CHAPTER 1

Hello Purr

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 (Brian is now a visiting scholar at Google working on the App Inventor team!). With App Inventor, even the simplest apps do more than just show messages: they play sounds and react when you touch the phone. 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 it.

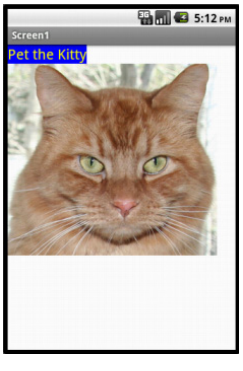

Figure 1-1. The HelloPurr app

What You'll Learn

The chapter covers the following topics:

- Building apps by selecting components and then telling them what to do and when to do it.
- Using the Component Designer to select components. Some components are visible on the phone screen and some aren't.
- Adding media (sounds and images) to apps by uploading them from your computer.
- Working in 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 phone step by step, even as you're building them.
- Packaging the apps you build and downloading them to a phone.

The App Inventor Environment

You can set up App Inventor using the instructions at *http://appinventor.googlelabs.com/learn/setup/*. App Inventor runs primarily through the browser, but you need to download some software to your computer's desktop and change some settings on your phone. Typically you can get set up in just a few minutes, though sometimes there are issues with setting up device drivers for particular Android phones. If you have any phone issues, we suggest you get started using the Android emulator that comes packaged with the App Inventor download.

The App Inventor programming environment has three key parts, all shown in Figure 1-2:

- The *Component Designer*, shown on the left side of Figure 1-2, runs in your browser window. You use it to select components for your app and specify their properties.
- The *Blocks Editor* runs in a window separate from the Component Designer—it is often easiest to arrange this to the right of the Component Designer on your screen while you are working on your app. You use the Blocks Editor to create behaviors for the components.
- A phone allows you to actually run and test your app as you are developing it. If you don't have an Android phone handy, you can test the apps you build using the Android emulator (shown in the bottom right of Figure 1-2) that comes integrated with the system.

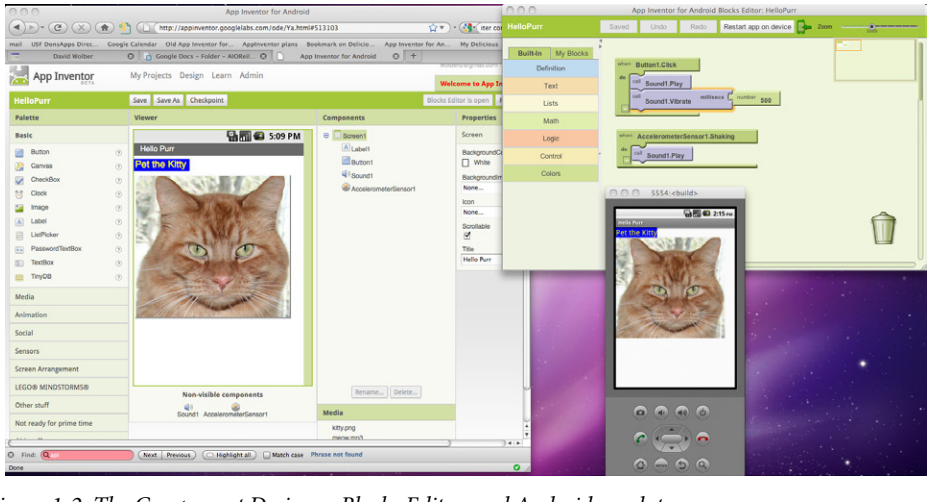

Figure 1-2. The Component Designer, Blocks Editor, and Android emulator

You start App Inventor by browsing to *http://appinventor.googlelabs.com*. If this is the first time you've used App Inventor, you'll see the Projects page, which will be mostly blank because you haven't created any projects yet. To create a project, click New at the top left of the page, enter the project name "HelloPurr" (one word with no spaces), and click OK.

The first window that opens is the Component Designer. When it appears, click Open Blocks Editor in the menu at the top right. The Blocks Editor comes up in a separate window, aided by a tool called Java Web Start. (You don't have to worry about all the Java messages—App Inventor is using Java, which should already be installed on your computer, to help launch the Blocks Editor.) This process usually takes about 30 seconds.

If everything is OK, the Blocks Editor will appear and you'll see two buttons near the top right of the screen, as shown in Figure 1-3.

Figure 1-3. Plug a phone into your computer or click "New emulator"; then, click "Connect to Device"

If you have an Android phone and a USB cable, plug the phone into the computer and select "Connect to Device." If instead you want to test the apps you build using an emulator, click "New emulator" and wait about 30 seconds while the Android emulator loads. When it is fully operational, click "Connect to Device" so that App Inventor will run your app in the emulator.

If all is well, you should see a window for the Component Designer, a window for the Blocks Editor, and the emulator window if you chose that option (your screen should look something like Figure 1-2, shown previously, but with the windows mostly empty). If you're having problems here, review the setup instructions at *http://appinventor.googlelabs.com/learn/setup/*.

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, a motion sensor that works like a Wii controller and detects when you move or shake the phone; 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.

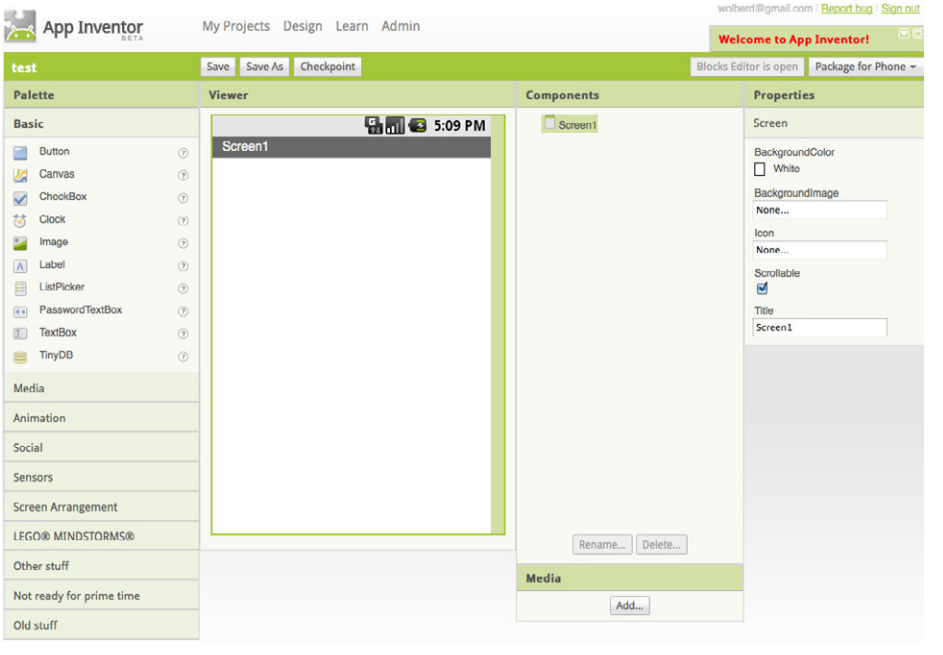

Figure 1-4. The App Inventor Component Designer

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 in your app than what you see in the Viewer. To see how your app will *really* appear, you'll need to either download the app to your phone (we'll go through how to do this a bit later, in the section "Packaging the App for Downloading") or view it in the emulator that comes with App Inventor.
- To the left of the Viewer is the *Palette*, which is a list of components you can select from. The Palette is divided into sections; at this point, only the Basic components are visible, but you can see components in other sections of the Palette by clicking the headers labeled Media, Animation, 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 show up in this list. Currently, the project has only one component listed: `Screen1`, which represents the phone screen 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.

At 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 (you can think of these as components 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 phone 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, click `Label` (which appears about five spots down in the list of components), and drag it to the Viewer. You'll see a rectangular shape appear on the Viewer, with 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. There's a property called `Text` about halfway down, 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.

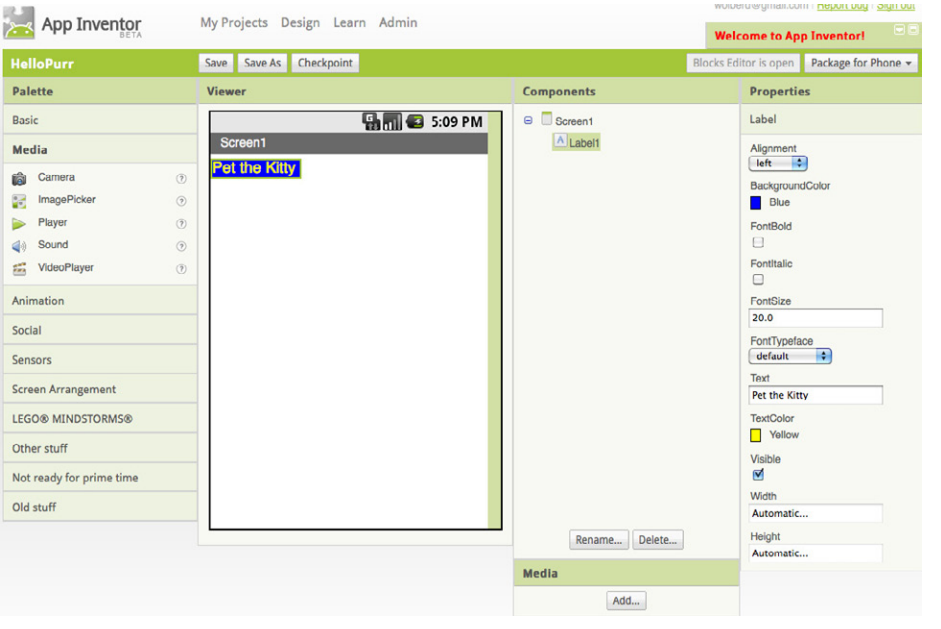

Figure 1-5. The app now has a label

Be sure you have your phone connected and the Blocks Editor open. You should see the label appear on the phone as you add it in the Designer. In App Inventor, you build the application on the phone as you pick the components in the Designer. That way, you can see right away how your application will look. This is called *live testing*, and it also applies to the behaviors you create for the components in the Blocks Editor, as you'll see shortly.

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. After about 10 seconds, the button should appear on the phone. Go ahead and tap the phone button—do you think anything will happen? It won't, because your app hasn't told 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 something happen with that component (we'll do that after we finish adding the components we need in the Designer).

Now we've got a button that we'll use to trigger the sound effect when someone clicks 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 from the site for this book at *http://examples.oreilly.com/0636920016632/*. The picture is the file called *kitty.png*. (*.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 like *.mpg* or *.mp3*.) You can also download the sound file we need, *meow.mp3*.
2. The Properties box should show the properties of the button. If it doesn't, click the image of the button in the Viewer to expose the button's properties on the right. In the Properties box, click the area under Image (which currently reads None). A box appears with a button marked Add.
3. Click Add and you'll see "Upload file." Click Choose File, browse to select the *kitty.png* file you downloaded to your computer earlier, and click OK.
4. You'll see a yellow message at the top of the screen: "Uploading kitty.png to the AppInventor server." After about 30 seconds, the message and the upload box will disappear, and *kitty.png* should be listed as the image property for the button. You'll also see this listed in the Media area of the Designer window, just below the Components list. And if you look at the phone, you'll see the kitty picture displayed—the button now looks like a kitty.
5. You may have also noticed that the kitty picture on your phone has the words "Text for button 1" displayed on it. You probably don't want that in your app, so go ahead and change the `Text` property of `Button1` to something like "Pet the Kitty," or just delete the text altogether.

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

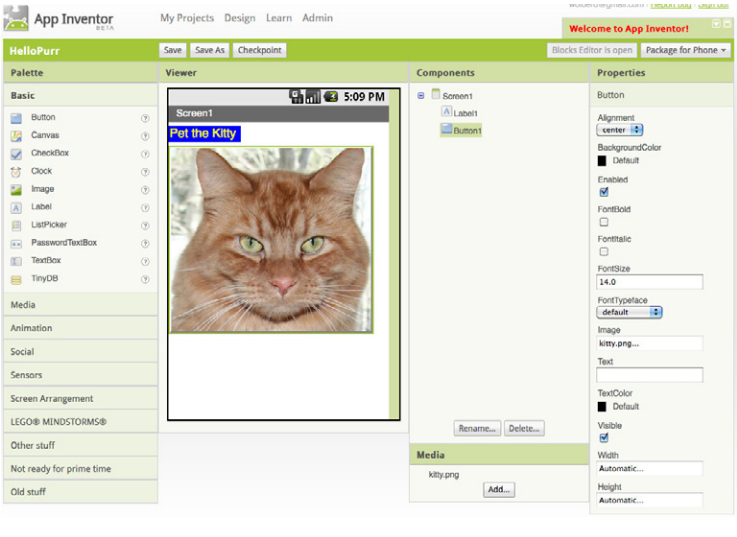

Figure 1-6. The app with a label and a button with an image on it

Adding the Meow Sound

In your app, the kitty will 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 at *http://examples.oreilly.com/0636920016632/*.
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. Wherever 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 of the app.
3. Click `Sound1` to show its properties. Set its `Source` to *meow.mp3*. You'll need to follow the same steps to upload this file from your computer as you did for the kitty picture. When you're done, you should see both *kitty.png* and *meow.mp3* listed in the Media section of the Designer.

You should now have the components depicted in Table 1-1.

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

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

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. If your Blocks Editor isn't yet open, click "Open the Blocks Editor" in the top right of the Component Designer.

Look at the Blocks Editor window. This is where you tell the components what to do and when to do it. You're going to tell 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, you'll see buttons labeled "Built-In" and "My Blocks." Click My Blocks, and you'll see a column that includes 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 you created. (Don't worry about the Built-In column for now—we'll get to that in Chapter 2.) Click the drawer for `Button1`. The drawer opens, showing a selection of blocks that you can use to tell the button what to do, starting with **Button1.Click** at the top, as shown in Figure 1-7.

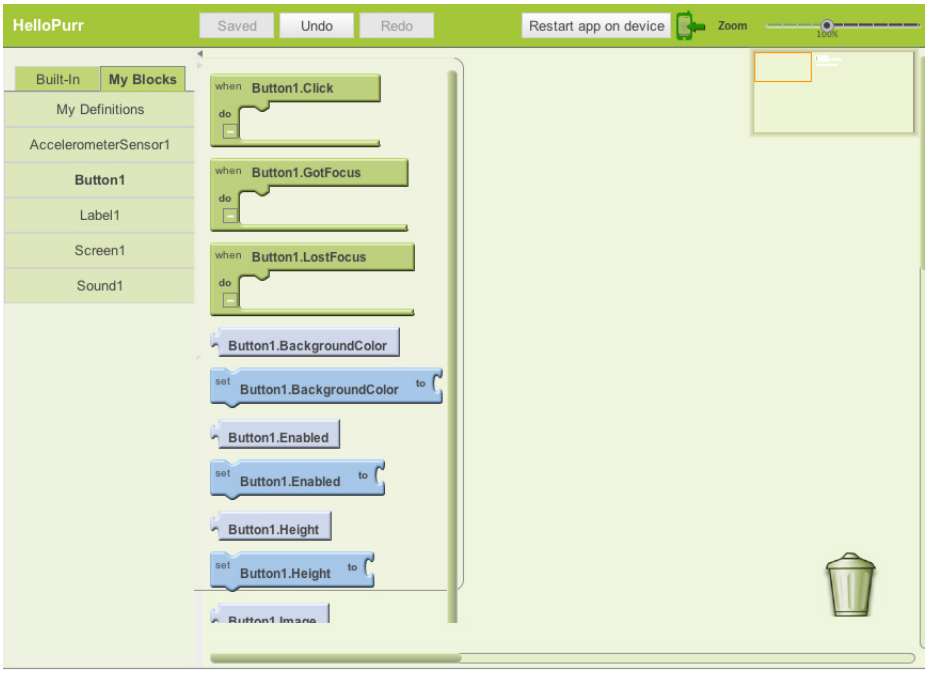

Figure 1-7. Clicking Button1 shows the component's blocks

Click the block labeled **Button1.Click** and drag it into the workspace. When you're looking for the block, you'll notice that the word "when" is smaller than **Button1.Click**. 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 clicks on the kitty (which is really a button), as shown in Figure 1-8. Next, we'll add some blocks to program what will happen in response to that event.

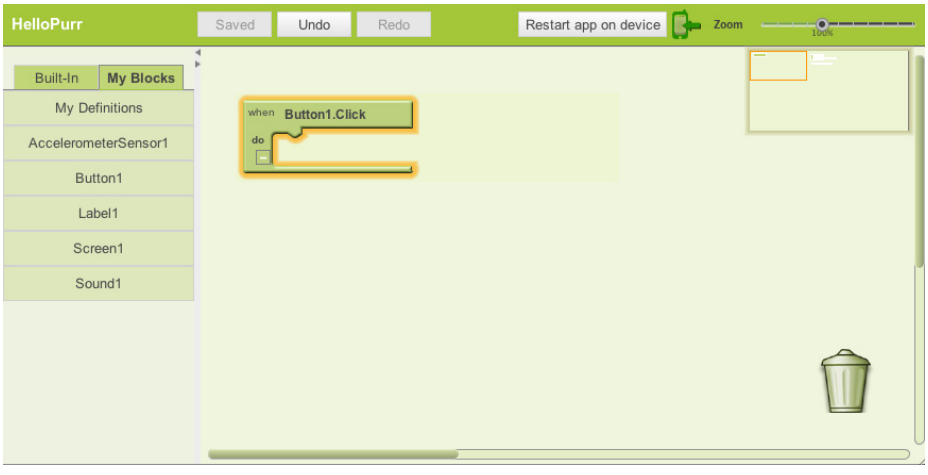

Figure 1-8. You'll specify a response to the user clicking within the Button.Click block

Click Sound1 in My Blocks to open the drawer for the sound component, and 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.) You may notice at this point that the **call Sound1.Play** block is shaped so 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" make components do things. The two blocks should snap together to form a unit, as shown in Figure 1-9, and you'll hear a snapping sound when they connect.

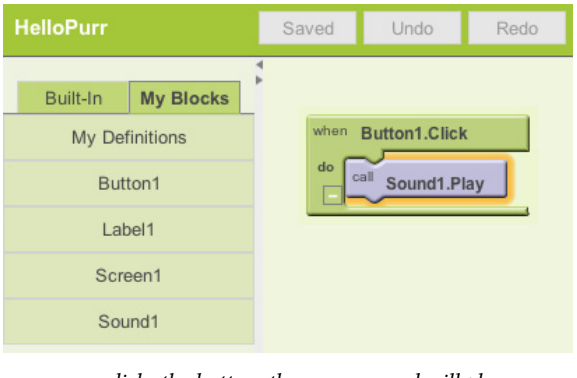

Figure 1-9. Now when someone clicks the button, the meow sound will play

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

Test your app. *Let's 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 phone (or click it 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 phone vibrate. That may sound hard, but in fact, it's easy to do because the Sound component we used to play the meow sound can make the phone vibrate as well. App Inventor helps you tap into this kind of core phone functionality without having to deal with *how* the phone actually vibrates. You don't need to do anything different in the Designer; you can just add a second behavior to the button click in the Blocks Editor:

1. Go to the Blocks Editor and click Sound1 in My Blocks 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 1-10. If it doesn't, try dragging it so that the little dip on the top of **call Sound1.Vibrate** touches the little bump on the bottom of **call Sound1.Play**.

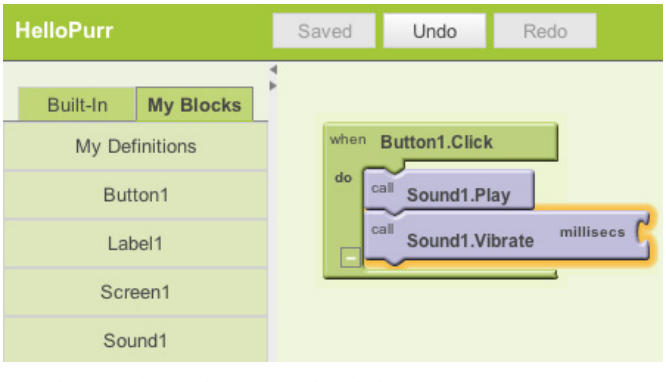

Figure 1-10. Playing the sound and vibrating on the Click event

3. You've likely noticed that the **call Sound1.Vibrate** block includes the text "millisecs" at the top right. An open slot in a block means you can 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 input this time in thousandths of a second (milliseconds), which is pretty common for many programming languages. So, to make the phone vibrate for half a second, put in

a value of 500 milliseconds. To put in a value of 500, you need to grab a number block. Click in an empty spot on the Designer screen, and then click the green Math button in the menu that pops up, as shown in Figure 1-11. You should see a drop-down list, with 123 as the first item; 123 indicates a block that represents a number.

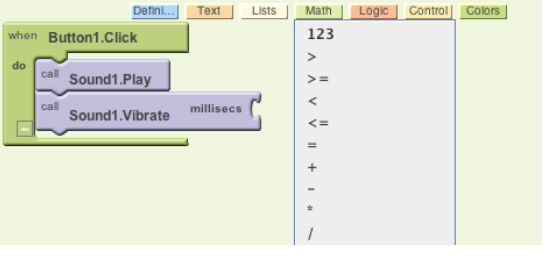

Figure 1-11. Opening the Math drawer

4. Click the 123 at the top of the list and you'll see a green block with the number 123, as shown in Figure 1-12.

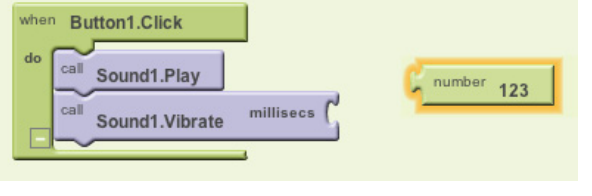

Figure 1-12. Choosing a number block (123 is the default value)

5. Change the 123 to 500 by clicking it and typing a new value, as shown in Figure 1-13.

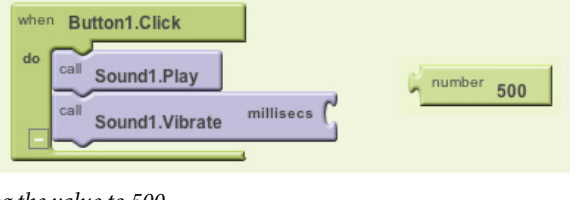

Figure 1-13. Changing the value to 500

6. Plug the 500 number block into the socket at the right of **call Sound1.Vibrate**, as shown in Figure 1-14.

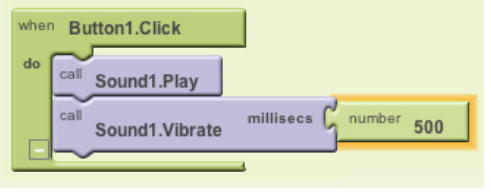

Figure 1-14. Plugging the 500 into the millisecs slot

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

Shaking the Phone

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

1. In the Designer, expand the Sensors area in the Palette components list 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 phone as a different, separate event from the button click. That means you need a new event handler. Go to the Blocks Editor. There should be a new drawer for `AccelerometerSensor1` under My Blocks. 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 phone.

Figure 1-15 shows the blocks for the completed HelloPurr app.

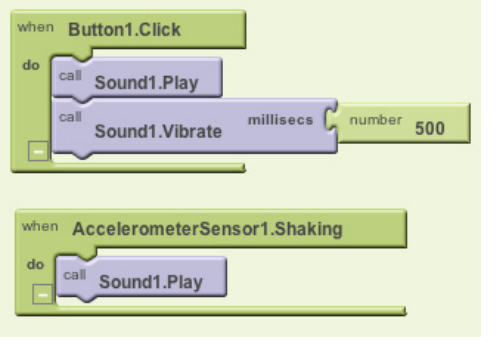

Figure 1-15. The blocks for HelloPurr

Packaging the App for Downloading

App Inventor is a *cloud computing* tool, meaning your app is stored on Google's online servers 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 a Word file or a music track. This also allows you to easily test the app while connected to your phone (what we call *live testing*), without having to download anything to your phone, either. The only problem is that if you disconnect your phone from App Inventor, the app running on the phone will stop, and you won't find an icon for it anywhere because it was never truly installed.

You can package up and install the completed app so that it works on any phone, even when it's not connected to the computer. First, make sure your phone allows apps to be downloaded from places other than the Android Market. Typically, you do this by going to Settings→Applications on your phone and checking the box next to "Unknown sources." Then, go to back into the Designer in App Inventor, click "Package for Phone," and select "Download to Connected Phone." You should see the messages "Saving" and then "Packaging," a process that takes up to a minute. After the "Packaging" message disappears, continue to wait for another 10–15 seconds while the finished app is downloaded to the phone. You'll get a download confirmation when everything is complete.

Once you've downloaded it, look at the apps available on your phone, 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 Phone application.) You can now unplug or even reboot the phone and kill all applications, 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 phone with the USB cable as before. But that won't

change the packaged app that is now installed on your phone. 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 phone.

Go ahead and package your HelloPurr app so you have it on your phone. Once you've done this, you can share it with your family and friends, too!

Sharing the App

You can share your app in a couple of ways. To share the executable app, first click "Package for Phone" and choose "Download to this Computer." This will create a file with a *.apk* extension on your computer. You need to upload this file so that it is accessible on the Web. Once the app is on the Web, other people can install it on their phones by opening the phone's browser and downloading it. Just let them know they need to allow "unknown sources" in their phone's Application settings in order to install apps that aren't from the Android Market.

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 you want to share (in this case, HelloPurr), and select More Actions→Download Source. The file created on your computer will have a *.zip* extension. You can email this file to someone, and she can open App Inventor, choose More Actions→Upload Source, and select the *.zip* file. This will give the user her own complete copy of your app, which she can then edit and customize without affecting your version.

The process of sharing apps will soon be easier and more fun—work is currently underway on a community sharing site.

Variations

Now that you've built a complete app and had the chance to play with it (and maybe download it to share with other people), you might have noticed a couple of things. Take a look at the following items and consider how you'd address them in your app. As you'll likely soon discover, you'll often build an app, find ways to improve and change it, and then go back into it to program those new ideas. Don't worry, that's a good thing—it means you're on your way to becoming a full-fledged app developer!

- As you shake the phone, 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, so the meows are overlapping. If you look at the `Sound` component in the Designer, you'll see a property called `Minimum interval`. That determines how close together successive sounds can start. It's currently set at a half-second (500 milliseconds), which is less than the duration of a single meow. By playing with the minimum interval, you can change how much the meows overlap.

- If you run the packaged app and walk around with the phone in your pocket, your phone 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 phone's menu button. You'll be offered an option to stop the application.

Summary

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

- You build apps by selecting components in the Designer and then telling them what to do and when to do it in the Blocks Editor.
- 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 like play sounds.
- You define components' behavior by assembling blocks in the Blocks Editor. You first drag out an event handler like **Button1.Click**, and then place command blocks like **Sound.Play** within it. Any blocks within **Button1.Click** will be performed when the user clicks the button.
- Some commands need extra information to make them work. An example is `Vibrate`, which needs to know how many milliseconds to vibrate. These values are called *arguments*.
- 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 phone is moved.
- You can package the apps you build and download them to the phone, where they run independently of App Inventor.