Foreword

Our consumer culture gives us all sorts of opportunities for entertainment, pleasure and sometimes even learning. However, by and large, these are passive activities. That's OK—we all like to kick back sometimes and be entertained—but it shouldn't be the whole picture. In addition to the appeal of consuming, there's the satisfaction of producing—that is, of creating. It's the joy and pride that results when we draw a picture, build a model airplane, or bake some bread.

The high-tech objects (like cell phones, tablet computers, TVs, etc.) that we use today to consume entertainment and information are black boxes to most of us. Their workings are incomprehensible and, while there are capabilities in some of them that enable the user to draw pictures, make videos, etc., they are not, in and of themselves, creative media. In other words, most people can't create the apps that run on these gadgets.

What if we could change that? What if we could take creative control of our everyday gadgets, like cell phones? What if building an app for your cell phone was as easy as drawing a picture or baking a loaf of bread? What if we could close the gap between the objects of our consumer culture and the media of our creative lives?

For one, it could demystify those objects. Rather than being black boxes, impenetrable to our sight, they become objects that can be tinkered with. They become objects capable of our understanding. We gain a less passive and more creative relationship to them, and we get to play with these devices in a much deeper, more significant way when we can actually build things for them.

When Hal Abelson first spoke to me about the idea that became App Inventor, we talked about the unique motivating force that cell phones could have in education. He wondered if we could use that motivating force to help introduce students to concepts in computer science. As we built it and tried it in classes like Dave Wolber's, we started to realize that something even more powerful was happening: App Inventor was starting to turn students from consumers to creators. Students thought it was fun and exhilarating to build apps for their phones! When one of Dave's students built the simple but powerful "No Texting While Driving" app, we really started to imagine what would happen if anybody, not just professional software engineers, could build an app.

So we worked hard to make App Inventor easier and more fun to use. We've worked to make it more powerful (but still simple) as well. And we're continuing this work—App Inventor is still a beta product and we have exciting plans for it.

The authors of this book are truly world-class educators and software engineers. I'd like to personally thank them for their work in building, testing, and documenting the App Inventor for Android product and, of course, for writing this wonderful book.

Now go, unleash your creativity and build an app!

—*Mark Friedman*
Tech Lead and Manager of the App Inventor for Android project, Google

Preface

You're on your regular running route, just jogging along, and an idea for the next killer mobile app hits you. All the way home, you don't even care what your time is, all you can think about is getting your idea out there. But how exactly do you do that? You're no programmer, and that would take years, and time is money, and…well, someone has probably done it already anyway. Just like that, your idea is dead in the water.

Now imagine a different world, where creating apps doesn't require years of programming experience, where artists, scientists, humanitarians, health-care workers, attorneys, firefighters, marathon runners, football coaches, and people from all walks of life can create apps. Imagine a world where you can transform ideas into prototypes without hiring programmers, where you can make apps that work specifically for you, where you can adapt mobile computing to fit your personal needs.

This is the world of App Inventor, Google's new visual programming tool for building mobile apps. Based on a visual "blocks" programming method that's proven successful even with kids, App Inventor dramatically lowers the barriers to creating apps for Android phones and devices. How about a video game where the characters look like you and your friends? Or a "did you pick up the milk?" app that reminds you if it's after 3 p.m. and you're near the grocery store? Or a quiz app you give your significant other that's in fact a surprise marriage proposal? "Question 4: Will you marry me? Press the button to accept by sending a text message." Someone really created an App Inventor app to propose marriage like this, and she said yes!

A Blocks Language for Mobile Phones

App Inventor is a visual, drag-and-drop tool for building mobile apps on the Android platform. You design the user interface (the visual appearance) of an app using a web-based graphical user interface (GUI) builder, then you specify the app's behavior by piecing together "blocks" as if you were working on a puzzle.

Figure 0-1 shows the blocks for an early version of an app created by Daniel Finnegan, a university student who had never programmed before. Can you tell what the app does?

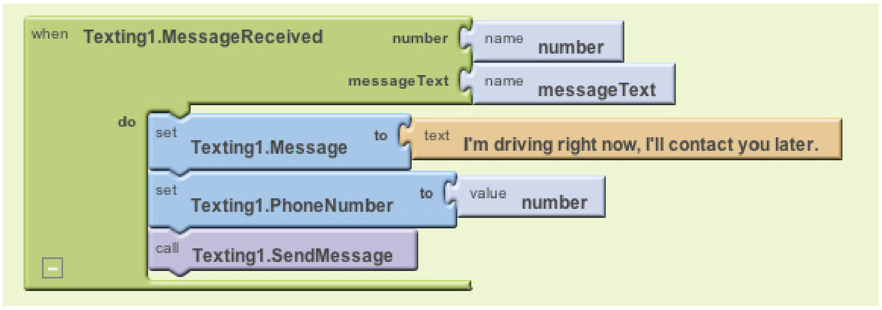
when Texting1.MessageReceived
number
name number
messageText
name messageText
do
set Texting1.Message to
text I'm driving right now, I'll contact you later.
set Texting1.PhoneNumber to
value number
call Texting1.SendMessage

Figure 0-1. App Inventor blocks specify the functionality of your app

The app is a text "answering machine." You launch it when you're driving and it auto-responds to the texts you receive.

Because the blocks are more understandable than traditional programming code, you're immediately drawn in, and the real-world utility gets you asking questions like: Can I make it so the received texts are spoken aloud? Can I make it so the response sent back could be customized? Can I write an app that lets people vote for something by text, like on American Idol? The answer to all these questions is "yes," and in this book, we'll show you how.

What Can You Do with App Inventor?

Play

Creating apps for your phone is fun, and App Inventor promotes exploration and discovery. Just open App Inventor in a web browser, connect your phone, and start putting together blocks like those in Figure 0-1. You can immediately see and interact with the app you're building on the phone. So you're programming, but you're also emailing your friend to send you a text to test your app, or you're controlling a LEGO NXT robot with the app you just built, or you're unplugging the phone and walking outside to see if your app is using the location sensor correctly.

Prototype

Have an idea for an app? Instead of writing it down on a napkin or letting it float off into the ether, build a quick prototype. Prototypes are incomplete and unrefined working models of your idea. Expressing an idea in text is like writing a to a friend or loved one with prose; think of an App Inventor prototype as poetry to a venture capitalist. In this way, App Inventor can serve as an electronic napkin for mobile app development.

Build apps with personal utility

In the current state of the mobile app world, we're stuck with the apps we're given. Who hasn't complained about an app and wished it could be personalized or adjusted in some way? With App Inventor, you can build an app exactly how you want it. In Chapter 3, you'll build a MoleMash game that lets you score points by touching a randomly moving mole. But instead of using the image of the mole in the tutorial, you can customize it so that you mash a picture of your brother or sister—something that only you might want to do, but who cares? In Chapter 8, you'll write a quiz app that asks questions about US Presidents, but you can easily customize it to ask questions on any topic you want, from your favorite music to your family history.

Develop complete apps

App Inventor is not just a prototyping system or an interface designer—you can build complete, general-purpose apps. The language provides all the fundamental programming building blocks like loops and conditionals, but in block form.

Teach and learn

Whether you're at a middle school, high school, or university, App Inventor is a great teaching and learning tool. It's great for computer science, but is also a terrific tool for math, physics, entrepreneurship, and just about any other discipline. The key is that you learn by creating. Instead of memorizing formulas, you build an app to, say, find the closest hospital (or mall!). Instead of writing an essay on Black History, you create a multimedia quiz app with video and speeches from Martin Luther King, Jr., and Malcolm X. We think App Inventor, and this book, can be a great tool in classes throughout the curriculum.

Why App Inventor Works

Most people say that App Inventor is easy to use because of its visual, drag-and-drop interface. But what does this mean? Why is App Inventor so easy to use?

You don't have to remember and type instructions

One of the biggest sources of frustration for beginning programmers comes from typing in code and having the computer spit back indecipherable error messages. This frustration discourages many beginners from programming before they even get to the more fun, logical problem solving.

You choose from a set of options

With App Inventor, the components and blocks are organized into drawers that are readily available to you. You program by finding a block—which helps specify the functionality you want to build—and dragging it into the program. You don't have to remember what the instructions are or refer to a programming manual.

Only some blocks plug in to each other

Instead of chastising programmers with cryptic error messages, App Inventor's blocks language restricts you from making many mistakes in the first place. For instance, if a function block expects a number, you can't plug in text. This doesn't eliminate all errors, but it sure helps.

You deal with events directly

Traditional programming languages were designed when programming was like working with recipes, or sets of instructions. But with graphical interfaces, and especially with mobile apps where events can happen at any time (for example, receiving a text message or phone call), most programs are not recipes, but are instead sets of event handlers. An event handler is a way of saying, "When this happens, the app does this." In a traditional language like Java, you have to understand classes, objects, and special objects called listeners to express a simple event. With App Inventor, you can say, "When a user clicks this button..." or "When a text is received..." by dragging out a "When" block.

What Kind of Apps Can You Build?

You can build many different types of apps with App Inventor. Use your imagination, and you can create all kinds of fun, useful apps.

Games

People often begin by building games like MoleMash (Chapter 3) or apps that let you draw funny pictures on your friend's faces (Chapter 2). As you progress, you can build your own versions of more complex games like Pac-Man and Space Invaders. You can even use the phone's sensors and move characters by tilting the phone (Chapter 5).

Educational software

App building is not limited to simple games. You can also build apps that inform and educate. You can create a quiz app (Chapter 8) to help you and your classmates study for a test, or even a create-a-quiz app (Chapter 10) that lets the users of your app create their own quizzes (think of all the parents that would love this one for those long road trips!).

Location-aware apps

Because App Inventor provides access to a GPS-location sensor, you can build apps that know where you are. You can build an app to help you remember where you parked your car (Chapter 7), an app that shows the location of your friends or colleagues at a concert or conference, or your own custom tour app of your school, workplace, or a museum.

High-tech apps

You can create apps that scan bar codes, talk, listen (recognize words), play music, make music (Chapter 9), play video, detect the phone's orientation and acceleration, take pictures, and make phone calls. Smartphones are like Swiss-Army knives for technology, and a group of Google engineers has dedicated themselves to making that technology easy to control through App Inventor.

SMS apps

"No Texting While Driving" (Chapter 4) is just one example of the SMS processing apps you can build. You can also write an app that periodically texts "missing you" to your loved ones, or an app like "Broadcast Hub" (Chapter 11) that helps coordinate large events. Want an app that lets your friends vote for things by texting, like on American Idol? You can build it with App Inventor.

Apps that control robots

Chapter 12 shows how to create an app that acts as a controller for a LEGO robot. You can use the phone as a remote control, or you can program it to be a "brain" that the robot carries around with it. The robot and phone communicate via Bluetooth, and App Inventor's Bluetooth components let you create similar apps that control other Bluetooth devices.

Complex apps

App Inventor dramatically lowers the entrance barrier to programming and lets you build flashy, high-tech apps within hours. But the language also provides loops, conditionals, and other programming and logic constructs necessary to build apps with complex logic. You'll be surprised at how fun such logic problems can be when you're trying to build an app.

Web-enabled apps

App Inventor also provides a way for your apps to communicate with the Web. You can write apps that pull in data from Twitter or an RSS feed, or an Amazon Bookstore Browser that lets you check the online cost of a book by scanning its barcode.

Who Can Build Apps?

App Inventor is freely available for anyone to use. It runs online (instead of directly on your computer) and is accessible from any browser. You don't even need a phone to use it: you can test your apps on an included Android emulator. As of January 2011, there were tens of thousands of active App Inventor users and hundreds of thousands of apps.

Who are these app builders? Were they already programmers when they started? Some of them were, but most were not. One of the most telling experiences has been the courses that coauthor David Wolber taught at the University of San Francisco. At

USF, App Inventor is taught as part of a general education computer science course targeting primarily business and humanities students. Many students take the course because they either hate or are afraid of math, and the course fulfills the dreaded Math Core requirement. The vast majority have never even dreamed of writing a computer program.

Despite their lack of prior experience, the students have been successful in learning App Inventor and building great apps. An English major created the first "No Texting While Driving" app; two communications majors created "Android, Where's My Car?"; and an International Studies major created the "BroadcastHub" app (Chapter 11). When an art major knocked on Wolber's office door one night well after hours, asking how to write a `while` loop, he knew that App Inventor had dramatically changed the landscape.

The media grasped the significance as well. The *New York Times* called App Inventor "Do-It-Yourself App Creation Software." The *San Francisco Chronicle* reported on the USF students' work in an article, "Google brings app making to the masses." Wired magazine featured Daniel Finnegan, the author of "No Texting While Driving," and wrote that "Finnegan's story illustrates a powerful point: It's time for computer programming to be democratized."

The cat is, as they say, out of the bag (your first app will involve a kitty, by the way). App Inventor is now used in high school courses; in the Technovation Challenge, a San Francisco Bay Area after-school program for high school girls; the Lakeside School in Seattle; and in new introductory courses at several universities. There are now thousands of hobbyists, businesspersons, marriage-proposers, and tinkerers roaming the App Inventor site and forum (*http://appinventor.googlelabs.com/forum/*). Want to get in on the action? No programming experience is required!

Conventions Used in This Book

This book uses the following typographical conventions:

Bold, green text
: Used to refer to program blocks that appear in App Inventor programs.

Italic
: Used to indicate email addresses, URLs, filenames, pathnames, and to emphasize terms when they're first introduced.

`Constant width`
: Indicates Python code and component, property, variable, and function names.

This icon signifies instructions for testing the app being developed.

This icon indicates a tip, suggestion, or general note.

How to Use This Book

This book can be used as a textbook for middle school, high school, and university courses or as a how-to guide for aspiring app developers. The book is split into two sections: a set of tutorials for building specific apps, and an Inventor's Manual section organized more like a typical programming textbook. The tutorials progress in complexity as you go, from "Hello Purr" in Chapter 1—which lets you click a cat to make it meow—to a web-enabled app that lets you scan a book to view information from the Amazon web service (Chapter 13).

Working through the tutorials in order is advantageous from a conceptual viewpoint, but as you start to feel comfortable with the system, you may want to jump around. The tutorials provide step-by-step instructions and snapshots of the blocks to help, and you'll be referred to chapters in the Inventor's Manual section to help solidify your understanding of the concepts.

One advantage of having a book at hand is that the App Inventor environment takes up most of your computer screen, so there's not really room for an on-screen tutorial window. We envision folks setting the book next to them as they walk through the tutorials and build each app. Then, we hope, people will be so engrossed that they'll use the book away from the computer, to read the more conceptual Inventor's Manual chapters.

For teachers and students, the book can serve as a textbook for an introductory computer science course, or as a resource for any course in which students learn by building. In our experience, a sequence of tutorial→discussion→creativity works best. So you might first assign the task of completing a couple of the apps in the tutorial chapters, with the minimal expectation of the students mechanically building the apps. Then you can assign a chapter from the Inventor's Manual section and slow the process down with some in-class discussion and lecture. The third phase encourages exploration: have the students build some of the suggested variations at the end of each tutorial, without detailed instruction, and then follow this up with a creative assignment in which students come up with their own ideas for apps and then implement them.

You can also download files for each chapter, along with complete code samples, here: *http://examples.oreilly.com/0636920016632/*.

Acknowledgments

The educational perspective that motivates App Inventor holds that computing can be a vehicle for engaging powerful ideas through active learning. As such, App Inventor is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s, and whose influence persists today through many activities and programs designed to support computational thinking.

App Inventor's design draws upon prior research in educational computing and upon Google's work with online development environments. The visual programming framework is closely related to the MIT Scratch programming language. The specific implementation here is based on Open Blocks, which is distributed by MIT's Scheller Teacher Education Program and derives from MIT thesis research by Ricarose Roque. We thank Eric Klopfer and Daniel Wendel of the Scheller Program for making Open Blocks available and for their assistance in working with it. The compiler that translates the visual blocks language for implementation on Android uses the Kawa Language Framework and Kawa's dialect of the Scheme programming language, developed by Per Bothner and distributed as part of the GNU Operating System by the Free Software Foundation.

The authors would like to thank Google and the App Inventor team for their support of our work and teaching efforts at USF, Mills College, and MIT. Special thanks go to App Inventor Technical Lead Mark Friedman, Project Manager Karen Parker, and engineers Sharon Perl and Debby Wallach.

We also owe a special thanks to our O'Reilly editors, Courtney Nash and Brian Jepson, as well as Kathy Riutzel, Brian Kernighan, Debby Wallach, and Rafiki Cai for their feedback and insights.

Finally, we'd like to acknowledge the support of our respective spouses: Ellen's husband, Keith Golden; Hal's wife, Lynn Abelson; Liz's husband, Kevin Looney; and David's wife, Minerva Novoa. New mother Ellen is also grateful for the help of nanny Neil Fullagar.