CHAPTER 1

Getting Started with the Android Mode

In this chapter, we will introduce the Processing software and its Android mode, the community behind these projects, and how we can start using Processing to create apps for Android devices.

What Is the Processing Project?

The Processing project is a community initiative with the goals of sharing knowledge, fostering education, and promoting diversity in computer programming. The Processing software is a central part of this initiative, guided by the Processing Foundation (https://processingfoundation.org/), whose main mission is to "promote software literacy within the visual arts and visual literacy within technology-related fields – and to make these fields accessible to diverse communities." This community grew off the original Processing software, a programming language and environment created in 2001 by Casey Reas and Ben Fry at the MIT Media Lab as a teaching and production tool in computational arts and design. The Processing software has been continuously developed since then and is available for download at https://processing.org/, with its source code released under free software licenses. From now on, we will simply refer to Processing when talking about the Processing software.

Processing consists of two complementary parts: the Processing programming language and the Processing development environment. Together, they form a "software sketchbook" that allows users to quickly implement, or "sketch," their ideas with code while also providing them with enough room to iteratively refine these ideas into fully developed projects. Processing has been used to create many beautiful and inspiring works in generative art, data visualization, and interactive installations, just to mention a few areas. There are several online resources listing some of these works, including CreativeApplications.Net (www.creativeapplications.net/category/processing/), OpenProcessing (https://openprocessing.org/), and "For Your Processing" on Tumblr (https://fyprocessing.tumblr.com/).

The Processing Language

The Processing language is a set of functions, classes, and constants we can use to program screen drawing, data input/output, and user interaction, originally implemented in Java and now available in Python, JavaScript, and other programming languages. The core team of people behind Processing (https://processing.org/people/) has carefully constructed this language, technically called "Application Programming Interface" or API, to simplify the development of graphical and interactive applications by means of a simple and consistent naming convention, unambiguous behavior, and a well-defined scope.

A. Colubri, *Processing for Android*, https://doi.org/10.1007/978-1-4842-9585-4_1

The Java implementation of the Processing language has been part of Processing dating back to 2001 and encompasses around 200 functions. A defining feature of this language is that it offers us the possibility of creating interactive graphics with very little code. It also includes several simplifications with respect to the Java language, with the purpose of making it easier to teach to people who are not familiar with computer code. The following program illustrates these features of the Processing language:

```
color bg = 150;

void setup() {
  size(200, 200);
}

void draw() {
  background(bg);
  ellipse(mouseX, mouseY, 100, 100);
}
```

The output of this program is a window of 200 by 200 pixels, with a white circle that follows the movement of the mouse on a gray background. The functions `setup()` and `draw()` are present in most Processing programs, and they drive its "drawing loop." All the initialization of the program should take place in `setup()`, which is called just once when the program starts up. The `draw()` function, containing all the drawing instructions, is then called continuously several times per second (by default, 60 times) so that we can animate the graphical output of our program. Another term specific to Processing is the word "sketch," which refers to a Processing program by highlighting the concept of code sketching we mentioned earlier.

The Processing Development Environment

The Processing Development Environment (PDE) is the software that provides users with a minimalistic code editor to write, run, and refine Processing sketches (Figure 1-1). The PDE also incorporates an uncluttered user interface to handle all the sketches created with it and to add libraries and other external components that extend the core functionality of the PDE, such as p5.js, Python, or Android modes.

The ease of use and simplicity of the PDE and the Processing language are the key elements to realize the concept of a "code sketchbook." A stumbling block for many people wanting to start working with code is the complexity of a modern development environment like Eclipse or IntelliJ, in terms of a lengthy installation and an overwhelming user interface. In contrast, the PDE addresses these issues by providing an easy install process and a clean interface, while the simple structure of a Processing sketch enables users to obtain visual feedback rapidly. Processing's aim is to enable an iterative development process analogous to sketching with pen and paper, where we can start with a simple idea and then refine by continuous code sketching.

■ **Note** The Processing language can be used outside of the PDE, for example, in advanced Integrated Development Environments or IDEs, such as Eclipse or IntelliJ. All of Processing's drawing, data, and interaction API can be used when writing a program with these IDEs; but many of the simplifications that the Processing language has with respect to Java will be lost.

Those of us who are familiar with Java have probably noticed that the code shown in Listing 1-1 is not a valid Java program. For example, there is no explicit definition of a main class encapsulating all the code, nor the additional instructions required in Java to initialize the graphical display and the user input. This

sketch needs to be run inside the PDE, which applies a "pre-processing" step to the Processing code to convert it into a complete Java program. However, this step occurs behind the scenes, and we do not need to worry about it at all. Only when using an advanced IDE like those noted previously, we need to make sure to incorporate the Processing API into a syntactically correct Java program.

We can download the latest version of Processing from the main website (https://processing.org/download). As we pointed out in the previous paragraph, installation is very easy, only requiring unpacking the zip (on Windows and macOS) or tgz (on Linux) package that contains the PDE and all other core files. We should be able to then run the PDE without any additional steps, from whichever location inside the home or applications folder.

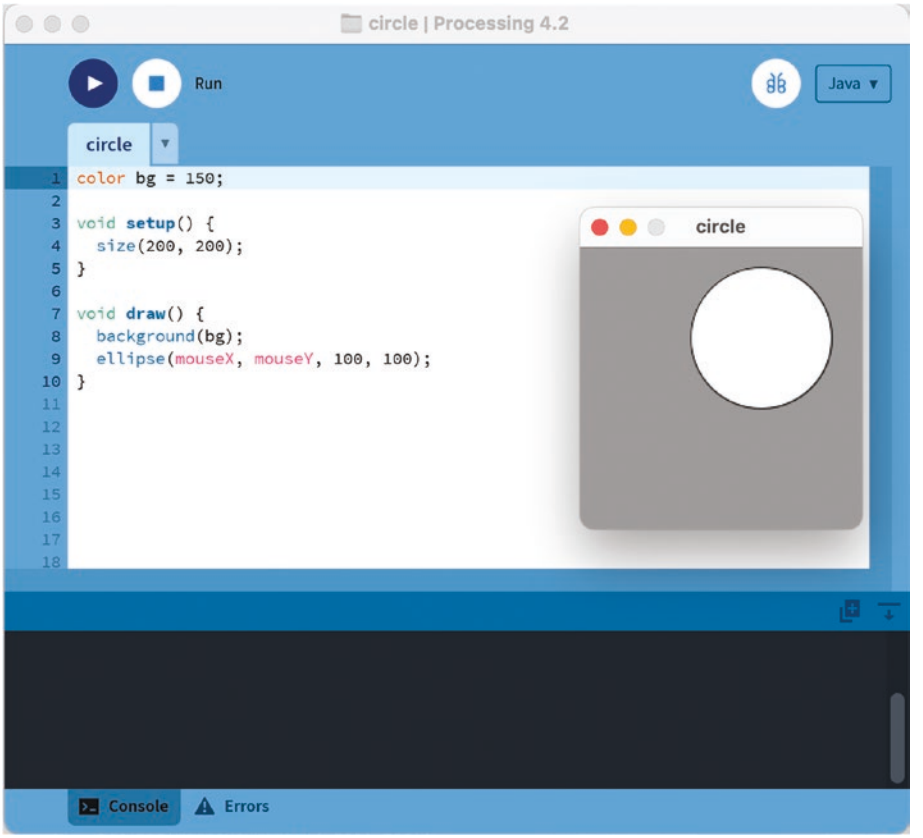

Figure 1-1. *The Processing Development Environment showing a running sketch in the Java mode*

The PDE saves our sketches inside a "sketchbook" folder. Each sketch is stored in a subfolder inside the sketchbook, which in turn contains one or more source code files with the pde extension. By default, Processing creates the sketchbook inside the documents folder located in the user's account (e.g., /Users/andres/Documents/Processing on macOS), but we can change this location by selecting the desired sketchbook folder in the Preferences window, available under the "Processing ➤ Settings" menu on macOS and "File ➤ Settings" on Windows and Linux (Figure 1-2). Notice the sketchbook location at the top.

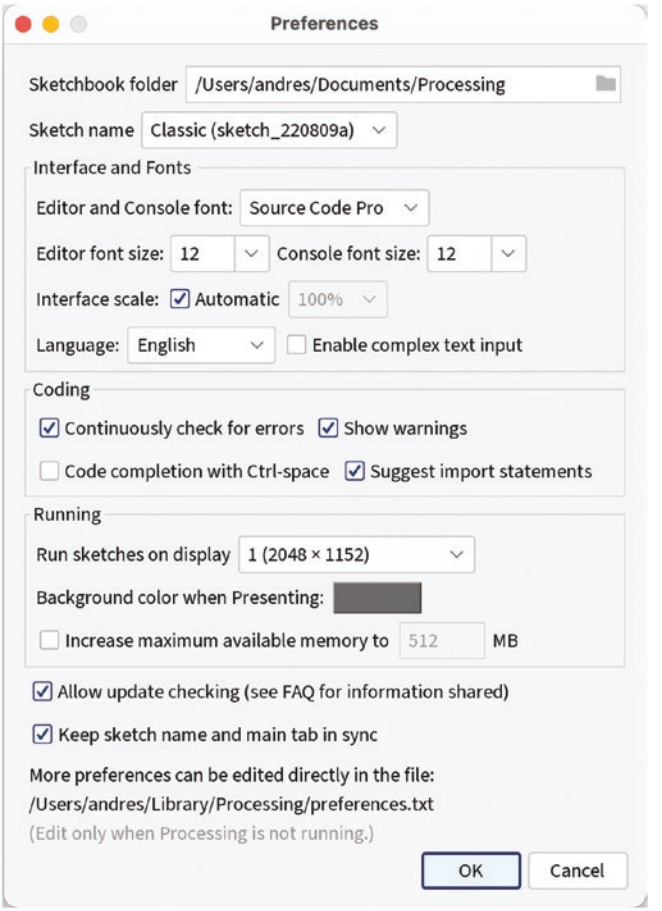

Figure 1-2. *The Preferences window on macOS*

Extending Processing

As we mentioned at the beginning, the Processing project is not only the PDE and the language but also, and very importantly, the community built around the use of the software and the "goal to empower people of all interests and backgrounds to learn how to program and make creative work with code, especially those who might not otherwise have access to these tools and resources," as stated in the home page of the Processing Foundation. Thanks to Processing's open nature and modular architecture, many people have contributed with improvements and extensions to the "core" software over the years. These contributions fall within one of the following four categories:

- Libraries: Modules (comprising one or more Java code files built into a jar package and additional documentation and example files) that make it possible to access new functionality in the sketches, for example, OpenCV library for computer vision applications on PC or Mac computers, or TensorFlow Lite for running machine learning tasks on Android.

- Programming modes: Alternative code editors and related PDE customizations that allow using an entirely different language within the PDE, for example, the Android mode itself! We will see in the next sections of this chapter how to install the Android mode on top of the default PDE.
- Tools: Applications that can only run from Processing and provide specific functionality to help while writing code, debugging, and testing the sketch, for example, the color picker (discussed in Chapter 2).
- Examples: Packages of contributed code sketches that can be used as learning material or reference, for example, the sketches from the *Learning Processing* book by Daniel Shiffman (http://learningprocessing.com/).

The possibility of extending Processing through contributed libraries, modes, tools, and examples has enabled the growth of Processing into application domains that were not part of the original software, such as mobile apps, computer vision, and physical computing, while keeping the core functionality simple and accessible for new programmers.

The Contribution Manager

By default, Processing includes one default mode, Java, where we can write and run sketches on Windows, Mac, and Linux computers using the Java implementation of the Processing language. Processing also bundles several "core" libraries, some of which are OpenGL (for drawing hardware-accelerated 2D and 3D scenes), PDF (to export graphics as PDF files), and data (which allows handling data files in formats such as CSV and JSON).

We can use the Contribution Manager (CM) to install additional contributions, which makes the entire process very seamless. A screenshot of the CM is shown in Figure 1-3. The CM has five tabs, the first four for each type of contribution – libraries, modes, tools, and examples – and the fifth tab for updates. All the contributions that are registered by their authors in a central repository are accessible through the CM and can also be updated through the CM when new versions become available.

■ **Note** Contributions that are not available through the CM can be installed manually. You need to download the package containing the library, mode, tool, or examples, typically in zip format, and extract it into the sketchbook folder. There are separate subfolders for libraries, modes, tools, and examples. See https://processing.org/reference/libraries/ for more info.

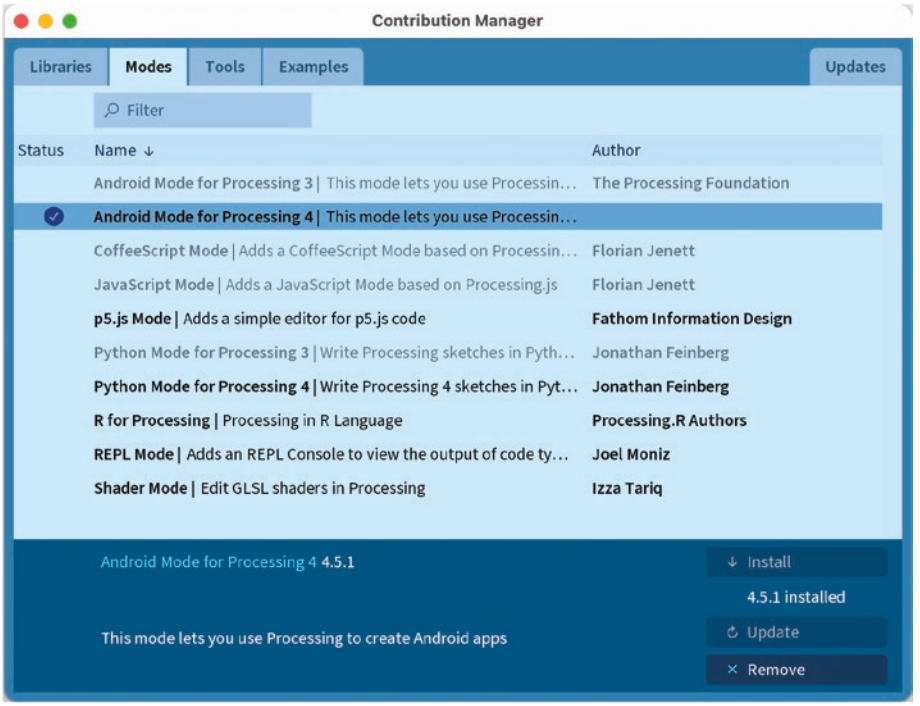

Figure 1-3. *The Contribution Manager in Processing showing the Modes tab*

Processing for Android

Processing for Android, just like the Processing software itself, is also several things. Primarily, it is a community effort that started in 2009 with the purpose of supporting the development of Android apps using Processing and translating some of the concepts of the project to the context of mobile apps: iterative sketching, simplicity, and accessibility.

If we look more closely into the technical details, Processing for Android is made of two parts: the processing-android library and the custom PDE programming mode itself. The library is the package that contains all the functions of the Processing API but re-implemented for the Android platform. The Android mode provides a customized version of the PDE that allows us to write Processing code and run it on an Android device or in the emulator. The Android mode includes the processing-android library, which we need for our Processing code to run without errors. However, we don't need to worry about these distinctions at this point, since Processing will let us install and use the Android mode and will make sure that all the required libraries are placed in the right places. But knowing about the difference between the android library and the mode would be important for those readers who may be planning to use Processing for Android in more advanced applications.

■ **Note** The processing-android library can be imported from an IDE like Android Studio, allowing using all the Processing functionality in a regular Android app. This advanced use is covered in Appendix A.

Installing the Android Mode

Once we have installed Processing in our computer, we should be able to open the PDE by running the Processing application and then to install the most recent release of the Android mode through the CM. The mode also requires the Android Software Development Kit (SDK) to work. The Android SDK is the set of libraries, tools, documentation, and other supporting files provided by Google to develop and debug Android apps. So let's follow these next steps to install the Android mode and, if needed, the SDK:

1. Open the CM by clicking the "Manage Modes…" option that appears in the drop-down menu in the upper-right corner of the PDE (Figure 1-4).
2. Select the entry for the Android mode in the Modes tab and click the "Install" button.
3. After installation is complete, close the CM and switch into the Android mode using the same drop-down menu from Figure 1-4.

When switching into the Android mode for the first time, it will ask us to locate or download the Android SDK (Figure 1-5). Because the SDK is very large (up to several GBs), it could be a good idea to use the one that is already installed to save disk space. However, if that SDK is also used by another development tool, such as Android Studio, it may get updated or changed outside Processing, which may lead to incompatibilities with the Android mode. If that's the case, then it may be better to allow Processing to download and install its own copy of the Android SDK, which will not interfere with the SDK in use by other development tools. If no valid Android SDK is detected, Processing will ask to manually locate an SDK, or automatically download it (Figure 1-5).

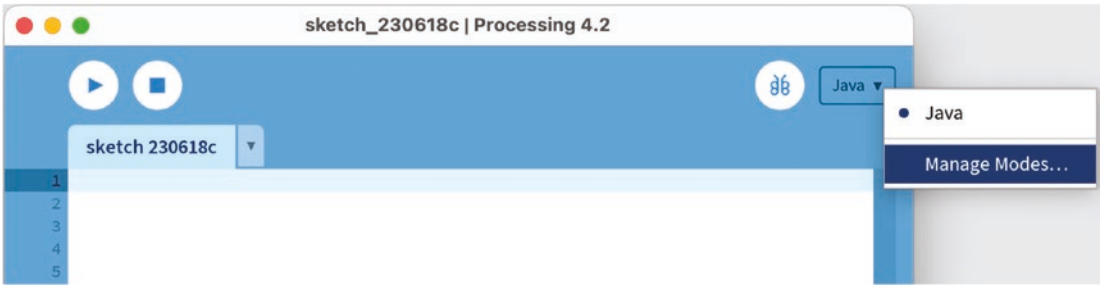

Figure 1-4. *Opening the Contribution Manager to add a new mode*

■ **Note** Version 4.5+ of the Android mode requires Android version 13, corresponding to API level 33 (https://source.android.com/docs/setup/about/build-numbers). The mode's automatic SDK download will retrieve this version from the Google servers.

Figure 1-5. *Pop-up dialog in the Android mode allowing us to search for an existing SDK in our computer manually or download a new copy of the SDK from Google's servers*

Finally, pre-releases of the Android mode, as well as older versions that are no longer are available through the CM, are all deposited in the GitHub releases page (`https://github.com/processing/processing-android/releases`). So, if we are interested in trying these versions out, we could install them manually by downloading the corresponding zip file and extracting it into the modes folder in the Processing's sketchbook.

Interface of the Android Mode

The editor in the Android mode is very similar to that of the Java mode. The toolbar contains the play and stop buttons to launch a sketch (on a connected device) and to stop its execution (on the device or in the emulator). Code autocompletion and integrated debugger in the editor are available as well. The main menu contains several Android-specific options as well (Figure 1-6). The "File" menu has options to export the current sketch as a package ready to upload to the Google Play Store, or as a project that can be opened with Android Studio. The "Sketch" menu contains separate options to run the sketch on a device or in the emulator, as well as a separate "Android" menu containing several options, among them the type of output to target with the sketch – regular app, wallpaper, watch face, VR, or AR app – and the list of Android devices currently connected to the computer. We will cover all these options in the next chapters!

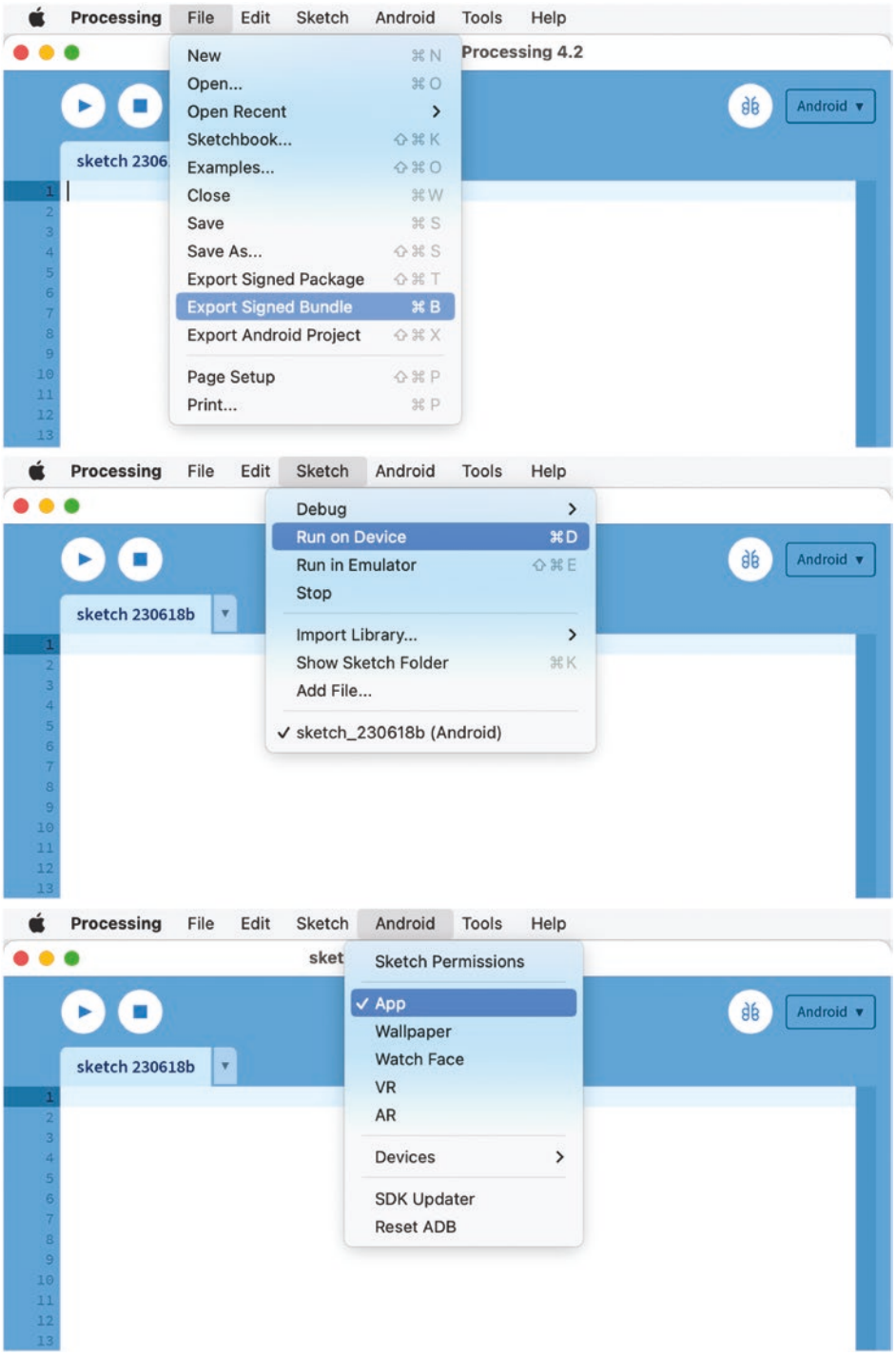

Figure 1-6. *Android-specific options in the interface of the Android mode*

Running a Sketch on a Device

Once we have written some sketch code with the PDE, we can then run it on an Android phone, tablet, or watch. The first thing we need to do is to make sure that "USB Debugging" is turned on on our device. The process to do this varies by device and which version of the Android OS we have installed on it. In most of the cases, this setting is in the Developer Options, under system Setting. On Android 4.2 and higher, the Developer Options are hidden by default, but we can enable them by following these instructions:

1. Open the Settings app.
2. Scroll to the bottom and select About phone.
3. Scroll to the bottom and tap Build number seven times.
4. Return to the previous screen to find Developer Options near the bottom.

After turning USB Debugging on (which we need to do only once), we need to connect the device to the computer through the USB port. Processing will then try to recognize it and add it to the list of available devices in the "Android" menu.

■ **Note** Version 4.5+ of the Android mode only supports devices that are running Android 4.2 (API level 17) or newer.

Let's use the code in Listing 1-1 as our first Processing for Android sketch. It is okay if we do not understand every line of code in it, as we will go over the Processing API in detail in the next chapters. It simply draws a black square in the half of the screen that receives a touch press.

Listing 1-1. Our first Processing for Android sketch

```
void setup() {
  fill(0);
}

void draw() {
  background(204);
  if (mousePressed) {
    if (mouseX < width/2) rect(0, 0, width/2, height);
    else rect(width/2, 0, width/2, height);
  }
}
```

It is possible to have several devices connected simultaneously to the computer, but only one can be selected in the Devices menu as the "active" device, which will be where our sketch will be installed and run. Figure 1-7 shows our first sketch already loaded in the PDE and the selected device to run it on.

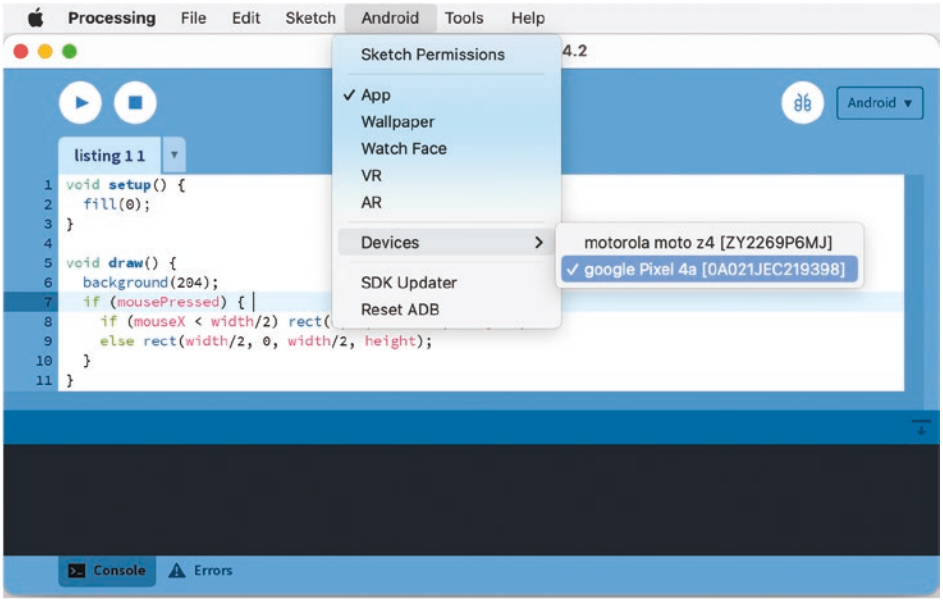

Figure 1-7. *Selecting the device to run the sketch on*

After we have picked the active device, we can hit the run button or select "Run on Device" under the "Sketch" menu. Then, we should see some messages scrolling down in the PDE's console, while Processing compiles the sketch, packages it as an Android app, and installs it on the device. One important detail is that the computer needs to be connected to the Internet the first time we run a sketch. The reason is that Processing uses a tool called Gradle to build the app from the sketch's source code, but Gradle is not part of the initial Android mode installation. The Android mode will automatically download the Gradle tool the first time it tries to run a sketch. It can be offline when running sketches after the first time. If everything goes well, the sketch should launch and show up on the screen of the device, as shown in Figure 1-8.

Figure 1-8. *Running a sketch on a connected phone*

■ **Note** If we are running Windows, we may need to install a special USB driver to connect to the device (https://developer.android.com/studio/run/oem-usb.html). In the case of Linux, we may need to install some additional packages depending on the distribution we are using (https://developer.android.com/studio/run/device.html). Also, make sure that the USB connection is not configured as "Charge Only."

Running a Sketch in the Emulator

If we do not have a device to run our sketch on, we can still use the emulator. The emulator is a program that creates a "software replica" of a physical device that runs on our computer. This replica is called an Android Virtual Device (AVD), and even though it is generally slower than a real device, it can also be useful to test our sketch on hardware we don't currently have.

The first time we run a sketch in the emulator, Processing will download the "system image" containing all the information needed to emulate the AVD in our computer (Figure 1-9). However, it will initially ask us if we want to use the "ARM" or the "x86" images. The reason for this is that Android devices use ARM CPUs (central processing units), while our desktop or laptop computers often have x86 CPUs (ARM and x86 denote different "architectures" that CPUs can have to process the instructions from our code). When emulating an AVD with an ARM image on an x86 CPU, the emulator will convert ARM instructions into x86 instructions one by one, which could be slow. But if we use the x86 image on an x86 CPU, our computer will be able to simulate the AVD's CPU directly and therefore much faster. One drawback of using x86 images is that we must install an additional software on Mac or Windows called HAXM. Processing already downloaded HAXM together with the Android SDK, so it will install it for us in case we decide to use x86 images. However, with the ARM-based Mac computers (M1, M2, etc.), we do not have this issue since their CPUs have the same ARM architecture as the emulated devices.

We should also keep in mind that HAXM is only compatible with Intel processors, so the emulator won't work with x86 images if our computer has an AMD CPU. Linux has its own AVD acceleration system and does not require HAXM, so we can use x86 images on a Linux computer with an AMD CPU. We would need to perform some extra configuration steps though, which are described here: https://developer.android.com/studio/run/emulator-acceleration.html#vm-linux.

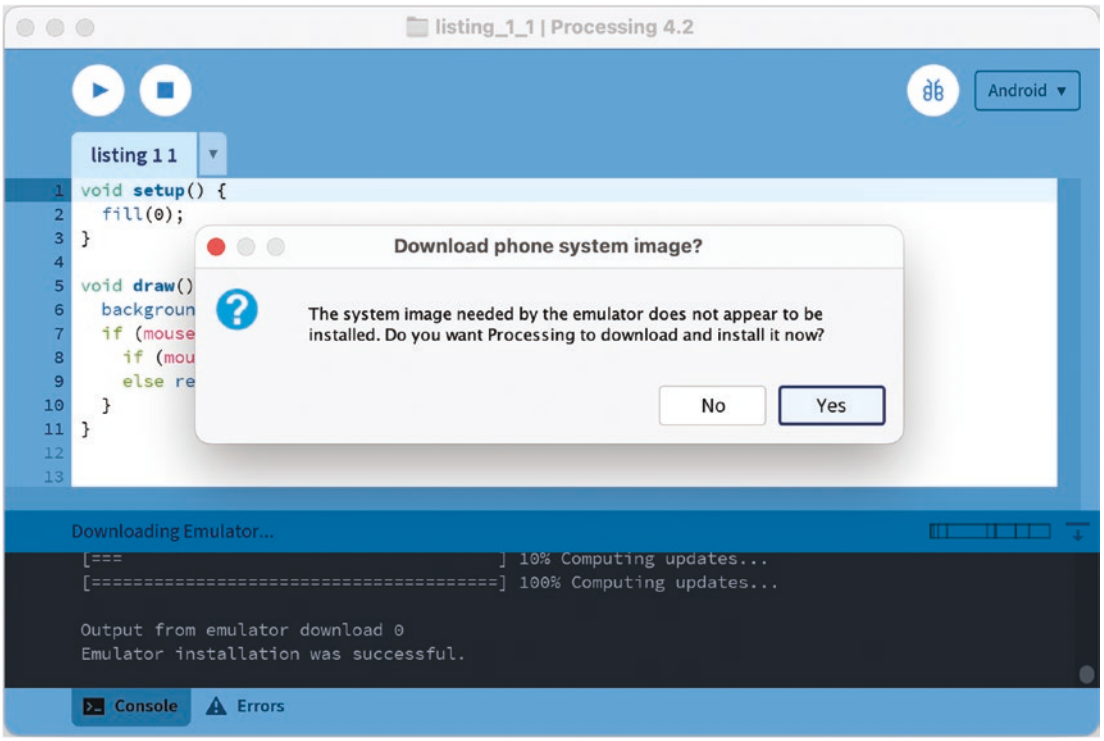

Figure 1-9. *System image download dialog in the Android mode*

After finishing the download, which can take several minutes depending on our Internet connection (the system images for the emulator are over 1GB in size), Processing will boot up the emulator and then will launch the sketch in it. Once our Listing 1-1 is running in the emulator, it should look like the one shown in Figure 1-10.

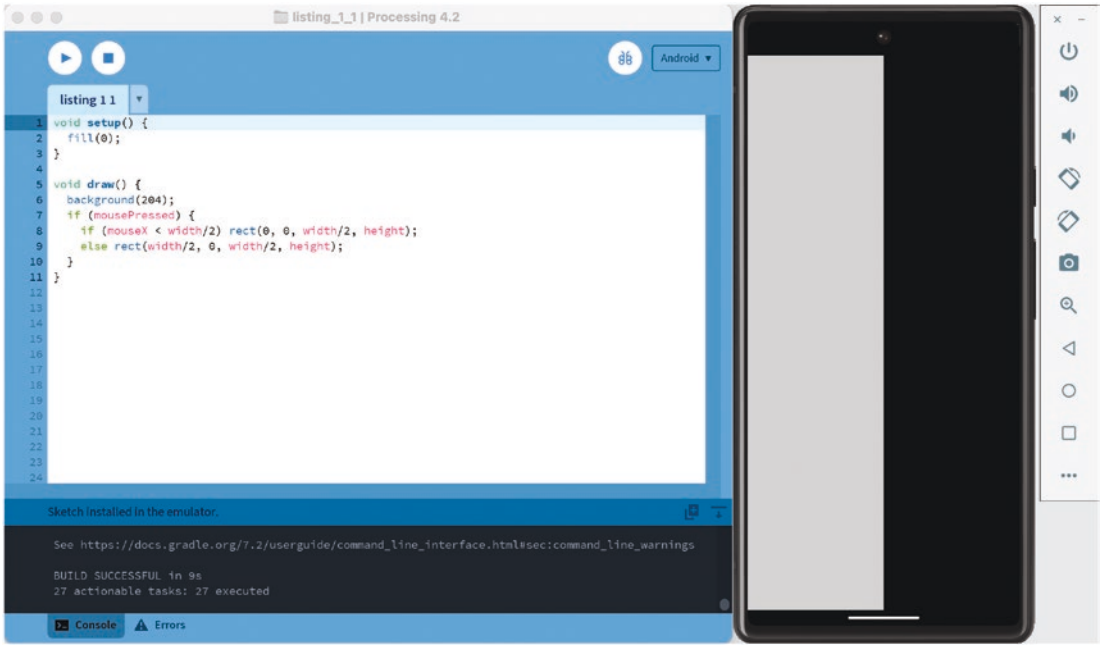

Figure 1-10. *Running our sketch in the emulator*

Summary

In this first chapter, we have learned what the Processing project and software are and how we can use Processing to create apps, thanks to its Android mode. As we saw, some of the main features of the Processing software are its minimal interface and the simple structure of a program, called a "sketch." These features allow us to start writing, testing, and refining our own sketches, either on a device or in the emulator, very quickly.