphics screen, as must all iOS apps. While the `PRINT` statement still works, it is tied to the Xcode console, not the iOS device. `PRINT` is great for debugging, but cannot be used in a running app.
3. You must have a copy of Xcode. Apple's Xcode IDE is the only way Apple® allows programs to be built and submitted to the App Store. Xcode can be downloaded from Apple's web site.
4. You must join Apple's developer program to get the certificates needed to install apps on any iOS device, even your own. This is true even if you do not plan to submit apps to the App Store. See Apple's web site for information on joining the developer program.
5. You need to know how to use Xcode and Apple's web site to submit apps to the app store or install them on your iOS devices. The specific steps and requirements change frequently; see Apple's web site for further information.

Differences between techBASIC on iOS and techBASIC Apps

For the most part, techBASIC is the same, regardless of where the program is executed. The key difference between techBASIC apps running from techBASIC and techBASIC apps running as stand alone apps is the absence of the development environment. Stand-alone apps have a single display mechanism corresponding to the graphics screen under techBASIC. The debugger, program list, and console are all missing. The graphics screen always displays in full-screen mode.

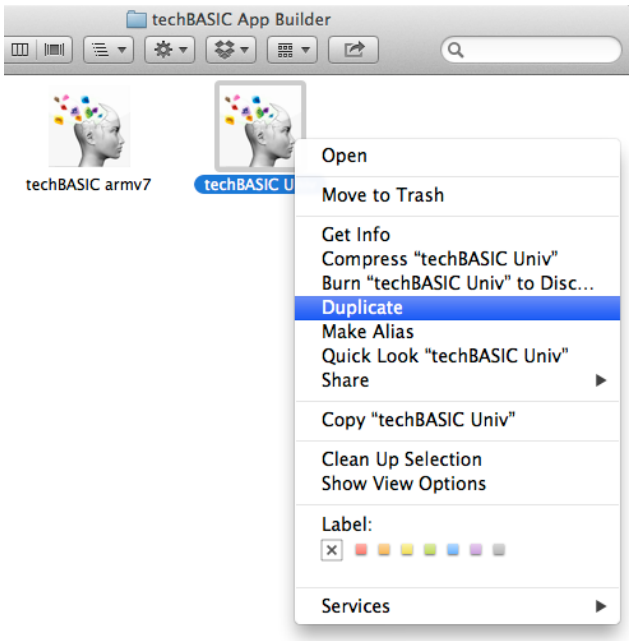
For the most part, this is an obvious difference. The one question might be what happened to the console. Does `PRINT` still print? As it happens, yes—but `PRINT` from an app sends the text to the Xcode console, not the app itself. This is exactly the same as using `printf` from Objective C apps, and is tremendously useful for debugging.

With the exception of the size of the graphics screen and the use of the console, techBASIC programs will behave exactly the same running from techBASIC or as stand-alone apps.

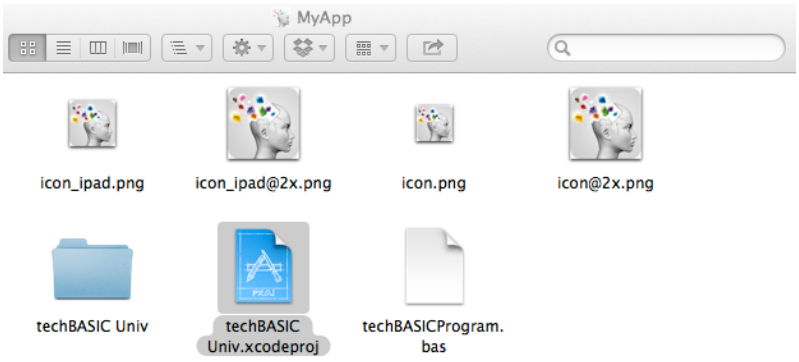
Building Your First techBASIC App

While there are a number of options, the basic steps for building and running a techBASIC app are really quite simple.

1. Unpack the techBASIC App Builder libraries that came with your purchase of techBASIC App Builder.
2. Make a copy of the folder techBASIC Univ. Rename it to something convenient.

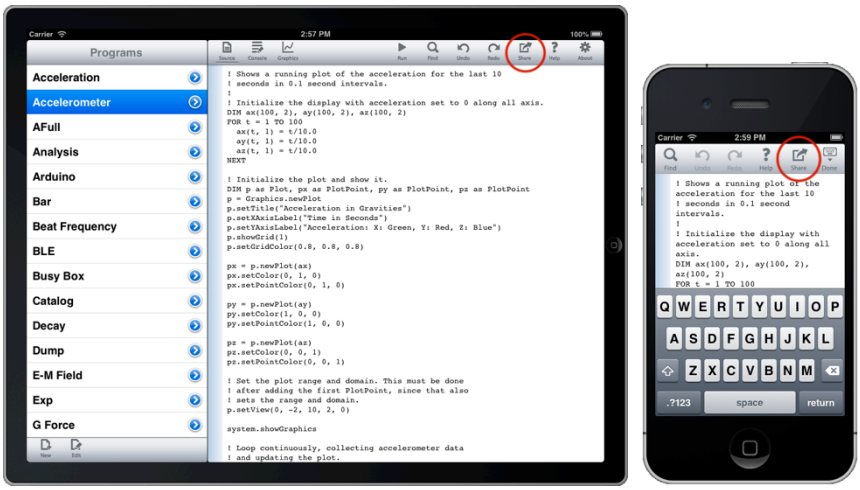


3. Open the folder and double-click techBASIC `Univ.xcodeproj` to start Xcode and open the new project.

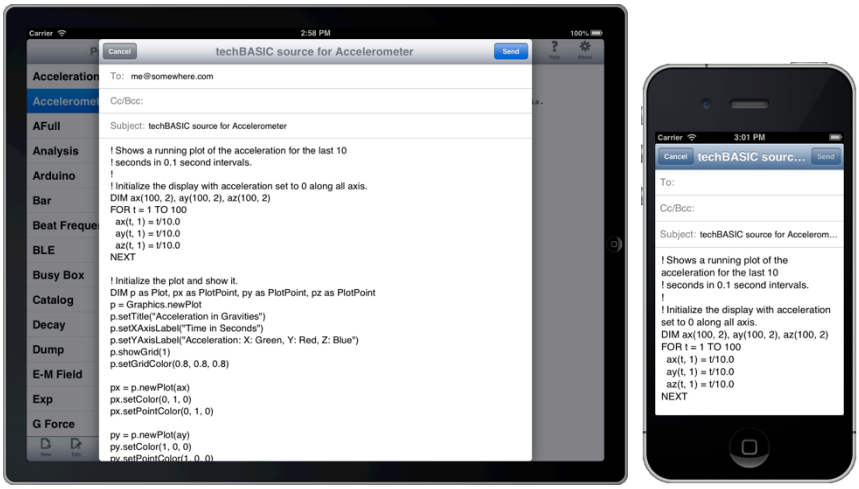


4. Replace the file `techBASICProgram.bas` with your own techBASIC program.

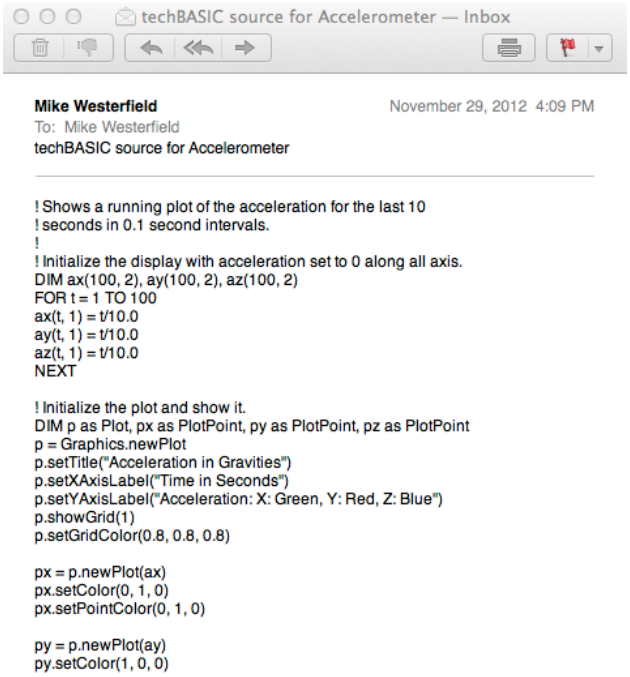
   a. Move the techBASIC source from your iOS device to your Macintosh using the Share button. From the iOS device, view the source you want to use, then tap the Share button to email the source.



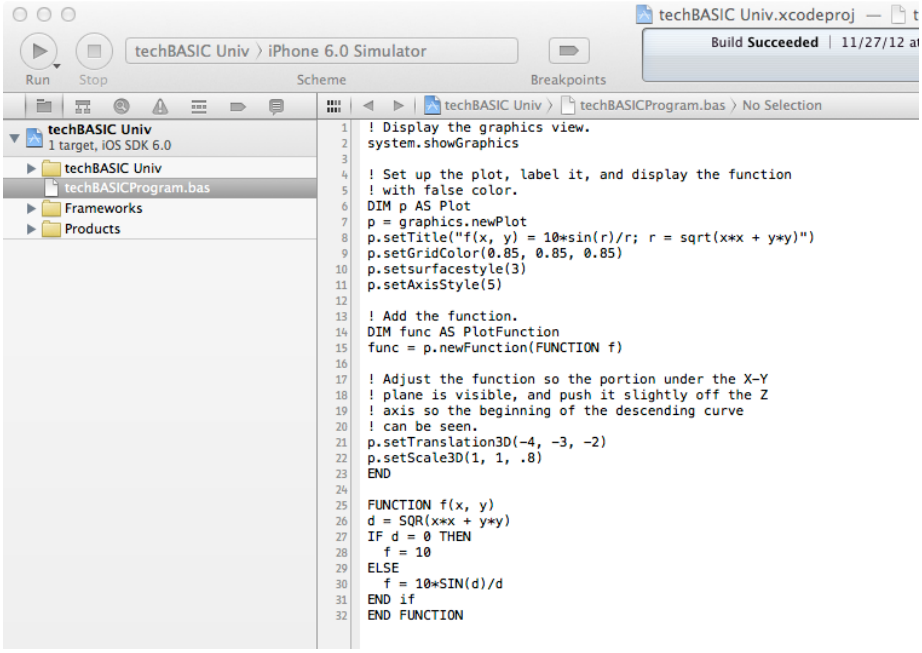
b. Send the source to yourself.



c. Copy the source from the email client.

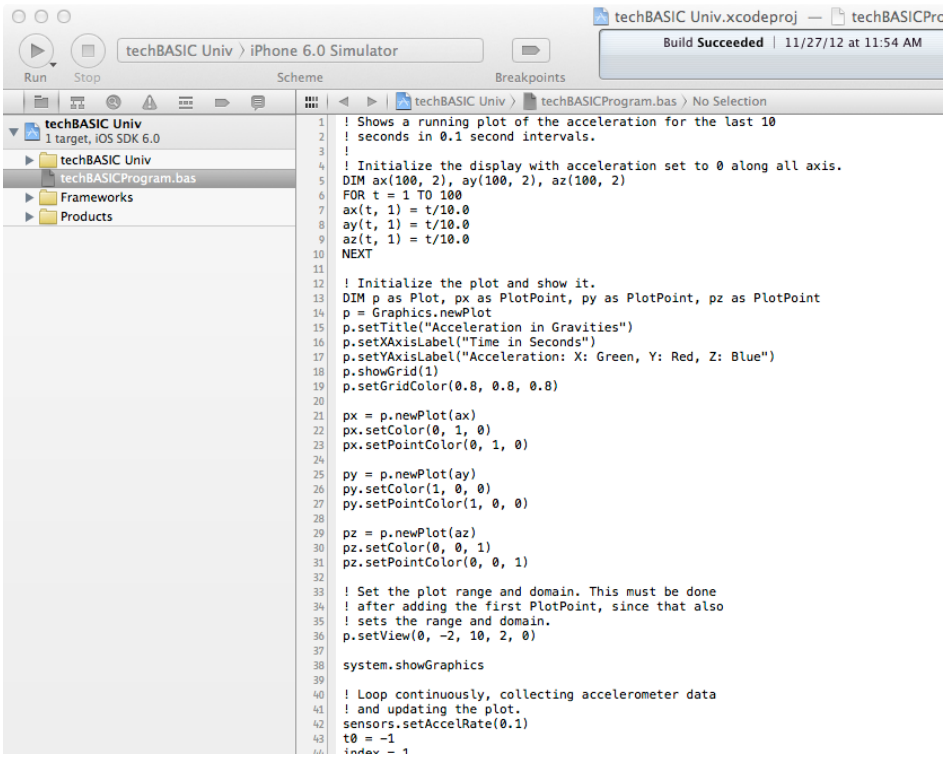


d. Click on `techBASICProgram.bas` from Xcode.



e. Select all of the existing source in this file and delete it.

f. Paste in your own source.



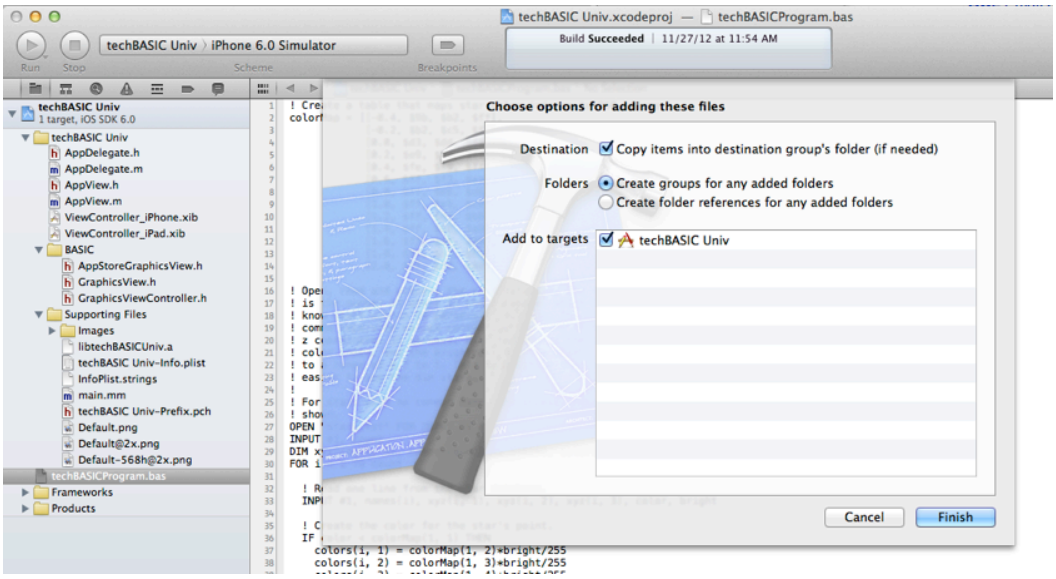
5. Copy any other files your program needs into the Files folder.

a. This can include images, data files, or anything else your techBASIC program needs.

b. Use iTunes to copy files to and from your iOS device. See the techBASIC Quick Start Guide (available on the Byte Works web site) if you are not familiar with copying data files with iTunes.

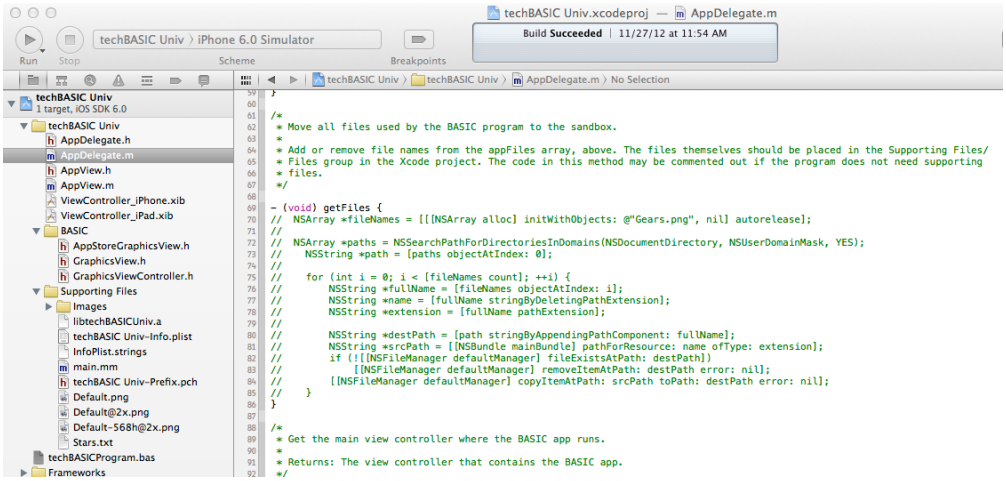
c. Drag the files into the Files folder in Xcode's project window.

d. Select the option Copy items into destination group's folder (if needed)

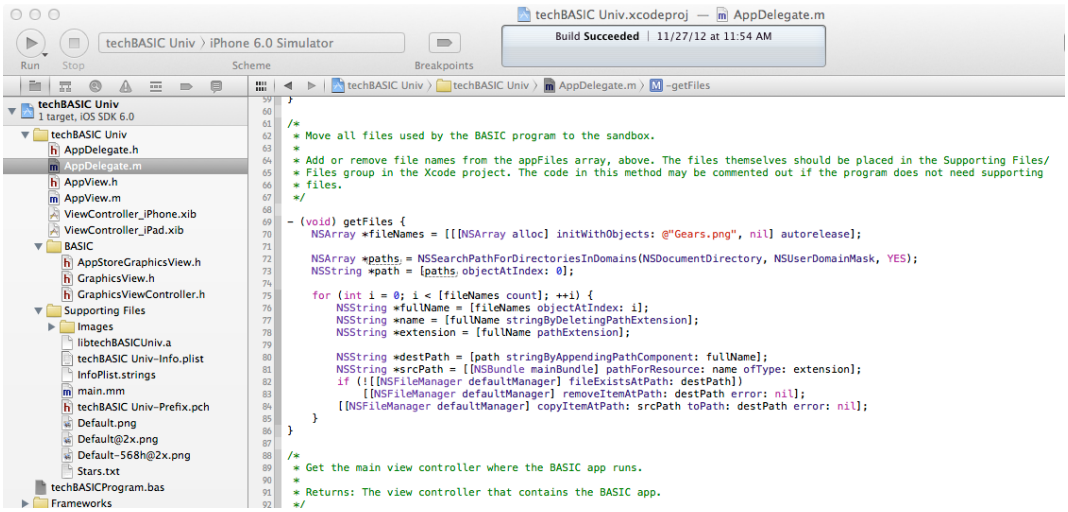


e. Click Finish.

f. Click the file `AppDelegate.m` from the file list in the Xcode project window to edit the file.



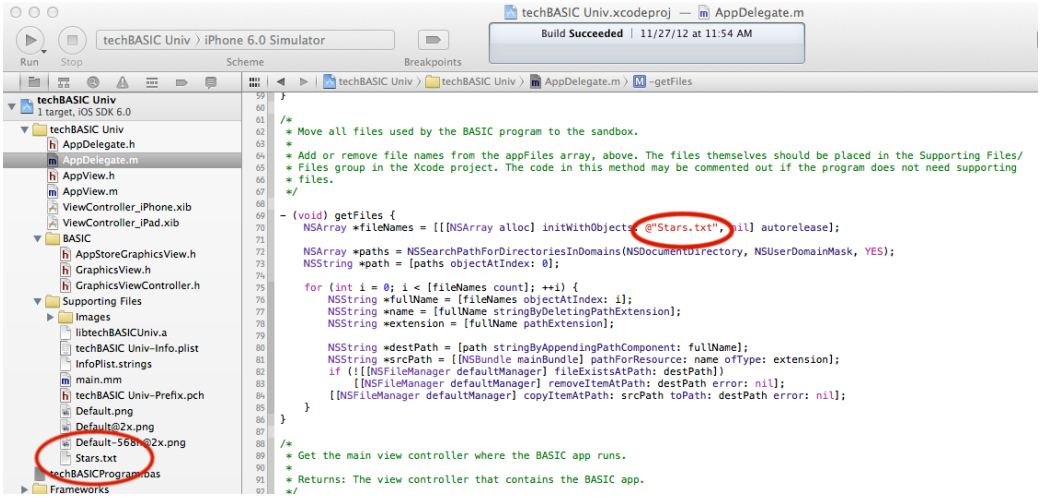
g. Find the method `getFiles`. All of the code in this method is commented out. Restore this code.



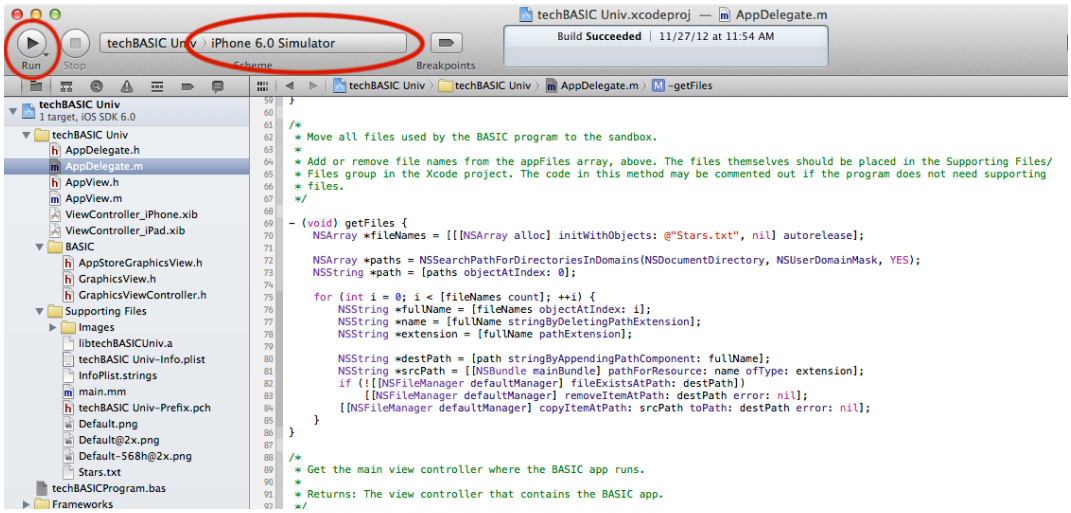
h. The first line of the method sets up an array with a list of the files needed by your program. Change `Gears.png` to the first file needed by your program, then add any additional files, separated by commas, like this. The file names are case sensitive on the iOS device, so be sure the letter case matches exactly.

```
NSArray *fileNames = [[[NSArray alloc] initWithObjects:
@"Gears.png", @"file2.txt", @"file3.bin", nil] autorelease];
```

In the screen shot, we added a file named `Stars.bas`, so that's the name that appears in the source code.



6. Select the iOS device or simulator from the Xcode project window.
7. Click Run to run your program.



Your program will compile and run. If you selected a simulator, the iOS Simulator will open automatically. If you selected an iOS device, the app will install on the device and then run.

Customizing Your App

There are a number of customizations you can perform on your project, and several you definitively should do for any project you plan to distribute beyond your own iOS device. The following sections go over a list of the basic customizations to consider.

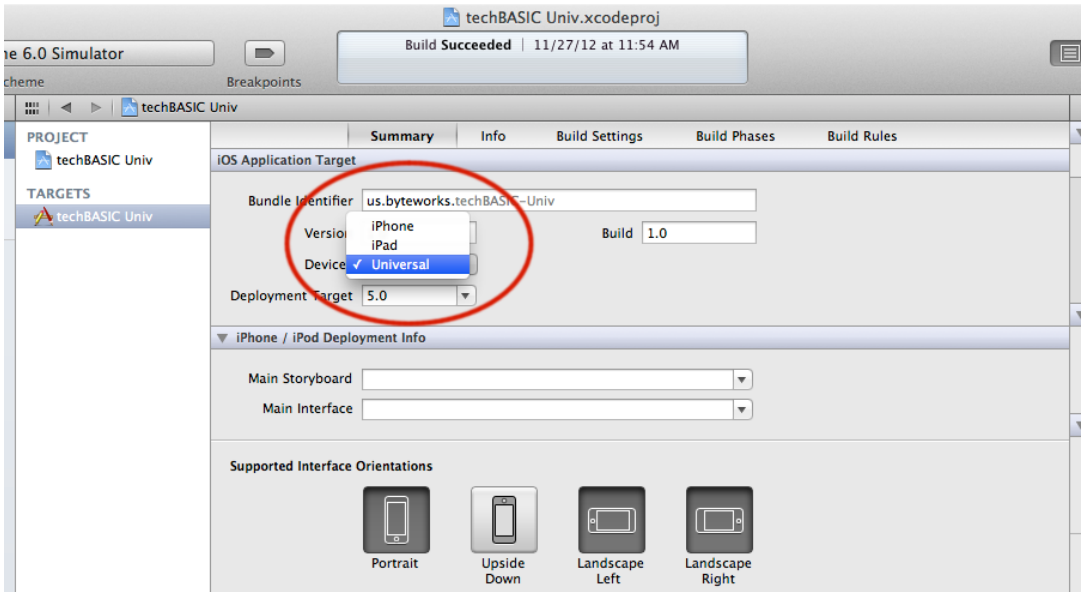
There are a number of other customizations you can do from Xcode that are not directly related to techBASIC. These are covered in Apple's documentation of Xcode.

Build for the iPhone or iPad

The default project is configured to run on both the iPhone and iPad. Change the Targeted Device Family setting if the techBASIC app is only designed to run on one or the other.

1. Select the techBASIC Univ project from the TARGETS list in the Xcode project window.
2. Select the Summary tab.

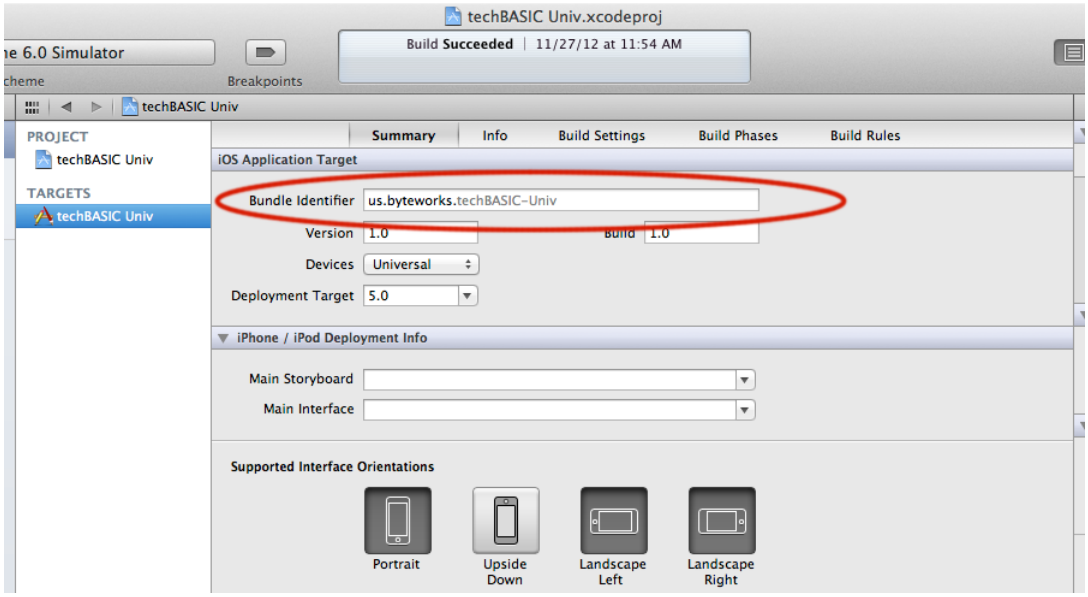
3. Select the appropriate device from the Devices pop-up.



Use Your Own Bundle Identifier

You created a bundle identifier when you signing up to distribute apps on the App Store. Be sure and change to your own bundle identifier.

1. Select the techBASIC Univ project from the TARGETS list in the Xcode project window.
2. Select the Summary tab.
3. Enter your own bundle identifier under iOS Application Target.



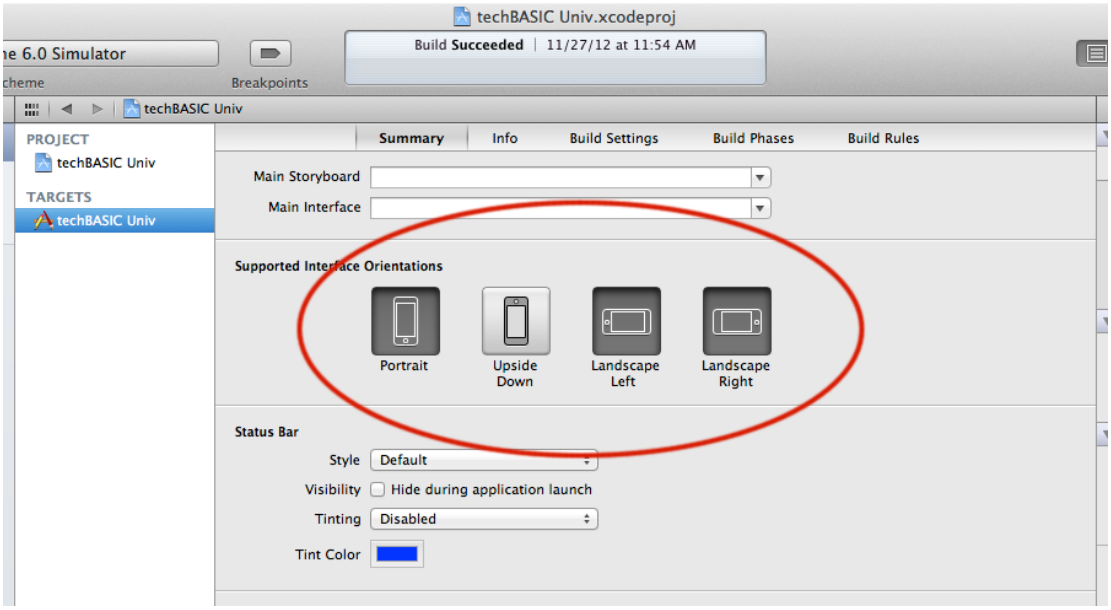
Select the Supported Device Orientations

There are two places to select the device orientations your app supports, one for the iPhone and one of the iPad. Click the appropriate settings.

In general, the iPad should support all orientations, while the iPhone should not support Upside Down.

techBASIC's `System.setAllowedOrientations` call can still be used to set the allowed device orientations while a program is running. This will override the setting in this panel. You should always make sure the device orientations selected here match the initial orientations specified in the `System.setAllowedOrientations` call. Changing the allowed orientations later will work exactly as it does in techBASIC.

1. Select the techBASIC Univ project from the TARGETS list in the Xcode project window.
2. Select the Summary tab.
3. Change the device orientation as needed.

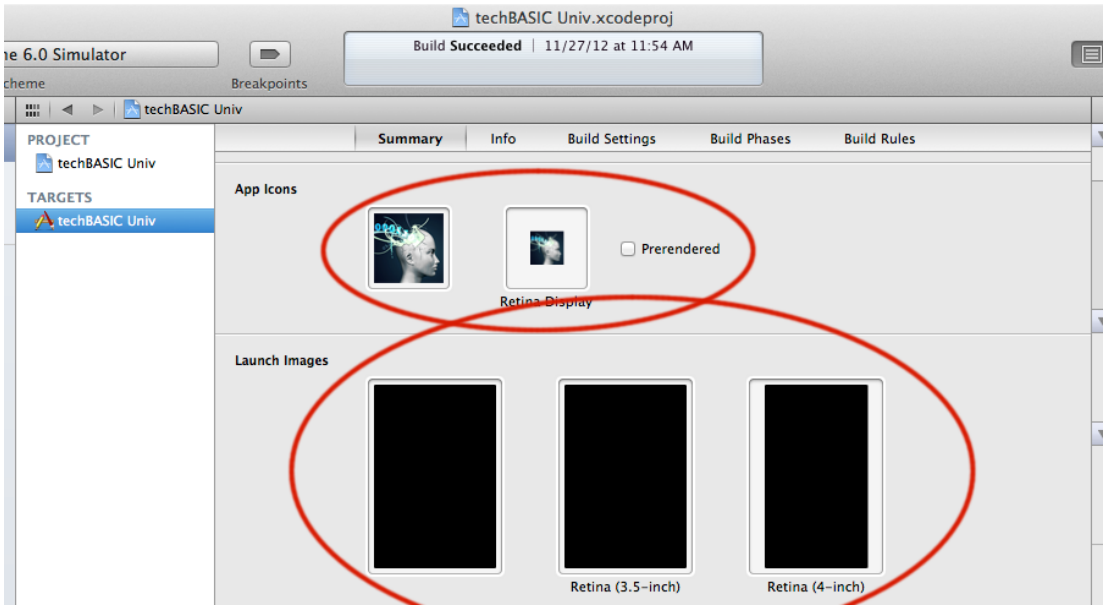


Use Your Own Icons and Launch Images

Create your own app icons and launch images and drag them to the appropriate spots in the target's deployment info section.

1. Select the techBASIC Univ project from the TARGETS list in the Xcode project window.
2. Select the Summary tab.

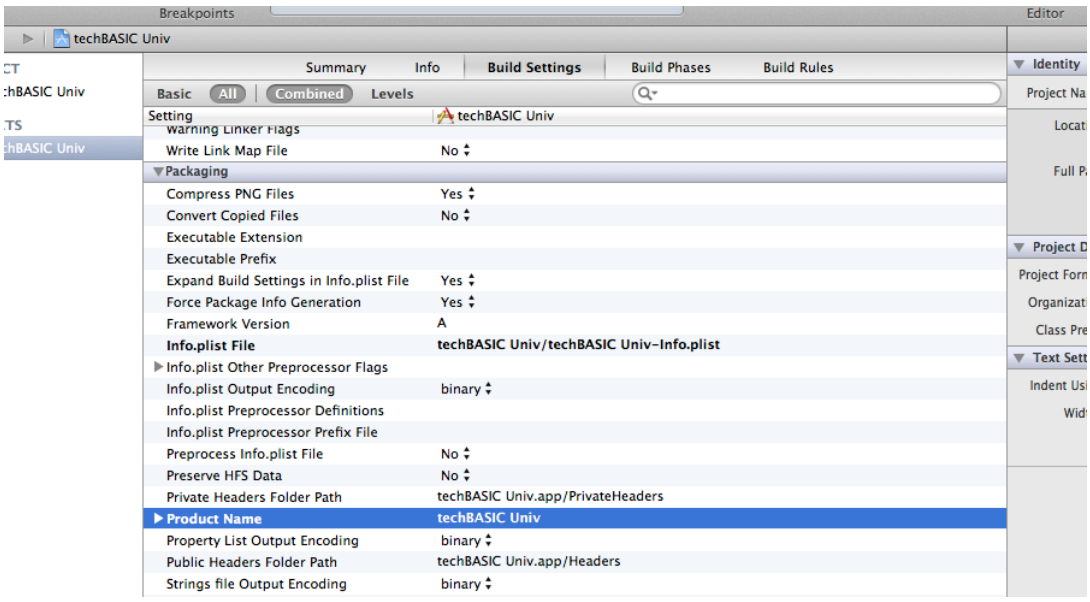
3. Drag new icons and launch images to the appropriate spots in the Xcode project window.



Change the Product Name

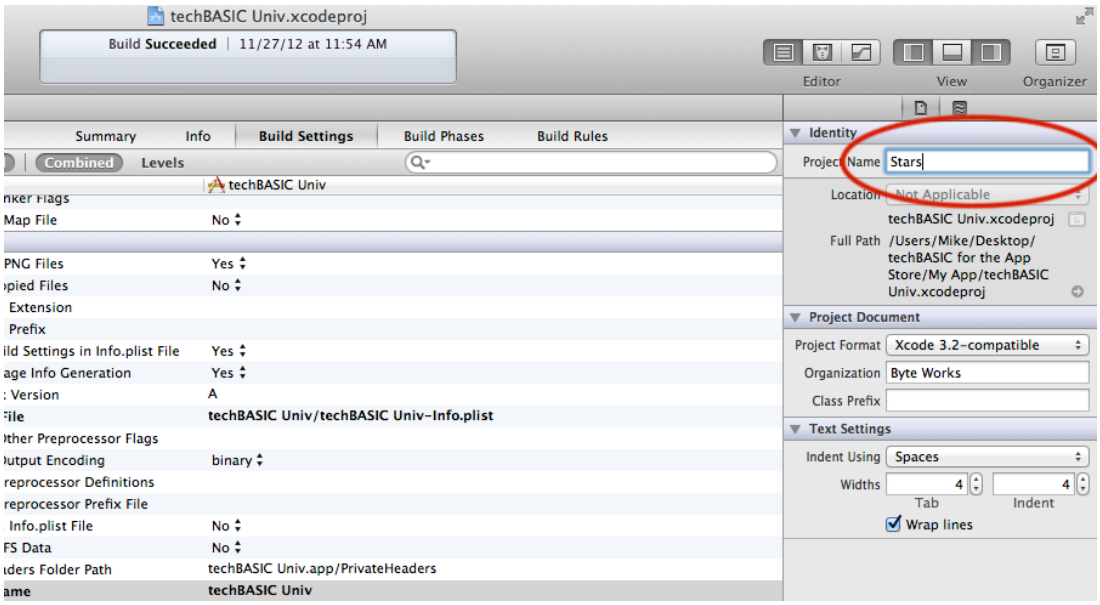
The iOS program will be called techBASIC Univ unless you change the product name. The simplest way is to change just the name of the app itself.

1. Select the techBASIC Univ project from the TARGETS list in the Xcode project window.
2. Select the Build Settings tab.
3. Under Packaging, change the product name from techBASIC Univ to the name of your app.



The alternative is to change the name of the entire project. This renames the `.xcodeproj` file and several displayed variables in Xcode itself.

1. Select the techBASIC Univ project from the PROJECT list in the Xcode project window.
2. Make sure the Inspector pane is visible by clicking the right button on the View options.
3. Click the folded page icon in the Inspector pane.
4. Change the Project Name field. You will be led through a number of options. Unless you are very sure about what you are doing, select the default option at each step. The project will be renamed.



Compile for iOS Only

You may have noticed that the apps created so far run on both iOS devices and the iOS Simulator on the Macintosh. This is a fantastic capability when you are debugging and testing an app, but it comes with a downside. Since Apple implemented the iOS Simulator as a simulator, not an emulator, apps compiled for the simulator are actually compiled for the Intel chips used in the Macintosh, not for the Arm chips used in iOS devices. That's a good thing—it means the simulator runs very fast, often faster than the real iOS device. It also means the techBASIC library is twice as large as it needs to be.

The size of the library isn't that big a deal when testing, or even if you are building apps for your own use. It becomes fairly wasteful, though, when you distribute apps through the App Store. For apps distributed through the App Store, start with a copy of the techBASIC armv7 folder instead of the techBASIC Univ folder. This uses a library that won't work on the simulator, but is half the size of the library that supports both devices.

A Tour of the Files

You don't really need to know what the various files in the project folder are to create and use apps from techBASIC, but we know you're curious, so here's a tour of the files. For the most part, you should leave these files untouched, but a few can be removed for certain kinds of techBASIC programs.

Files not listed here are created by Xcode. They should not be modified unless you are very familiar with the inner workings of Xcode projects.

AppDelegate

The `AppDelegate.h` and `AppDelegate.m` files are created by Xcode, and represent the top level of control for the app. They are required. Other than adding your list of files (covered earlier), don't modify them unless you know Objective C well.

AppView

The `AppView.h` and `AppView.m` files are used by the two ViewController nib files. They link the iOS app to the techBASIC graphics display.

ViewController_iPhone.xib

This is the GUI layout for the iPhone and iPod. It can be deleted for iPad-only apps.

ViewController_iPad.xib

This is the GUI layout for the iPad. It can be deleted for iPhone-only apps.

BASIC

The three .h files in the BASIC file group expose the parts of the techBASIC library needed to link the iOS app to the techBASIC core. All three are needed, and should not be modified.

iconxxx.png

There are four icon files in the Supporting Files/Images group. These are the techBASIC icons that appear by default when the app is viewed from the home screen on the iOS device. All four can be deleted after adding your own icons for the app.

cp.png

This image is used by the techBASIC color picker control. It is not needed in apps that do not use color picker controls.

pan-zoomxxx.png

There are four files used to present the user with the option of panning, zooming and rotating either the plot or the axis when manipulating 3D plots. They are not needed in apps that do not display 3D plots.

libtechBASICUniv.a

This is the library containing the techBASIC compiler and runtime libraries. It must be present.

techBASICProgram.bas

This is the techBASIC program itself, which you replaced with your own program.