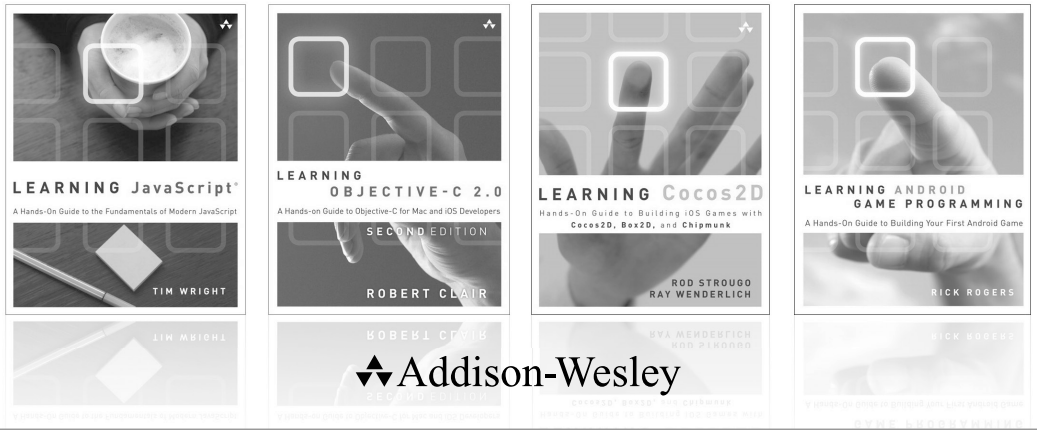
LEARNING MIT App Inventor

A Hands-On Guide to Building Your Own Android Apps

DEREK WALTER
MARK SHERMAN

Learning MIT App Inventor

Addison-Wesley Learning Series



Visit informit.com/learningseries for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

Learning MIT App Inventor

A Hands-On Guide to Building Your Own Android Apps

Derek Walter
Mark Sherman

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2014950962

ISBN-13: 978-0-133-79863-0
ISBN-10: 0-133-79863-1

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.

First printing: December 2014

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Senior Development Editor
Chris Zahn

Managing Editor
Kristy Hart

Project Editor
Elaine Wiley

Copy Editor
Krista Hansing
Editorial Services, Inc.

Indexer
Lisa Stumpf

Proofreader
Debbie Williams

Technical Reviewers
Tom Stokke
Arta Szathma
Janet Brown-Sederberg

Editorial Assistant
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
Nonie Ratcliff

❖

from Derek

This book is dedicated to my incredible wife, Candy.

from Mark

This book is dedicated to Stacy (depending on what she says).

❖

This page intentionally left blank

Table of Contents

Acknowledgments

from Derek

I want to thank my amazing wife, Candy, who supported me during the crucible of writing a book. Your strength and encouragement kept me going through the late nights, exhausting weekends, and challenges that came with this project.

I would like to thank the MIT App Inventor team members for their support and for continuing such an important project that is democratizing computer programming. Also, thank you to Laura Lewin and the Pearson team for their guidance with this first-time author.

from Mark

I want to thank the AI team at MIT, the AI Master Trainers program organizers, Lyn Turbak, and especially Fred Martin. All of you helped me get to this point, and all of this accrued knowledge is thanks to you.

I want to thank my close friends, all of whom endured my writing and working through many events, and often were kind enough to pull the laptop off my sleeping face. I appreciate it. I especially want to thank Stacy for taking care of me every step of the way. Stacy, will you marry me?

About the Authors

Derek Walter is a freelance writer specializing in the mobile ecosystem. He contributes regularly to PCWorld, Macworld, Greenbot, and other sites devoted to consumer technology. He also blogs about mobile apps and other topics in technology at theapplanet.com. His undergraduate degree is in mass communication/journalism, and he holds a master's degree in educational technology from The George Washington University. Derek has also worked in education for the last 15 years as a classroom teacher and adjunct university instructor.

Mark Sherman is a researcher in computer science education and has taught computing, programming, and robotics to undergraduates in the U.S., India, and China. He is an MIT App Inventor Master Trainer, and he has taught students mobile app design with App Inventor and trained teachers and faculty on best practices and pedagogy of the same. He holds a bachelor's degree in computer engineering and a master's degree in computer science, both from UMass Lowell.

Preface

The smartphone is the ultimate personal computer. Mobile devices are always with us and have become an essential part of personal productivity and lifestyle needs. We use them for messaging, social media, Google searches, games, picture taking, and, of course, phone calls.

The Android operating system powers most of the world's smartphones, bringing an extensive app catalog to these devices. According to Google, more than 1 billion active devices are currently running Android.

Perhaps you have reached the point at which using mobile apps on your smartphone isn't enough—it is time to create one. You might just want to tinker and program a simple app, or maybe you have thought of a new concept that doesn't exist yet. Whatever the case, MIT App Inventor is an excellent place to start.

App Inventor is an easy-to-use tool for building both simple and complex Android applications. The apps can easily be ported to your phone, shared with others, or even sent to the Google Play Store for distribution to all Android devices worldwide.

For those looking to learn a programming language, MIT App Inventor can serve as an excellent bridge to acquiring more complex coding skills. Instead of presenting new users with frustrating messages and unfamiliar commands, App Inventor has a visually friendly interface that uses the methods of dragging, dropping, and connecting puzzle pieces to program applications (see Figure P.1).

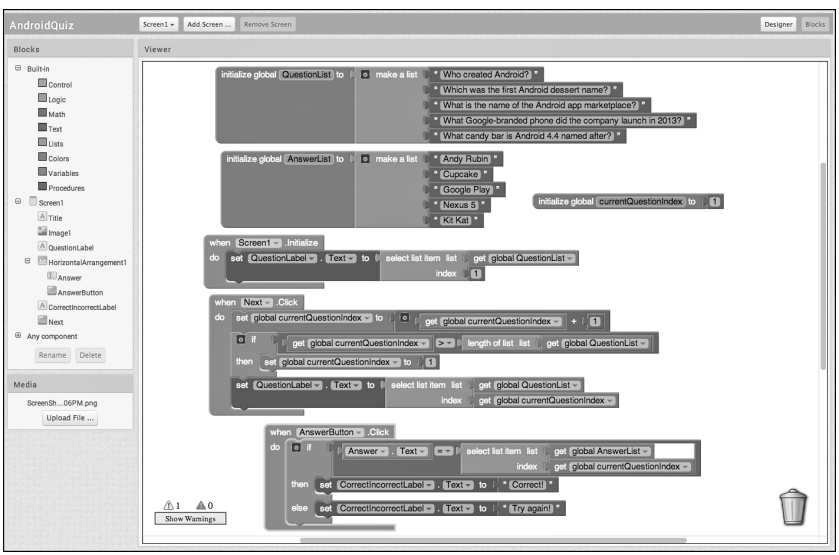


Figure P.1 The MIT App Inventor Blocks Editor. The visual programming is designed to help beginners learn the ropes of building mobile applications.

Even though App Inventor does not require using code, it builds on the same kinds of principles that successful programmers need to write good applications. Whether you go no further with programming or you use App Inventor to launch a new career, using App Inventor can be a highly engaging and challenging experience. Additionally, the open and flexible nature of Android makes it the perfect place to start.

What Is MIT App Inventor?

MIT App Inventor is a web-based tool for building Android apps (see Figure P.2). This is often referred to as visual programming, which means the user is able to perform programming tasks without entering any computer code.

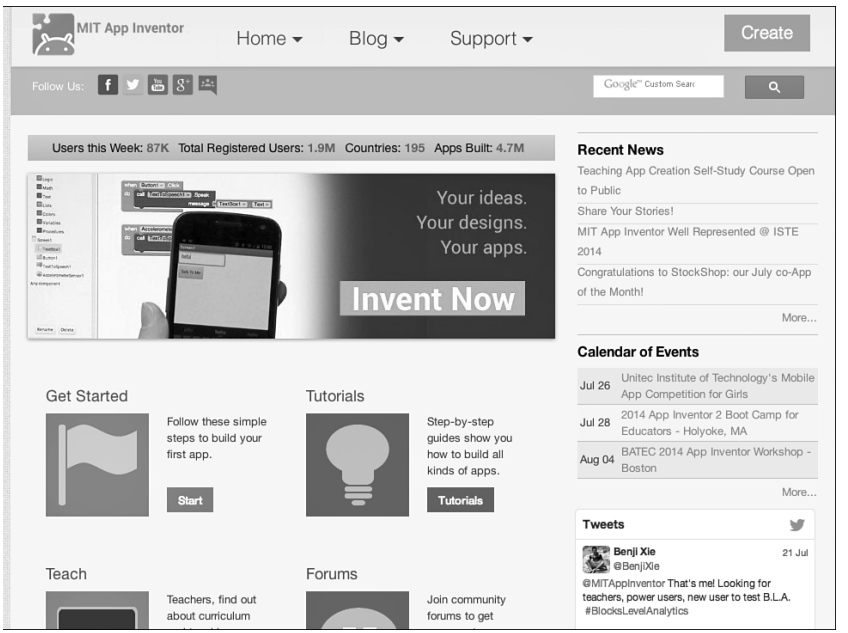


Figure P.2 The MIT App Inventor home page. This is where the app-creation journey begins.

App Inventor is actively managed and developed by MIT's Mobile Learning Lab (the project was originally built by Google). App Inventor is growing in popularity among educators as a way to introduce those with no programming experience to the principles of computer science and app development. It also serves as a great first step for those dabbling with programming or looking to increase their knowledge of how smartphone apps work.

The work takes place in two key sections of App Inventor: the Designer and the Blocks Editor. In the Designer, you decide what actions the app will perform and how it will look (see Figure P.3).

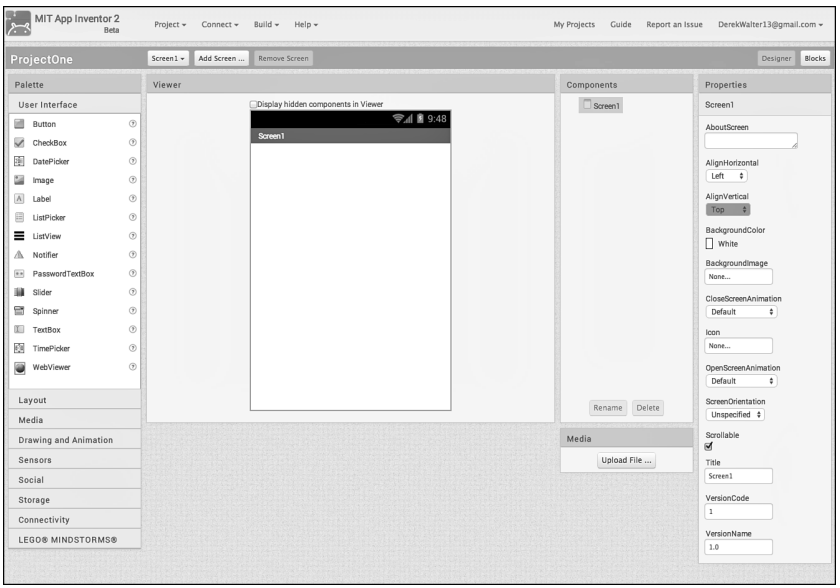


Figure P.3 In the App Inventor Designer, you design the app and add its key functionality.

The programming takes place in the Blocks Editor. There you tell the app what it should do and give specific instructions for making that happen.

The specific capabilities are programmed through connecting puzzle pieces. Over time, you will learn what each block does and find multiple ways for them to interact with one another. The pieces that do not interact will not connect with each other—another helpful way for beginners to get a sense of introductory programming principles.

MIT released App Inventor 2 in December 2013, creating a more powerful and easier-to-use tool. The most significant improvement is that all the work takes place within the browser (the previous version required a software download for some of the capabilities).

This improvement most impacts the onscreen emulator, which places a virtual Android device screen on your computer. Using this emulator provides a perspective on how the app will look and function when put to use. This is especially useful for those without an Android device or anyone in an education setting who wants to monitor student progress by viewing app builds on computer screens.

App Inventor also offers a method for using the app in real time while performing work on it: the AI Companion app (see Figure P.4). With this free download from Google Play, you can see

your app change and develop while working on it. The Companion app also works wirelessly, so you don't need to physically connect your phone to a computer while working in App Inventor.

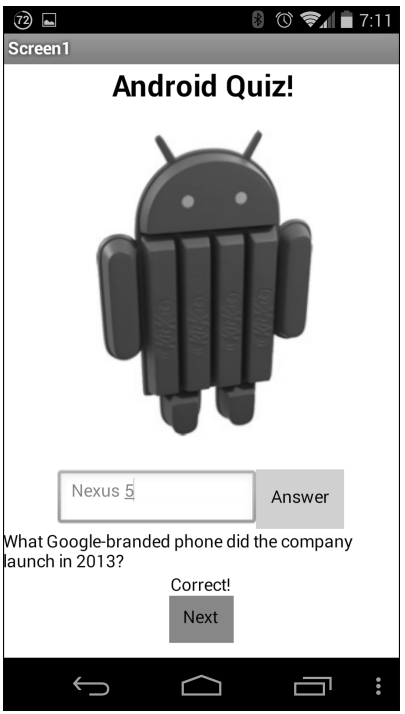


Figure P.4 An App Inventor app as viewed through the AI2 Companion. This lets you see both how the app is performing and how it looks while it is still being developed.

Why Android?

Android is not only the most popular operating system—it also is the most extendable. It is found on a wide variety of flagship devices from major handset makers, such as Samsung, HTC, LG, and Motorola. App Inventor is built to take advantage of the customization and flexibility that Android offers.

App Inventor is also a tool that is designed with those who have little to no programming experience in mind. Other platforms have a fairly high barrier of entry, but with App Inventor, you can more easily learn the essential skills for building an app with the world's most popular mobile platform.

Although many apps you create are likely to be used for practice or to share with others (see Figure P.5), MIT App Inventor is capable of creating apps that can be uploaded for distribution in the Google Play Store. With only a one-time fee of $25, anyone can put his or her skill set to work and become a registered Android developer. Chapter 12, "Distributing an App," discusses this process and walks through how to accomplish it.

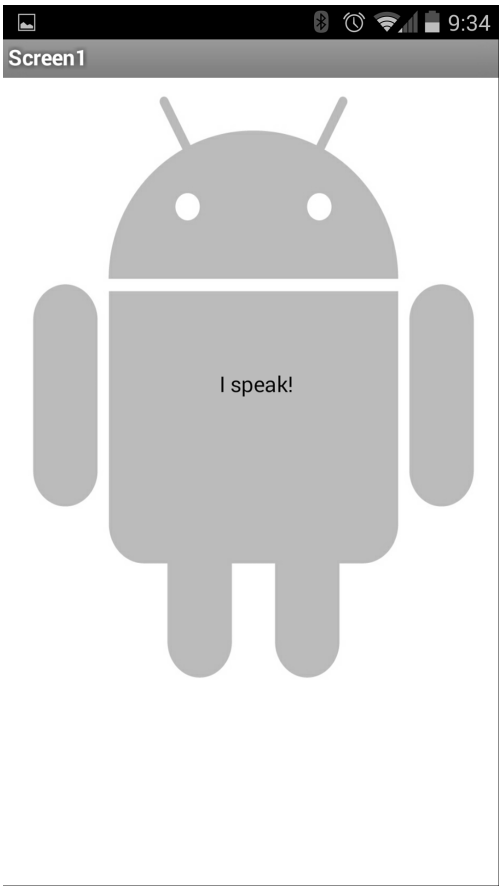


Figure P.5 An App Inventor app in the AI2 Companion.

What This Book Covers

So what exactly does this book discuss? The following sections provide a preview of the key highlights from the upcoming 12 chapters.

Chapter 1: An Introduction to Programming

Before getting started with MIT App Inventor, it is important to get some background in key computer science principles. This chapter addresses the key parts of a mobile operating system and how computer programmers should think about creating applications.

Chapter 2: Building with MIT App Inventor

This chapter provides the first detailed walkthrough of the key pieces of App Inventor. The Blocks Editor and Designer get fuller attention here, and you get to build your first app: Sherlock Is Watching.

Chapter 3: App Inventor Toolkit

App Inventor has some powerful yet easy-to-use tools for designing and building Android apps. In this chapter, the focus is on the different components available in the Designer. This serves as a good resource on the different capabilities of the Designer and will be a chapter worth referencing often.

Chapter 4: Variables

Variables are one of the key pieces of App Inventor; almost any app that you build will use them in some way. Chapter 4 covers the essentials of variables and provides some strategies for their effective use, particularly in the context of building the sample app for the chapter.

Chapter 5: Procedures

With some basic app building under your belt, it is time to take the next step and use procedures. Procedures make your life easier when it comes to building larger, more complicated apps because they enable you to group pieces of code together and recall it later.

Chapter 6: Working with Lists

As with variables, lists are a core piece of most apps that you will build with App Inventor. Lists store large pieces of data or information. The chapter culminates with a quiz app that provides some good practice in using lists.

Chapter 7: Games and Animations

Work and productivity alone are no fun. This chapter teaches the basics of the gaming and animation capabilities of App Inventor. It concludes with a simple game that could be a springboard for you to use App Inventor for other basic or more complex games.

Chapter 8: Multiple Screens and Debugging Techniques

Apps typically have multiple screens, giving users greater clarity and more streamlined access to the content of an app. This chapter focuses on strategies for using multiple screens and explores how to build them into applications. It also covers some debugging techniques for App Inventor.

Chapter 9: Using Media

Most of the smartphone apps that people use are media rich. Here you get some exposure to and practice in building media capabilities into your own apps, and you learn what is possible in App Inventor.

Chapter 10: Sensors

Many apps are location aware, letting users find specific information or customize their interaction based on location. This chapter shows you how to build some of these tools into your own apps and illustrates how they can improve a user's experience.

Chapter 11: Databases

Databases might not sound exciting, but they are a core feature of any good app that relies on storing information. This chapter looks at how to use databases effectively in different scenarios.

Chapter 12: Distributing an App

Keeping an app that you have built all to yourself is no fun. It is time to share it with others. This can be as simple as sending the file to friends and family or placing it in the Google Play Store for worldwide distribution. Whichever path you choose, this chapter assists you in getting to your destination.

Next Steps

Using App Inventor is an excellent way to build an Android app (see Figure P.6). As with many other skills in computer science, building a mobile application is an exercise in both creativity and logical thinking. You need to solve rational, complex problems while simultaneously building out a creative vision. Although you can learn App Inventor's basics rather quickly, you can build more powerful and complex applications with additional time and practice.

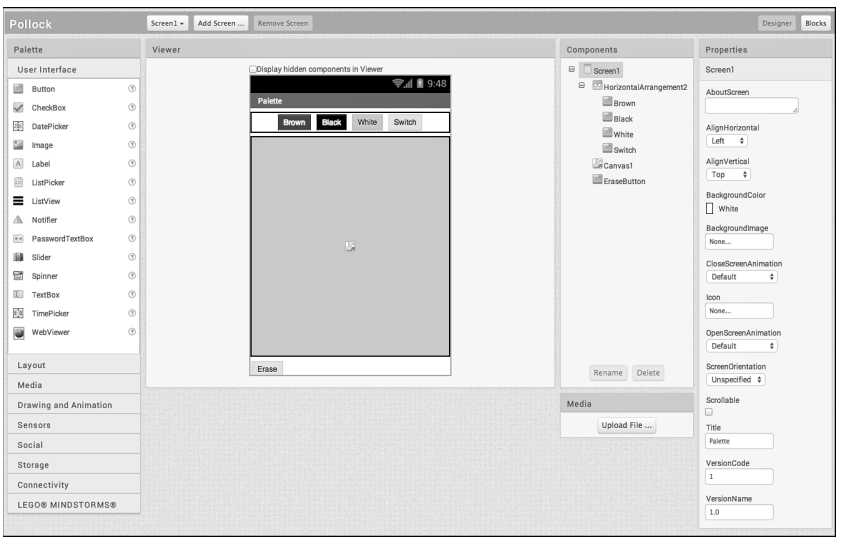


Figure P.6 You can create a variety of application types with App Inventor.

Chapter 1 begins with some essential computer science principles. Understanding how an operating system functions and what developers are actually doing when building software will give you a solid foundation in effective programming. With this established, you will be ready to build a variety of sample Android apps as you follow along in the book and then extend this skill set to your own Android apps.

This page intentionally left blank

This page intentionally left blank

2

Building with MIT App Inventor

Understanding the functionality of an app is only one part of programming. The programmer also has to focus on specific features and how to implement them.

Before the serious work of building apps begins, a brief overview of how applications perform is useful. Let's get beyond the pretty screen and graphics that you interact with and start to look at what is really happening and how to make an app perform the way you envision. After doing this, you will be able to understand how apps can request information, pull in data from the Internet, and interact with other applications.

The MIT App Inventor Site

MIT App Inventor lives on the Web, just like other online productivity tools such as Gmail and Google Drive. You do not need to download any software or save work to your hard drive before you use App Inventor (see Figure 2.1).

The choice of web browser is very important; the App Inventor team recommends using Google Chrome or Firefox. Choosing a different browser, such as Internet Explorer, could result in errors or other complications when working with App Inventor.

Exploring the App Inventor site is a good way to get a feel for what is available. To begin, launch your browser and go to appinventor.mit.edu (see Figure 2.2). The home page includes the portal to the App Inventor tool, along with many online tutorials and other helpful materials.

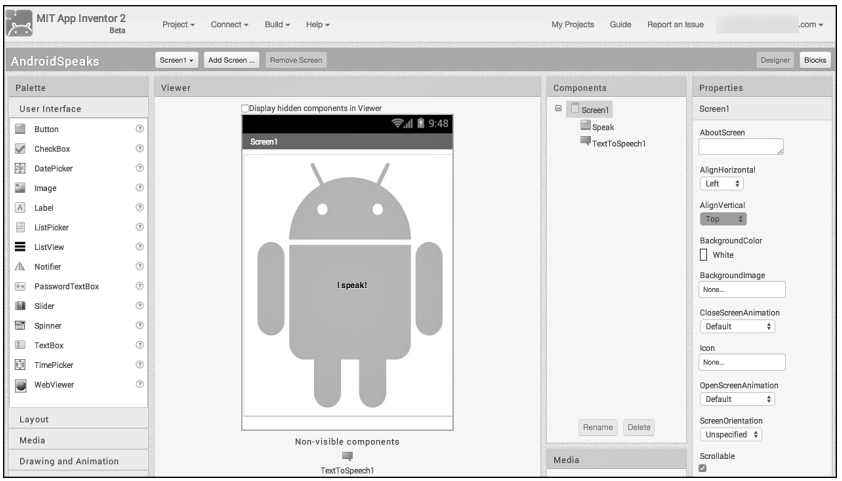


Figure 2.1 The Designer interface in MIT App Inventor.

Signing In

To begin a session with App Inventor, click the Create button at the top of the home page (see Figure 2.2).

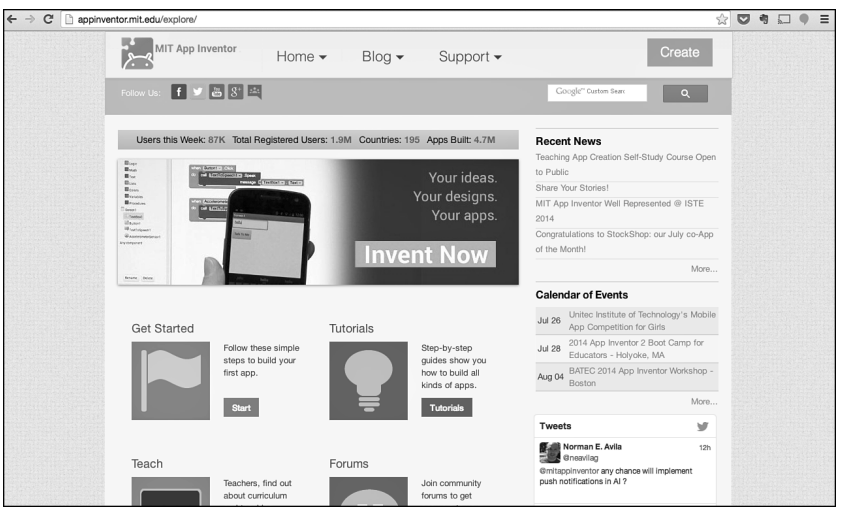


Figure 2.2 The App Inventor home page—click Create to get started.

Next, App Inventor asks permission to connect to your Google account. This can be a personal Google account (one that ends with an @gmail.com address) or a Google apps account managed by a university, business, or other type of organization (see Figure 2.3).

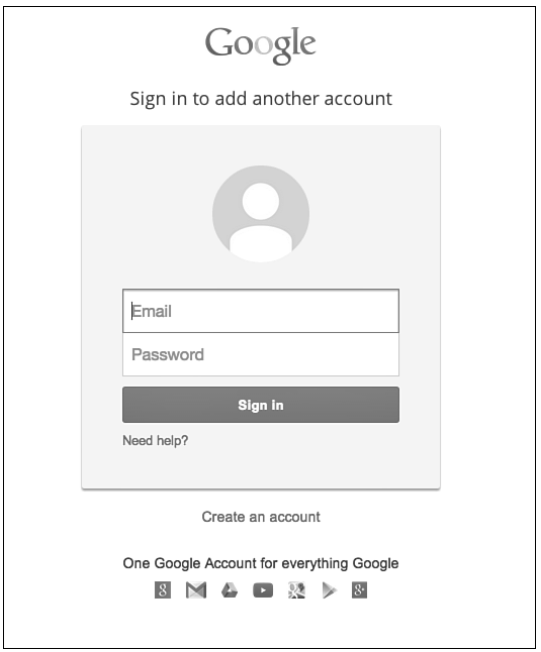


Figure 2.3 Add your Google account to connect to App Inventor.

After signing in with your Google account, you must authorize App Inventor to access your Google account so that it can verify your login information. If you select Remember This Approval for the Next 30 Days, then you will not need to repeat this step when you return to work on apps (see Figure 2.4). At the end of the 30 days, you simply need to reauthorize access.

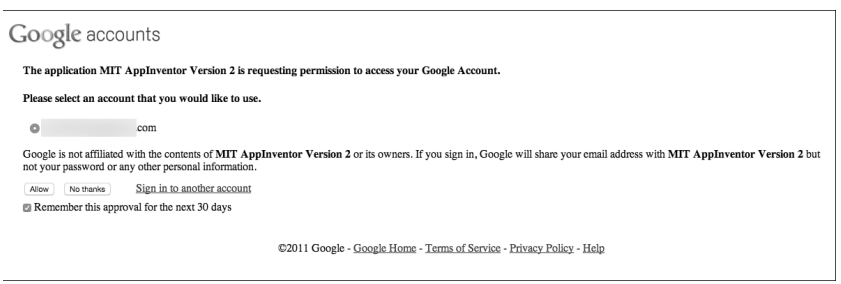


Figure 2.4 Authorize your Google account.

The next screen is the file system where App Inventor projects are stored. As you create more projects over time, you can find them there, just as a folder on your computer holds a list of all your documents saved to that location.

Next, click New Project and then type **CatWatch** into the box (spaces are not allowed). This takes you to the Designer screen.

Designer

App building begins in the Designer. Here you create the user interface, or the "look and feel" of the app. You also add the components needed to receive input from the user, as well as the components needed to display output or information to the user. The Designer also is where you specify which nonvisible components the app will use, such as the dialer, GPS, or SMS. Notice that because we are in Designer, the Designer button in Figure 2.5 is slightly grayed out in the top-right corner of the screen. This button, along with the one next to it, labeled Blocks, indicates which editor you are using.

The left side of the screen features the Palette (see Figure 2.5), which, as the name implies, is the space for all the creation tools (the next chapter details the full suite).

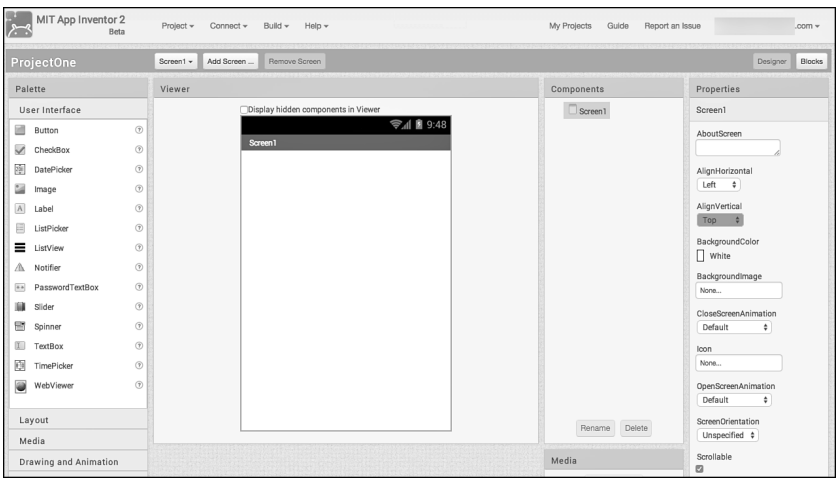


Figure 2.5 The App Inventor Designer screen.

Blocks Editor

The Blocks Editor is where you will be programming an app's behavior (see Figure 2.6). Here you will add the commands that do the work of the app. As just noted, you access it from the Blocks button at the top right.

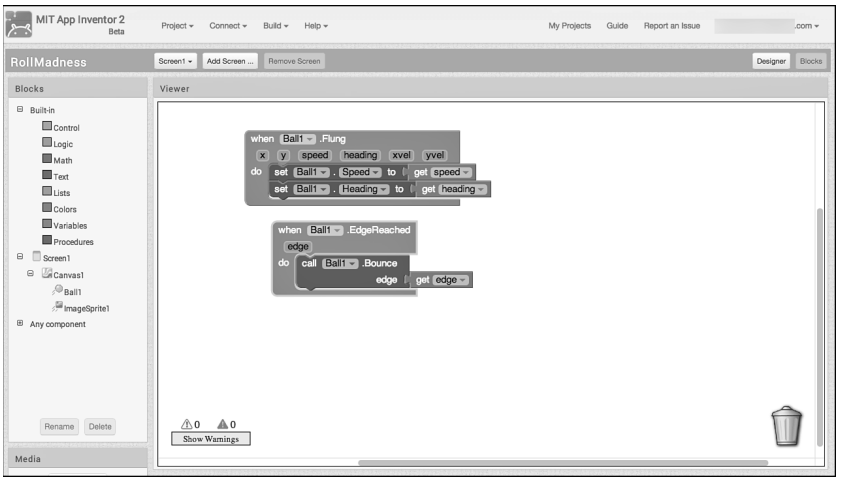


Figure 2.6 More specific programming takes place in the Blocks Editor.

MIT App Inventor uses the metaphor of drawers containing puzzle pieces for programming. Each item in the Blocks palette under Built-in is considered a drawer. The drawers contain the puzzle-looking pieces. The programming is accomplished by connecting the puzzle-looking pieces. Despite its seeming simplicity, App Inventor has many powerful capabilities that enable the user to build complex applications.

To better understand what programming an app entails, it is useful to understand what is going on inside an application.

The AI2 Companion App

App Inventor has a useful tool for continuously seeing your app in real time on an Android device during each step of the development process.

You can find the MIT AI2 Companion app (see Figure 2.7) in the Google Play Store by performing a search for "MIT AI2 Companion."

When you are building your app, the computer and Android device must be connected to the same wireless network (the desktop machine might have a wired connection). To connect your app to your device in App Inventor on your computer, click AI Companion on the Connect tab (see Figure 2.8).

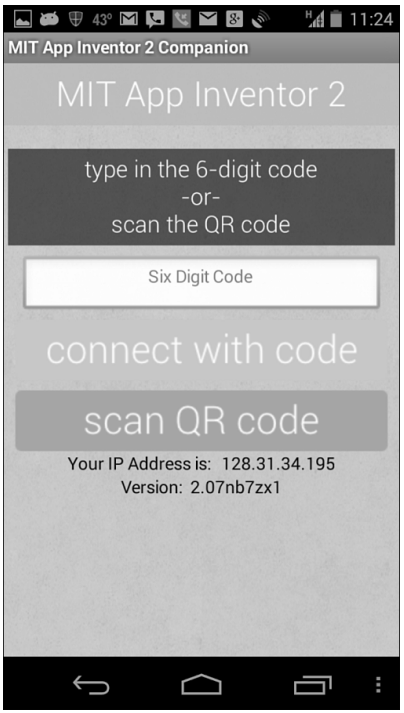


Figure 2.7 The AI2 Companion Android app.

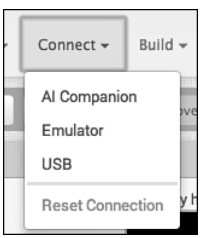


Figure 2.8 Connecting to the Companion app.

You can then type in a six-digit code or scan the QR code with your device (see Figure 2.9), using the App Inventor app. Doing so brings up a live view of your app. As you add elements to it with the MIT App Inventor software, those changes are reflected in real time on your device.

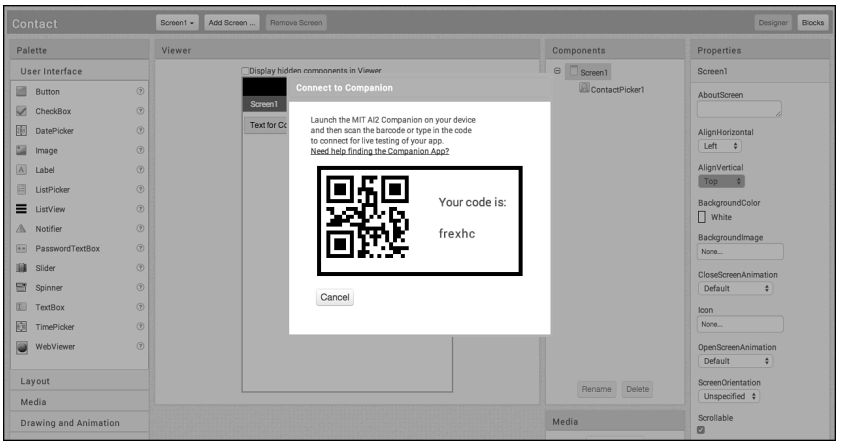


Figure 2.9 Connecting to the Companion app.

The Android Emulator

As another option, App Inventor has an Android emulator that puts a simulated Android screen on the computer desktop. This enables you to view the app's progression if you do not have an Android device. It also is useful for anyone using App Inventor in a classroom environment.

An installer package is available for Windows or Mac. Choose the proper platform from the App Inventor site and download it to your computer (see Figure 2.10).

To view your app, choose Emulator from the same menu in Figure 2.8.

USB Connection to Android Device

Another option is to connect your device to the computer with a USB cable. This method provides the benefit of seeing the app on your Android device just as if you were using the MIT App Inventor Emulator application. This option also does not require a wireless network connection (see Figure 2.11).

First, you need to install the App Inventor setup software to your Mac or Windows PC. Many Android devices also require the installation of driver software (available at the device manufacturer's website). Your device might require other changes to the device's settings. Android has a web page that describes the potential changes and implications at http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html.

After you have properly configured your device, select Connect and choose USB from the menu. After a few moments, the app should appear live on your device.

Figure 2.10 The emulator download package. The emulator can be downloaded to a Mac or a Windows PC.

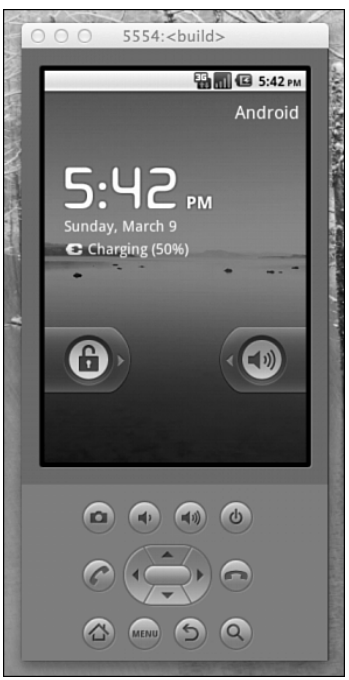


Figure 2.11 The Android emulator.

Getting Inside an App

Apps have an internal design, the programming that works with the user interface. Effective programming entails knowing how to use the internal components to effectively implement what is visible to the user.

A good way to think about the internal pieces of an app is to focus on components and actions. The components are the various tools you find in App Inventor to create tasks. Think about the Design Editor and all the onscreen tools: buttons, images, drawings, and so on. In the Design Editor, you pull into your app all the pieces that make up the user interface, or what the users of your app will see on the device's screen.

Event Handlers

All of your blocks, the pieces that make your app perform tasks, will be connected with an event handler. Events are created whenever something in the real world happens to the app, such as when the user clicks a button, the phone's location changes, or the phone receives a text message. Blocks exist for just about everything you want to do in an app; taking a picture, checking the GPS location, displaying text, changing the color of a component, finding out what the user entered into a text box, and so on. You can add (or remove) these blocks from an event handler, allowing the programming to determine the precise set of actions to take when the user presses a button.

When any event happens, App Inventor runs whatever blocks are inside the **event handler** block for that event. For example, Figure 2.12 shows a button labeled Speak; the block **when Speak.Click do** is the event handler for when that button is clicked.

Imagine that we wanted to write an app to speak written text. We would need a button to start the process. To do this, we would have to drag the **action** block into the event handler. Whenever the button is clicked, that block will be called, and the device will move to whatever blocks have been placed inside the event handler. The event handler will grow and shrink as needed to accommodate whatever blocks are placed inside it (in Figure 2.13, this involves speaking some text). The way it is right now, however, the app has a button and a text-to-speech component, but it won't do anything because the event handler is empty. After we drag in some action blocks, it will do those actions whenever the button is clicked. Note that the component used to initiate the action (such as a button) is usually different than the component that does the requested action (such as taking a picture or sending a text message).

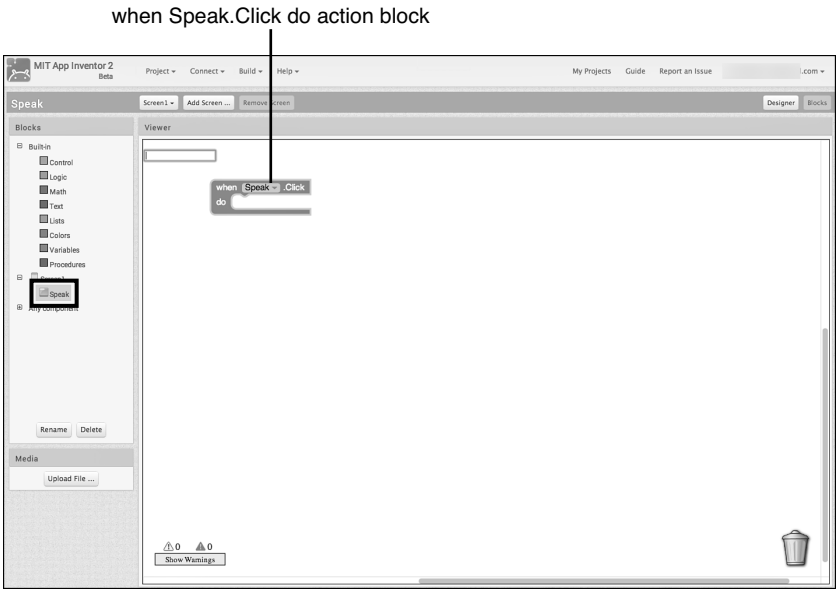


Figure 2.12 An event handler in the Blocks Editor.

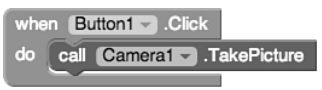


Figure 2.13 An event handler.

Doing One Thing at a Time

Modern smartphone applications perform a large number of tasks simultaneously—or at least appear to do so. All computers, including smartphones, are so good at switching between different tasks so quickly that it appears they are doing two unrelated things simultaneously.

Many computers have multiple cores, or processing units. These computers really can do more than one thing because each core can work on a separate task. But those cores are also switching very quickly between many tasks that the system and user are trying to do. You might have two, or four, or even more separate cores in your computer or phone, but between the operating system and the apps, the processor has hundreds of small tasks to work on.

Most of the things you will do in App Inventor are single task, meaning that only one task is actually running at a time. However, they can happen quickly and get out of the way for other things to run. In the code blocks, *only one event handler can run at a time*. So when an event happens and the event handler begins executing the blocks you put in it, all other event handlers must wait until this one is done.

Events that take place while a handler is already running are put in a queue and will run when it is their turn. Most event handlers run much faster than events are generated, so this is often not an issue. The most common actions, such as updating the text in a label or looking at the state of a check box, occur nearly instantly. However, other actions, such as working through a large collection of data, might take a long time, and the app will appear frozen until the process finishes.

While an event handler is running, the display isn't updated. Again, this isn't an issue most of the time, but if you have an event handler that is taking long enough for a user to notice, the display will appear frozen until the event handler is complete.

You might notice that some features in App Inventor take time, such as playing sounds or music. Other features have to wait for something on the Internet to respond, which can cause an unpredictable amount of delay time. But the display doesn't freeze when you play music, and the app continues to work while waiting for a web page to load. Android provides the means to allow some tasks, such as playing music, to run in the background without disrupting the normal actions your app performs. Android provides other means of dealing with actions that could take a long time, or that might never finish, such as loading a web page or waiting for the user to take a picture, without affecting the normal functioning of your app. The actions for music and sound are made possible by the Android system. Your app simply hands off the sound file to the phone's operating system and tells the phone to handle it. The music plays while the app continues to work.

Later in the text, we provide more details on how App Inventor switches tasks. You can use that knowledge to make better apps.

Exercise: Sherlock Is Watching

Next we create an app that uses some of the basic functionality described in this section. As with the other apps you will build in this book, there is flexibility in the specifics. Feel free to experiment after following the steps to get a feel for how all the pieces work and the type of customizations possible. Learning to build apps is a process that involves both following the step-by-step directions and branching out on your own.

1. Navigate back to your projects by clicking My Projects at the top of the page (see Figure 2.14).
2. If you created a CatWatch app earlier, select it from your list of projects; otherwise, create a new project called CatWatch.
3. Click User Interface in the Palette. The Palette then expands to reveal several choices, such as Button, Checkbox, and Clock. Click and hold the Button choice, and then drag it onto the Android home screen in the Viewer (see Figure 2.15). A button appears in the Viewer, indicating that an element has been added to the screen. The button's name also is added to the Components tree.

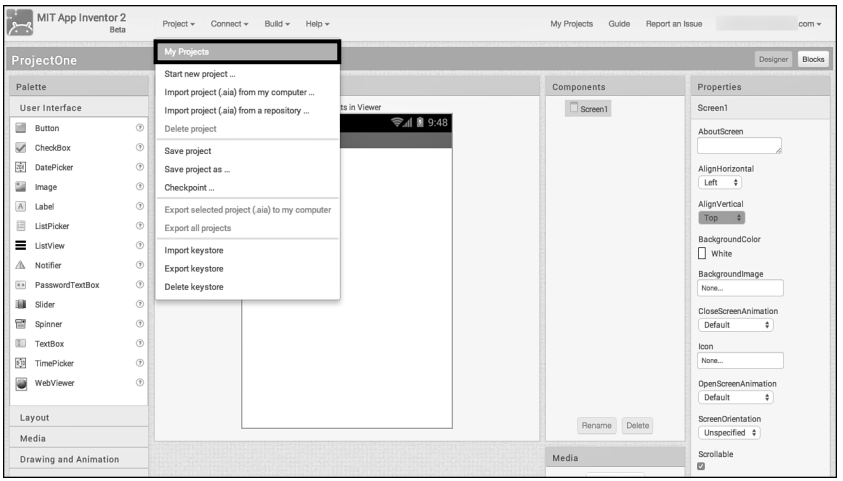


Figure 2.14 The My Projects button is for creating new projects and accessing existing ones.

Figure 2.15 Adding a button to the Viewer.

4. The button shows with the following default text: Text for Button1. Let's change the name of the button. Click the label in either the Viewer or the Components tree and then click the Rename button. In the box that launches, give it another name, such as Meow.

5. Notice that changing the name of the button does not change the button's text. To change the displayed text, click the button and then find the box labeled Text inside the Properties pane. Then highlight the text and type Meow. The text will change in the Viewer.

Adding an Image

Images are an effective way to add some visual polish to an app. Next, we insert an image into the app.

1. If you have not done so already, download the CatIsWatching image from the book's InformIT page.
2. Find the box labeled Media, which is just below the Components box. Click the Upload File button (see Figure 2.16) and then upload the CatIsWatching file. This adds the image to the app, making it available to any component that uses the image. Note that the filename appears in the media box.

Figure 2.16 Uploading an image.

3. Click User Interface in the Palette. Then click and hold the image and drag it onto the Android home screen in the Viewer. A blue bar shows where you can place the image. By moving the mouse, you can place the image above or below the Button. When the blue bar is below the Button, drop the image. Next click the Image1 button in the Components box. The Properties pane updates to display the properties associated with the image.

4. Click the box labeled Picture. All available images in your app are listed; select the CatIsWatching image. Click OK to see the image appear in the Viewer (see Figure 2.17).

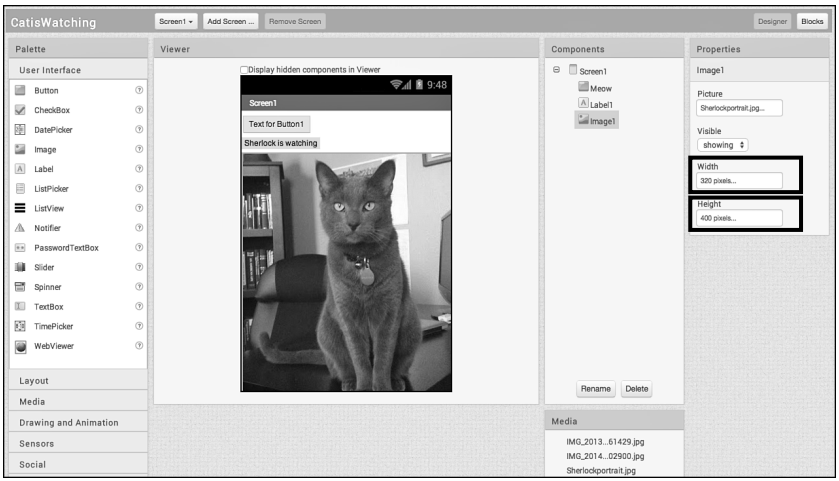


Figure 2.17 Uploading an image.

5. The image of Sherlock the cat needs to be tweaked to properly fill the screen. Click in the width box and then click the box that reads Pixels. Type **320** pixels.
6. The procedure is the same for the height: Click the height box and then type **400** pixels.
7. This app requires one more element. From User Interface, click and drag the Label component. Again, watching the blue insertion bar, drop the Label between the button and the Image components. Next, select Label1 in the Components box. Click the button underneath the BackgroundColor label, which is currently set to None. Then choose Green.
8. It is time to write some text inside the box. Go to the Text box and type **Sherlock is watching**. Choose TextAlignment and change this selection to Center. Notice that this does not change the position of the label—it changes only the text inside it (see Figure 2.18).
9. Click Blocks in the upper-right corner. You should see Meow, Label1, and Image1 underneath Screen1. This is where you would drag these components into the Blocks Editor for further programming.

This first app is relatively simple, but it should make you feel more familiar with the core pieces of App Inventor. Later apps will make more extensive use of the interface elements and how they can be programmed. We will also explore how to put this app on your own device and interact with it.

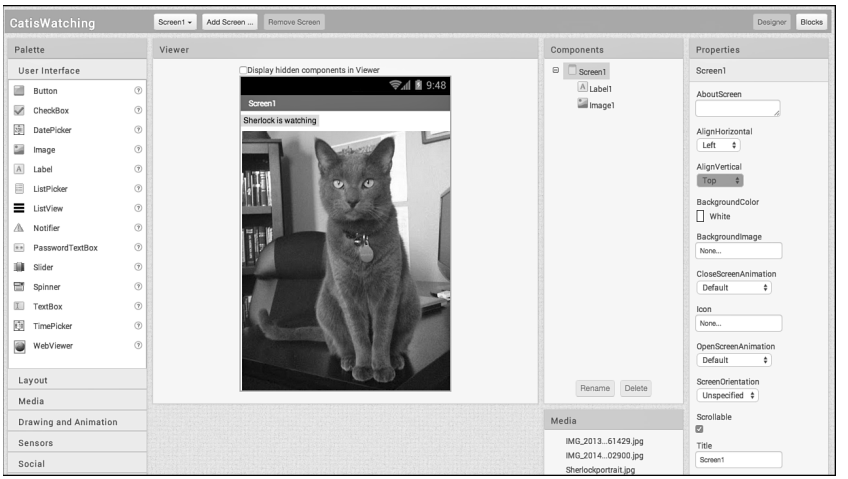


Figure 2.18 The completed image and label.

What Can You Build?

This first exercise demonstrates a little of what is possible with App Inventor. Throughout this book, you will learn how to maximize the power of App Inventor to build a variety of apps.

The skills you gain will also empower you to begin your own experimentation and build apps beyond the walk-throughs provided in this text.

The following sections preview some of the other apps and exercises we will be exploring throughout the book. With careful attention to detail and some creativity, you will be able to build these apps and have the foundations for creating your own set of applications.

Speak, Android!

Give your Android device a voice. This simple app (see Figure 2.19) teaches you how to enable an image to respond to touch and speak on command. You can also use it to explore other ways to work with images.

Pollock

Named after Jackson Pollock, the American artist who helped popularize abstract art, this app turns an Android device into a canvas for color (see Figure 2.20). You will learn how to turn buttons into paint and use the Canvas component.

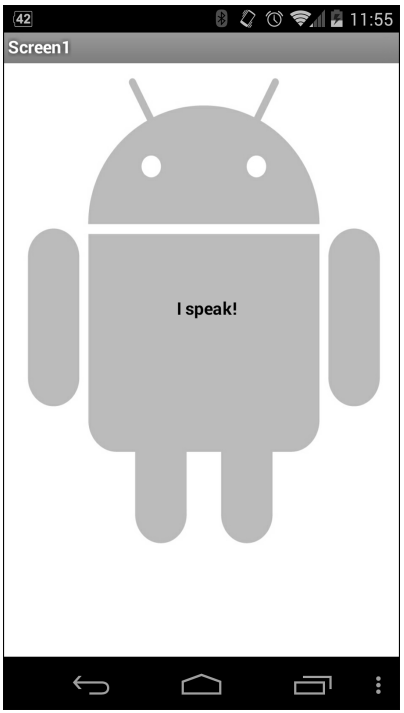


Figure 2.19 The Speak, Android app.

Figure 2.20 The Pollock app.

Fore

MIT App Inventor has some surprisingly powerful tools for creating games. Various motion-enabled commands enable you to create some powerful games. The game you will be building will show how you can use the canvas and various sprites to create a game field and objects that can be manipulated while playing (see Figure 2.21).

Android Quiz

Games can be fun to create, but imagine being able to use an app for your own productivity. Android Quiz (see Figure 2.22) demonstrates that you can create an actual assessment app.

Uploading to Google Play

Later chapters discuss several ways to share your app with others. However, the ultimate step is uploading your app to Google Play for distribution to other Android users.

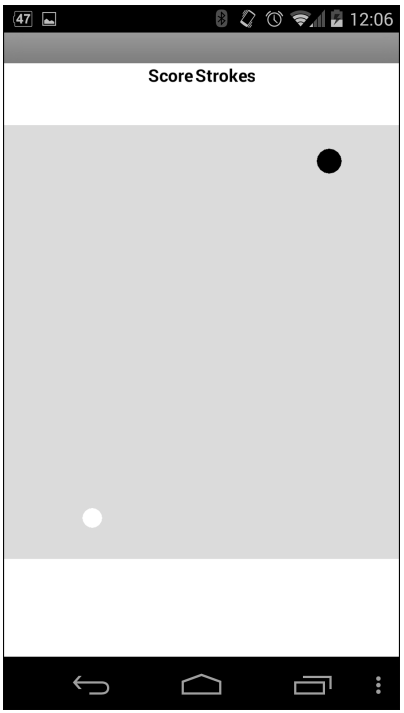


Figure 2.21 The Fore golf game.

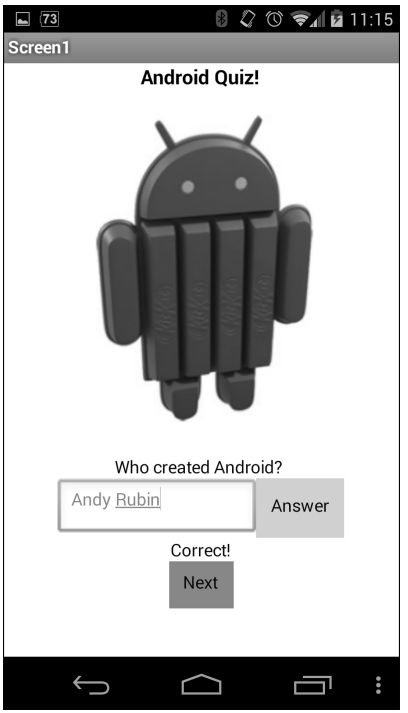


Figure 2.22 Android Quiz.

Summary

MIT App Inventor is a powerful tool that beginning coders or anyone dabbling with mobile technology can use to build Android apps. In this chapter, we looked at the kind of apps that are possible with this cloud-based tool.

Computer science is applied reasoning using both art and science. It requires the ability to communicate ideas through a combination of language and powerful technology. Hopefully this first app has demonstrated that App Inventor is an excellent starting point for anyone looking to create with computer technology, whether professionally or recreationally.

In the next chapter, we build an app that includes both an image and sound, and we look at how to see the app on your own device.

Exercises

1. Add another button to the Viewer. Change the text, trying different configurations to see how they fit in the Viewer.
2. Upload a different image. Download one from the Web or upload an image on your computer. Try different configurations for size in relation to the rest of the app. Decide on the optimal size for images in the context of the rest of the content.
3. Change the text and color scheme in the label. Try labels in different locations of the app. Pay attention to how the Designer works in terms of the placement of components and how you can customize the components.

This page intentionally left blank

Index

A

B

C

D

E

F

G

M

N

O

P

Q

R

S

T

U

V

W-X

Y-Z