Corona SDK Mobile Game Development Beginner's Guide  Second Edition

Corona is a software development kit to build mobile applications for iPhone, iPad, and Android devices. This book will take you through the journey of developing games right from installing Corona SDK and creating Hello World as your first app. Going further, you will learn how to script in Lua and learn some techniques to apply this in Corona SDK with demonstration. Towards the end of the book you'll build the final game, which involves physical objects and also uses features such as the accelerometer. Moreover, learn to make your game socially available on Twitter and Facebook.

By the end of the book, you’ll learn to deploy your iOS and Android games to the App Store and Google Play Store.

What this book will do for you...

- Learn how to script in Lua and get a better understanding of command flow, functions, and objects
- Load images and add a background image to your games
- Apply more functionality to your game using more APIs
- Use sprite sheets and create more complex game logic for the game
- Track your data and save the scores of your games
- Make your game socially aware by sharing messages and scores on Facebook and Twitter

The Beginner’s Guide approach...

- Clear step-by-step instructions for the most useful tasks
- Learn by doing, start working right away
- Leave out the boring bits
- Inspiring, realistic examples give you ideas for your own work
- Tasks and challenges encourage experimentation

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with Corona SDK'
- A synopsis of the book’s content

About the Author

Michelle M. Fernandez is a mobile game developer and cofounder of MobiDojo (http://www.mobidojo.com). She is also a mentor to aspiring artists and programmers trying to break into the game industry. After experimenting with several game engines over the years, she was introduced to Corona SDK in late 2010 and has been an avid user of the program ever since. She enjoys creating tutorials and new game frameworks for others to learn from. When Michelle is not developing games, she is spending time with friends and family, playing video games, traveling, and constantly learning new design paradigms.
This book is designed to introduce you to the basic standards of using the Corona SDK across iOS and Android platforms. You will enhance your learning experience by building three unique games in easy-to-follow steps. Aside from developing games, you will also dive into learning about social network integration, In-App Purchasing, and shipping your applications to the Apple App Store and/or Google Play Store.

What This Book Covers

*Chapter 1, Getting Started with Corona SDK*, begins by teaching you how to install Corona SDK on both the Mac OS X and Windows operating systems. You will learn how to create your first program in just two lines of code. Lastly, we'll go through the process of building and loading an application to an iOS or Android device.

*Chapter 2, Lua Crash Course and the Corona Framework*, dives into the Lua programming language that is used to develop in Corona SDK. We'll go over the basics of variables, functions, and data structures in Lua. This chapter will also introduce how to implement a variety of display objects within the Corona framework.

*Chapter 3, Building Our First Game – Breakout*, discusses the first half of building your first game, Breakout. You'll learn how to structure game files in a Corona project and create game objects that will be displayed on screen.

*Chapter 4, Game Controls*, continues with the second half of building your first game, Breakout. We'll cover game object movement as well as collision detection between objects in the scene. You will also learn how to create a scoring system that will implement the win and lose conditions of the game.

*Chapter 5, Animating Our Game*, explains how to animate a game using sprite sheets. This chapter will go in-depth with managing motion and transitions while creating a new game framework.

*Chapter 6, Playing Sounds and Music*, provides information on how to apply sound effects and music to your applications. It is vital to include some type of audio to enhance the sensory experience of your game's development. You will learn how to incorporate audio through loading, executing, and looping techniques with the Corona Audio System.
Chapter 7, *Physics – Falling Objects*, covers how to implement the Box2D engine in Corona SDK using display objects. You will be able to customize body construction and work with the physical behavior of falling objects. In this chapter, we'll apply the uses of dynamic/static bodies and explain the purpose of post collisions.

Chapter 8, *Operation Composer*, discusses how to manage all your game scenes with Composer API. We'll also go into detail on menu design, such as creating a pause menu and main menu. In addition, you'll learn how to save high scores within your game.

Chapter 9, *Handling Multiple Devices and Networking Your Apps*, provides information about integrating your applications with social networks such as Twitter or Facebook. This will enable your app to reach a bigger audience globally.

Chapter 10, *Optimizing, Testing, and Shipping Your Games*, explains the application submission process for both iOS and Android devices. This chapter will guide you on how to set up a distribution provisioning profile for the Apple App Store and manage your app information in iTunes Connect. Android developers will learn how to sign their applications for publication so they can be submitted to the Google Play Store.

Chapter 11, *Implementing In-App Purchases*, covers monetization of your game by creating consumable, nonconsumable, or subscription purchases. You will apply for In-App Purchases in the Apple App Store using Corona's store module. We'll take a look at testing purchases on a device to check whether transactions have been applied using the Sandbox environment.

Appendix, *Pop Quiz Answers*, covers all the answers enlisted in the pop quiz sections of the book.
Getting Started with Corona SDK

Before we jump right into coding some simple games, we need to install and run the necessary programs that will make our applications come to life. Corona SDK is primarily a 2D development engine. If you’ve had experience developing for iOS or Android, you will find the experience of working with Corona refreshing. It is also simple to use. In no time, you’ll be creating finished products to distribute through the Apple App Store and Google Play Store.

In this chapter, we will:

- Set up Corona SDK on Mac OS X and Windows
- Install Xcode for Mac OS X
- Create a Hello World program in two lines
- Add devices in the iOS Provisioning Portal
- Load an application to an iOS device
- Load an application to an Android device

Downloading and installing Corona

You have the option of choosing the Mac OS X or Microsoft Windows operating system to develop on. Keep in mind the following system requirements that are needed to run the program. The most compatible version used for this book is Build 2014.2511.
If you are installing Corona for Mac OS X, be sure that your system has the following features:

- Mac OS X 10.9 or later
- An Intel-based system that runs Lion, Mountain Lion, Mavericks, or Yosemite
- A 64-bit CPU (Core 2 Duo)
- OpenGL 2.0 or higher graphics system

If you are running Microsoft Windows, be sure that your system has the following features:

- Windows 8, Windows 7, Vista, or XP (Service Pack 2) operating system
- 1 GHz processor (recommended)
- 80 MB of disk space (minimum)
- 1 GB of RAM (minimum)
- OpenGL 2.1 or higher graphics system (available in most modern Windows systems)
- The 32-bit (x86) version of the Java Development Kit (JDK)
- The Android SDK is not required to create Android device builds with Corona on Mac or Windows

### Time for action – setting up and activating Corona on Mac OS X

Let’s begin by setting up the Corona SDK on our desktop:

1. If you haven’t downloaded the SDK, please do so from http://www.coronalabs.com/downloads/coronasdk. You will have to register as a user before you can access the SDK.

2. The file extension for any Mac program should end in `.dmg`; this is known as an Apple disk image. Once you’ve downloaded the disk image, double-click on the disk image file to mount it. The name should be similar to `CoronaSDK-XXXX.XXXX.dmg`. Once it is loaded, you should see the mounted disk image folder, as shown in the following screenshot:
3. Next, drag the CoronaSDK folder into the Applications folder. This will copy the contents of the Corona folder into /Applications. You will be prompted to enter an administrator password if you are not the main administrator of the account. You will be able to see the CoronaSDK folder in /Applications once it has been successfully installed. For easy access to the folder contents, create an alias by dragging the CoronaSDK folder to the dock of your Mac desktop:
Getting Started with Corona SDK

First-time Corona SDK users will have to perform a quick and easy one-time authorization process before it is accessible. You must be connected to the Internet to complete the authorization process.

1. Launch the Corona simulator in the SDK folder.
2. Assuming this is your first time, you will be presented with an End-user License Agreement (EULA). Once you have accepted the agreement, enter the e-mail you used to register for Corona and the password to activate the SDK. Otherwise, click on Register to create an account.

   If you register with Corona as a single developer, there is no fee to develop on iOS and/or Android devices.

3. Upon successful login, you will get a confirmation dialog to show that the SDK is ready to use:
4. Click on the **Continue** button, and you'll be presented with the Welcome to Corona screen:

![Corona SDK Welcome Screen](image)

**What just happened?**

Setting up Corona SDK on your Mac operating system is as simple as installing any other dedicated Mac program. After you have authorized the SDK on your machine and logged in with your e-mail and password, it will be ready for use. From here on out, every time you launch Corona, it will automatically log in to your account. You will notice that you are greeted with a Corona SDK screen when this happens.

**Time for action – setting up and activating Corona on Windows**

Let’s set up the Corona SDK on our desktop using the following steps:

1. Download the Corona SDK from [http://www.coronalabs.com/downloads/coronaskdd](http://www.coronalabs.com/downloads/coronaskdd). You will have to register as a user before you can access the SDK.

2. The file extension for the Windows version of Corona should end in `.msi`, which is known as a Windows Installer, a component of Windows made to install programs by Microsoft. Double-click on the file. The filename should be similar to `CoronaSDK.msi`.

3. Follow the onscreen directions for installation.
4. Corona will be installed directly into your Programs folder by default. On Microsoft Windows, select Corona Simulator from the list of programs in your Start menu or double-click on the Corona icon on the desktop. Upon successful activation, you should be greeted with the following screen:

5. The process to activate the SDK should be the same as the Mac procedure once you launch Corona for the very first time.

   If you run into issues with images not displaying properly, check to see whether you’re using an up-to-date OpenGL graphics driver, 2.1 or higher.

   Be aware that Corona SDK on Windows can only build for Android devices, not for iOS devices (iPhone, iPad, or iPod Touch). A Mac can build not only for iOS, but also for Android devices in Corona.

6. To create device builds, you need to install Java 6 SDK on your PC. You will need to go to the Oracle website at http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html for the JDK download and click on the Java SE Development Kit 6u45 link.

7. On the next page, select the Accept License Agreement radio button and then click on the Windows x86 link to download the installer. You will be asked to log in or create a user account on the Oracle website if you don't already have one.

8. As soon as the JDK is downloaded, run the installer. Once installed, you'll be able to create device builds for Android on your PC.
What just happened?

Installing the SDK on Windows is a different setup process compared to on the Mac OS X. While executing the installer file, Windows will automatically present a designated location to install the application, such as the Programs folder, so that you won't have to manually choose a destination. Upon successful installation, you will see the Corona SDK icon on your desktop for easy access, or it may be highlighted in the programs list in the Start menu, assuming that you’re accessing it for the first time. When you authorize Corona on your machine and sign in with your login information, it is ready for you to use and will log in automatically every time it is launched.

Using the simulator on Mac and Windows

On Mac OS X, launch Corona SDK by either selecting the Corona terminal or Corona simulator from the Applications directory. Both selections will access the SDK. The Corona simulator will only open the simulator. The Corona terminal will open both the simulator and the terminal window. The terminal is helpful to debug your programs and display simulator errors/warnings and print() messages.

On Microsoft Windows, select the Corona SDK folder and click on Corona Simulator from the list of programs in your Start menu or double-click on the Corona icon on the desktop. The simulator and terminal are always opened together if you are using Windows.

Let’s go over the useful contents contained in the Corona SDK folder (located in Applications/Corona SDK on Mac and Start/All Apps/Corona SDK on Windows):

- **Debugger (Mac)/Corona debugger (Windows):** This is a tool to find and isolate issues in your code.
- **Corona simulator:** This is the environment used to launch your application for testing. It simulates the mobile device you’re developing for on your local computer. On Windows, it will open both the simulator and terminal.
- **Corona terminal:** This launches the Corona simulator and opens a terminal window to display error/warning messages and print() statements. It is very helpful to debug your code, but is only available on a Mac.
- **Simulator:** This has the same properties as the Corona terminal, but is called from the command line and is only available on a Mac.
- **Sample code:** This is a set of sample applications to get you started with Corona. It contains code and art assets to work with.

The Corona SDK window opens automatically when you launch the simulator. You can open a Corona project in the simulator, create a device build for testing or distribution, and view some example games and apps to get yourself familiar with the SDK.
Time for action – viewing a sample project in the simulator

Let’s take a look at the HelloPhysics sample project in the simulator:

1. Click on Corona Simulator in the Corona SDK folder.
2. Click on the Samples link on the Corona SDK window when it launches. In the Open dialog that appears, navigate to Applications/CoronaSDK/SampleCode/Physics/HelloPhysics (Mac) or C:\Program Files (x86)\Corona Labs\Corona SDK\Sample Code\Physics\HelloPhysics (Windows). On a Mac, click on Open, and it will automatically open main.lua. On Windows, double-click on main.lua to open the file. The HelloPhysics application opens and runs in the simulator.

What just happened?

Accessing the SDK through the Corona terminal or Corona simulator is a matter of preference. Many Mac users prefer to use the Corona terminal so that they can track messages outputted to the terminal. When you launch the SDK through the Corona simulator, the simulator will be displayed, but not the terminal window. When Windows users launch the Corona simulator, it will display both the simulator and the terminal window. This is nice to use when you want play around with any of the example applications that are provided by Corona.

The main.lua file is a special filename that tells Corona where to start in a project folder. This file can also load other code files or other program resources such as sounds or graphics.

When you launch the HelloPhysics application in Corona, you will observe a box object fall from the top of the screen in the simulator and collide with a ground object. The transition from launching the main.lua file to viewing the results in the simulator is almost immediate.

Have a go hero – use a different device shell

As you start getting familiar with the Corona simulator, whether you’re in Windows or Mac OS X, a default device is always used when you launch an application. Windows uses the Droid as the default device, while the Mac OS X uses the regular iPhone. Try launching the sample code in a different device shell to view the differences in screen resolution between all the devices the simulator has available.

When porting builds to more than one platform, you’ll have to consider the variety of screen resolutions in both iOS and Android devices. A build is a compiled version of all your source code that is converted into one file. Having your game build configured for multiple platforms broadens the audience reach of your application.
Choosing a text editor

Corona does not have a designated program editor to code in, so you will have to find one that suits your needs.

For Mac OS, TextWrangler is a good one, and it is free too! You can download it from http://www.barebones.com/products/textwrangler/download.html. Other text editors such as BBEdit at http://www.barebones.com/thedeck and TextMate at http://macromates.com/ are great, but you will need to purchase them in order to use them. TextMate is also compatible with the Corona TextMate Bundle, which is available at http://www.ludicroussoftware.com/corona-textmate-bundle/index.html.

For Microsoft Windows, Notepad++ is recommended and can be downloaded from http://notepad-plus-plus.org/.

The following are the text editors that are compatible with both Mac OS and Microsoft Windows:

- Sublime Text (http://www.sublimetext.com)
- Lua Glider (http://www.mydevelopersgames.com/Glider/)
- Outlaw (http://outlawgametools.com/outlaw-code-editor-and-project-manager/)

Any text editor, such as TextEdit for Mac or Notepad for Windows, that is already included in the operating system works as well, but it’ll be easier to work with one that is designed for programming. For Corona, using an editor that supports the Lua syntax highlighting will work the best when coding. Syntax highlighting adds formatting attributes to keywords and punctuation in a way that makes it easier for the reader to separate code from text.

Developing on devices

It is not necessary to download Apple's developer kit, Xcode, or the Android SDK if you only want to use the Corona simulator. In order to build and test your code on an iOS device (iPhone, iPod Touch, and iPad), you will need to sign up as an Apple developer and create and download the provisioning profiles. If you want to develop on Android, you don't need to download the Android SDK unless you want to use the ADB tool to help with installing builds and viewing debug messages.

The Corona SDK starter version allows you to build Adhoc (for iOS) and debug builds (Android) for testing on your own devices. Corona Pro users also get the benefit of special features, such as access to daily builds, premium features, all plugins, and premium support.
Time for action – downloading and installing Xcode

In order to develop any iOS application, you will need to enroll in the Apple Developer Program, which costs $99 a year, and create an account on the Apple website at http://developer.apple.com/programs/ios/ by following these steps:

1. Click on the Enroll Now button and follow Apple’s instructions to complete the process. When adding a program, select iOS Developer Program.

2. When you have completed your enrollment, click on the iOS link under the section marked Dev Centers.

3. Scroll down to the Downloads section and download the current Xcode, or you can download Xcode from the Mac App Store.

4. Once you have fully downloaded Xcode, double-click on Xcode from /Applications/Xcode. You will be asked to authenticate as an administrative user:
5. After you have entered your credentials, click on the OK button to complete the installation. You will see the following screen:

![System Component Installation](image)

6. When you have installed the Xcode developer tools, you can access the documentation by launching Xcode and choosing any of the items in the Help menu. Developer applications such as Xcode and Instruments are installed in /Applications/Xcode. You can drag these app icons to your dock for convenient accessibility.

What just happened?
We just went through the steps of how to install Xcode for Mac OS X. By enrolling in the Apple Developer Program, you will have access to up-to-date development tools on the website. Remember that to continue being an Apple developer, you have to pay a fee of $99 annually to keep your subscription.

The Xcode file is quite large, so it will take a bit of time to download, depending on how fast your Internet connection is. Once your installation is complete, Xcode will be ready to go.
Getting Started with Corona SDK

Time for action – creating a Hello World application in two lines of code

Now that we have the simulator and text editors set up, let’s start making our very first Corona program! The first program that we will make is called Hello World. It is a traditional program that many people learn when starting a new programming language.

1. Open your preferred text editor and type the following lines:
   ```lua
   textObject = display.newText( "Hello World!", 160, 80, native.systemFont, 36 )
   textObject: setFillColor ( 1, 1, 1 )
   ```

2. Next, create a folder on your desktop called Hello World. Save the preceding text as a file named main.lua to the location of your project folder.

3. Launch Corona. You will be greeted with the Corona SDK screen. Click on Open and navigate to the Hello World folder you just created. You should see your main.lua file in this folder:

![Image of the Corona SDK screen with the Hello World folder selected.]
4. On a Mac, click on the **Open** button. On Windows, select the `main.lua` file and click on the **Open** button. You'll see your new program running in the Corona simulator:

---

**Downloading the example code**

You can download the example code files for all Packt Publishing books you have purchased from your account at [http://www.packtpub.com](http://www.packtpub.com). If you purchased this book elsewhere, you can visit [http://www.packtpub.com/support](http://www.packtpub.com/support) and register to have the files e-mailed directly to you.
Time for action – modifying our application

Before we dive into more complex examples, let's alter some minor things in our program by performing the following steps:

1. Let's alter the second line of `main.lua` to display as follows:

   ```lua
   textObject = display.newText( "Hello World!", 160, 80, native.systemFont, 36 )
   textObject:setFillColor( 0.9, 0.98 ,0 )
   ```

2. Save your file and go back to the Corona simulator. The simulator will detect a change from your file and automatically relaunch with the changes. If the simulator doesn't relaunch automatically upon saving your file, press `Command + R` (Mac) / `Ctrl + R` (Windows). You will see the following output on the screen:
As you continue learning more Corona functions, you'll notice that some of the text values will be optional. In this case, we need to use five values.

### Time for action – applying a new font name to our application

Now, let's play around with the font name by performing the following steps:

1. Change the first line to the following line of code:
   ```lua
   textObject = display.newText( "Hello World!", 160, 80, "Times New Roman", 36 )
   ```

2. Be sure to save your `main.lua` file after making any alterations; then, press `Command + R` (Mac) / `Ctrl + R` (Windows) in Corona to relaunch the simulator to view the new font. If you're using a Mac, usually, the simulator automatically relaunches after saving your file, or it may ask you if you want to relaunch the program. You can see the new font in the simulator:
What just happened?

You have now made your first complete mobile application! What’s even more amazing is that this is a completed iPhone, iPad, and Android application. This two-line program will actually install and run on your iOS/Android device if you were to create a build. You have now seen what the basic workflow in Corona is like.

If you take a look at line 2 in your main.lua file, you will notice that setFillColor alters the color of the text for Hello World!

Colors are made up of three sets of RGB numbers that represent the amount of red, green and blue contained within a color. They are displayed with three numbers, with values ranging from 0 to 1. For example, the color black would be (0,0,0), blue would be (0,0,1), and the color white (0.6, 0.4, 0.8).

Continue playing around with different color values to see the different results. You can see the alterations to the code in the simulator when you save your main.lua file and relaunch Corona.

When you view the first line from the main.lua file, you will notice that newText() is called by the display object. The returning reference is textObject. The newText() function returns an object that will represent the text on the screen. The newText() function is a part of the display library.

When you want to access the display properties of newText, type in display.newText. The two numbers after Hello World! control the horizontal and vertical positions of the text on the screen in pixels. The next item specifies the font. We used the name native.systemFont, which, by default, refers to the standard font on the current device. For example, the iPhone’s default font is Helvetica. You can use any standard font name, such as Times New Roman that is used in the preceding example. The last number used is the font size.

Have a go hero – adding more text objects

Now that you’re starting to get a taste of coding, try following these steps in your current project file:

1. Create a new display object using a different font and text color. Ensure it displays below the Hello World! text. Make sure that your new display object has a different object name.

2. Continue changing the values of the current display object, textObject. Alter the x and y coordinates, the string text, font name, and even the font size.
3. While `object:setFillColor( r, g, b )` sets the color of the text, there is an optional parameter you can add that controls the opacity of the text. Try using `object:setFillColor( r, g, b [, a] )`. The values available for `a` also range between 0 to 1 (1 is opaque, which is the default value). Observe the results of your text color.

**Testing our application on an iOS device**

If you are only interested in testing the application on an Android device, skip past this section of the chapter to *Testing our application on an Android device*. Before we can upload our first Hello World application on an iOS device, we need to log in into our Apple developer account so that we can create and install our signing certificates on our development machine. If you haven’t created a developer account yet, do so by going to [http://developer.apple.com/programs/ios/](http://developer.apple.com/programs/ios/). Remember that there is a fee of $99 a year to become an Apple developer.

The Apple developer account is only applied to users developing on Mac OS X. Make sure that your version of Xcode is the same or newer than the version of the OS on your phone. For example, if you have version 5.0 of the iPhone OS installed, you will need Xcode that is bundled with the iOS SDK version 5.0 or later.

**Time for action – obtaining the iOS developer certificate**

Make sure that you’re signed up for the developer program; you will need to use the Keychain Access tool located in `/Applications/Utilities` so that you can create a certificate request. A valid certificate must sign all iOS applications before they can be run on an Apple device in order to do any kind of testing. The following steps will show you how to create an iOS developer certificate:

1. Go to Keychain Access | Certificate Assistant | Request a Certificate From a Certificate Authority:

![Keychain Access screenshot]
2. In the User Email Address field, type in the e-mail address you used when you registered as an iOS developer. For Common Name, enter your name or team name. Make sure that the name entered matches the information that was submitted when you registered as an iOS developer. The CA Email Address field does not need to be filled in, so you can leave it blank. We are not e-mailing the certificate to a Certificate Authority (CA). Check Saved to disk and Let me specify key pair information. When you click on Continue, you will be asked to choose a save location. Save your file at a destination where you can locate it easily, such as your desktop.
3. In the following window, make sure that 2048 bits is selected for the **Key Size** and **RSA** for the **Algorithm**, and then click on **Continue**. This will generate the key and save it to the location you specified. Click on **Done** in the next window.
4. Next, go to the Apple developer website at http://developer.apple.com/, click on iOS Dev Center, and log in to your developer account. Select Certificates, Identifiers & Profiles under iOS Developer Program on the right-hand side of the screen and navigate to Certificates under iOS Apps. Select the + icon on the right-hand side of the page. Under Development, click on the iOS App Development radio button. Click on the Continue button till you reach the screen to generate your certificate:

5. Click on the Choose File button and locate your certificate file that you saved to your desktop, and then, click on the Generate button.

6. Upon hitting Generate, you will get the e-mail notification you specified in the CA request form from Keychain Access, or you can download it directly from the developer portal. The person who created the certificate will get this e-mail and can approve the request by hitting the Approve button.
7. Click on the Download button and save the certificate to a location that is easy to find. Once this is completed, double-click on the file, and the certificate will be added automatically in the Keychain Access.

**What just happened?**

We now have a valid certificate for iOS devices. The iOS Development Certificate is used for development purposes only and valid for about a year. The key pair is made up of your public and private keys. The private key is what allows Xcode to sign iOS applications. Private keys are available only to the key pair creator and are stored in the system keychain of the creator’s machine.
Adding iOS devices

You are allowed to assign up to 100 devices for development and testing purposes in the iPhone Developer Program. To register a device, you will need the **Unique Device Identification (UDID)** number. You can find this in iTunes and Xcode.

**Xcode**

To find out your device’s UDID, connect your device to your Mac and open Xcode. In Xcode, navigate to the menu bar, select **Window**, and then click on **Organizer**. The 40 hex character string in the **Identifier** field is your device’s UDID. Once the **Organizer** window is open, you should see the name of your device in the **Devices** list on the left-hand side. Click on it and select the identifier with your mouse, copying it to the clipboard.

Usually, when you connect a device to **Organizer** for the first time, you'll receive a button notification that says **Use for Development**. Select it and Xcode will do most of the provisioning work for your device in the iOS Provisioning Portal.
iTunes
With your device connected, open iTunes and click on your device in the device list. Select the Summary tab. Click on the Serial Number label to show the Identifier field and the 40-character UDID. Press Command + C to copy the UDID to your clipboard.

Time for action – adding/registering your iOS device
To add a device to use for development/testing, perform the following steps:

1. Select Devices in the Developer Portal and click on the + icon to register a new device. Select the Register Device radio button to register one device.

2. Create a name for your device in the Name field and put your device’s UDID in the UDID field by pressing Command + V to paste the number you have saved on the clipboard.
3. Click on Continue when you are done and click on Register once you have verified the device information.

Time for action – creating an App ID

Now that you have added a device to the portal, you will need to create an App ID. An App ID has a unique 10-character Apple ID Prefix generated by Apple and an Apple ID Suffix that is created by the Team Admin in the Provisioning Portal. An App ID could looks like this: 7R456G1254.com.companyname.YourApplication. To create a new App ID, use these steps:
1. Click on App IDs in the Identifiers section of the portal and select the + icon.

![App ID Description](image)

2. Fill out the App ID Description field with the name of your application.

3. You are already assigned an Apple ID Prefix (also known as a Team ID).
4. In the App ID Suffix field, specify a unique identifier for your app. It is up to you how you want to identify your app, but it is recommended that you use the reverse-domain style string, that is, `com.domainname.appname`. Click on Continue and then on Submit to create your App ID.

You can create a wildcard character in the bundle identifier that you can share among a suite of applications using the same Keychain access. To do this, simply create a single App ID with an asterisk (*) at the end. You would place this in the field for the bundle identifier either by itself or at the end of your string, for example, `com.domainname.*`. More information on this topic can be found in the App IDs section of the iOS Provisioning Portal at https://developer.apple.com/ios/manage/bundles/howto.action.

**What just happened?**

All UDIDs are unique on every device, and we can locate them in Xcode and iTunes. When we added a device in the iOS Provisioning Portal, we took the UDID, which consists of 40 hex characters, and made sure we created a device name so that we could identify what we’re using for development.

We now have an App ID for the applications we want to install on a device. An App ID is a unique identifier that iOS uses to allow your application to connect to the Apple Push Notification service, share keychain data between applications, and communicate with external hardware accessories you wish to pair your iOS application with.

**Provisioning profiles**

A **provisioning profile** is a collection of digital entities that uniquely ties apps and devices to an authorized iOS Development Team and enables a device to be used to test a particular app. Provisioning profiles define the relationship between apps, devices, and development teams. They need to be defined for both the development and distribution aspects of an app.
Time for action – creating a provisioning profile

To create a provisioning profile, go to the Provisioning Profiles section of the Developer Portal and click on the + icon. Perform the following steps:

1. Select the iOS App Development radio button under the Development section and then select Continue.
2. Select the App ID you created for your application in the pull-down menu and click on Continue.
3. Select the certificate you wish to include in the provisioning profile and then click on Continue.
4. Select the devices you wish to authorize for this profile and click on Continue.
5. Create a Profile Name and click on the Generate button when you are done:

![Provisioning Profile Creation](image-url)
6. Click on the Download button. While the file is downloading, launch Xcode if it's not already open and press Shift + Command + 2 on the keyboard to open Organizer.

7. Under Library, select the Provisioning Profiles section. Drag your downloaded .mobileprovision file to the Organizer window. This will automatically copy your .mobileprovision file to the proper directory.

![Organizer window](image)

**What just happened?**

Devices that have permission within the provisioning profile can be used for testing as long as the certificates are included in the profile. One device can have multiple provisioning profiles installed.

**Application icon**

Currently, our app has no icon image to display on the device. By default, if there is no icon image set for the application, you will see a light gray box displayed along with your application name below it once the build has been loaded to your device. So, launch your preferred creative developmental tool and let's create a simple image.
The application icon for standard resolution iPad2 or iPad mini image file is 76 x 76 px PNG. The image should always be saved as Icon.png and must be located in your current project folder. iPhone/iPod touch devices that support retina display need an additional high resolution 120 x 120 px and iPad or iPad mini have an icon of 152 x 152 px named as Icon@2x.png.

The contents of your current project folder should look like this:

```
Hello World/       name of your project folder
 Icon.png           required for iPhone/iPod/iPad
 Icon@2x.png   required for iPhone/iPod with Retina display
 main.lua
```

In order to distribute your app, the App Store requires a 1024 x 1024 pixel version of the icon. It is best to create your icon at a higher resolution first. Refer to the Apple iOS Human Interface Guidelines for the latest official App Store requirements at http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html.

Creating an application icon is a visual representation of your application name. You will be able to view the icon on your device once you compile a build together. The icon is also the image that launches your application.

**Creating the Hello World build for iOS**

We are now set to build our Hello World application for our device. Since we have our provisioning profiles in place, the build process from here on out is pretty simple. Make sure that you are connected to the Internet before creating a device build. You can build your application for testing in the Xcode simulator or on a device.

**Time for action – creating an iOS build**

Follow these steps to create a new iOS build in Corona SDK:

1. Open the Corona simulator and select Open.
2. Navigate to your Hello World application and select your main.lua file.
3. Once the application is launched in the simulator, go to the Corona Simulator menu bar and navigate to File | Build | iOS or press Command + B on your keyboard. The following dialog box will appear:

4. Create a name for your app in the Application Name field. We can keep the same name, Hello World. In the Version field, keep the number at 1.0. In order to test the app in the Xcode simulator, select Xcode Simulator from the Build For drop-down menu. If you want to build for the device, choose Device to build an app bundle. Next, select the target device (iPhone or iPad) from the Supported Devices drop-down menu. From the Code Signing Identity drop-down menu, choose the provisioning file you created for the specified devices you are building for. It is the same name as Profile Name in the iOS Provisioning Portal on the Apple developer website. In the Save to folder section, click on Browse and choose where you would like your application to be saved.

If all the information has been confirmed in the dialog box, click on the Build button.

It is more convenient to set your application to save on the Desktop; this way, it is easy to find.
What just happened?

Congratulations! You have now created your very first iOS application file that can be uploaded to your device. As you start developing apps for distribution, you will want to create new versions of your application so that you can keep track of the changes in every new build you make. All the information from your Provisioning Profile was created in the iOS Provisioning Portal and applied to the build. Once Corona has finished compiling the build, the application should be located in the folder you saved it under.

Time for action – loading an app on your iOS device

Select your Hello World build you created and choose either iTunes or Xcode to load your app onto your iOS device. They can be used to transfer the application file.

If using iTunes, drag your build into your iTunes Library and then sync your device normally, as shown in the following screenshot:
Another way to install your app onto your device is to use Xcode, since it provides a convenient method to install iOS device applications. Perform the following steps:

1. With the device connected, open Xcode’s Organizer from the menu bar by going to Window | Organizer and navigate to your connected device under the Devices list on the left-hand side.

2. If a proper connection is established, you will see a green indicator. If it is yellow after a couple minutes, try powering the device off and on again or disconnect the device and connect it again. This will usually establish a proper connection.

3. Simply drag your build file and drop it into the Applications area of the Organizer window, and it will automatically install on your device.
What just happened?

We just learned two different ways of loading an application build to an iOS device: using iTunes and using Xcode.

Using iTunes provides a simple drag and drop feature into your library and then allows you to transfer the build as long as your device is synced up.

The Xcode method is probably the easiest and most common way to load a build to a device. As long as your device is connected properly and ready to use in the Organizer, you drag and drop the build to applications and it loads automatically.

Testing our application on an Android device

Creating and testing our builds on an Android device does not require a developer account like how Apple does for iOS devices. The only tools you need to build for Android are a PC or Mac, Corona SDK, JDK6 installed, and an Android device. If you plan on submitting an app to the Google Play Store, you’ll need to sign up as a Google Play Developer at https://play.google.com/apps/publish/signup/. There is a one-time $25 registration fee you have to pay if you want to publish software on the Google Play Store.

Creating the Hello World build for Android

Building our Hello World application is fairly simple since we do not have to create a unique keystore or key alias for debug builds. When you’re ready to submit an application to the Google Play Store, you’ll need to create a release build and generate your own private key to sign your app. We’ll discuss in further detail about release builds and private keys later on in this book.

Time for action – creating an Android build

Follow these steps to create a new Android build in the Corona SDK:

1. Launch the Corona Simulator and select Simulator.
2. Navigate to your Hello World application and select your main.lua file.
3. Once your application is running in the simulator, go to the Corona Simulator menu bar and navigate to File | Build For | Android (Windows) / Shift + Command + B on your keyboard (Mac). The following dialog box will appear:

![Build for Android dialog box](image)

4. Create a name for your app in the Application Name field. We can keep the same name, Hello World. In the Version Code field, set the number to 1 if that is not already the default number. This specific field must always be an integer and is not visible to users. In the Version Name field, keep the number at 1.0. This attribute is the string shown to users. In the Package field, you will need to specify a name that uses the traditional Java scheme, which is basically the reverse format of your domain name; for example, com.mycompany.app.helloworld would work as a package name. Project Path shows the location of your project folder. Minimum SDK Version currently supports Android 2.3.3 and newer devices running the ArmV7 processor. In the Target App Store pull-down menu, the default store can stay as Google Play. In the Keystore field, you will be signing your build with the Debug keystore that is already provided in Corona. In the Key Alias field, select androiddebugkey from the pull-down menu if it's not selected. In the Save to Folder section, click on Browse and choose where you'd like your application to be saved to.
5. If all the information has been confirmed in the dialog box, click on the Build button.

For more information on Java package names, see the section on Unique Package Names in the Java documentation at http://java.sun.com/docs/books/jls/third_edition/html/packages.html#40169.

What just happened?

You have created your very first Android build! See how simple that was? Since the Corona SDK already provides the Debug keystore and androiddebugkey key alias in the engine, most of the signing work is already done for you. The only requirement from you is to fill out the build information for your application and click on the Build button to make a debug build. Your Hello World application will save as an .apk file at the location you designated. The file name will appear as Hello World.apk.

Time for action – loading an app on your Android device

There are several ways to load your Hello World build to your Android device, and these ways do not require you to download the Android SDK. Here are some simple methods.

A convenient method is through Dropbox. You can create an account at https://www.dropbox.com. Dropbox is a free service that lets you upload/download files on your PC/Mac and mobile devices. Perform the following steps to load the Hello World build using Dropbox:

1. Download the Dropbox installer and install it on your computer. Also, download the mobile app from the Google Play Store (which is also free) on your device and install it.

2. Log in to your Dropbox account on your computer and mobile device. From your computer, upload your Hello World.apk file.

3. Once it has finished uploading, go to the Dropbox app on your device and select your Hello World.apk file. You will be greeted with a screen that will ask you if you want to install the application. Select the Install button. Assuming that it installs correctly, another screen will appear saying Application installed, and you can launch your Hello World app by pressing the Open button that is available.
Another method to upload an .apk file onto your device is to transfer it to an SD card via the USB interface. If your device doesn't come with some kind of file manager application, a great one you can download from the Google Play Store is ASTRO File Manager, which can be found at https://play.google.com/store/apps/details?id=com.metago.astro. You can always do a normal search for the preceding app or similar apk installers through the Google Play app on your device. To transfer the .apk file to the SD card, perform the following steps:

1. In your device's Settings, select Applications and then select Development. Tap on USB Debugging if the mode is not active.

2. Go back a couple of screens to the Applications section. Enable Unknown Sources if it is not already active. This will allow you to install any non-market application (that is, debug builds). Select the home button on your device when done.

3. Connect the device to your computer with a USB cable. You will see a new notification that a new drive has connected to your PC or Mac. Access the SD drive and create a new folder. Name the folder to something you can identify easily for your Android builds. Drag and drop your Hello World.apk file from the desktop to the folder.

4. Eject the drive from your desktop and disconnect your device from the USB cable. Launch ASTRO File Manager or use whichever app you decided to download from the Google Play Store. In ASTRO, select File Manager, search for the folder you added to your SD card, and select it. You will see your Hello World.apk file. Select the file, and a prompt will appear asking you to install it. Select the Install button, and you should see your Hello World application appear in the Apps folder of your device.
One of the easiest methods is through Gmail. If you don't already have a Gmail account, create one at https://mail.google.com/. Perform the following steps to send the .apk file on your Gmail account:

1. Log in to your account, compose a new e-mail, and attach your Hello World.apk file to the message.
2. Address the recipient of the message to your own e-mail address and send it.
3. On your Android device, make sure you have your e-mail account linked. As soon as you receive the message, open the e-mail. You’re given the option to install the application on your device. There will be an Install button or something similar displayed.

What just happened?

We just learned several ways of loading an .apk file to an Android device. The preceding methods are some of the easiest ways to load an application quickly without running into any problems.

Using the file manager method allows you to access your .apk files easily without requiring any carrier data or Wi-Fi connection. Using a USB cable that is compatible with your device and connecting it to your computer is a simple drag and drop procedure.

The Dropbox method is the most convenient once you have it set up on your computer and your mobile devices. All you have to do is drag and drop your .apk file to your account folder, and it’s instantly accessible to any device with the Dropbox app installed. You can also share your files through a download link, which is also another great feature provided by Dropbox.

Setting up a Gmail account and sending your .apk files as an attachment to yourself is simple if you don't want to download any file managers or other programs to your device and computer. The only thing you have to remember is that you can’t e-mail an attachment over the size of 25 MB in Gmail.

Pop quiz – understanding Corona

Q1. What is true about using the Corona simulator?

1. You need a main.lua file to launch your application.
2. The Corona SDK only runs on Mac OS X.
3. The Corona terminal doesn’t launch the simulator.
4. None of the above.
Q2. How many iOS devices can you use for development in the iPhone Developer Program?

1. 50.
2. 75.
3. 5.
4. 100.

Q3. What does the version code have to be when building for Android in Corona SDK?

1. A string.
2. An integer.
3. It has to follow the Java scheme format.
4. None of the above.

Summary

In this chapter, we covered some of the necessary tools needed to start developing applications for Corona SDK. Whether you are working on a Mac OS X or Microsoft Windows, you will notice the similarities in working on both operating systems and how simple it is to run the Corona SDK.

To further familiarize yourself with Corona, try doing the following:

- Take time to look into the sample codes provided by Corona to view the capabilities of the SDK
- Feel free to alter any of the sample codes to your liking to get a better understanding of programming in Lua
- Whether you’re working on iOS (if you’re a registered Apple developer) or Android, try installing any of the sample codes on your device to see how the applications work outside a simulator environment
- Take a look at the Corona Labs Forums at http://forums.coronalabs.com/ and browse through the latest discussions on Corona development by fellow Corona SDK developers and personnel

Now that you understand the process of how to display objects in Corona, we'll be able to dive into other functions that will help create an operational mobile game.

In the next chapter, we'll look at further details of the Lua programming language, and you will learn simple coding techniques that are similar to the sample code in Corona. You will get a better understanding of the Lua syntax and notice how fast and easy it is to learn compared to other programming languages. So, let's get started!
Where to buy this book


Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.