Extracted from:

Build Location-Based Projects for iOS

GPS, Sensors, and Maps

The Pragmatic Bookshelf

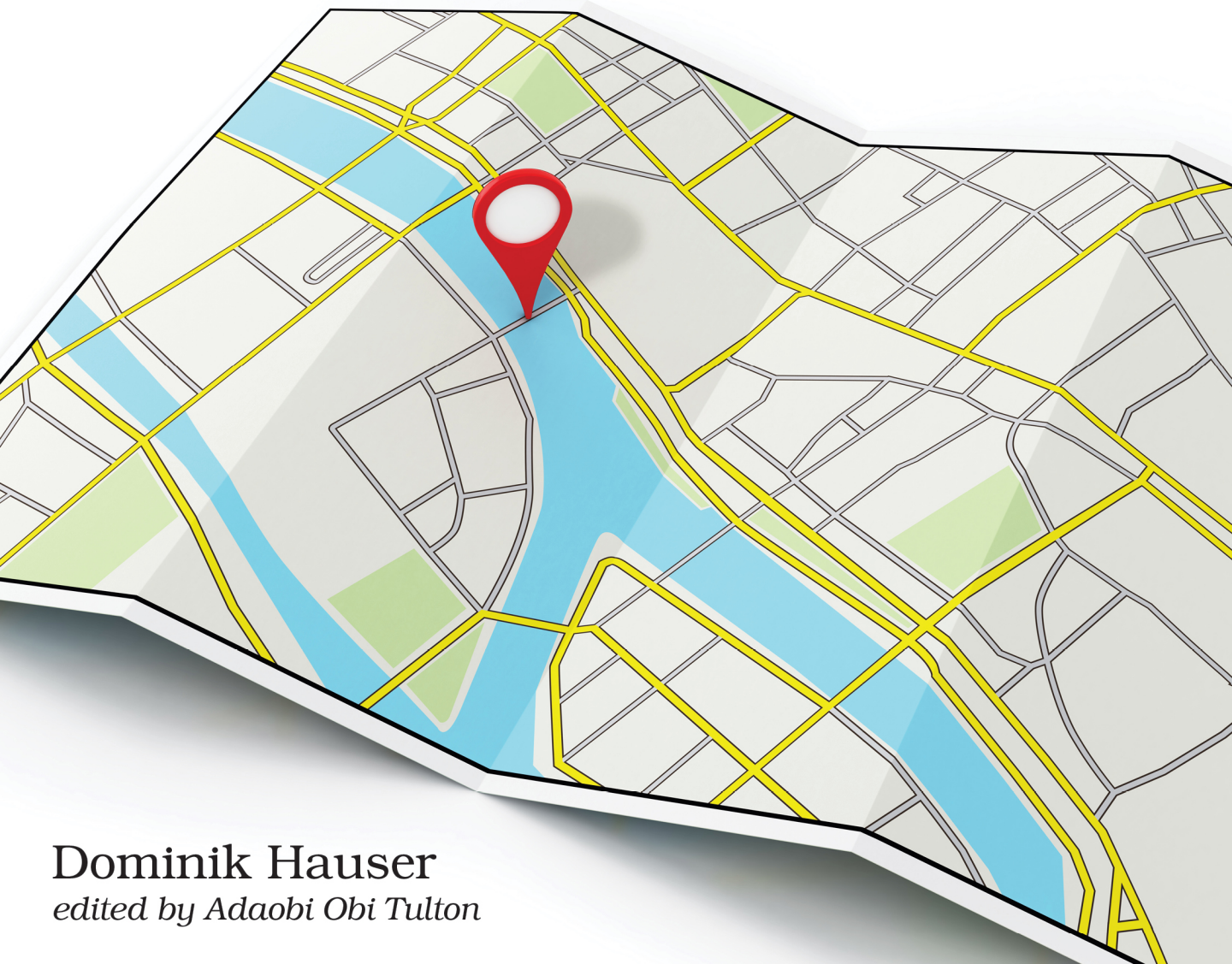
Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Build Location-Based Projects for iOS

GPS, Sensors, and Maps



Dominik Hauser

edited by Adaobi Obi Tulton

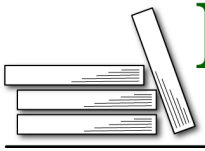
Build Location-Based Projects for iOS

GPS, Sensors, and Maps

Dominik Hauser

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Copy Editor: Rachel Monaghan
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-781-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—August 2020

Removing the Storyboard

The user interface of the app we're going to build is quite simple. This is a good opportunity to practice building apps without storyboards. In the next chapter we'll use storyboards again.

Pro Tip: Always Practice

A good developer can build apps with storyboards, with XIBs, in code, and with SwiftUI. You should practice your abilities in all these different approaches. Only when you are proficient in each approach can you decide which is best for the project at hand.

When setting up a new app with Xcode, you can choose only between storyboard and SwiftUI for building the user interface. If you want to build the user interface in code instead, you have to choose one of the two options and then remove the files and settings you don't need. What you choose mainly depends on your preferences. The amount of work to remove the storyboard is comparable to that for removing SwiftUI support.

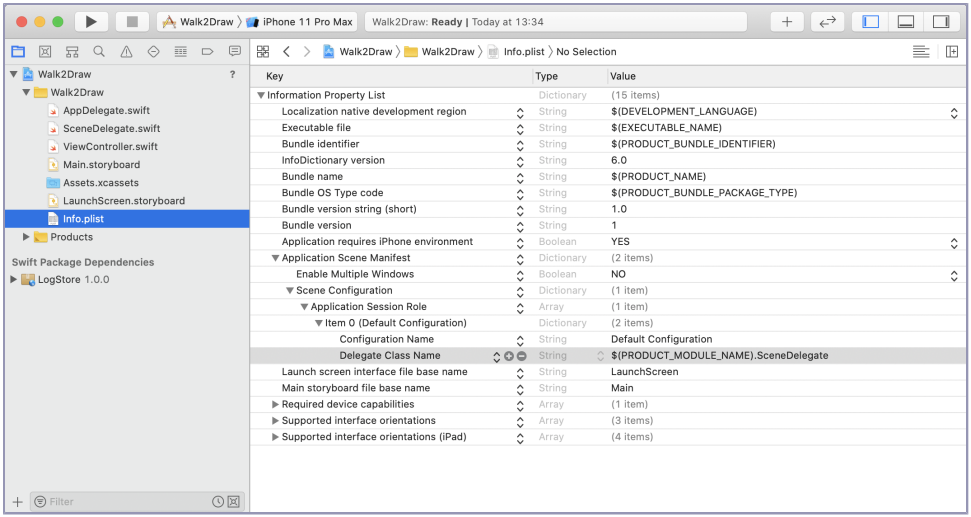
The first entry in the project navigator, with the blue icon and the name Walk2Draw, is the project itself. Click it to open the project settings. Next, select the target in the TARGETS section to open the target settings. Select the General tab if it's not already selected. Xcode defines the initial storyboard of an app in the section Deployment Info, next to Main Interface. Remove *Main* from the text field to tell Xcode that the app shouldn't load a storyboard when it launches.

As of Xcode 11, the storyboard is defined in a second place, so we'll have to make changes there as well. Open the file Info.plist and navigate to Application Scene Manifest > Scene Configuration > Application Session Role > Item 0. Delete the whole line with the setting for Storyboard Name. The result should look like what's shown in the image on page 2.

Once we've made the changes, the storyboard won't be used anymore, so we can delete it. Select the file Main.storyboard and remove it from the project by deleting it.

If you think removing a storyboard from a project should be easier, I feel you. I hope Apple adds the option to use neither a storyboard nor SwiftUI in a future version of Xcode, but I won't hold my breath.

Now that the storyboard is gone, we need to create the window and assign its rootViewController when the app loads. Open SceneDelegate.swift, look for the



method scene(_:willConnectTo:options:), and replace the underscore (which acts as a placeholder) in the following line of code with scene:

Map/Walk2Draw/Walk2Draw/SceneDelegate.swift

```
guard let _ = (scene as? UIWindowScene) else { return }
```

Afterward, the line should look like this:

Map/Walk2Draw/Walk2Draw/SceneDelegate.swift

```
guard let scene = (scene as? UIWindowScene) else { return }
```

With this line we try to cast the scene variable passed into scene(_:willConnect-To:options:) to the type UIWindowScene. If it fails, we return from this method. Now change the contents of scene(_:willConnectTo:options:) so that it looks like the following code. The changed lines are highlighted.

Map/Walk2Draw/Walk2Draw/SceneDelegate.swift

```
func scene(_ scene: UIScene,
           willConnectTo session: UISceneSession,
           options connectionOptions: UIScene.ConnectionOptions) {

  guard let scene = (scene as? UIWindowScene) else { return }

➤ window = UIWindow(windowScene: scene)
➤ window?.rootViewController = ViewController()
➤ window?.makeKeyAndVisible()

  #if DEBUG
  trigger = LogTrigger(in: window)
  #endif
}
```

In this code, we first instantiate a new instance of UIWindow using the scene. As of iOS 13, a scene represents the user interface—or more precisely, an instance of the user interface. If we don't provide the window with the scene, the screen remains black. Next we assign an instance of ViewController to the rootViewController property of the window to define the initial view controller of our app. Then we call makeKeyAndVisible() to tell the window that it should become visible and that it is the *key window*. The key window of an app is responsible for handling nontouch and keyboard-related events. Remember, when creating the window of the application in code, we always have to call makeKeyAndVisible().

Setting up the user interface in code might be something you haven't done a lot. Let's see if our changes work. We can use an easy trick to figure out if Xcode launched the correct view controller.

Open ViewController.swift and add the following line at the end of viewDidLoad():

Map/Walk2Draw/Walk2Draw/ViewController.swift

```
view.backgroundColor = .red
```

Select your favorite simulator and build and run the code by clicking the play button in Xcode. After the app is loaded, you should see a red screen in the simulator. If you don't, retrace your steps and compare them with your code and settings.

Pro Tip: Alternate Simulators

Alternate the simulators you use during development to find user-interface bugs or layout problems related to false assumptions as early as possible.