Learning Mobile Application & Game Development with 
Corona SDK

Brian G. Burton, Ed.D.
Table of Contents
About the Author ............................................................................................................. viii
Dedication ....................................................................................................................... viii
Preface .............................................................................................................................. x
Who This Book Is For ..................................................................................................... x
How This Book Is Organized ....................................................................................... xi
Conventions Used In This Book .................................................................................. xi
Using Code Examples and Fair Use Laws ................................................................... xi
Why didn’t I use _____ for ____ ................................................................................ xii
Appendices ................................................................................................................... xii
How to Contact Us ....................................................................................................... xii
Why I Chose to Indie-Publish ...................................................................................... xii
Chapter 1 Introduction to Mobile App Development .................................................... 1
Learning Objectives ..................................................................................................... 1
Introduction to Mobile Application Development ....................................................... 1
First, an assumption .................................................................................................... 1
Mobile Operating Systems ......................................................................................... 1
  Google Android ........................................................................................................ 2
  Apple iOS .................................................................................................................. 2
  Blackberry ................................................................................................................ 2
  Windows 8 ................................................................................................................ 3
Cross-Platform Development ...................................................................................... 3
One Last Note Before We Get Started ....................................................................... 3
Developing Mobile Applications ................................................................................... 3
  Software That You Will Need .................................................................................. 5
  Setting Up Your Software ....................................................................................... 6
Setting up Your Hardware ........................................................................................... 6
Test Devices .................................................................................................................. 6
  Android .................................................................................................................... 7
  iOS ............................................................................................................................ 7
Book Examples and Graphics ...................................................................................... 7
Why become certified? ........................................................................................................51
About the Author

Brian Gene Burton, Ed.D. is a teacher, author, and game developer. Beside writing “Beginning Mobile App Development with Corona” and contributing to several academic books on serious games and learning in virtual worlds, Dr. Burton has created game development degrees at two universities and enjoys researching and playing virtual environments. Dr. Burton presents and publishes internationally on his research and enjoys sharing what he has learned about game and mobile development. When not traveling or teaching, he can be found at his home in the Ozark Mountains of Missouri with his beautiful wife of over 25 years, Rosemary.

Dedication

I dedicate this book to my loving wife whose support and encouragement kept me focused and writing. Thank you for keeping me focused and not running off on rabbit trails!

A special thank you to my students and the Corona community for their support and requests for specific details and editorial comments that helped so much with the development of this book.

A big thank you to Michael Kelly for providing great advice on how to make several of the chapters more understandable and a better learning experience.

All other graphics (unless specified) and cover designed by Brandon Burton (http://www.geeklyentertainment.com).

Copyediting and formatting assistance provided by Brianna Burton (http://www.LiteraryDiaries.com).
Preface

Welcome

Welcome to Learning Mobile Application Development! This book is the result of years of developing and teaching mobile application development. When I began writing *Beginning Mobile App Development with Corona* in 2011, it was with a specific audience in mind; my current and future students who are experienced programmers. But before I had even finished that textbook, I had begun hearing from another vocal group, (those who did not have any programming experience but had fantastic ideas for apps that they wanted to develop). Agreeing with Corona Labs founder Walter Luh that everyone should be able to develop mobile apps, I began working on tutorials and lessons for those in this underserved audience.

I selected the Corona SDK to use as the method for teaching programming and mobile app development for several good reasons: Corona by Corona Labs ([http://www.CoronaLabs.com](http://www.coronalabs.com)) was developed from the beginning around the concept that anyone can make mobile apps. Using the Lua scripting language, Corona is easy to learn yet powerful enough to allow you to create great, powerful apps and fast, responsive games. Finally, the international community that has formed around Corona is one of the best, developer friendly environments that I have ever experienced in my 25+ years of programming. Corona User Groups and meet-ups are happening all over the world thanks to a devoted network of ambassadors. If there isn't a Corona group in your community, contact Corona Labs and ask about starting one! I hope that you enjoy learning to develop your own mobile apps!

Best wishes and looking forward to seeing your app in the stores,

Brian G. Burton, Ed. D.

Who This Book Is For

While my focus and impetus for writing this book is that it be used as a textbook, I have also written it with the understanding that many (hopefully) are just interested in learning more about the Corona SDK and want to develop for multiple mobile devices at the same time. As I wrote this book, it was with the expectation that this is your first time programming or you are not an experienced programmer. If you are an experienced programmer and would like to learn Corona, I would recommend my first textbook, *Beginning Mobile App Development with Corona*. 
How This Book Is Organized
While writing this book, I have kept the traditional 16-week U.S. college semester in mind, assuming one chapter per week. I have included additional chapters to meet specific state requirements at the high school level. These final chapters should allow faculty to create a course that fits their specific needs or allow high school classes to make use of the textbook for the entire year. While that doesn’t work for everyone, it should be enough for most people to get started with mobile development using the Corona SDK. My first draft ended up with more than 28 chapters. After reorganizing content and continuing to develop, we are now down to 19 chapters with an additional chapter on great resources and a couple of appendices for the specific installation and app publishing requirements for Apple or Android. Chapters 17 and 18 are to meet specific Texas state requirements for teaching mobile application development at the high school level. Chapters 17 and 18 were developed so that they can be covered at any time after Chapter 7.

Conventions Used In This Book
Throughout the book I will use Courier New font to denote code that should be typed in exactly. When you find examples that are in Courier New, Italic, you will need to enter your own value.

Using Code Examples and Fair Use Laws
This book was written to help you learn to develop applications and games with the Corona SDK. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission for reproducing a portion of the code. You don’t need to ask permission to write an app that uses large chunks of code.

Now, on the other extreme, if apps appear that exactly reproduce the examples from this book, I will not be a happy camper and will contact the app store that the offending app is a violation of copyright. I don’t have issues with using the examples as a starting point, but take the app much further; be original! Answering questions by citing this book or quoting examples does not require permission (but I would appreciate the citation).

I reserve all rights for selling or distributing the examples in any format provided in this book. If you’re not sure if your use falls outside of the fair use laws, please feel free to contact me at: DrBurton@BurtonsMediaGroup.com.
Why didn’t I use ____ for ____
There are a lot of great products available that can help the budding programmer/developer get their work done much faster (see chapter 17 for a short list). As this book is aimed at high school and college students, or people just getting started in app development, I tried not to use outside tools. If a tool was required to get the project done, I tried to use only free or low cost tools. If I didn’t use one of your favorites, I either 1) didn’t know the tool existed; 2) was unable to get an evaluation copy of the software in a timely fashion; or 3) just didn’t like that tool (probably the first or second option). If you know of a great tool that can save time and money to developers, please share it with the world in the discussion board on this books site: http://www.BurtonsMediaGroup.com/forum.

Appendices
Appendix A discusses how to download and install Corona SDK. Appendix B covers configuring xCode and setting up provisioning profiles with Apple. In Appendix C, I cover installing Android APK and configuring Keystores for provisioning your Android based apps. Appendices D and E were provided by Corona Labs (with minor copy editing provided by Dr. Burton) and covers the Lua scripting language in greater detail. The final appendix, F, is a review for the Corona Certification examination which was written by the author.

How to Contact Us
Please address any comments or questions to the books website: http://www.BurtonsMediaGroup.com/books or email DrBurton@BurtonsMediaGroup.com. You will find discussion forums for this and other books at http://www.BurtonsMediaGroup.com/forum.

Why I Chose to Indie-Publish
The decision to Indie-publish was reached after a great deal of consideration. While there were numerous publishers interested (both academic and technical), I decided to release this first edition without the use of traditional publishers. There are many reasons why I made this decision, even though it will most likely lead to fewer sells.

First among my concerns was the price of the final book. I am sick of seeing textbooks at $100+. I feel such pricing places an undue burden upon students and schools. While publishers have cut the price slightly with the advent of eBooks and eTextbooks, it hasn’t
been enough in my opinion. By indie-publishing, I am only at the mercy of Apple, Amazon, Google, and Kobo.

My second concern was how rapidly software environments change. I personally hate having to purchase a new book for each major revision of software. I have stacks of books that are now completely useless. I decided to publish this as an eTextbook, which allows me to update and provide it to you, the reader, more rapidly. I will provide the updates between editions to the eBook to everyone who purchases the eTextbook through my website: http://www.BurtonsMediaGroup.com/books/book-update/

However, if you received a copy of this book, either through a torrent or a friend, please purchase your own copy through my website. This will provide you with the most recent version of the textbook and encourage me to continue to keep it updated. While I am doing this to help my students, I have bills to pay, and my wife is really good at keeping my ‘honey-do’ list up-to-date. Help me to avoid that list by buying a legitimate copy of this book (I don’t have to work on her list if I’m writing or editing).

On the downside of indie-publishing, I do NOT have a team of people to proof and double check everything in this book. I am sure that typos were entered by gremlins during the night. To make things more challenging, I have dyslexia. I did hire a person to proof the final version of the book, but having read many books that were published by major companies and finding errors in their books, I am sure that errors remain in this one. Please let me know if you find a typo via email or through the book's forum site: http://www.burtonsmediagroup.com/forum and I will make sure that it is fixed in the next update.
Chapter 1 Introduction to Mobile App Development

Learning Objectives
In chapter 1 we will learn:

- The different mobile operating systems
- The life-cycle of a mobile app project
- Software needed to make a mobile app
- How to make your first app
- Troubleshooting basics
- About Objects, Methods, and Properties

Introduction to Mobile Application Development
You have been working on your killer mobile app idea for days. It is completely original; no one has done anything like what you have planned before! Just one problem... How do you get your idea on the tablet or smart phone?

Don’t worry! You are in the right place! This book was specifically written for you! In the following pages we will walk through all the decisions and processes that you will need to address to develop and sell your app.

To begin, we will examine the various options you have for developing your app.

First, an assumption...
This textbook is designed for the person who has no or very limited previous programming experience. If you are an experienced programmer, I would recommend that you use one of my other books such as "Beginning Mobile App Development with Corona," a textbook that makes the assumption that you understand the fundamentals of programming and are comfortable creating your own loops, decision statements, and functions. If that sounds like a foreign language to you, then you are in the right place!

Mobile Operating Systems
The smart-phone and tablet world are divided by the operating system (OS) that runs on the device. An operating system handles all of the directions from the apps that are running and what the user is tapping on as well as connecting to the internet, handling text messages and phone calls. It is a busy system! As we begin to develop applications for smart phones and tablets, it is important that we keep in mind the devices that our apps will be running on. Below I have listed the four most popular Operating Systems for mobile devices:
Google Android
The Android OS was developed by Google. The first beta was released in 2007 and it has been regularly updated every few months. At the time of this writing Jelly Bean (v.4.1) is the current stable release.

Native development for Android devices is done with Google’s APK (Android Programming Kit). But many tools (including Corona, which we will be using) also allow you to build android applications.

The Android OS is available on smart phones and tablets. It is the foundation of Amazon’s Kindle Fire and Barnes & Noble’s Nook tablets and enjoys a devoted following. Android Apps can be sold on Amazon, Barnes & Noble, and Google Play. Watch for Android to begin showing up on gaming consoles such as the Ouya.

Apple iOS
Apple’s iOS (previously known as iPhone OS) was first released in 2007. Apple iOS can only run on Apple hardware according to the licensing agreement that you sign when you download the software. At the time of writing v.6 of iOS is now available and has been widely adopted by the Apple community.

Apple iOS runs on the iPhone, iPod Touch, and iPad. Apple iOS apps can only be sold through the iTunes store.

Apple iOS native app development is done with xCode. However, like Android, there are many tools available (including Corona, which will be the primary tool that we use in this book) that allow you to build for multiple operating systems/devices at the same time.

Blackberry
At one time, Blackberry smartphones by Research In Motion were the smartphone to own. They controlled a significant market share prior to the advent of the iPhone and Android smartphones. It is difficult to say the impact that Blackberry will have in the future; they are releasing a new SDK called Blackberry 10 and it is supposed to allow Android apps to run on the platform.

At this point in time, only a few multi-platform app development tools do support Blackberry.
**Windows 8**
Microsoft has dabbled in the smartphone and tablet arena for years (one of the first tablets used Microsoft). However, it has not found much traction. Microsoft hopes to change all of that with Windows 8 and the release of Surface tablet (which Microsoft refers to as a PC). This is a new Surface tablet. Previously Microsoft called their interactive tabletop Surface. Native app development for Windows 8 devices is done in C# using Visual Studio.

**Cross-Platform Development**
Perhaps I am just lazy, but I don't like to do things twice. When I have created a great app (or even a not-so-great app) for the iPhone or iPad, I don’t want to spend weeks completely re-writing all of the programming code just so that it can be deployed to a different set of devices. When I first got started in app development, this is exactly what you had to do. Fortunately, there are now many tools that allow the developer to create apps for more than one operating system.

We will be using the Corona SDK by Corona Labs (http://www.CoronaLabs.com) for the majority of our development. Corona Labs was created in 2008 as a venture-backed company in Palo Alto, California. Before Corona, the Corona Labs team was responsible for creating many of the industry standard tools that I am sure you are familiar with. In the time that I have been developing apps with Corona, I have found Corona Labs to be one of the most friendly and helpful businesses that I have had the pleasure of working with. In addition, online community is unusually friendly and supportive. If you decide to join the Corona community, be sure to continue this great spirit of helpfulness!

At the time of this writing, Corona supports iOS & Android app development, with additional operating system support planned for the near future.

**One Last Note Before We Get Started**
Corona Labs has recently begun offering the option of becoming a Corona Certified Developer (CCD). As the author of the test and study guide, I want to let you know that if you successfully complete this course, you will have covered everything you need to become a CCD except one; publishing your own app to a store, but we’ll cover how to do that so that you can earn your CCD.

**Developing Mobile Applications**
From concept to store, an app development project goes through eight stages:
1) **Design phase** – This is the entry point that many apps never get past. You have an idea for what you want the app to do. The first thing to determine is if the device you hope to place the app on can even do the tasks that you are requiring of it. If it can, is the project feasible? There are many considerations including development cost, software cost, and time; I think you get the picture. If the project appears to be capable of being completed, then you move on to the next stage. Remember to include in this initial design phase how to make money from your app and include any social networking/marketing ideas. The most successful apps plan for marketing and sales from the beginning. One final consideration in the design phase is legal. Be sure to investigate intellectual property and regional/national laws where you hope to sell your app.

2) **Gather Requirements** – This phase is about detailing exactly what functionality the app will contain and the design of what the various views will look like (also called storyboarding). This is an essential phase. If you, the developer, are not sure what the app will look like when it is completed, how can you communicate what you are developing to others? Be sure to keep these initial designs and develop a webpage around the development process. It will help with your marketing efforts!

3) **Code and Graphic development** – This is where you get started programming and developing the graphics for your project. The best teams are composed of programmers and artists.

4) **User Testing** – Too many people skip the testing phase or do not conduct a thorough enough test of their app. Deploy your app to a few test devices and have people in your target audience use the app. Listen to their feedback (remembering that they might be overly kind) and implement their suggestions.

5) **System Tests** – Before releasing your app into the wild, run a systems test. Is your app connecting to a remote server or the cloud? Facebook? Twitter? Make sure that all of these features and services are capable of supporting the additional demand your app might place on it.

6) **Documentation/Marketing** – Before you can release your work to an app store, you must have a supporting webpage in place with contact emails and screen shots. This is also part of your marketing effort so make sure everything is perfect!

7) **Production** – Create your app for release and place it in the stores in which you would like to sell it.

8) **Maintenance** – If you haven’t noticed, the operating systems for devices are constantly changing with more, newer devices becoming available on almost a daily basis. You should expect to refresh and update your app at least every few months at the very least. On the bright side, in most app stores, releasing a new version could get you a higher ranking in the search engine!
The above list is based upon the traditional software development lifecycle. I have made one adjustment. In the traditional lifecycle, Documentation/Marketing is placed after Production. In app development, that would be a mistake. Documentation and marketing are too important and their not being completed will delay you being able to submit your project to the app stores.

**Software That You Will Need**
It's no surprise that you will need the Corona SDK to get started. For learning, I recommend signing up and downloading the Starter version. Just head over to the on the Corona Labs website [http://www.CoronaLabs.com/store](http://www.CoronaLabs.com/store). Click on the buy button and register (whether you are purchasing the Pro subscription or using the free Starter version). If you are a student or faculty, you can get a discount on a Pro subscription by going to [http://www.coronalabs.com/store/corona-for-education/](http://www.coronalabs.com/store/corona-for-education/).
Setting Up Your Software
The process varies depending on whether you are installing Corona SDK on a Macintosh or a Windows computer system. A full tutorial for installing the software on both machines is provided in Appendix A. In Appendix A we will also discuss what software is required to publish to devices for different operating systems.

Setting up Your Hardware
Corona isn’t too demanding on your development computer. As long as you are running at least OSX 10.7 (Lion) or later on the Mac side, or Windows XP with a 1 GHZ processor on the PC side, you will be fine.

If you are planning to develop and deploy to iPhone, iPod Touch, and/or iPad, then you must have a Mac of some type to publish your apps. This is an Apple requirement. To keep in everyone’s good graces, Corona will only publish for an iOS device if you are using a Mac computer to deploy the app. You will also be able to develop and deploy your Android based app from a Mac.

If you only have a windows system, you will be able to develop and deploy for Android based devices. You will also be able to develop for iOS devices. You just cannot deploy your finished app to an iOS device (or the iTunes store). I use both a Mac laptop and a PC, regularly switching back and forth during the app development process.

Development Hardware Matrix:

<table>
<thead>
<tr>
<th>Development Hardware</th>
<th>Android OS</th>
<th>Apple iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Develop</td>
<td>Deploy</td>
</tr>
<tr>
<td>Macintosh</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Windows PC</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Test Devices
If you are going to develop and sell apps for mobile devices, you should have a mobile device to test your creation. I have been on projects where I was required to develop for hardware that I didn’t have. It was like herding cats. Using just the app simulator will get you 75% of the way home, but it won’t allow you to spot all potential problems. On one of the aforementioned projects, the app worked fine on the simulator, but crashed on the mobile device and was rejected by Apple. The experience was more than just a little frustrating and taught me a valuable lesson: If you are developing for a platform, have test devices!
Android
Corona only builds for Android OS 2.2 and newer. Any devices that you plan to develop for must use the ARM V7 processor. There are plenty of devices that meet this requirement, so you shouldn’t have any problem finding one to perform your tests.

iOS
For deploying to an iOS device, you will need a developers license and either an iPhone, iPod Touch, or iPad. Obviously, having an older phone or iPad is a good idea for testing FPS (Frames Per Second) for graphically intensive apps. It is recommended that you use the newest iOS on your devices. To be able to deploy to an iOS device, you will need a Mac computer system and a Standard, Enterprise, or University developers account from Apple.

Book Examples and Graphics
If you don’t want to create your own graphics or you would like to double check what you have programmed against what I have coded, I have created a repository of code samples, graphics and other tools that you might want to use with the projects that are listed in this book. They are all available at http://www.burtonsmediagroup.com/books/learning-mobile-application-development/.

Editors
The editor that you decide to use is a personal decision. Corona isn’t impacted by the editor selection, so you need to use an editor that you are comfortable with. I recommend one that allows the integration of Lua to make your editing easier.

Some of the most popular editors in use with Corona include (but are not limited to) BBEdit, Eclipse, Notepad++, TextMate, TextWrangler, and Xcode. Of course you can ignore all of these editors and use notepad or textedit if you so desire.

**BBEdit** (Mac) by Bare Bones software, $99.99.
BBEdit does a nice job for a multitude of editing needs. BBEdit has built in configurations (including Lua), which easily allows you to set the editor to the language you are developing in. http://www.barebones.com

**Outlaw** (Mac/Win) by J.A. Whye, free or $29.99.
Outlaw has a built in editor and is the primary tool I use for editing and project management. Coupled with its ability to greatly simplify tracking your Corona project, the
cost of Outlaw is well worth it. See Chapter 20 for a coupon code to save on Outlaw. http://outlawgametools.com/

**Eclipse** (Mac/Win) Open source, $0.
Eclipse is the editor I use when working on my PC. Eclipse has a large community of support. Though Eclipse was originally designed as a Java IDE (Integrated Development Environment), it is now the bases for many editors on the market. A Lua/Corona plugin is available. http://eclipse.org

**Lua Glider IDE** (Mac/Win) M.Y. Developers, $39.99
Glider is a popular integrated development environment (IDE) with some really nice features for troubleshooting and app development. Features include pause and continue, the ability to set break points and stepping through code. A really great tool! http://www.mydevelopersgames.com/Glider/

**Notepad++** (Win) Open source, $0

**TextMate** (Mac) by Micromates, €39 (about $57).
Textmate is very popular in the Corona community with a Corona plugin available on the Corona Labs website. http://macromates.com

**TextWrangler** (Mac) by Bare Bones Software, $0.
TextWrangler has the advantage of being a free editor for your Mac. Though it doesn’t have all the bells and whistles as BBEdit, it will get the job done for those on a budget and offers integrated Lua support. http://www.barebones.com

**Xcode** (Mac) by Apple, $0*.
Xcode is an integral part of the iOS SDK. If you are used to developing using Objective-C, Xcode is a natural choice. While Xcode is included with iOS SDK, it is only free if you are already a standard developer with Apple. If you register for a free account, the iOS SDK (which includes Xcode) is $4.99.

**Our First Project: Hello World**
The first time you launch the Corona Terminal or Simulator it will ask you to login with your registration information that you used on the Corona Labs website. Complete this one time authentication and you will be ready to go.
Corona Developer Registration

You should always launch the Corona Debugger on a Macintosh instead of the Simulator for performing application builds and testing. On a Windows system, launching the Corona Simulator also launches the Corona Simulator Output window (commonly referred to as the terminal window). The Corona Terminal gives you important feedback when you are building your apps and allows for easier troubleshooting. The Corona Terminal will automatically launch the Corona Simulator.

Project 1.0: Hello World

I personally always hated programming books and classes that spent the first chapter or week just getting all the details taken care of. I purchased the book or took the class because I wanted to program, not to go over some syllabus or a review of all the different ages of computer development. So let’s skip all of that and make an app that will help you learn your way around Corona: a “Hello World” project.

Stop with the rolling of eyes! Before I lose you, let me guarantee that you will get a very valuable resource out of this Hello World project, something that you will use the rest of the time you develop in Corona.

Was that enough to get your attention? Then let’s get started!

Project Setup

If you follow this process each time you start a new project, it will make your life a lot easier:
First, create a project folder called “Hello World”. This can be on your desktop or wherever you like to organize your work. I keep all of my project folders together in a folder called “Corona Projects”.

Create the Hello World folder for your project

Open your editor of choice (I’m using BBEdit in these initial screen shots). Create a blank file and save it as “main.lua” to your Hello World folder that you just created. The main.lua file is the first file that the Corona simulator will look for when it is run. If there is no main.lua file present, nothing will happen.
Save the main.lua file to your Hello World folder

There should now be a main.lua file in your Hello World folder.

Back in your editor type:

print("Hello World")

and save your file as main.lua.
Hello World project in the editor

Next, you will need to launch Corona. If you are on a Microsoft Windows system, launch the Corona Simulator. On a Macintosh, launch Corona Terminal.

An Important Macintosh Note: Throughout the book I recommend using the Corona Terminal to launch Corona instead of the simulator if you are using an Apple computer. The Corona Terminal can be found through your Finder under Applications > CoronaSDK > Corona Terminal. On Windows, the terminal windows will open automatically when you launch the simulator.

Corona at startup on a Macintosh – don’t use the new project button yet!

On launch, you will see the Terminal window and the Welcome to Corona dialog box. Select “Open a Project” from the Welcome to Corona dialog and navigate to the Hello World folder that was created earlier. Your initial window might be different based upon the version of Corona that you are using.
Open Hello World & select device to simulate

When you are opening a project, you will be able to select which device you would like to simulate in the Corona Simulator. For now select either iPhone or Droid and click on the Open button. Selecting other devices could give you different results than what are in the screen shots that have been included.

As soon as you open the project, the simulator will run the project.

Did you notice? That’s right, nothing happened...in the simulator. Look in the Terminal window.

On the last line of the text in the Terminal you should see your Hello World displayed.
Hello World in the Corona Terminal window

Congratulations! You just made your first Corona app! Now before you become disappointed, you just learned a very important tool for troubleshooting your applications. When something doesn’t seem to be working correctly or displaying the way you want, you can send yourself messages through the Corona Terminal window. Believe me when I tell you that this one command will save you hours of troubleshooting headaches!

I am sure you also noticed that Corona generates a great deal of additional information before giving you the results of your print command. The first few lines provide information about the version of Corona and the location of the simulation files.

Note: If you didn’t see anything, there are two areas that people commonly make a mistake: 1) they didn’t save their main.lua file (I still make this mistake) or 2) when saving the main.lua file, it wasn’t saved as a text file type.

Debugging
One of the key methods for debugging is liberal use of the print() command. With the print() command, you can pass a variety of text or variable values to the NSLog (a file that tracks events for Apple devices) or terminal window (depending on your version of Corona SDK). To be a successful app developer, you will quickly come to depend on using print throughout your program.

Project 1.1: Hello World (v2.0)

Well, that was frustrating! I wanted to make something appear on the screen! Let us make a second attempt at getting something on the simulator screen. Back in your editor (you can use the same main.lua file) type:
local textObj = display.newText("Hello World", 50, 50, native.systemFont, 24)
textObj:setTextColor(255, 255, 255)

Lua, the language behind Corona, is case sensitive. So newText is a different word than newtext or NewText. Try newtext and look at the error that appears in the Terminal window.

**Note:** Be sure to use " " and not “” in your apps. “” will cause an error!

So what do you make of everything you just typed? Take a moment and savor the possibilities. What will the screen look like when it launches? Why?

Of course we expect “Hello World” to be emblazoned somewhere on the screen, but where? Perhaps those two 50’s have something to do with where it will be placed on the screen? What will it look like? Looks like some sort of default font will be used for the text, and we are setting the color of the text, but what color shall it be?

I’m sure you have a lot of questions on what you typed in. Time to save and launch your app, and then we will look at what happens.

Save the file, and then launch your simulator. You should now see Hello World displayed in the simulator:
Hello World on the Droid simulator

Looks like it’s an equal distance from the top and the side: 50. 50? 50 pixels either way, right. It’s definitely a workhorse system font, probably Helvetica, and those three 255’s somehow add up to white. Okay, so I’m presuming you just came out of a cave and know nothing about RGB values. Well, just in case you did, let’s change a couple of those 255’s to 0's and see what happens:

<table>
<thead>
<tr>
<th>Hello World</th>
<th>Hello World</th>
<th>Hello World</th>
</tr>
</thead>
<tbody>
<tr>
<td>(255, 0, 0)</td>
<td>(0, 255, 0)</td>
<td>(0, 0, 255)</td>
</tr>
</tbody>
</table>

And of course when you add Red, Green, and Blue together in the additive color system, you get White.

Now that final number - 24. Let’s change it to a larger number such as 48. What do you think will happen?

What if you want to change the font to say, Times. What would you do?

Well, first off, you’d obviously have to get rid of the offending native.systemFont, but could you just type in Times? Let’s try it, change native.systemFont to Times.
Hmmmmm. Nothing. No error in the terminal window, but no change on the display either.

I will give you a clue: you can quote me.

We have to convert *Times* to a string (i.e. place it inside of quotes) for it to be understood. Fonts are stored by their literal names and can only be recognized if you tell Corona that this is a name, not a variable. Anything placed in quotes is referred to as a string (which we will discuss in greater detail later). The only way the computer in our smartphone knows that we are looking for the name of a font is if we place it in quotes. Try changing the font to "*Times*" and see what happens.

Now for the technical explanation of what we just accomplished:

First we created a local variable called `textObj`. A **variable** is just a place holder for other things in our program. Often times we aren’t sure what the value or information will be when we are writing our apps, so it is necessary to use a variable to hold our information. Remember back in math (or if you are from the U.K., maths) when you would use x or y to solve an equation? Well, a variable is the same thing, except we are going to name our variables much better than x or y. Good variable naming will make our lives much easier when we get to more complex apps that might have 10 to 20 (or more) variables. It might mean more typing, but you will really appreciate it when you go to revise or update the program at a later date. We do not have to use the variable name `textObj`, we could use `fred` for the variable name but after a couple of days we might forget what `fred` represents.

We set `textObj` equal to the **object** (which we will discuss in just a minute) that we create by calling `display.newText`. `display.newText` is a command that Corona understands. When Corona sees `display.newText`, it knows that we are going to type something to the screen by telling it what we want to type, where we want it placed (the numbers 50, 50, which are the top and left corner of the text), what font we want to use, and how big the text will be.

The `display.newText` parameters are:

```
display.newText(text, left corner, top corner, font, text size)
```

or

```
display.newText("Hello World", 50, 50, native.systemFont, 24)
```

In the second command line,
```
textobj:setTextColor(255, 255, 255)
```
we set the color of the textObj that was just changed using the R, G, B color system (each color (red, green, blue) having a value between 0 – 255) to white (which is 255, 255, 255; to get black, we would set it at 0, 0, 0; red is 255, 0, 0; green is 0, 255, 0; blue is 0, 0, 255):

textObj:setTextColor(R, G, B)

By default, the text object is white, so we didn’t really accomplish anything by setting the textObj to white. But I want to get you in the practice of setting the text color when you create a text object. Later we will look at how to fade the text object out (or in).

Now you have made your first REAL mobile app!

**Warning:** If you copy code from a website (or even from this book), sometimes the quotation marks will change from straight quotation marks to smart quotes. This WILL cause an error in Corona. Make sure your quotes are always " " and not “ ”.

---

**Introducing Objects**

You may have noticed the use of the term **object** sprinkled throughout the book thus far. When I use the term ‘object’ it is to represent anything that is used in our project. Text, buttons, or sounds; they are all objects. Just as in the real, physical world, I can move or interact with an object (a lamp, table, or car). An object in your software is anything that you or the people using your app can interact with, including viewing, tapping, dragging, listening to, or just a pretty picture that is on the display.

Real world objects all have **properties** that help to describe the object’s location, color, or anything that can be changed about the object. If I have a car, I might describe the car’s location by its longitude and latitude.

In programming (including Corona), we are able to interact with each object’s properties to make changes; such as when the textObj was created, we set the left, top, font, and size properties as well as the string that would be displayed.

Most objects can have their property changed just by setting it to a new value:

textObj.x = 100

would move the Hello World that was displayed on the screen to pixel location 100 (or to the right 50 pixels of the original location). Properties always have a period between the object name and the name of the property.

A few valid properties for display.newText include:
object.size – set the font size of the text
object.text – set or change the text
object.x – set or change the x location of the object (based upon the center of the text)
object.y – set or change the y location of the object (based upon the center of the text)

Objects can also have **methods**. A method is something that changes the current state of an object. Think of a lamp. A lamp can be turned on or off. If we were going to have a method for a lamp, we might call it setLight so that we could have it on or off.

To use a method, we put a colon between the object’s name and the method we are going to use. In the case of our text, the primary method that we are concerned about right now is `setTextColor`. To change the color of the text we would use the command

```
object:setTextColor(R, G, B)
```

Okay, that is enough for now! If it seems confusing, do not worry about it, it is confusing when you are first getting started! Give yourself a little bit of time to get used to the idea. Remember: 75% of any new skill is learning the vocabulary. If you get used to the idea that an object can be anything in our app and a variable is just the name that we are going to use to refer to that object, you are most of the way there already!

**Summary**
This has been a busy chapter! Corona should now be installed on your system, you have been introduced to editors, hardware considerations, and publishing information. We even managed to develop two apps (okay, maybe not saleable apps, but they are apps)! The first app introduced the critically important print command; the second app actually displayed text to the Corona simulator, our original goal. Finally, the concept of a variable and an object in programming was briefly introduced. If the idea of an object and variable doesn’t seem natural yet, don’t worry, it will make more sense as we learn more material.

**Programming Vocabulary:**
Method
Object
Property
Questions:
1. What is a method? Specify its importance and give an example.
2. What is an object? Specify its importance and give an example.
3. What is a property? Specify its importance and give an example.
4. How are objects and properties related to one another?
5. What is a variable? Specify its importance and give an example.
6. What is the programming language that Corona uses?
7. True or false: I can publish for an iOS device using a PC.
8. True or false: Corona allows me to publish to multiple mobile operating systems.
9. List and summarize the eight steps of app development.

Assignments
1. Try various typos to see the resulting error messages in the terminal window.
   a. Make a typo in newText. What is the result?
   b. Make a typo in native.systemFont. What is the result?
   c. Try setTextColor. What is the result?
2. Change the text object to red in the Hello World (v2) project.
3. Reposition the text to the bottom of the simulator without letters going off the bottom by changing the x and y values of display.newText in the Hello World (v2) project.
4. Place 5 different messages in different places on the screen, each in a different font, size, and color. Note that fonts will depend upon your system. Remember that the font name must be enclosed in quotation marks. Don't worry, if the font is not available, the system will switch to a default font.
Chapter 2  Introduction to Functions

Learning Objectives
In chapter 2, we will learn

- about Variables
- the difference between a local and global variable
- how to place a comment in your program
- how to determine the screen size of a device
- how to create a routine that can be used later
- what a listener is and how to use one
- about the API and how to use it

Name that Object
In our last chapter we learned about objects. Now it is time to name the object. Whenever we assign an object to a variable, we are essentially giving that object a name. Everything that the object is will be stored and referred to through that assigned name. Just as you are called by a name, so too are the objects within our app. When I assign text that will be on the screen to the variable name textObj, we have given it a 'name'. Using this name (in this case textObj), we can give it further instructions later in the app.

Local vs. Global Variables
Most of us have the experience at one point or another where we are given a 'temporary' name. Perhaps during school there were two students with the same first name in a class, so they would have their last initial also used: thus two Heathers became HeatherA and HeatherB. Or maybe you were given a nickname in school or in athletics. Usually these different names were short lived or only used in limited situations.

There are a few rules to the naming of variables:

- A variable can be any combination of letters, numbers, or underscores
- A variable must not begin with a number
- A variable cannot contain a space or any symbol except underscore
- Variables are case-sensitive. myVariable is not the same as MyVariable
In programming we have two types of variables: local and global. As we progress through the rest of this book (and any other programming language that you ever learn) think of the differences this way: a local variable is a short-lived name given to an object, much like that short-lived nickname in school. A global variable is a long-lasting name and can be used throughout a program, anywhere in a program.

How do you tell them apart? Easy. A local variable will always have the word `local` in front of it the first time it is used in an app. A global variable will never have the word local.

Declaring a local variable:

```
local textObject
local myNewPicture
local backgroundImage
```

Declaring a global variable:

```
textObject
myNewPicture
backgroundImage
```

The preference in programming is to always use local variable whenever possible. Local variables use less memory and will help you avoid naming problems in more complex programs.

One last thing before we move on: while you can place text and images on the screen without using a variable, you won’t be able to move, hide, change, or remove them later. It is best to always use a variable name for your objects.

```
display.newText("Can’t Touch This", 10, 10, nil, 16)
local myObject = display.newText("Can Touch This", 10, 50, nil, 16)
myObject.y = 100
```
How to Code Comments
If you have never written a program before, placing comments might seem like a silly waste of time, especially when you are working on a simple program. Let me assure you, comments are as important as any line of code that you write! While good commenting is needed in all software development, it is especially critical in mobile app development. Mobile apps have a very short cycle before they have to be updated. Usually you will be updating a successful app every 6 to 12 months; just enough time to completely forget why you wrote a line in the program to do a specific operation or how you used a special command to fix a bug.

Comments are a gift you give the future you (or the programmer who comes after you to update the software). Taking a few minutes to leave good comments in your program will potentially save you hours of work later. Learn the habit of commenting now. Believe me, you will thank yourself later!

There are two ways to comment in Corona: line comment and block comment.

Line comments convert the remainder of the current line into a comment. By placing two dashes: -- you tell Corona that everything after the dashes is a comment and can be ignored.

A block comment also begins with two dashes and is followed by two brackets: --[[ You end the comment with: --]] usually placed on its own line in the editor. Everything between the brackets will be ignored by the compiler.

Block commenting is a great way to turn on or off a section of program when you are doing testing.

```
-- This is a comment.
local myObject = display.newText("Your Text Here", 10, 40, nil, 20) -- everything after the dashes is a comment

--[[
This is a block comment. Everything between the double brackets is a comment.

--]]
```

Device Boundaries
Developing an app for mobile devices is a little different than traditional programming. One of our considerations, especially when we are developing apps for a number of devices is that each one has a different screen size and pixel count. Even if you are just going to publish to Apple, you have four different resolutions that need to be supported. If you add
all the different Android based devices, you could drive yourself crazy making all of the
adjustments so that each view that is created looks good.

One way around this problem is to use variables to store the device resolution. Corona has
built-in commands that will return the display width and height. The command
\textit{display.contentWidth} and \textit{display.contentHeight} will return the width and height. These are
just two of the commands included in the Corona SDK Application Programming Interface
(usually abbreviated as API). All programming languages and development kits include an
API so that programmers know what commands are available and how to properly use
those commands.

\textbf{Project 2.1 Display Size}

For our first project in this chapter, let’s create a simple program that will give the screen
size of our device. Specifically, I would like to know what the x, y location is for each corner
of my screen. For this program, I would like it to display at each corner what that screen
location is, shown in the examples below:
The iPhone 4, Kindle Fire, and iPad 2 display resolutions (not to scale)

To get started, we will need to create a new folder for our app. The folder can be any place on your computer, even a flash drive. But you should always create a new folder for each app. Once you have your folder, create a new main.lua file with your editor.

To get started we first need to hide the status bar. This is done using the display.setStatusBar API command. To make the status bar hidden, we need to pass it a parameter. A parameter is any information that is given to an API command. In this case, the parameter is display.HiddenStatusBar. This makes the command to hide the status bar from view during the execution of your app: display.setStatusBar( display.HiddenStatusBar )

main.lua

display.setStatusBar( display.HiddenStatusBar )

Now that the status bar is hidden, it is time to find out the height and width of the display. We can store the display’s width and height in variables for easy reuse with the display.contentWidth and display.contentHeight commands:

local myWidth = display.contentWidth
local myHeight= display.contentHeight

We now know (well, our app knows) the maximum width and height of the display.
Displaying it is just a matter of using the display.newText, right? Remember, the top, left corner will be our starting point, so it has an x of 0 and a y of 0.

One last thing: instead of using system.nativeFont as the fourth parameter of display.newText, I have used a short cut: nil. Nil is nothing, zero, nada. It is a shortcut way of telling Corona that I don’t care what system font is used, so Corona will automatically use the system’s native font.

```
local topLeft = display.newText("0, 0", 0, 0, nil, 24)
local topRight = display.newText(myWidth .. ", 0", myWidth, 0, nil, 24)
local bottomLeft = display.newText("0, " .. myHeight, 0, myHeight, nil, 24)
local bottomRight = display.newText(myWidth .. ", .. myHeight, myWidth, myHeight, nil, 24)
```

Did you notice my use of two periods in the display.newText (myWidth.. ", 0")? By placing two periods together, we are telling Corona that we want to concatenate or take two strings and make them one string. In this case, we are taking myWidth and concatenating it with ", 0" which will place the width beside the comma in the string (for example: if the width is 360 pixels, the resulting string would be “360, 0”). We will talk about concatenation more in chapter 4.

Save your app and run it with the Corona Simulator.

Hmmm, while the top, left corner text is showing, none of the other corners have any text.
Why? As we saw in Chapter 1, display.newText positions the top left corner of the text at the values we give. So when we assign topRight to myWidth, 0, the myWidth positions the left off of the screen. Same situation for myHeight at the bottom of the screen, it causes the text to be displayed below the bottom. To fix this problem, we need to move the text back on the display by subtracting from the myWidth & myHeight values. Remember that the measurement is in pixels, not characters. Let’s replace the last four lines with the following:

```lua
local topLeft = display.newText("0, 0", 0, 0, nil, 24)
local topRight = display.newText(myWidth.. ",", 0, myWidth-100, 0, nil, 24)
local bottomLeft = display.newText("0, ".. myHeight, 0, myHeight-50, nil, 24)
local bottomRight = display.newText(myWidth ..",", ".. myHeight, myWidth-120, myHeight-50, nil, 24)
```

Run the app. You can change the device shown by the Simulator by clicking on the View Menu, then View As. Doesn’t that look better? Try changing the values in left and top parameters of the display.newText to see how the text changes location on the screen.
We hope you have enjoyed this sample chapter of *Learning Mobile Application & Game Development with Corona*. This and other books on mobile and game development are available on our website:  
http://www.BurtonsMediaGroup.com/books
Chapter 3 Animation and Orientation

Learning Objectives
In chapter 3 we are going to begin working with animation and handling when the user changes the orientation of their mobile device. To do this we will learn:

- Methods of animation
- How to program a decision
- How to program a loop
- Fading objects in and out
- Handling orientation change
Chapter 4 Working with Data

**Learning Objectives**

In chapter 4 we will begin working with various types of data:

- How to enter data using textfield and textbox
- Using the Math API
- Using the String API
- Using your own fonts
- How to publish to your test device
Learning Objectives

One of the things that many people find appealing about Corona is how easy it is to create and load graphics into the mobile environment. In this chapter we are going to:

- Create vector based graphics
- Load bitmap graphics
- An introduction to sprite sheets
- Review associated graphic properties
- Identify App Store Icon requirements
- Explore additional animation methods
Learning Objectives
In this chapter we will be learning about creating a good user experience. This includes learning:

- How to hide and show the status bar
- How to load custom fonts
- How to group objects
- How to create and use external modules
- How to use storyboard API to create additional views for the user
Learning Objectives

Sound, music, photos, and movies make up a very important part of the mobile device culture. In this chapter we are going to look at using, playing, and recording all four of these types of media. Specifically, we will learn to:

- the difference between loading and streaming audio
- load and play sound effects
- load and play long sound files such as music files
- use the built-in camera on a device
- play movies in your app
Learning Objectives
The physics in Corona is just plain fun (or phun). In this chapter we will examine the basics of using physics in Corona. This includes:

- Setting gravity
- Types of bodies
- Detecting collisions
- Working with joints
Chapter 9 Mobile Game Design

Learning Objectives

One of the most exciting areas of mobile devices is being able to develop and play games. The concepts behind mobile game development are appropriate for an entire book or series of books. Consider this chapter a very brief introduction to some of the basic programming considerations. In this chapter we will learn:

- How to use timers
- Handling drag events
- Using frame base animation
- How to make a game
Chapter 10: Tables and Arrays

Learning Objectives
In this chapter we are going to begin working with Tables. Tables have become a critical part of mobile application development, with Apple having spent enormous amount of effort in the creation and refinement of tables for the iPhone and iPad. Tables are one of the simplest ways to store large quantities of data.

In our examination of tables we will:
- Clarify the term table and array
- Examine the tools available for tables
- Create a simple table
Chapter 11: Going Native - Working with Widgets

Learning Objectives

Time to jump into tools that will make your apps user interface appear native. These tools or widgets allow us to add picker wheels, buttons, scroll-bars and sliders with ease. We will also examine tools for creating mock-ups. Specifically we will learn:

- Examine various mock-up or pre-design tools
- How to use of widgets in app development
- How to use widget themes
- Review the use of the different types of widgets
- How to properly remove a widget
Learning Objectives
When developing apps to sell on the app stores, it is good programming practice to develop your app to properly handle things going wrong (of course, it won’t be YOUR app that messes up ;-)). In this chapter we will examine some of the system event and system resources that are available. We will examine:

- How to handle and catch system events
- How to use the accelerometer
- How to use the GPS
- How to use Alerts for notification
Learning Objectives
I know, I know, you have been wanting to ask the following question for some time now, “How do I save information in my app?” With a few great API resources we can take care of this problem easily. In this chapter we will learn:

- Where to store files on the smartphones and tablets
- The difference between explicit and implicit file input and output
- How to read data from a file
- How to write data to a file
- Using JSON to store and retrieve app data
Learning Objectives
In this chapter we will examine several ways to read and save data to a mobile device. The ability to access external information that is located on your device is critical to many types of data-intensive applications. For the sake of simplicity, we shall keep our focus limited to files that are already located on the device. To that end, we will examine:

- File location considerations
- Reading from a file
- Writing to a file
- XML/JSON
- Reading from a SQLite database
- Writing to a SQLite database
Learning Objectives
How can we discuss mobile app development without addressing the ability to communicate with a network? In this chapter we will examine the basic methods used to create network connectivity and how to connect with some of the most popular services. We shall:

- Create a network connection
- Check network status
- Connect to a webserver
- Download/upload to a webserver
- Connect to Facebook
- Introduce inMobi functions for in-app ads
- Introduce pubnub for multi-player apps
Chapter 16 Head in the Cloud

Corona Cloud

Analytics
Chapter 18 Web-based app development
Chapter 19: Facebook Mobile API
Chapter 20 Next Steps and 3rd Party Resources

Where do we go from here?
I think you will believe me when I tell you that we have only scratched the surface with what you can do with mobile application & game development! There are many great resources that you can use to continue your education. I recommend that you start with the Corona Labs website. Watch the forums, blogs, tutorials and updates. I am sure that they will spark many creative ideas and help you to complete your personal and professional projects.

A new feature that will soon be available (or is already available, depends on when you purchased the textbook) is the graphics 2.0. Graphics 2.0 will allow Corona to implement shaders, 2.5 graphics, and dramatically improved graphic improvements. I have already seen a few demos created with the new graphic tools set, and to say its performance was impressive would be an understatement. Be sure to

3rd Party Resources
If you haven’t notice, one of the strongest features of the Corona SDK is the great community that uses it to develop their apps. Over the past several years a few great resources have become available that you should be aware. I’ve not attempted to list every resource, but here are a few that might help you with your projects.
Note: all prices are in USD and were accurate at the time of writing. Software is listed in alphabetical order.
Appendix A: Installation of Corona SDK

Appendix B: Publishing to Apple - Installing xCode & Apple Provisioning
Appendix C: Publishing to Android - Installation and Configuring Keystores
Appendix D: The Lua Language
Appendix E: Advanced Lua Language
Appendix F: The Path to Certification

What is a Corona Certified Developer?
A Corona Certified Developer is someone who has distinguished themselves in the development of cross-platform mobile applications. They have successfully published applications developed with the Corona SDK. If you have worked through the entirety of this book, you have been introduced to every concept that is covered on the Corona Certified Developer exam.

Why become certified?
The Corona Certified Developer (CCD) is able to advertise their app development services with a certified developer; showing that they are experienced and knowledgeable in using the Corona SDK to develop applications.
We hope you have enjoyed this sample of *Learning Mobile Application & Game Development with Corona*. This and other books on mobile and game development are available on our website: [http://www.BurtonsMediaGroup.com/books](http://www.BurtonsMediaGroup.com/books)