CHAPTER 1

HelloPurr

This chapter gets you started building apps. It presents the key elements of App Inventor, the Component Designer and the Blocks Editor, and leads you through the basic steps of creating your first app, HelloPurr. When you're finished, you'll be ready to build apps on your own.

A typical first program with a new computer system prints the message "Hello World" to show that everything is connected correctly. This tradition goes back to the 1970s and Brian Kernighan's work on the C programming language at Bell Labs. With App Inventor, even the simplest apps do more than just show messages: they play sounds and react when you touch the device. So, we're going to get started right away with something more exciting: your first app (as shown in Figure 1-1) will be "HelloPurr," a picture of a cat that meows when you touch it and purrs when you shake the device on which it's being viewed.

Figure 1-1. The HelloPurr app

What You'll Learn

The chapter covers the following topics:

- Building apps by selecting components and specifying their behavior.
- Using the Component Designer to select components. Some components are visible on the device's screen and some aren't.
- Adding media (sounds and images) to apps by uploading them from your computer.
- Using the Blocks Editor to assemble blocks that define the components' behavior.
- Testing apps with App Inventor's *live testing*. This lets you see how apps will look and behave on the device, step by step, even as you're building them.
- Packaging the apps you build and downloading them to a device.

The App Inventor Environment

You can begin programming with App Inventor by opening a browser to *ai2.appinventor.mit.edu*. This opens the newest version of App Inventor, which was released in December, 2013. Some people call it App Inventor 2, but it is formally just named *App Inventor*, and the previous version is called *App Inventor Classic*. In this book, you'll be using the new version.

The App Inventor programming environment has three key parts:

- The *Component Designer* (Figure 1-2). You use it to select components for your app and specify their properties.
- The *Blocks Editor* (Figure 1-3). You use it to specify how the components will behave (e.g., what happens when a user clicks a button).
- An Android device with which you can actually run and test your app as you are developing it. If you don't have an Android device handy, you can test the apps you build by using the Android emulator that comes with the system.

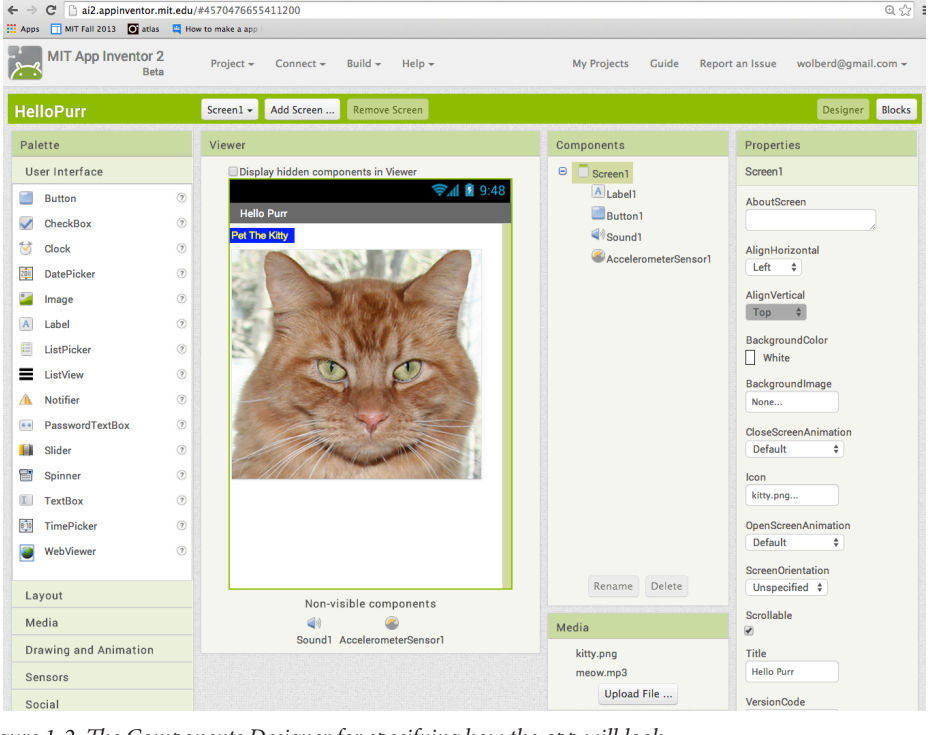

Figure 1-2. The Components Designer for specifying how the app will look

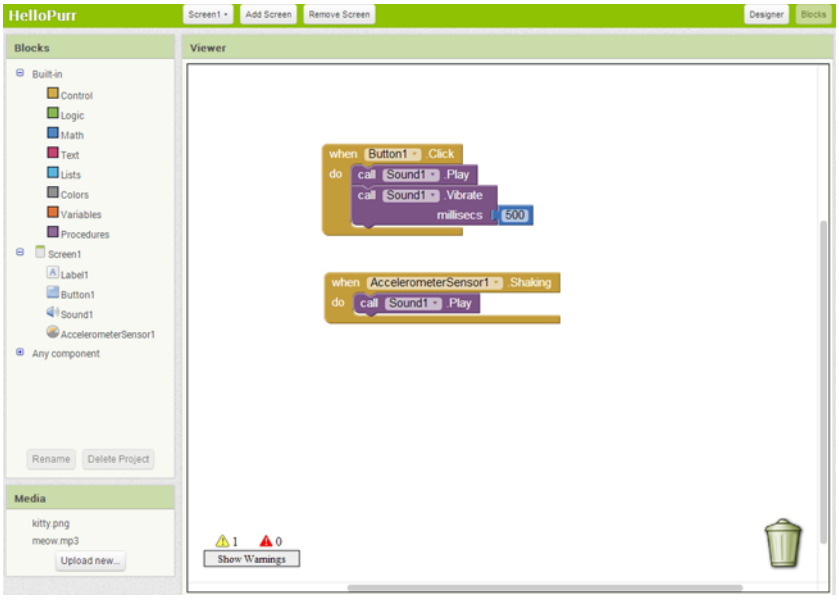

Figure 1-3. The Blocks Editor for specifying how the app will behave

The first time you browse to *ai2.appinventor.mit.edu,* you'll see the Projects page, which will be mostly blank because you haven't created any projects yet. To create a project, at the upper left of the page, click "New Project," enter the project name "HelloPurr" (one word with no spaces), and then click OK.

The first window that opens is the Component Designer. The Blocks Editor is available by clicking on the "Blocks" button in the upper-right corner of the window.

App Inventor is a *cloud computing* tool, meaning that your app is stored on an online server as you work. So if you close App Inventor, your app will be there when you return; you don't have to save anything on your computer as you would with, for example, a Microsoft Word file.

Designing the Components

The first tool you'll use is the Component Designer (or just Designer). Components are the elements you combine to create apps, like ingredients in a recipe. Some components are very simple, like a `Label` component, which shows text on the screen, or a `Button` component, which you tap to initiate an action. Other components are more elaborate: a drawing `Canvas` that can hold still images or animations; an *accelerometer*, which is a motion sensor that detects when you move or shake the device; or components that make or send text messages, play music, and video, get information from websites, and so on.

When you open the Designer, it will appear as shown in Figure 1-4.

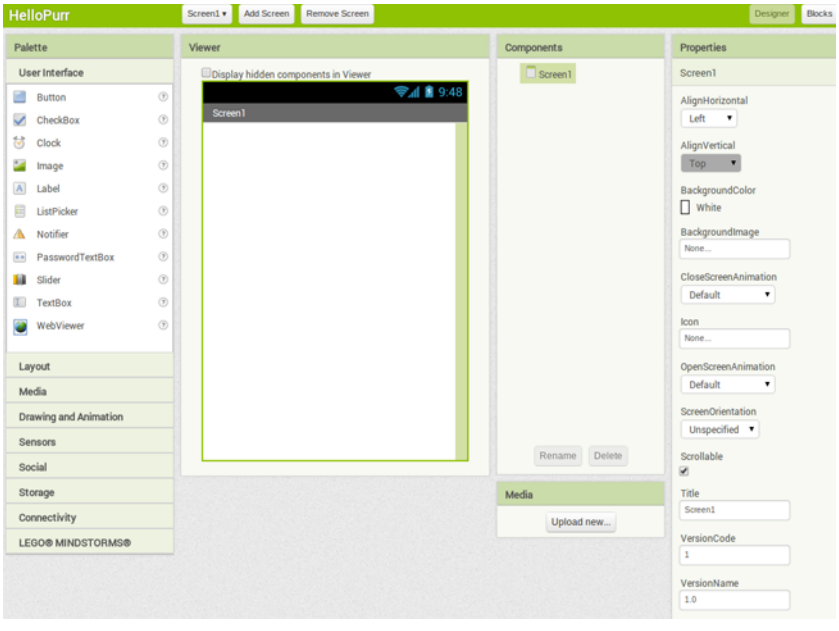

Figure 1-4. The App Inventor Component Designer

The Designer is divided into several areas:

- Toward the center is a white area called the *Viewer*. This is where you place components and arrange them to map out what you want your app to look like. The Viewer shows only a rough indication of how the app will look, so for example, a line of text might break at a different place on your device than on the Viewer. To see how your app will *really* appear, you'll need to test it on your device or the emulator (we'll show you how to do this shortly).
- To the left of the Viewer is the *Palette*, which is a list of components from which you can select. The Palette is divided into sections; at this point, only the User Interface components are visible, but you can see components in other sections of the Palette by clicking the headers labeled Layout, Media, and so on.
- To the right of the Viewer is the *Components* list, which lists the components in your project. Any component that you drag into the Viewer will also show up in this list. Currently, the project has only one component listed: `Screen1`, which represents the screen of the device itself.
- Under the Components list is an area that shows the *Media* (pictures and sound) in the project. This project doesn't have any media yet, but you'll be adding some soon.
- To the far right is a section that shows the Properties of components; when you click a component in the Viewer, you'll see its Properties listed here. Properties are details about each component that you can change. (For example, when clicking

on a `Label` component, you might see properties related to color, text, font, and so on.) Right now, it shows the properties of the screen (called `Screen1`), which include a background color, a background image, and a title.

For the HelloPurr app, you'll need two *visible* components (think of these as components that you can actually see in the app): the `Label` component reading "Pet the Kitty" and a `Button` component with an image of a cat in it. You'll also need a *non-visible* `Sound` component that knows how to play sounds, such as "meow," and an `Accelerometer` component for detecting when the device is being shaken. Don't worry—we'll walk you through each component, step by step.

Making a Label

The first component to add is a `Label`:

1. Go to the Palette, open the User Interface drawer if it is not open, click `Label` (which appears about six spots down in the list of components), and drag it to the Viewer. You'll see a rectangular shape appear on the Viewer, containing the words "Text for Label1."
2. Look at the Properties box on the right side of the Designer. It shows the properties of the label. About halfway down, there's a property called Text, with a box for the label's text. Change the text to "Pet the Kitty" and press Return. You'll see the text change in the Viewer.
3. Change the `BackgroundColor` of the label by clicking the box, which currently reads None, to select a color from the list that appears. Select Blue. Also change the `TextColor` of the label to Yellow. Finally, change the `FontSize` to 20.

The Designer should now appear as shown in Figure 1-5.

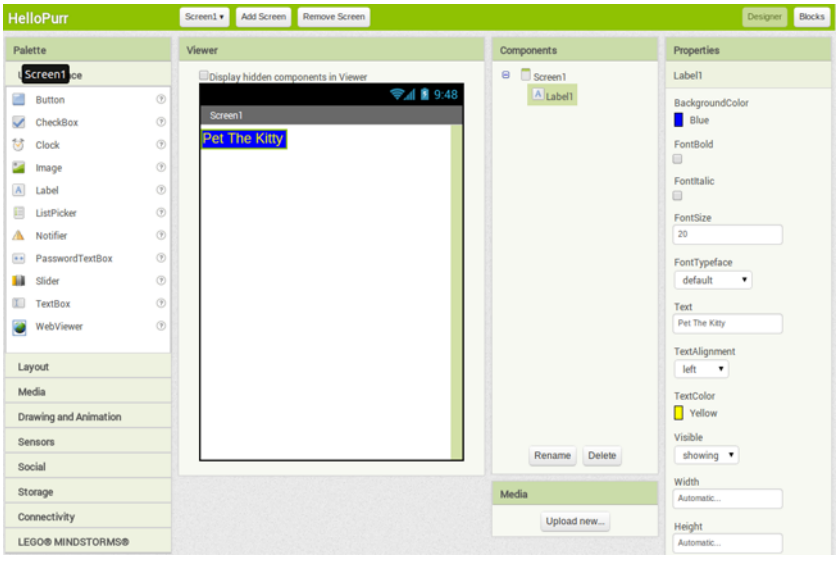

Figure 1-5. The app now has a label

Adding the Button

The kitty for HelloPurr is implemented as a `Button` component—you create a normal button and then change the button image to the kitty. To make the basic button first, go to the Palette in the Designer and click `Button` (at the top of the list of components). Drag it onto the Viewer, placing it below the label. You'll see a rectangular button appear on the Viewer.

Now you've got a button that you'll use to trigger the sound effect when someone taps it, but we really want it to look like the picture of the kitty, not a plain, old rectangle. To make the button look like the kitty:

1. First, you need to download a picture of the kitty and save it on your computer desktop. You can download it at *http://appinventor.org/bookFiles/HelloPurr/kitty.png*. The extension *.png* is a standard image format similar to *.jpg* and *.gif*; all of these file types will work in App Inventor, as will most standard sound files such as *.mpg* or *.mp3*. You can also download the sound file you'll need to make the kitty meow at *http://appinventor.org/bookFiles/HelloPurr/meow.mp3*.
2. The Properties box should display the properties of the button. If it doesn't, click the button in the Viewer to reveal the button's properties on the right. In the Properties box, click the area under Image (which currently reads "None...").
3. Click "Upload file." Then, click "Choose File" and browse to select the *kitty.png* file you downloaded to your computer earlier, and then click OK.
4. After the image uploads, *kitty.png* should be listed as an option for the image property of the button. Click OK to choose it. You'll also see the file listed in the

Media area of the Designer window, just below the Components list. And if you look at the button in the designer, you'll see the kitty picture displayed—the button now looks like a kitty.

5. You might have also noticed that the kitty picture still has the words "Text for Button 1" displayed on it. You probably don't want that in your app, so go ahead and blank out the `Text` property of `Button1`.

Now, the Designer should appear as shown in Figure 1-6.

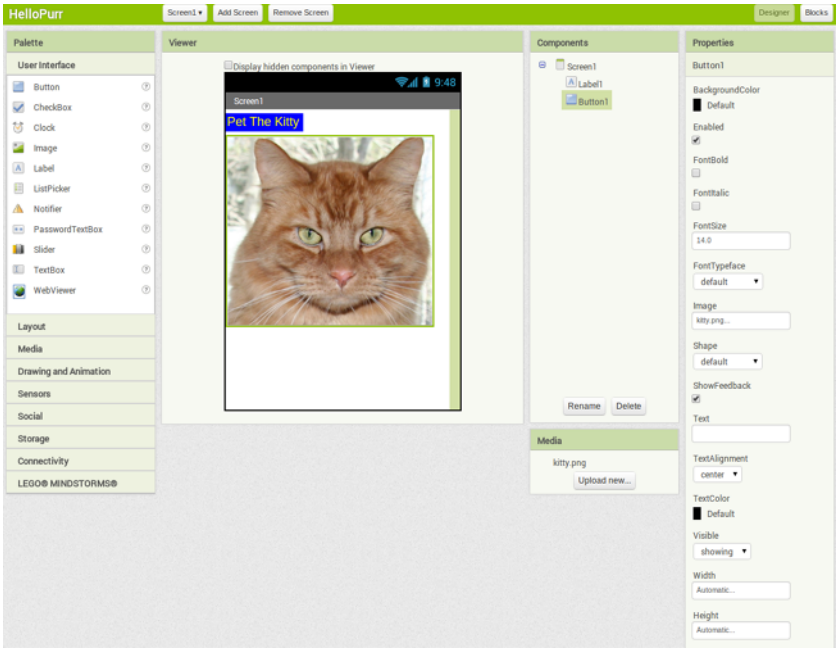

Figure 1-6. The app with a label and a button with an image on it

Adding the Meow Sound

In your app, you want the kitty to meow when you tap the button. For this, you'll need to add the meow sound and program the button behavior to play that sound when the button is clicked:

1. If you haven't downloaded the *meow.mp3* file to your computer's desktop, do so now by using this link: *http://appinventor.org/bookFiles/HelloPurr/meow.mp3*.
2. Go to the Palette at the left of the Designer window and click the header marked Media to expand the Media section. Drag out a `Sound` component and place it in the Viewer. No matter where you drop it, it will appear in the area at the bottom of the Viewer marked "Non-visible components." Non-visible components are objects that do things for the app but don't appear in the visual user interface.

3. Click `Sound1` to show its properties. Click the `Source` property and then go through the steps to upload and choose the *meow.mp3* file you downloaded earlier. When you're done, you should see both *kitty.png* and *meow.mp3* listed in the Media section of the Designer.

Table 1-1 lists the components that you've gathered for your app so far.

Table 1-1. The components you've added to the HelloPurr app

Component type	Palette group	Name of component	Purpose
`Button`	User Interface	`Button1`	Press to make the kitty meow.
`Label`	User Interface	`Label1`	Shows the text "Pet the Kitty."
`Sound`	Media	`Sound1`	Play the meow sound.

Live Testing

With App Inventor, you can view and test your app on an Android device as you create it. Testing your app in an incremental manner is a practice used by effective software developers and will save you hours of work.

If you have an Android device and an internet connection with WiFi, you can set up live testing in minutes, and you don't have to download any software to your computer (just an app on your phone). If you don't have an Android device, you'll need to perform some additional setup in order to use the emulator, the details of which are covered at *http://appinventor.mit.edu/explore/ai2/setup.html*.

If you have an Android device, do the following:

1. On your device, download and install the "MIT AI2 Companion" app from the Google Play Store. Launch the app when it's installed.
2. Connect both your computer and your device to the same WiFi connection.
3. In App Inventor (in the browser), from the top menu, select Connect and then choose AI Companion, as shown in Figure 1-7.

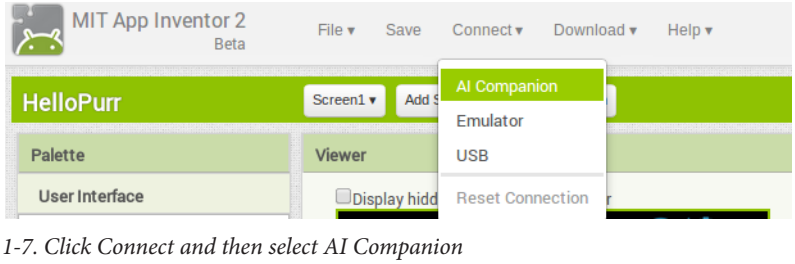

Figure 1-7. Click Connect and then select AI Companion

4. On your device, launch the app you installed, the MIT AI2 Companion, as shown in Figure 1-8. Select "Scan QR Code" and then hold your device up to the QR code on the computer screen to scan it.

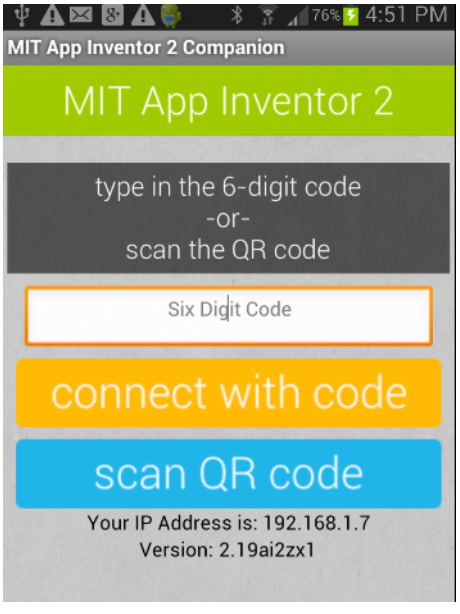

Figure 1-8. On your device, open the Companion app and click "Scan QR Code"

If all goes well, you should see the HelloPurr app running on your device, including all of the components you added. As you make changes in the App Inventor Designer or Blocks Editor, those changes will also appear on the device, as well.

Live testing setup If you have trouble setting up live testing, visit *http://appinventor.mit.edu/explore/ai2/setup.html*.

If your app does appear on the device, go ahead and tap the button. Do you think anything will happen? It won't, because you haven't instructed the button to do anything yet. This is the first important point to understand about App Inventor: for every component you add in the Designer, you have to move over to the Blocks Editor and create the code to make that component do whatever it is that you want it to do.

Adding Behaviors to the Components

You've just added `Button`, `Label`, and `Sound` components as the building blocks for your first app. Now, let's make the kitty meow when you tap the button. You do this with the Blocks Editor. In the top right of the Component Designer, click "Blocks."

Look at the Blocks Editor window. This is where you instruct the components what to do and when to do it. You're going to direct the kitty button to play a sound when the user taps it. If components are ingredients in a recipe, you can think of blocks as the cooking instructions.

Making the Kitty Meow

At the top left of the window, beneath the Blocks header, you'll see a column that includes a Built-in drawer and a drawer for each component you created in the Designer: `Button1`, `Label1`, `Screen1`, and `Sound1`. When you click a drawer, you get a bunch of options (*blocks*) for that component. Click the drawer for `Button1`. The drawer opens, showing a selection of blocks that you can use to build the button's behavior, starting with `Button1.Click` at the top, as shown in Figure 1-9.

Figure 1-9. Clicking Button1 shows the component's blocks

Click the block labeled `Button1.Click` and drag it into the workspace. You'll notice that the word "when" is included on the `Button1.Click` block. Blocks including the word "when" are called *event handlers*; they specify what components should do *when* some particular event happens. In this case, the event we're interested in happens when the app user taps the image of the kitty (which is really a button), as

shown in Figure 1-10. Next, you'll add some blocks to program what will happen in response to that event.

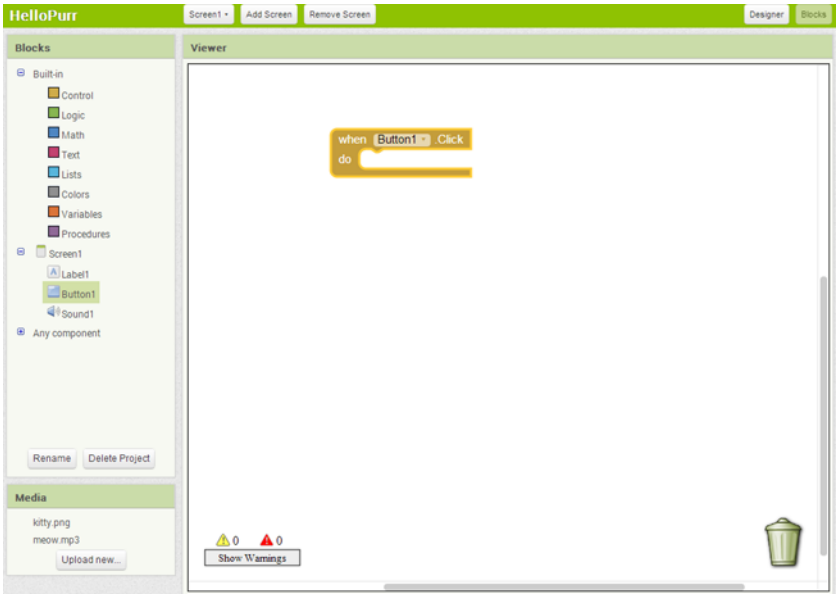

Figure 1-10. You'll specify a response to the user clicking within the Button.Click block

Click `Sound1` to open the drawer for the sound component, and then drag out the `call Sound1.Play` block. (Remember, earlier we set the property for `Sound1` to the meow sound file you downloaded to your computer.) At this point, you might have noticed that the `call Sound1.Play` block is shaped so that it can fit into a gap marked "do" in the `Button1.Click` block. App Inventor is set up so that only certain blocks fit together; this way, you always know you're connecting blocks that actually work together. In this case, blocks with the word "call" cause components to do things. The two blocks should snap together to form a unit, as shown in Figure 1-11, and you'll hear a snapping sound when they connect.

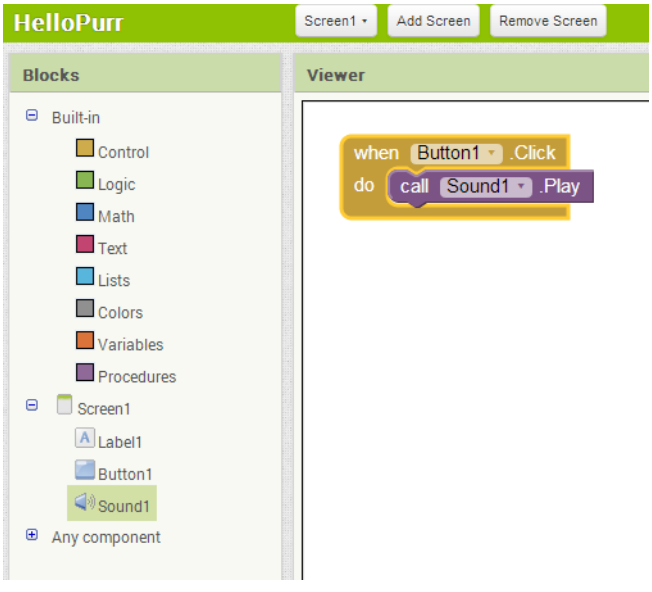

Figure 1-11. Now when someone clicks the button, the meow sound will play

Unlike traditional programming code (which often looks like a jumbled mess of gobbledygook "words"), the event-response blocks in App Inventor spell out the behaviors you're trying to create in a plain, understandable fashion. In this case, we're essentially saying, "Hey, App Inventor, when someone taps the kitty button, play the meow sound."

Test your app *Check to make sure everything is working properly—it's important to test your app each time you add something new. Tap the button on the device (or click it if you are using the emulator). You should hear the kitty meow. Congratulations, your first app is running!*

Adding a Purr

Now we're going to make the kitty purr *and* meow when you tap the button. We'll simulate the purr by making the device vibrate. That might sound hard, but in fact, it's easy to do because the `Sound` component we used to play the meow sound can make the device vibrate, as well. App Inventor helps you tap into this kind of core device functionality without having to deal with *how* the device actually vibrates. You don't need to do anything different in the Designer; you can just add a second function call block to the button click in the Blocks Editor:

1. Go to the Blocks Editor and click `Sound1` to open the drawer.

2. Select `call Sound1.Vibrate` and drag it under the `call Sound1.Play` block in the `Button1.Click` slot. The block should click into place, as shown in Figure 1-12. If it doesn't, try dragging it so that the little notch on the top edge of `call Sound1.Vibrate` touches the little bump on the bottom of `call Sound1.Play`.

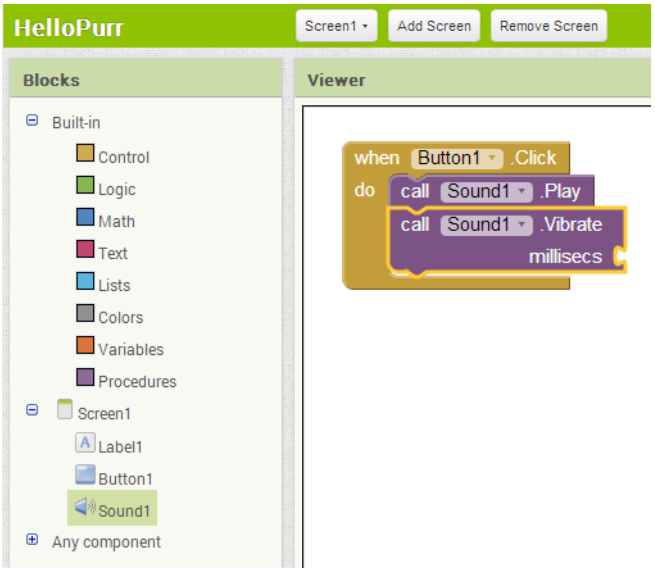

Figure 1-12. Playing the sound and vibrating on the Click event

3. You might have noticed that the `call Sound1.Vibrate` block includes the text "millisecs" at the lower right, and alongside it is an open socket protruding inward from the block's edge. An open socket in a block means that you need to plug something into it to specify more about how the behavior should work. In this case, you must tell the `Vibrate` block how long it should vibrate. You need to specify this time in thousandths of a second (milliseconds), which is pretty common for many programming languages. So, to make the device vibrate for half a second, you need to enter a value of 500 milliseconds. To do that, you need to grab a number block. Click the Math drawer and you'll see a list of blue blocks appear, as shown in Figure 1-13.

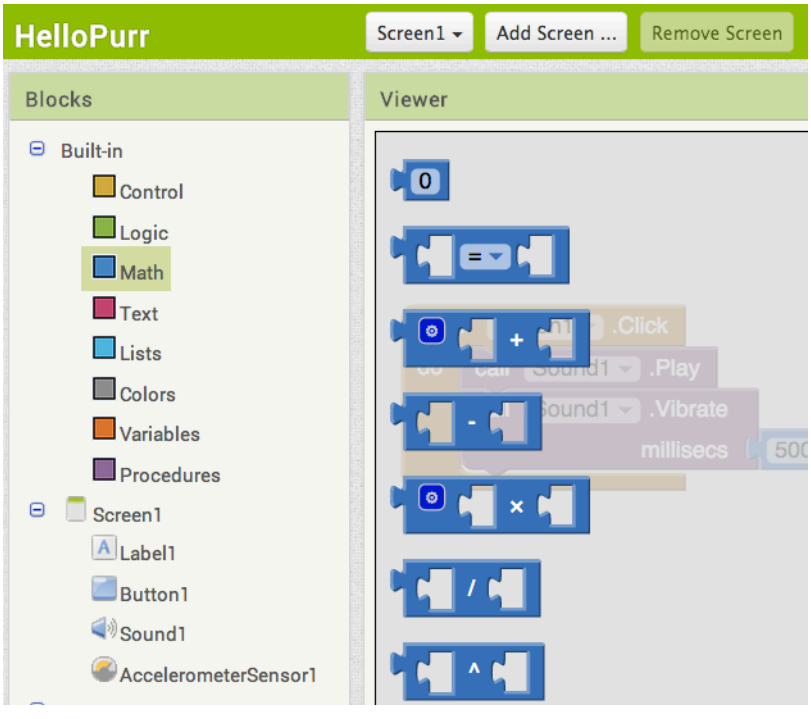

Figure 1-13. Opening the Math drawer

4. At the top of the list, you should see a block with a "0" in it. You can drag this block out and then change the 0 to any number you want. Go ahead and drag out the number block, as shown in Figure 1-14.

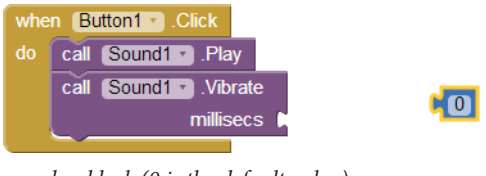

Figure 1-14. Choosing a number block (0 is the default value)

5. Click the 0 and type the new value, 500, as shown in Figure 1-15.

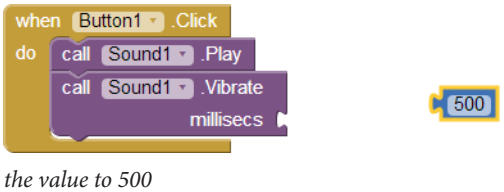

Figure 1-15. Changing the value to 500

6. Plug the 500 number block into the socket on the right side of `call Sound1.Vibrate`, as shown in Figure 1-16.

Figure 1-16. Plugging the value 500 into the millisecs socket

Test your app *Try it! Tap the button on the device, and you'll feel the purr for half a second.*

Shaking the Device

Now, let's add a final element that taps into another cool feature of Android: making the kitty meow when you shake the device. To do this, you'll use a component called `AccelerometerSensor` that can sense when you shake or move the device around.

1. In the Designer, in the Palette components list, expand the Sensors area and drag out an `AccelerometerSensor`. Don't worry about where you drag it. As with any non-visible component, no matter where you place it in the Viewer, it will move to the "Non-visible components" section at the bottom of the Viewer.
2. You'll want to treat someone shaking the device as a different, separate event from the button click. This means that you need a new event handler. Go to the Blocks Editor. There should be a new drawer for `AccelerometerSensor1`. Open it and drag out the `AccelerometerSensor1.Shaking` block. It should be the second block in the list.
3. Just as you did with the sound and the button click, drag out a `call Sound1.Play` block and fit it into the gap in `AccelerometerSensor1.Shaking`. Try it out by shaking the device.

Figure 1-17 shows the blocks for the completed HelloPurr app.

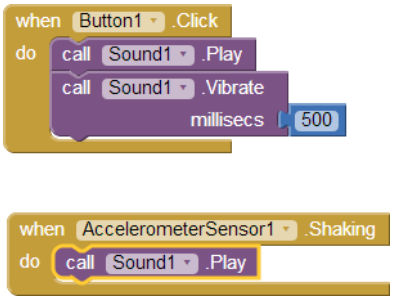

Figure 1-17. The blocks for HelloPurr

Downloading the App to Your Android Device

App Inventor's live testing feature allows you to easily test the app while connected to your device. The only problem is that if you disconnect your device from App Inventor, the app running on the device will stop, and you won't find the app anywhere on the device because it was never truly installed; it was just running within the App Inventor Companion app.

You can download and install the completed app so that it works on any device, even when it's not connected to the computer. To get ready for this, first set the app's icon so that when you install it on a device, it will appear with a distinguishing picture in the list of apps. You can do this in the Designer by selecting the `Screen` component, clicking its `Icon` property, and then uploading an image file as the icon (e.g., the picture of the kitty).

Next, ensure that your device allows apps to be downloaded from places other than the Android Market. For most Android devices, you do this by going to Settings→Applications, and then checking the box next to "Unknown sources."

Then, back in App Inventor, in the Designer, click Build, and select "App (provide QR code for .apk)." You should see a "Progress Bar" message in the window, a process that takes up to a minute. When the QR Code for the finished app is displayed, scan it onto your device with a Barcode Scanner app.[1] After scanning the QR Code, the device might prompt you to enter your password for your Google account. When you finish entering your password, your app will begin downloading to your device and you'll see a download icon in your device's notifications. Go to your notifications, wait until the download completes, and then choose the app to install it.

After you've installed it, look at the apps available on your device, and you'll now see HelloPurr, the app we just built. You run it just like any other app. (Make sure that you run your new app, not the App Inventor Companion application.) You can now stop

1 There are many QR Code scanners for Android. If you don't have one on your device, go to the Play Store and install one.

the Companion app or unplug your device from the computer, and your new packaged application will still be there.

It's important to understand that this means your packaged app is now separate from the project on App Inventor. You can do more work on the project in App Inventor by connecting the device with the AI Companion as before. But that won't change the packaged app that is now installed on your device. If you make further changes to your app in App Inventor, you'll want to package the result and download the new version to replace the old one on the device.

Sharing the App

You can share your app in a couple of ways. To share the executable app (the .apk file), first click Build and choose "Application (save to my computer)." This will create a file with a *.apk* extension on your computer. You can share this file with others by sending it to them as an email attachment, which they'll then open with their email app on their device. Or you can upload the .apk file somewhere on the Web (e.g., on Dropbox). Just be sure to let the people installing your app know that they need to allow "unknown sources" in their device's Application settings in order to install apps that aren't from the Android Market.

You can also create a QR code for the app so that people can scan it onto their device from the Web or even a physical poster. There are numerous tools that will generate a QR code from a URL (e.g., check out *qrcode.kaywa.com*). You can then cut and paste the QR code into a web page or document for printing and posting.

You can also share the *source code* (blocks) of your app with another App Inventor developer. To do this, click My Projects, check the app that you want to share (in this case, HelloPurr), and then select Project→Export Selected Project. The file created on your computer will have a *.aia* extension. You can send this file by email to someone, and they can open App Inventor, choose Project→Import project, and then select the *.aia* file. This will give the user a complete copy of your app, which can then be edited and customized without affecting your version.

App Inventor will soon have its own App gallery where you can share your apps and remix the apps from developers all over the world.

Variations

After you build the apps in this book, you'll likely think of ways to improve them. At the end of each chapter, we'll also suggest customization ideas for you to try. Customizing the apps will lead you to explore the available components and blocks, and learn to program on your own without the detailed instructions provided in the tutorials.

Here are a couple of things to try for HelloPurr:

- As you shake the device, the meows will sound strange, as if they are echoing. That's because the accelerometer sensor is triggering the shaking event many times a second in response to each individual up and down movement, so the meows are overlapping. If you look at the Sound component in the Designer, you'll see a property called `Minimum Interval`. This property determines how close together successive sounds can start. It's currently set at a little under half a second (400 milliseconds), which is less than the duration of a single meow. By adjusting the minimum interval, you can change how much the meows overlap.
- If you run the packaged app and walk around with the device in your pocket, your device will meow every time you move suddenly, something you might find embarrassing. Android apps are typically designed to keep running even when you're not looking at them; your app continues to communicate with the accelerometer and the meow just keeps going. To really quit the app, bring up HelloPurr and press the device's menu button. You'll be offered an option to stop the application. Select this to close the app completely.

Summary

Here are some of the concepts we covered in this chapter:

- You build apps by selecting components in the Designer, and then in the Blocks Editor, you tell the components what to do and when to do it.
- Some components are visible and some aren't. The visible ones appear in the user interface of the app. The non-visible ones do things such as play sounds.
- You define components' behavior by assembling blocks in the Blocks Editor. You first drag out an event handler, such as `Button1.Click`, and then place command blocks like `Sound.Play` within it. Any blocks within `Button1.Click` will be performed when the user taps the button.
- Some commands need extra information to make them work. An example is `Vibrate`, which needs to know how many milliseconds to vibrate for. These values are called *arguments* or *parameters*.
- Numbers are represented as number blocks. You can plug these into commands that take numbers as arguments.
- App Inventor has sensor components. The `AccelerometerSensor` can detect when the device is moved or shaken.
- You can package the apps you build and download them to the phone, where they run independently of App Inventor.