# 模块 1

# 初识 Android

**学习目标**

(1)了解 Android 基础知识,掌握 Android 体系结构。

(2)认识 Android 应用程序编程接口(application programming interface,API)并熟悉其适用范围。

(3)掌握 Android 开发环境搭建的方法。

## 1.1 Android 基础知识

Android 是 Google 公司基于 Linux 平台开发的手机、平板电脑等的操作系统,自问世以来,受到了前所未有的关注,并成为移动平台最受欢迎的操作系统之一。

### 1.1.1 Android 起源与发展

说起 Android,就不得不提安迪·鲁宾(Andy Rubin),也就是所谓的“Android 之父”。早在 2003 年,Andy Rubin 就同其他三位创始人成立了“Android 公司”,开发一种基于数码相机的系统,这便是 Android 的雏形。

随后,Android 转而开发手机操作系统,并于 2005 年被谷歌收购,Rubin 和其他创始人留在谷歌,开始了真正意义上的智能手机操作系统开发。此时,Android 系统的方向已经确定,如基于 Linux、开源、免费供手机厂商使用等,2007 年 11 月,Google 与 84 家硬件制作商、软件开发商及电信运营商组建开放手机联盟,共同研发改良 Android 系统。随后 Google 以 Apache 开源许可证的授权方式,发布了 Android 的源代码,2008 年 9 月发布第 1 个版本 Android 1.1 版本。

Android 系统一经推出,版本升级非常快,几乎每隔半年就有一个新的版本发布。从 Android 1.5 版本开始,每一个版本都用一个甜点的名字作为代号。具体版本如下:

2009 年 4 月 30 日，Android 1.5 Cupcake(纸杯蛋糕)正式发布。

2009 年 9 月 15 日，Android 1.6 Donut(甜甜圈)正式发布。

2009 年 10 月 26 日，Android 2.0/2.1 Eclair(泡芙)正式发布。

2010 年 5 月 20 日，Android 2.2/2.2.1 Froyo(冻酸奶)正式发布。

2010 年 12 月 7 日，Android 2.3 Gingerbread(姜饼)正式发布。

2011 年 2 月 2 日，Android 3.0 Honeycomb(蜂巢)正式发布。2011 年 5 月 11 日，Android 3.1 Honeycomb(蜂巢)正式发布。2011 年 7 月 13 日，Android 3.2 Honeycomb(蜂巢)正式发布。严格来说，蜂巢并不算是甜点，这也意味着 3.0 版本的独特。它是一款专为平板设计的 Android 分支。

2011 年 10 月 19 日，Android 4.0 Ice Cream Sandwich(冰淇淋三明治)正式发布。它直接适配不同设备，不论手机和平板均可使用，另外还可以使用照片解锁。

2012 年 6 月 28 日至 2013 年 7 月，发布的 Android 4.1～4.3 都被称为果冻豆，其中的改进包括全面支持谷歌 Chrome 浏览器，增强搜索功能，提升触摸速度，支持 HDR 摄影等。

2013 年 9 月 4 日，Android 4.4 KitKat(奇巧巧克力)正式发布。

2014 年 10 月 15 日，Android 5.0 Lollipop(棒棒糖)正式发布。大人小孩都爱的棒棒糖成为 Android 5.0 的代号，也预示着 Android 的普及和流行。全新的界面设计、更丰富的功能，让该系统大受欢迎。另外，5.1 版本还加入了双 SIM 卡、高清语音通话、设备锁等新功能。Android 也从此开始进入一年一个大更新的进程。

2015 年 9 月 30 日，Android 6.0 Marshmallow(棉花糖)正式发布。新功能包括垂直滚动的应用抽屉 UI、指纹识别、USB-C 接口和 Android Pay 等。

2016 年 8 月 22 日，Android 7.0 Nougat(牛轧糖)正式发布。功能进化明显，包括针对大屏手机的分屏模式、更好的多任务模式，以及 Vulkan 3D 渲染 API 等内核改进。

2017 年 8 月 22 日，Android 8.0 Oreo(奥利奥)正式发布。

2018 年 8 月 7 日，Google 官方发布了 Android 9.0 Pie(馅饼)。

2019 年 9 月 4 日，Google 推送了 Android 10.0(Android Q)的正式版。

### 1.1.2 Android 体系结构

Android 系统采用分层架构，由高到低分为四层，依次是应用程序层(application)、应用程序框架层(application framework)、核心类库(libraries)和 Linux 内核(Linux kernel)，如图 1-1 所示。

下面以系统启动过程为主线，从进程的视角来剖析 Android 体系结构。

**1. 应用程序层**

应用程序层是一个核心应用程序的集合，所有安装在手机上的应用程序都属于这一层。例如，系统自带的短信、浏览器、通讯录等，或者下载的微信、QQ、支付宝等。

**2. 应用程序框架层**

应用程序框架层是 Android 为开发者提供的开放平台，位于应用程序层的下一层，主要提供构建应用程序时用到的各种 API。Android 提供的是一组服务和系统，在开发应用程序层应用时会直接用到。

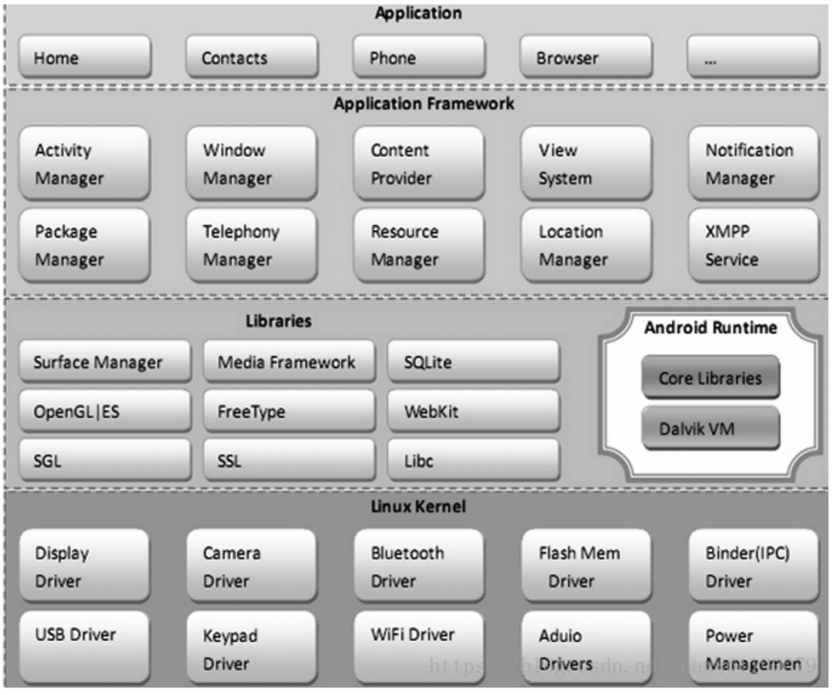


图 1-1　Android 体系结构

(1)视图系统(view system):构建应用程序的界面。

(2)内容提供者(content provider):允许应用程序访问其他应用程序的数据或者共享数据。

(3)通知管理器(notification manager):允许应用程序在状态栏上显示定制的提示信息。

(4)活动管理器(activity manager):管理应用程序的生命周期,提供一个通用的导航回退功能。

(5)资源管理器(resource manager):提供对非代码资源的管理。

### 3. 核心类库

核心类库包含了系统库和 Android 运行库。系统库主要包括一组 C/C++库,用于 Android 系统中不同的组件,这些功能通过 Android 应用程序框架对开发者开放。例如,多媒体库(media framework),基于 OpenCore 多媒体开源框架,支持多种视频、音频文件;WebKit 库提供了对浏览器内核的支持。

Android 运行时库(Android runtime)主要提供了一些核心库,能够允许开发者使用 Java 语言来编写 Android 应用程序。另外,Android 运行时库中还包括了 Dalvik 虚拟机,它使得每一个 Android 应用都能运行在独立的进程当中,并且拥有一个自己的 Dalvik 虚拟机实例。相比于 Java 虚拟机,Dalvik 虚拟机是专门为移动设备定制的,它针对手机内存、CPU 性能等做了优化处理。

### 4. Linux 内核

Android 依赖于 Linux 相应版本的核心系统服务,为 Android 设备的各种硬件提供底层的驱动,如显示驱动、音频驱动、照相机驱动、蓝牙驱动、电源管理驱动等。

### 1.1.3 Android API

API 是一些预先定义的函数，目的是为应用程序与开发人员提供基于某软件或硬件得以访问一组例程的能力，而又无须访问源码或理解内部工作机制的细节。程序员通过使用 API 函数开发应用程序，从而可以避免编写无用程序，以减轻编程任务。

Android 适用于各种各样的手机，从最低端直到最高端的智能手机。核心的 Android API 在每部手机上都可使用，但仍然有一些 API 接口有一些特别的适用范围。

#### 1. API 级别

一个 Android 平台提供的框架 API，被指定一个整数标识符，称为 API 级别。它唯一标识一个具体版本的 Android 平台及其框架的 API 的版本。Android 平台提供的一套框架 API，使得应用程序可以与系统底层进行交互。该框架 API 由以下模块组成：

(1)一组核心的包和类。

(2)清单(manifest)文件的 XML 元素和属性声明。

(3)资源文件的 XML 元素和属性声明及访问形式。

(4)各类意图(Intents)。

(5)应用程序可以请求的各类授权，以及系统中包含的授权执行。

每个 Android 平台的后续版本都会包括它提供的更新的 Android 应用程序框架的 API，使高版本的 API 与早期版本兼容。在极少数情况下，旧版本的 API 部分可能被修改或删除，通常这种变化是为了保障 API 的稳定性及应用程序或系统的安全。表 1-1 说明了具体平台版本和支持的 API 级别的对应关系。

表 1-1 Android 版本与 API 级别的对应关系表

| 平台版本 | API | 平台版本 | API | 平台版本 | API | 平台版本 | API |
|---|---|---|---|---|---|---|---|
| Android 1.0 | 1 | Android 2.3 | 9 | Android 4.1，4.1.1 | 16 | Android 6.0 | 23 |
| Android 1.1 | 2 | Android 2.3.3 | 10 | Android 4.2，4.2.2 | 17 | Android 7.0 | 24 |
| Android 1.5 | 3 | Android 3.0 | 11 | Android 4.3 | 18 | Android 7.1/7.1.1 | 25 |
| Android 1.6 | 4 | Android 3.1 | 12 | Android 4.4 | 19 | Android 8.0 | 26 |
| Android 2.0 | 5 | Android 3.2 | 13 | Android 4.4W | 20 | Android 8.1 | 27 |
| Android 2.0.1 | 6 | Android 4.0/4.01/4.0.2 | 14 | Android 5.0 | 21 | Android 9.0 | 28 |
| Android 2.1 | 7 | Android 4.0.3/4.0.4 | 15 | Android 5.1 | 22 | Android 10.0 | 29 |
| Android 2.2 | 8 | | | | | | |

#### 2. API 分类

(1)WiFi API。WiFi API 为应用程序提供了一种与那些带有 WiFI 网络接口的底层无线堆栈相互交流的手段。几乎所有的请求设备信息都是可利用的，包括网络的连接速度、IP 地址、当前状态等，还有一些其他可用网络的信息。一些可用的交互操作包括扫描、添加、保存、结束和发起连接。

(2)定位服务。定位服务允许软件获取手机当前的位置信息。这包括从全球定位系统

卫星上获取地理位置，但相关信息不限于此。例如，未来其他定位系统可能会运营，届时，与其对应的 API 接口也会被加入系统中。

(3)多媒体 API。多媒体 API 主要用于播放媒体文件。这同时包括对音频(如播放 MP3 或其他音乐文件及游戏声音效果等)和视频(如播放从网上下载的视频)的支持，并支持播放 URI 地址模式——在网络上直接播放的流媒体。从技术上来说，多媒体 API 并不是“可选的”，因为它总是要用到。但是不同的硬件环境上面可能有不同的编解码的硬件机制，因而它又是“可选的”。

(3)基于 OpenGL 的 3D 图形。Android 的主要用户接口框架是一个典型的面向控件的类继承系统。在它下面是一种非常快的 2D 和 3D 相组合的图形引擎，并且支持硬件加速。用来访问平台 3D 功能的 API 接口是 OpenGL ES API。和多媒体 API 一样，OpenGL 也不是严格意义上的“可选”，因为这些 API 总是存在并且实现那些固定的功能。但是，一些设备可能有硬件加速环节，使用它时就会影响应用程序的表现。

## 1.2　Android 开发环境搭建

在开发 Android 程序之前，首先要在系统中搭建开发环境。目前使用的是 Android Studio 环境，Android Studio 提供了集成的 Android 开发工具，拥有更强大的功能和更高效的性能。本书使用的是 Google 公司 2018 年 8 月 7 日发布的 Android Studio 3.2(Android 9.0)版本。

### 1.2.1　系统要求

要想保证 Android Studio 的运行速度，开发用的计算机最低配置要求如下：

(1)内存：最低要求 4 GB，推荐 8 GB 或 16 GB。

(2)CPU：要求 1.5 GHz 以上。

(3)处理器：要求 i5、i7。

(4)硬盘：要求系统盘剩余空间 10 GB 以上。

(5)如果操作系统是 Windows，那么至少为 Windows 7，不支持 Windows XP。

### 1.2.2　Android 依赖的包

Android Studio 作为 Android 应用的开发环境，仍然需要依赖于 JDK(Java development kit)、软件开发工具包(software development kit，SDK)和原生开发工具包(native development kit，NDK)三种开发工具。

**1. JDK**

JDK 是 Java 语言的编译器，即 Java 语言的软件开发工具包。主要用于移动设备、嵌入式设备上的 Java 应用程序。JDK 是整个 Java 开发的核心，它包含了 Java 的运行环境(JVM＋Java 系统类库)和 Java 工具。因为 Android 应用采用 Java 语言开发，所以开发机上要先安装 JDK。JDK 建议安装 1.8 及以上版本，因为不同的 Android 版本对 JDK 有相应的要求。

### 2. SDK

SDK 被软件开发工程师用于为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具的集合，主要包括：

(1)build-tools 目录，存放各版本 Android 的各种编译工具。

(2)docs 目录，存放开发说明文档，包括开发指南、API 参考、资源等。

(3)add-ons 目录，Android 开发需要的第三方文件和软件库。

(4)extras\android 目录，存放兼容低版本的新功能支持库，如 android-support-v4. jar、v7 的各种支持库、v13 以上的兼容库等。

(5)platforms 目录，分版本存放安装下载的所有 Android 资源文件。

(6)platform-tools 目录与 tools 目录，存放常用的开发辅助工具，如数据库管理工具 sqlite3. exe、模拟器管理工具 emulator. exe 等。

(7)samples 目录，存放各版本 Android 常用功能的 demoi 源码。

(8)system-images 目录，存放模拟器各版本的系统镜像与管理工具。

(9)sources 目录，存放各版本 Android 的 API 开放接口源码。

SDK 可以单独安装，也可以与 Android Studio 一起安装，单独安装的下载页面入口地址是 http://sdk. android-studio. org。一般通过 Android Studio 安装 SDK，因为这样可以避免一些兼容性与环境设置问题。无论是单独安装还是一起安装，SDK 安装完成后都要在环境变量的系统变量中添加 ANDROID_HOME，取值为 SDK 的安装目录，如 E:\Android\sdk，并在系统变量 Path 末尾添加“;%ANDROID_HOME%\tools”。

### 3. NDK

NDK 是一组可以让开发者在 Android 应用中利用 C 和 C++代码的工具，主要供 JNI 接口使用，先把 C\C++代码编译成 so 库，然后由 Java 代码通过 JNI 接口调用 so 库。

Android 程序运行在 Dalvik 虚拟机中，NDK 允许用户使用类似 C / C++之类的原生代码语言执行部分程序。NDK 包括以下方面：

(1)从 C / C++生成原生代码库所需要的工具和 build files。

(2)将一致的原生库嵌入可以在 Android 设备上部署的应用程序包文件(application packages files，即. apk 文件)中。

(3)支持所有未来 Android 平台的一系列原生系统头文件和库。

## 1.2.3 安装 Android Studio 3.2

Google 公司 2018 年 8 月 7 日发布 Android Studio 3. 2(Android 9. 0)版本，国内开发者可直接在其官网 http://www. android-studio. org 下载。Android Studio 3. 2 安装包 android-studio-ide-181. 5014246-windows. exe 的大小为 923 MB，安装包不带 SDK。

下载完成后直接双击安装，其具体步骤如下：

**【步骤 1】** 进入 Android Studio 安装欢迎界面，单击“Next”按钮继续安装，如图 1-2 所示。


图 1-2　Android Studio 安装欢迎界面

【步骤 2】 已经默认选中“Android Studio”复选框和“Android Virtual Device”复选框，如果不需要安装虚拟设备，则不需要选中此项。单击“Next”按钮继续安装，如图 1-3 所示。

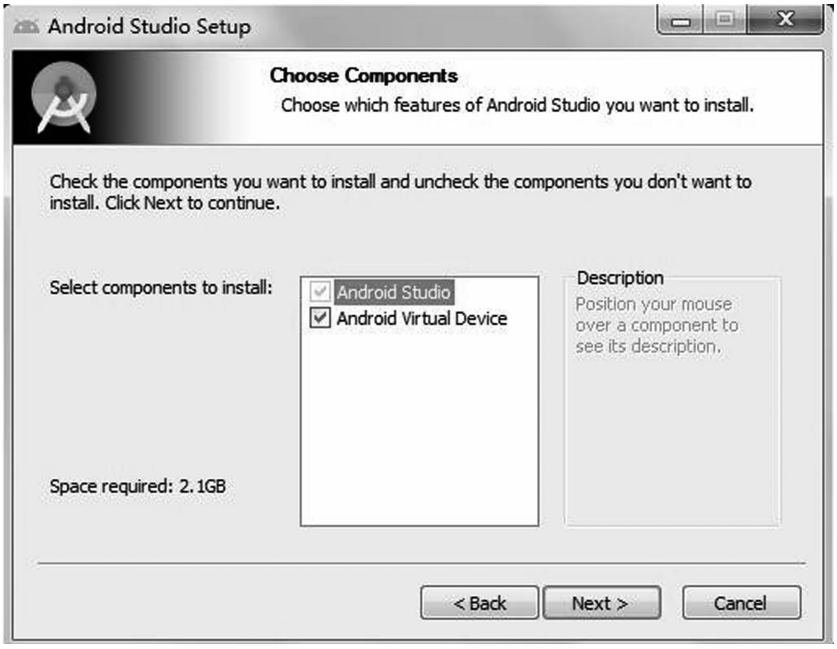


图 1-3　选择要安装的组件

【步骤 3】 进入如图 1-4 所示的界面，设置 Android Studio 的安装位置。单击“Browse”按钮，可以在弹出的对话框中修改 Android 的安装位置。

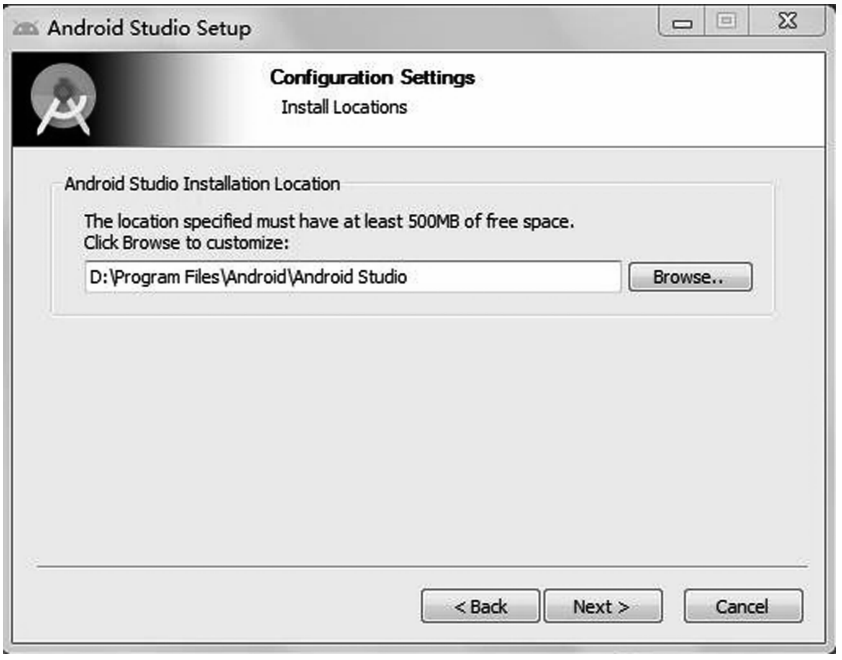


图 1-4　设置 Android Studio 的安装位置

【步骤 4】 选择开始菜单文件夹，安装默认在桌面创建快捷方式，选中“Do not create shortcuts”复选框，则不需要创建快捷方式，如图 1-5 所示。单击“Install”按钮开始正式安装，此时要从官网下载安装文件，需要等待一段时间。


图 1-5　选择开始菜单文件夹

【步骤 5】 安装完成后单击“Next”按钮。如图 1-6 所示，这里 Android Studio 应用程序虽然安装完毕，但是还需要继续对其进行配置；选中“Start Android Studio”复选框，单击“Finish”按钮将会启动 Android Studio，然后将会进行一些第一次启动前的初步设置。


图 1-6　启　　动

【步骤 6】 Android Studio 第一次启动，需要进一步对环境进行设置。弹出如图 1-7 所示的对话框，是在询问是否导入以前的配置文件，在这里可以直接选中“Do not import settings”单选按钮，不导入以前的设置，单击“OK”按钮进入启动界面。


图 1-7　询问是否导入以前的配置文件

【步骤 7】 现在的 Android Studio 安装包仅包含其本身，并不包含 Android SDK。所以会弹出如图 1-8 所示的对话框，提示无法访问 Android SDK 加载项列表，这里先不理会，单击 Cancel 按钮，暂时忽略，进入欢迎界面，后续会下载 SDK。


图 1-8　提示没有 SDK

【步骤 8】 进入欢迎界面，如图 1-9 所示。单击“Configure”下拉按钮，在弹出的下列菜单中选择“SDK manager”选项，打开“Default Settings”对话框，开始设置 Android SDK。

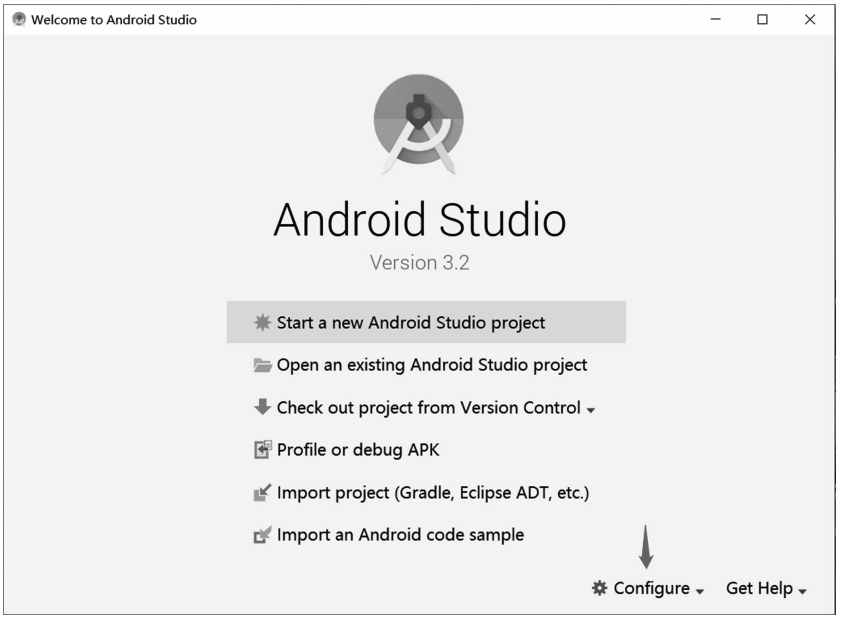


图 1-9　Android Studio 3.2 启动界面

【步骤 9】 如图 1-10 所示，在左侧窗格中依次展开“Appearance&Behavior”→“System Settings”选项，选择“Android SDK”选项，进入“Android SDK”管理界面。单击“Android SDK Location”栏右侧的“Edit”链接，在弹出的对话框中设置 SDK 下载后的保存路径。在“SDK Platforms”选项卡的列表框中选中需要安装的项目对应的复选框，建议全部选中。每个 Android SDK 平台包都包含 Android 平台和属于某个 API 级别的源代码，安装后 Android Studio 会自动检查更新，设置完毕后单击“OK”按钮进行安装。

【步骤 10】 切换到“SDK Tools”选项卡，如图 1-11 所示。单击“OK”按钮将会下载 Android SDK 相应的资源，在线升级编译工具 Build-Tools、平台工具 Platform-Tools，以及开发者需要的其他工具。

至此已经完成 Android Studio 的下载安装及初步设置。接下来就可以使用 Android Studio 进行 Android 开发了。

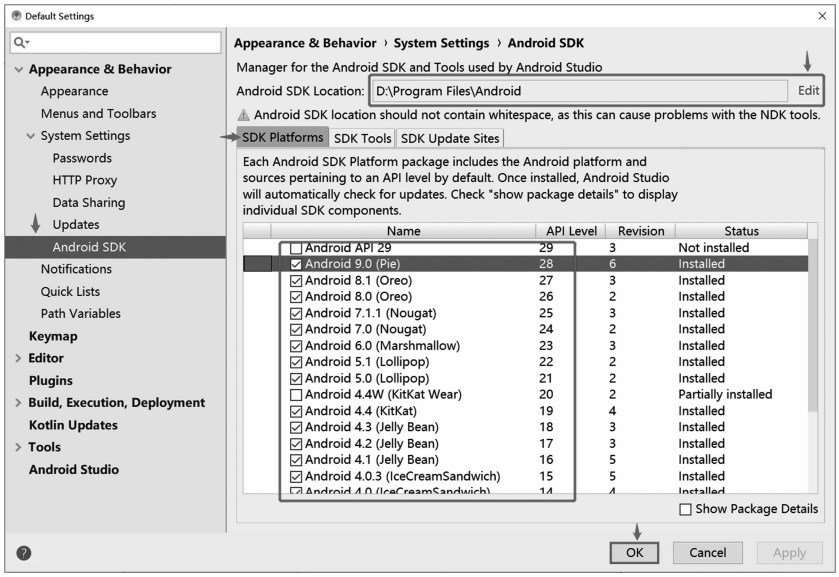


图 1-10　“Android SDK”管理界面

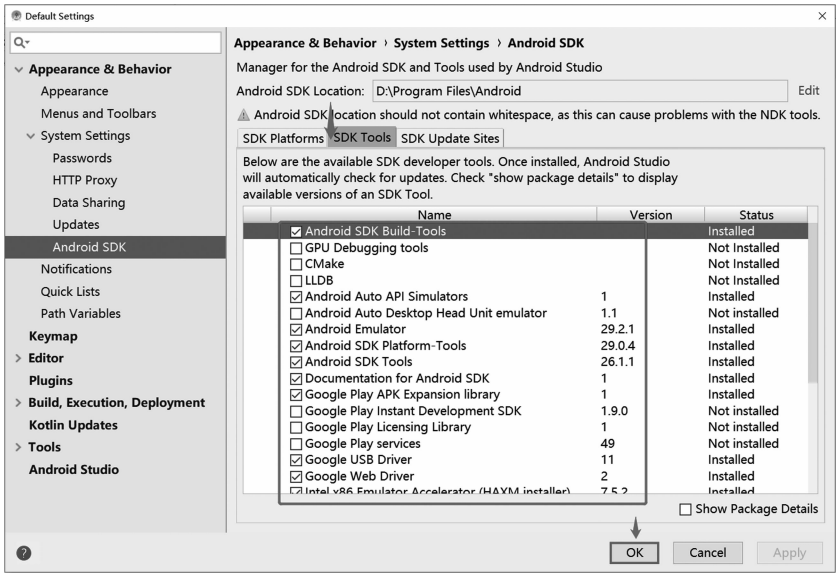


图 1-11　“SDK Tools”选项卡

## 1.3　开发第一个 Android 程序

成功安装 Android Studio 后，会发现 Android Studio 开发环境非常复杂，为了尽快熟悉这个开发环境，了解 Android 项目的结构，本书第一时间从实战入手，创建一个经典的 HelloWorld 应用程序。

### 1.3.1 创建 HelloWorld 程序

【步骤 1】 启动 Android Studio，选择“Start a new Android Studio project”选项，创建一个新 Android 工程，如图 1-12 所示，其中：

(1)“Application name”用于设置应用名称，该命名要求第一个字母必须大写。

(2)“Company domain”用于设置公司域名，可以通过它来设置包名，一般不用修改。

(3)“Project location”用于设置工程存放的路径，可以创建自己的工作目录用于存放 Android Studio 工程，也可以不用更改，使用默认路径。

(4)“Package name”即包名，由应用名称与公司域名自动生成，可以不用更改。

设置完成后单击“Next”按钮。

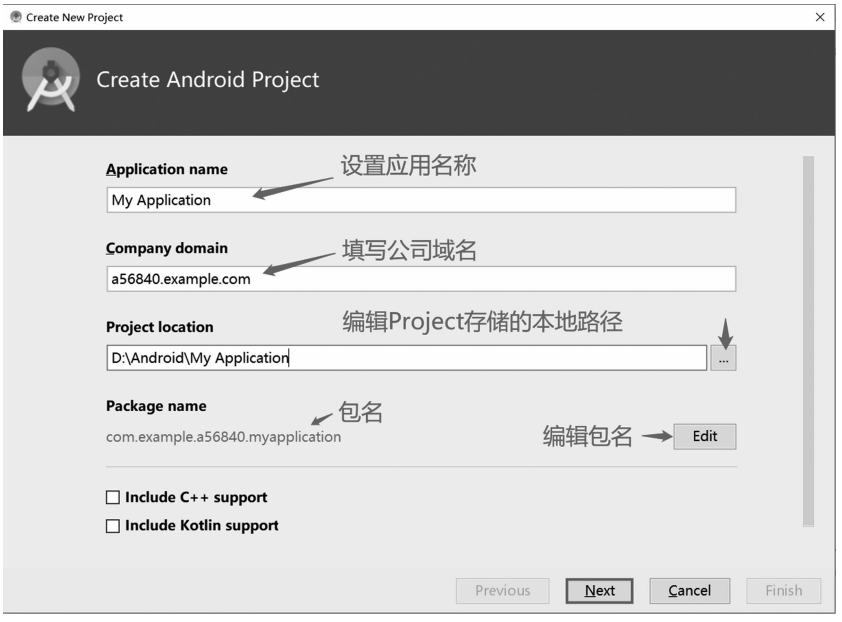


图 1-12 创建新工程

【步骤 2】 设置目标 Android 设备。通过选中复选框来确定应用开发的目标设备，如图 1-13 所示。选中“Phone and Tablet”复选框，表示适用于手机和平板电脑；选中“Wear OS”复选框，表示适用于可穿戴设备；选中“TV”复选框，表示适用于智能电视；选中“Android Auto”复选框，表示适用于智能车载设备；选中“Android Things”复选框，表示适用于其他 Android 设备。

通过设置工程适用的最小 API 级别来选择应用程序适用的设备范围。较低的 API 级别适用于更多的设备，但提供较少的 API 特性。这里选择默认的 API 21，当使用机型的 API 小于这个级别时，将不再适用。

【步骤 4】 单击“Next”按钮，进入“Installing Requested Components”界面，即进入安装工程所需组件的环节，这一环节需要一段时间，当自动安装完毕后单击“Next”按钮继续。

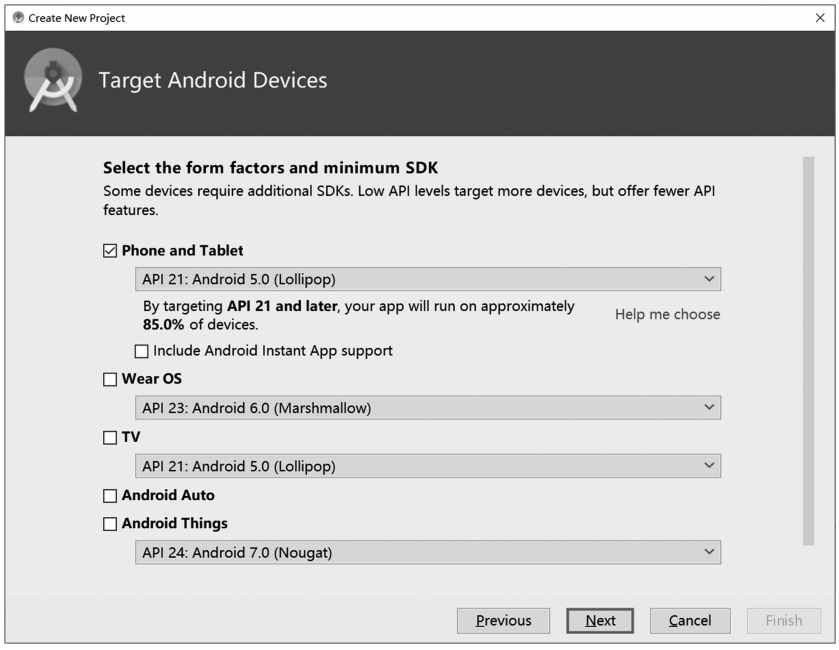


图 1-13　设置目标 Android 设备

【步骤 5】　进入“Add an Activity to Mobile”界面，如图 1-14 所示。选择不同的 Activity(活动界面)类型，为了更好地理解工程代码，选择“Empty Activity”选项，这样可以避免工程中出现冗余的代码，单击“Next”按钮继续。

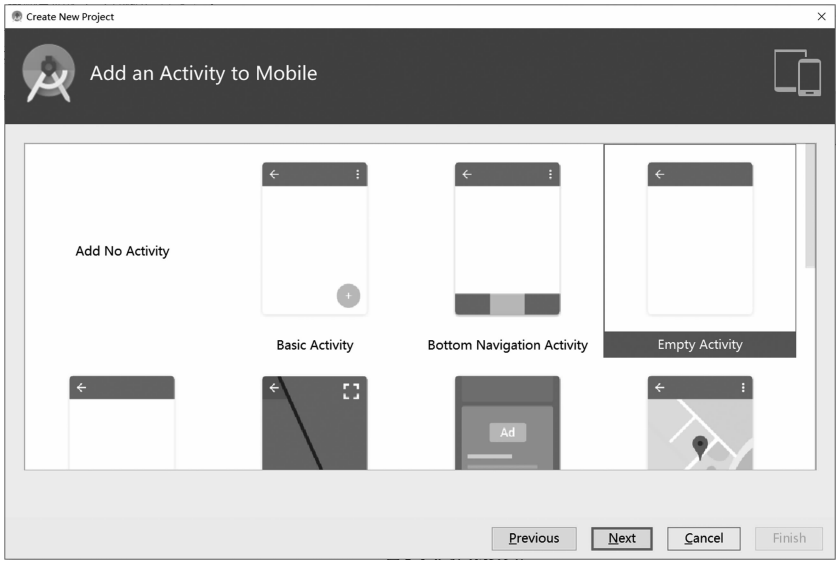


图 1-14　“Add an Activity to Mobile”界面

【步骤 6】　创建一个新的空 Activity，如图 1-15 所示。设置“Activity Name”及“Layout Name”；选中图 1-15 中所示的复选框，设置是否生成布局文件及是否向后兼容。

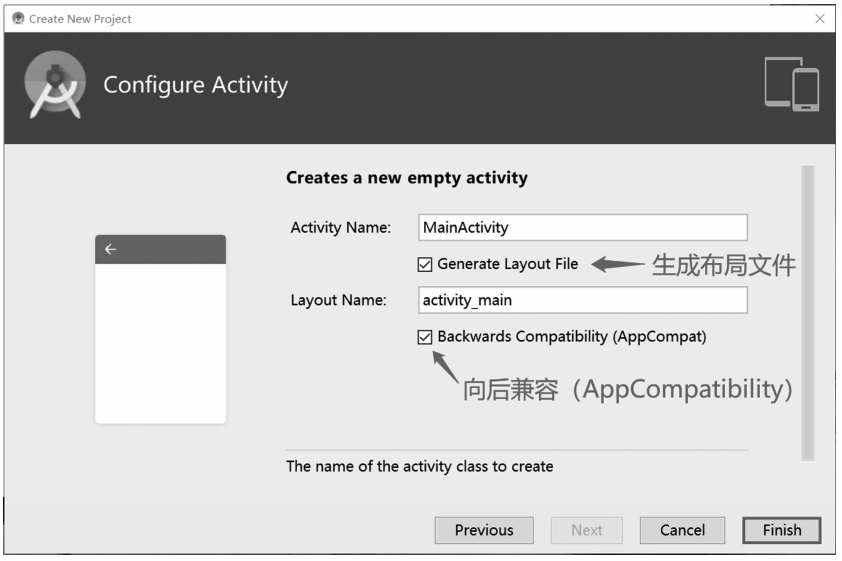


图 1-15　创建一个新 Activity

【步骤 7】 在图 1-15 所示的对话框中单击“Finish”按钮完应用程序的创建，进入应用开发环境，此时在 Android Studio 中会显示创建好的 HelloWorld 程序，如图 1-16 所示。至此，HelloWorld 程序的创建已经全部完成。

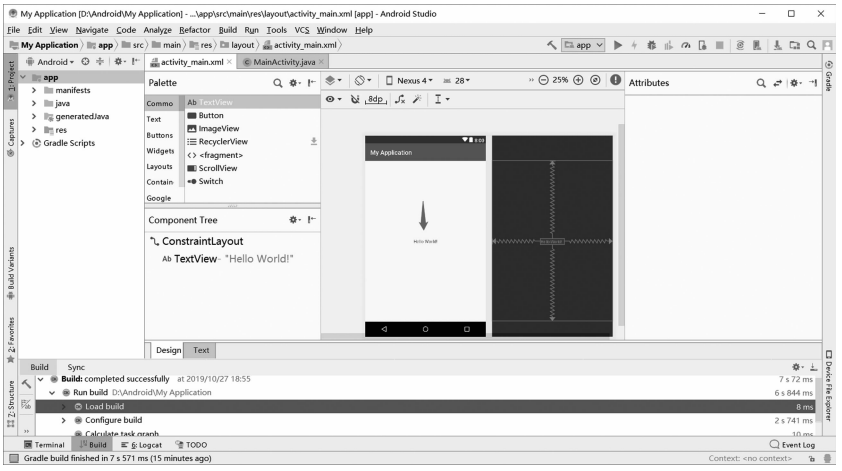


图 1-16　HelloWorld 程序

## 1.3.2　认识程序中的文件

当 HelloWorld 项目创建成功后，Android Studio 会自动生成两个默认文件，即布局文件和 Activity 代码文件，另外，还有清单文件与配置文件等。

### 1. 布局文件

布局文件也就是界面布局文件，是一个 XML 文件。在该布局中可以添加任意按钮和文本框或其他组件，让 App 界面变得美观、友好。下面是 HelloWorld 程序的布局文件代码：

```
//表示按照 1.0 版本的 XML 规则进行解析,传输数据的字符编码采用 utf-8 编码格式
<?xml version="1.0" encoding="utf-8"?>
//根节点布局方式
<android.support.constraint.ConstraintLayout
//定义这个 XML 文件所使用的命名空间
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
//定义该布局宽高属性
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
//定义文本视图,用于显示"Hello World!"
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Android 中常见的命名空间包括 android、tools、app(自定义命名空间)这几个常见的命名空间。

(1)android。

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

在 Android 布局文件中必须在根元素上定义这样一个命名空间,接下来对这行代码进行逐一说明。

①xmlns:即 xml namespace,声明开始定义一个命名空间;

②android:称作 namespace-prefix,它是命名空间的名字;

③http://schemas.android.com/apk/res/android:这个 URL 是一个不可访问的地址,只是一个统一资源标识符,它的值固定不变,相当于一个常量。使用这行代码,可以引用命名空间中的属性。例如,在这个布局中,只要以 android:开头的属性便是引用了命名空间中的属性,android 是赋予命名空间一个名字,与我们平时定义变量一样。

(2)tools。

```
xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity"
```

可以把它理解为一个工具(tools)的命名空间,它只作用于开发阶段,当 App 被打包时,所有关于 tools 的属性都会被摒弃掉。tools:context 是在开发中查看 Activity 布局效果,使

用 context 属性，可以在开发阶段中看到设置在 Activity 中的主题效果。

(3)app。

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

app 是个自定义命名空间，设置格式为在 res/后面填写包名。通常自定义命名空间是和自定义 View 分不开的，当 Android 自带的控件不能满足需求时，可以自己去绘制一些 View，而要为自定义 View 加上自定义属性时，就需要创建自定义命名空间。

命名空间里面存放的是特定属性的集合，这样一来，思路就很清晰了，也就是说自定义命名空间的实际过程就是自定义属性。

### 2. Activity 代码文件

XML 布局文件通过 Activity 的 setContentView 方法进行渲染之后才会转换成 View 对象显示在界面上。Activity 文件是由 Java 代码实现的，定义布局的目的、功能，并包含各种方法的实际代码。下面是 HelloWorld 程序的第一个 Activity 代码文件(MainActivity.java)的内容：

```
//指定 App 的包名
package com.example.a56840.myapplication;
//引用包，是引用 Android SDK 的过程，直接引用开发者写好的方法可以提高效率
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
//声明 MainActivity 类继承 AppCompatActivity 类
public class MainActivity extends AppCompatActivity {
//重写在 Java 里面继承来的方法，覆盖掉原来的 onCreate 方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
//调用一个对象的父对象上的函数，也就是 AppCompatActivity
        super.onCreate(savedInstanceState);
//启动内容视图，调用布局文件 activity_main.xml
        setContentView(R.layout.activity_main);
    }
}
```

可以看出，MainActivity.java 的代码内容很简单，只有一个 MainActivity 类，该类下面只有一个函数 onCreate()。注意 onCreate()内部的 setContentView()方法直接引用了布局文件的名字 activity_main，该方法的意思是向 App 界面填充 activity_main.xml 的布局内容。

### 3. 清单文件

每个 Android 程序创建成功后，都会自动生成一个清单文件 AndroidManifest.xml，存放于 manifests 文件夹中。该文件是整个项目的配置文件，程序中定义的四大组件都需要在该文件中进行注册。下面就来看一下清单文件中的默认内容，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
//根节点,指定 Android 命名空间与 App 包名
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.a56840.myapplication">
//该节点用于指定 App 的自身属性
    <application
//用于指定是否允许备份,开发阶段设置为 true,上线时设置为 false
        android:allowBackup="true"
//用于指定该 App 在手机屏幕上显示的图标(普通图标)
        android:icon="@mipmap/ic_launcher"
//用于指定该 App 在手机屏幕上显示的名称
        android:label="@string/app_name"
//用于指定该 App 在手机屏幕上显示的图标(圆形图标)
        android:roundIcon="@mipmap/ic_launcher_round"
//是否支持阿拉伯语、波斯语这种从右往左的文字排列顺序
        android:supportsRtl="true"
//指定该 App 的显示风格
        android:theme="@style/AppTheme">
//子节点,注册一个 Activity
        <activity android:name=".MainActivity">
//设置 Activity 属性,表示当前 Activity 最先启动
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
//当前应用显示在桌面程序列表中
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

### 4. 编译配置文件

项目中一般会出现两个或者多个 build.gradle 文件,一个在根目录下,另一个在 App 目录下。根目录下项目级别的 build.gradle 一般无须改动,一般只须关注 App 目录下模块级别的 build.gradle,下面是在初始的 build.gradle 文件中补充文字注释,有利于读者更好地理解每个参数。

```
//指定 App 的包名
apply plugin:'com.android.application'
android {
//指定编译用的 SDK 版本号,Android 9.0 使用版本 28
    compileSdkVersion 28
    defaultConfig {
```

```
            applicationId "com.example.a56840.myapplication"
    //可运行应用的最低版本的 Android 平台
            minSdkVersion 21
    //指定运行应用的目标 API 级别
            targetSdkVersion 28
    //指定 App 的应用版本号
            versionCode 1
    //指定 App 的应用版本名称
            versionName "1.0"
            testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        }
        buildTypes {
            release {
    //指定是否开启代码混淆功能
                minifyEnabled false
    //指定代码混淆规则文件的文件名
                    proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
            }
        }
    }
    //指定 App 编译的依赖信息
    dependencies {
    //指定引用 jar 包的路径
        implementation fileTree(dir: 'libs', include: ['*.jar'])
    //指定编译 Android 的高版本支持库,AppCompatActivity 必须指定编译 appcompat-v7 库
        implementation 'com.android.support:appcompat-v7:28.0.0'
        implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    //指定单元测试编译用的 junit 版本号
        testImplementation 'junit:junit:4.12'
        androidTestImplementation 'com.android.support.test:runner:1.0.2'
        androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    }
```

### 1.3.3 运行 Android 程序

如何运行 Android 程序取决于两件事情,是否有一个 Android 设备和是否正在使用 Android Studio 开发程序。运行 Android 程序的设备可以是真实的 Android 设备,也可以是虚拟的 Android 模拟器。本节将以在模拟器上运行为例说明 Android 程序的运行。

首先确定当前开发机系统支持 VT-X 运行,若系统提醒 BIOS 未支持 VT-X,需要手动在“BIOS-configuration—Virtual Technology”界面中打开。条件满足后可以执行以下步骤:

**【步骤 1】** 单击工具栏上的“运行”按钮▶,打开“Select Deployment Target”对话框,选

择程序部署目标模拟器。如图 1-17 所示，在虚拟设备列表中选择虚拟设备，单击 OK 按钮启动虚拟机。

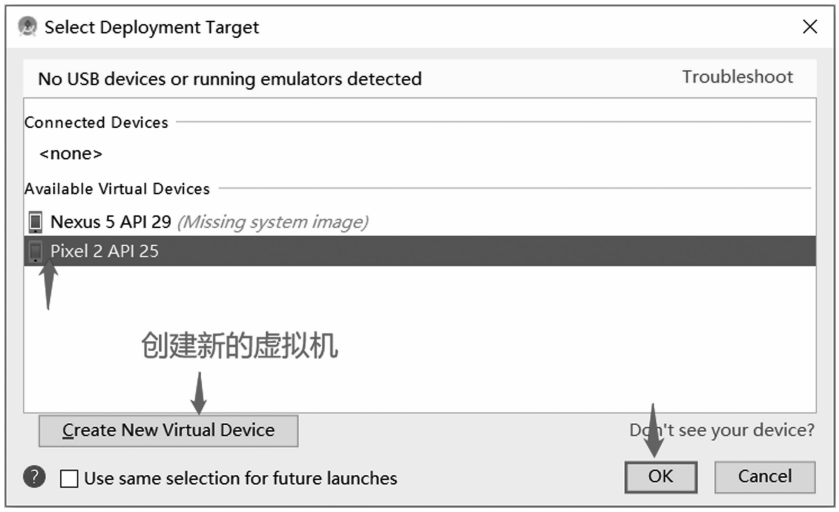


图 1-17　选 择 虚 拟 机

【步骤 2】　如果没有可选类型，则可以单击图 1-17 中的“Create New Virtual Device”按钮创建新的虚拟设备。如图 1-18 所示，选择设备类别、设备型号，单击 Next 按钮继续。

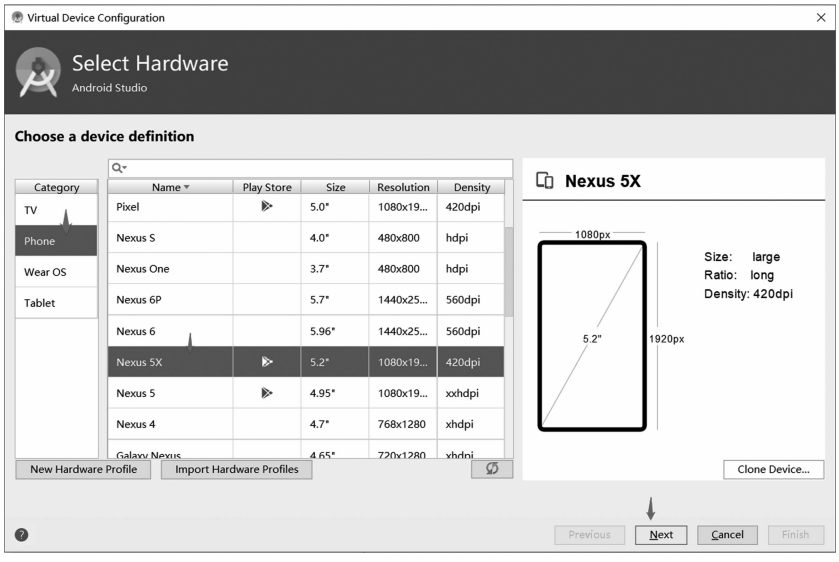


图 1-18　可 选 类 型

【步骤 3】　在“Virtual Device Configuration”对话框中对所选虚拟设备进行配置，如图 1-19 所示，选择 Android 版本及 API 级别，单击“Next”按钮继续。在打开的对话框中单击“Finish”按钮完成虚拟机的新建。

【步骤 4】　选择刚刚创建的虚拟机，单击“OK”按钮进行启动。第一次启动该设备，需要下载并安装相应版本的 SDK，经过一段时间后，即可打开模拟机显示 HelloWorld 程序，如图 1-20 所示。

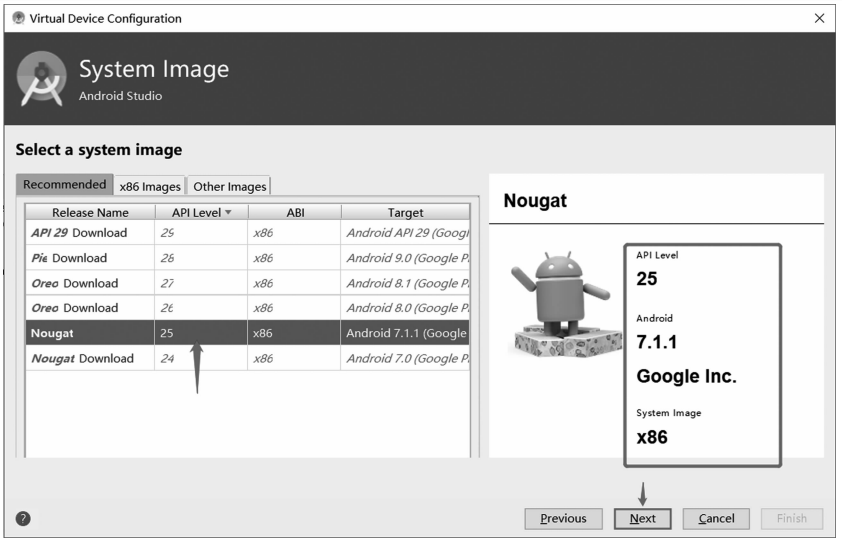


图 1-19　选择 Android 版本及 API 级别

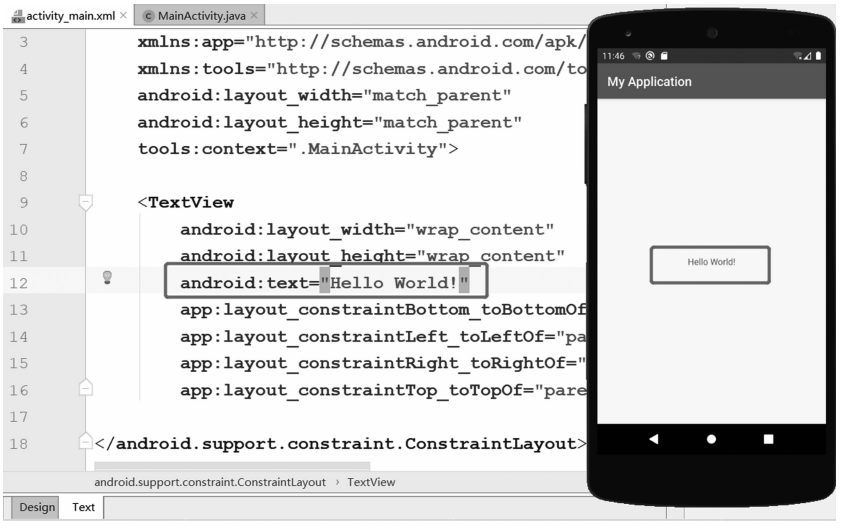


图 1-20　模拟机显示 HelloWorld 程序

### 1.3.4　Android 程序打包

Android Studio 应用程序开发、调试完成后，需要生成 APK 文件进行发布，为用户提供安装文件，这个过程称为打包，这个包就是要生成的 APK 文件。包分为 debug 版和 release 版，通常所说的打包指生成 release 版的 APK，release 版的 APK 会比 debug 版的小，release 版的 APK 还会进行混淆和用自己的 Key Store 签名，以防止别人反编译后重新打包替换成自己的应用。以 release 版包为例，其具体操作步骤如下：

**【步骤 1】** 在当前的工程编辑状态下执行“Build”→“Generate Signed Bundle/APK”命

令，打开如图 1-21 所示的“Generate Singned Bundle or APK”对话框，选中“APK”单选按钮，单击“Next”按钮继续，构建可以布署到设备的签名 APK。


图 1-21　选择生成 APK 包

**【步骤 2】** 在打开的新对话框中，单击“Create new”按钮后打开“New Key Store”对话框；单击…按钮设置新建新密钥库的存储路径及密钥的名称，如图 1-22 所示。这个 jks 文件很重要，相当于 APK 的身份证。如果开发者用同一代码生成两个 jks 文件包，那就代表的是两个软件。


图 1-22　设置密钥库路径及名称

【步骤 3】 继续回到“New Key Store”对话框，为密钥和密钥库配置以下信息，如图 1-23 所示。其中，“Key store path”用于创建密钥库的位置；“Password”选项是为密钥库创建一个安全的密码，“Confirm”是为密钥创建并确认安全的密码，此密码应当与密钥库的密码不同；“Alias”选项为密钥输入一个标识名、别名；“Validity (years)”选项以年为单位设置密钥的有效时长，密钥的有效期应至少为 25 年，以便开发者可以在应用的整个生命期内使用相同的密钥签署应用更新；“Certificate”为证书输入一些关于开发者自己的信息，包括开发者姓名、组织单位、所在城市或地区、州或省及国家代码等，此信息不会显示在应用中，但会作为 APK 的一部分包含在开发者的证书中。


图 1-23 密钥和密钥库配置信息

【步骤 4】 密钥库和密钥生成后，相关信息会自动填充到“Generate Singned Bundle or APK”对话框中，如图 1-24 所示。单击“Next”按钮就可以用来签署 APK 了。

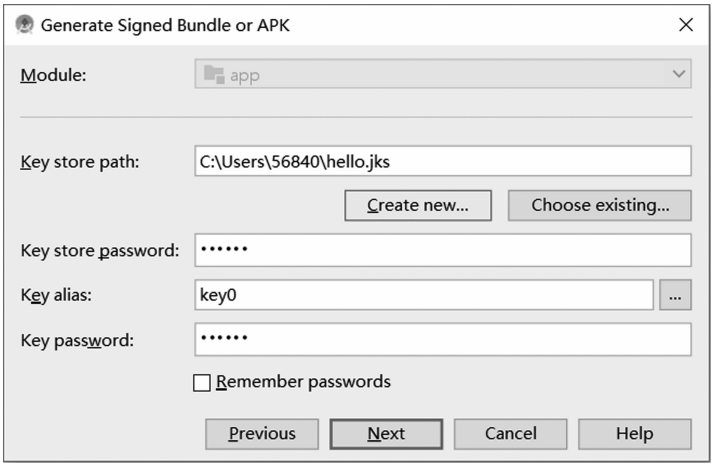


图 1-24 完成密钥设置

【步骤 5】 如图 1-25 所示，设置 APK 文件存储的位置；选择构建类型，这里使用默认的 release 类型；设置该应用发布的版本号，完成后单击“Finish”按钮完成程序打包操作。


图 1-25　设置包文件存储路径

【步骤 6】 在本地按照 D:\android\MyApplication\app\release 的路径，就可以找到 APK 文件，如图 1-26 所示的 app-release. apk 文件，可以进行改名操作，推送到 Android 系统的手机后就可以安装了。

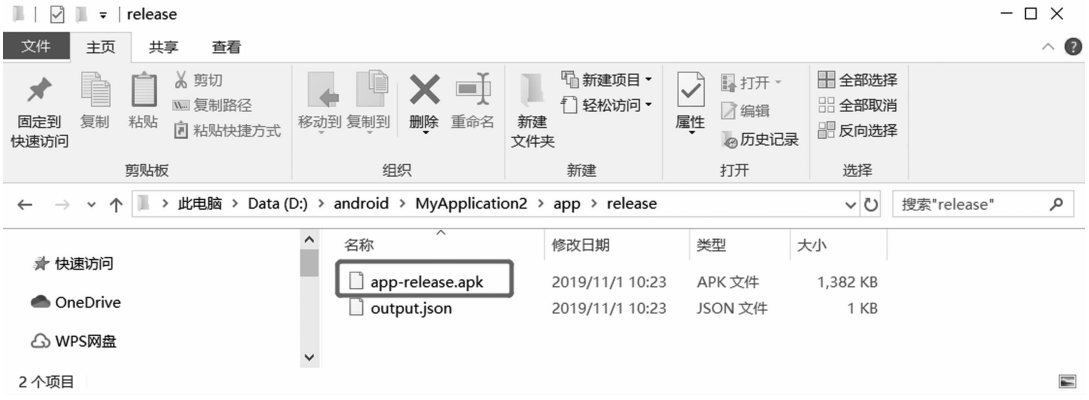


图 1-26　找到 APK 文件

## 1.4　Android Studio 环境认识

Android Studio 应用开发环境比较复杂，内容也很多，初次接触 Android Studio，首先需要对工程界面、Android 的项目结构、程序文件等有一个简单的了解，下面进行简单的介绍。

### 1.4.1　Android Studio 工程界面

Android Studio 应用开发环境默认是以 Android 形式显示工程目录的，下面主要了解如图 1-27 所示的五个主要区域。

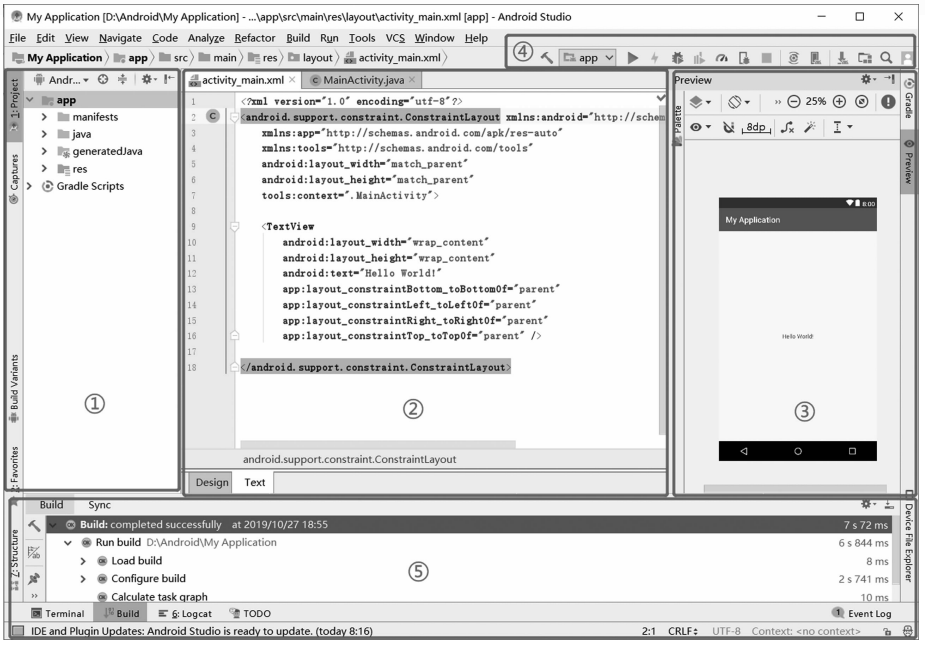


图 1-27　Android Studio 工程界面

(1)第一个区域为项目工程区域,主要是工程文件资源等相关的操作。展示项目中文件的组织方式,默认是以 Android 方式展示的,可选择 Project、Packages、Scratches、Project Files、Problems 展示方式。平时用得最多的就 Android 和 Project 两种。该区域右上角的按钮分别用于定位当前打开文件在工程目录中的位置、关闭工程目录中所有的展开项及一些额外的系统配置等。

(2)第二个区域为布局与代码区,主要是用来编写代码和设计布局。图 1-17 中的该区域包括两个文件,一个是 activity_main. xml 布局文件,另一个是 MainActivity. java 程序文件,单击文件名所在位置,可以在文件之间进行切换。布局设计有两种模式,分别是 Design 设计模式和 Text 文本模式,布局编辑模式可以切换,新手一般更愿意使用 Design 编辑布局,编辑后再切换到 Text 模式,对于学习 Android 布局设计很有帮助。

(3)第三个区域为 UI 布局预览区域,可以实时查看布局设计的效果。其上方的按钮可调整预览大小比例等。

(4)第四个区域是运行、调试及 Android 设备和虚拟机的操作按钮区。前七个按钮分别负责编译、运行、调试 App 中的模块,测试 App 中显示的模块代码覆盖率、调试 Android 进程及重新运行、停止运行等;后边按钮则负责虚拟设备管理、同步工程的 Gradle 文件、项目结构及与项目相关的属性配置、Android SDK 管理等。

(5)第五个区域大部分是用来查看一些输出信息的。例如,查看终端、监控、工程编译的一些输出信息、应用运行后的一些相关信息、标有 TODO 注释的列表、事件日志,还可以了解 Gradle 构建应用时的一些输出信息等。

### 1.4.2　Android Studio 常规设置

对于 Android Studio 环境可以根据个人的使用习惯进行一些常规设置,包括主题设置、

代码字体大小设置、导包方式设置等。

### 1. 更换主题

界面主题有 IntelliJ 默认明亮风格和 Darcula 暗黑炫酷风格，可以通过外观进行设置。如图 1-28 所示，选择“File”→“Settings”→“Appearance&Behavior”→“Appearance”选项，进入外观设置界面，通过“Theme”下拉列表框设置界面主题，通过“Name”下拉列表框设置系统字体，通过“Size”下拉列表框设置系统字体大小，单击 OK 按钮完成设置。

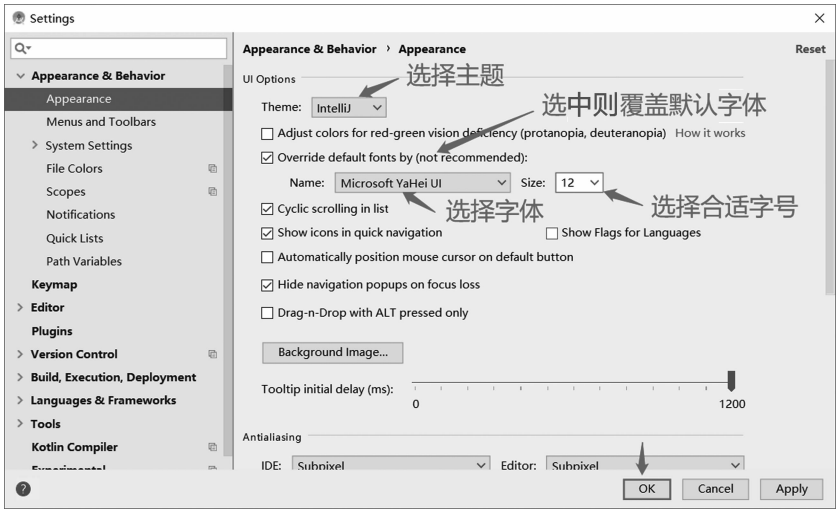


图 1-28　Android Studio 外观设置

### 2. 修改代码字体大小

选择“File”→“Settings”→“Editor”→“Colors&Fonts”→“Font”选项，可修改“Primary font”选项和“Size”选项，调整字号大小，单击“OK”按钮，如图 1-29 所示。


图 1-29　修改代码字体大小

### 3. 设置自动导包

选择“File”→“Settings”→“Editor”→“General”→“Auto Import”选项，选中“Optimize imports on the fly(for current project)”和“Add unambiguous imports on the fly”两个复选框，单击“OK”按钮，如图 1-30 所示。

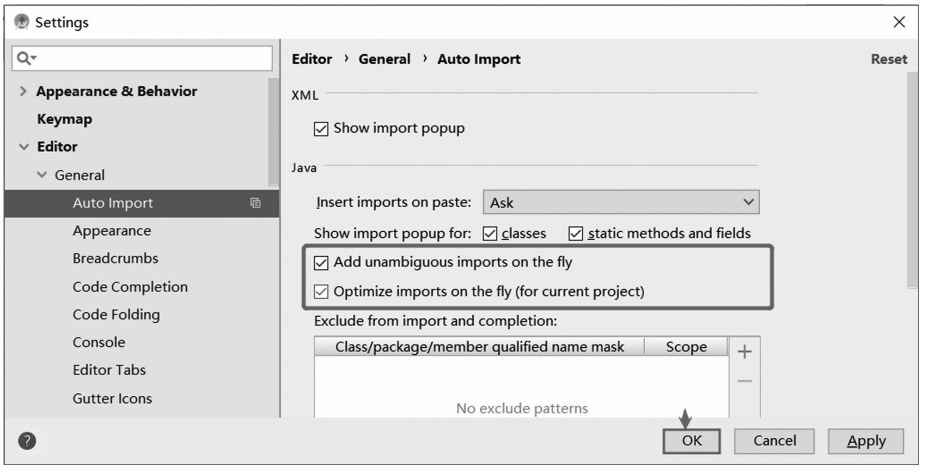


图 1-30　自动导包设置

### 4. 关闭自动检查更新

选择“File”→“Settings”→“Appearance&Behavior”→“System Settings”→“Updates”选项，取消自动检查更新，单击“OK”按钮，如图 1-31 所示。

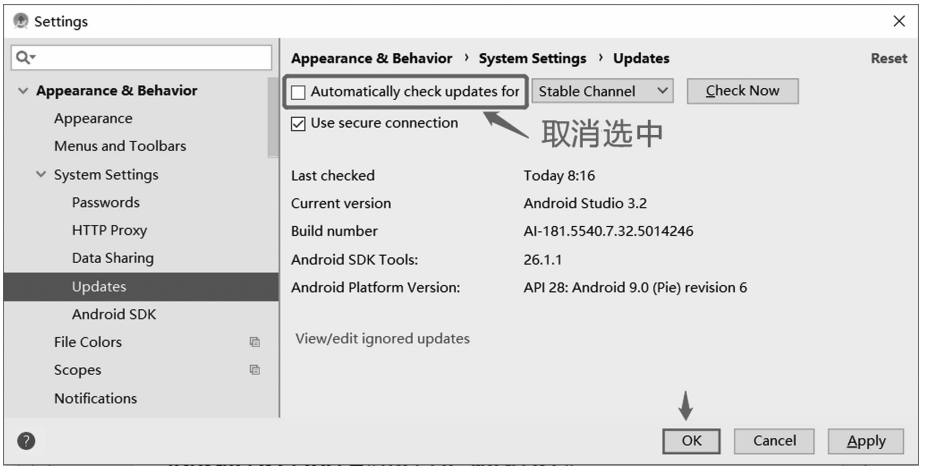


图 1-31　自动检查更新设置

### 5. 禁止自动打开上次的工程

选择“File”→“Settings”→“Appearance&Behavior”→“System Settings”选项，取消选中“Reopen last project startup”复选框，单击“OK”按钮，如图 1-32 所示。

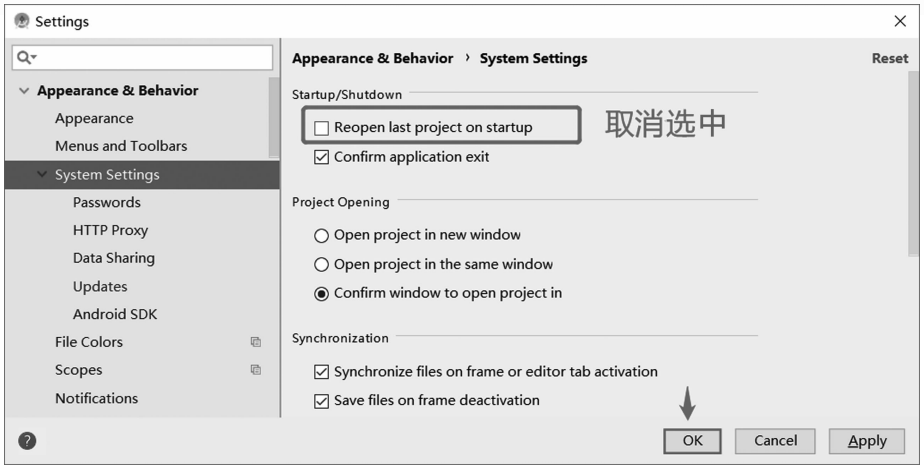


图 1-32　禁止自动打开上次的工程

## 1.4.3　Android 程序结构

Android 工程在创建时，Android Studio 就为其构建了基本结构，设计者可以在此结构上开发应用程序，因此，掌握 Android 程序结构是很有必要的。以 HelloWorld 程序为例，如图 1-33 所示，可以看到一个 Android 程序由多个文件及文件夹组成，这些文件分别用于不同的功能，具体分析如下：

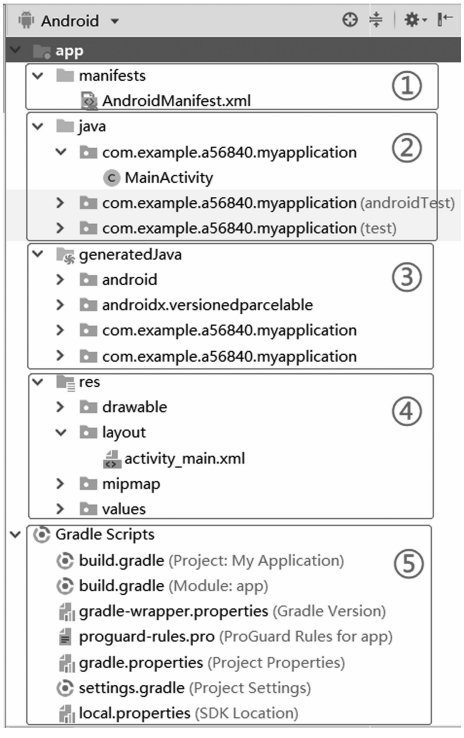


图 1-33　Android 程序结构

(1)manifests。用于存放 AndroidManifest. xml 文件,又称为清单文件,是整个项目的配置文件。在程序中定义的四大组件都需要在这个文件中注册,另外,在该文件中还可以给程序添加权限。在清单文件中配置的信息会添加到 Android 系统中,当程序运行时,系统会找到清单文件中的配置信息,然后根据配置信息打开相应组件。

(2)java。其中包括三个包文件夹,第一个包存放的是 App 工程的 Java 源代码,后面两个包存放测试用的 Java 代码。在该文件夹中可以创建多个包,每个包中可以存放不同的文件或 Activity。

(3)generatedJava。Android Studio 自动生成的,存储一些动态备份内容。

(4)res。用于存放 Android 程序所用到的资源,如图片文件、布局文件、字符串等。

①drawable 目录用于存放图片及 XML 文件;

②layout 目录用于存放布局文件;

③mipmap 目录用于存放应用程序启动图标,系统会根据手机分辨率(hdpi/mdpi/xhdpi/xxhdpi/xxxhdpi)匹配相应大小的图标;

④values 目录用于放置定义的字符串。例如,字符串常量 strings. xml、样式风格定义文件 styles. xml、像素常量 dimens. xml、颜色常量 colors. xml 等。

(5)Gradle Scripts。用于存放项目创建的编译配置文件,一般不需要修改。包括:

①build. gradle:该文件分为项目级与模块级两种,用于描述 App 工程的编译规划;

②proguard-rules. pro:该文件用于描述 Java 文件的代码混淆规划;

③gradle. properties:该文件用于配置编译工程的命令行参数,一般无须改动;

④settings. gradle:配置哪些模块在一起编译;

⑤local. properties:项目的本地配置,一般无须改动,该文件是在工程编译时自动生成的,用于描述开发者本机的环境配置,如 SDK 的本地路径、NDK 的本地路径等。

## 课后提升

**1. 判断题**

(1)Android 是微软公司基于 Linux 平台开发的用于手机和平板电脑的操作系统。 (　　)

(2)Android 系统采用分层架构,由低到高分为四层,依次是应用程序层、应用程序框架层,以及核心类库和 Linux 内核。 (　　)

(3)Android 依赖于 Linux 相应版本的核心系统服务,为 Android 设备的各种硬件提供底层驱动。 (　　)

(4)核心类库是一个核心应用程序的集合,所有安装在手机上的应用程序都属于这一层。 (　　)

(5)应用程序层包含系统库和 Android 运行库。 (　　)

**2. 选择题**

(1)(　　)用于存放 Android 程序所用到的资源,如图片文件、布局文件、字符串等。

A. generatedJava　　B. res

C. manifests　　D. java

(2)(　　)目录用于存放应用程序启动图标,系统会根据手机分辨率(hdpi/mdpi/

xhdpi/xxhdpi/xxxhdpi)匹配相应大小的图标。

A. layout　　B. drawable

C. mipmap　　D. values

(3)项目中一般会出现两个或者多个 build. gradle 文件,一个在根目录下,另一个在(　　)目录下。

A. App　　B. Android

C. Project　　D. Drawable

(4)Google 以 Apache 开源许可证的授权方式发布了 Android 的源代码,(　　)发布第 1 个版本即 Android 1. 1 版本。

A. 2012 年 1 月　　B. 2010 年 6 月

C. 2009 年 10 月　　D. 2008 年 9 月

(5) Google 公司于 2018 年 8 月 7 日发布 Android Studio 3. 2,其 Android 版本是(　　)。

A. Android 10. 0　　B. Android 9. 0

C. Android 8. 0　　D. Android 7. 0

**3. 实操题**

创建一个工程,修改“HelloWorld!”为自己的学校名称,并运行和打包程序,推送到手机进行安装、启动。

模块 2

# Android UI 界面布局

**学习目标**

(1)了解 Android 开发布局的基础知识。
(2)掌握 Android 六大布局的常用属性及使用规则。
(3)能够灵活运用各种布局方式,完成复杂界面布局。

Android 系统为开发人员提供了六大常用布局,以适应不同的界面风格和实际需要。这其中包括:LinearLayout(线性布局)、RelativeLayout(相对布局)、FrameLayout(帧布局)、TableLayout(表格布局)、GridLayout(网格布局)、ConstraintLayout(约束布局)。通过这些布局我们可以实现各种各样的 UI(user interface)界面。本章介绍的内容主要围绕 Android 布局及优化,来探讨在日常开发中的一些常用布局方法和需要注意的一些事项。

## 2.1 认识布局

布局,顾名思义就是对页面的文字、图形或表格进行排布、设计。优秀的布局需要对页面信息进行完整的考虑。既要考虑用户需求、用户行为,也要考虑信息发布者的目的、目标。为了完美呈现 UI 设计师的设计效果,Android 开发的任务就变得非常艰巨。

在 Android 开发中 UI 布局常用的有两种设计方式,一种是利用可视化工具来进行,允许通过拖曳控件来进行布局;另一种是通过编写 XML 文档来进行布局,这两种方法可以随时相互转换。另外,用户所能见到的所有界面布局,都是使用 View 和 ViewGroup 对象建立的。

### 2.1.1 视图 View

在 Android 中 View 类是最基本的一个 UI 类,所有控件和布局都是直接或者间接由这个基本视图派生而来,也就是说 View 是所有 UI 组件的基类。一个 View 在屏幕上占据了

一块矩形区域，它负责渲染这块矩形区域。View 类的基本属性和方法也是各控件和布局通用的属性，在 XML 布局文件中常用的属性见表 2-1。

表 2-1　View 常用属性定义说明

| 属性名称 | 定义说明 |
| --- | --- |
| id | 唯一地标识该视图的 ID |
| layout_width | 指定该视图的宽度，取值可以是具体的数值，单位是 dp；可以是 match_parent，表示与其父窗体同宽；也可以是 wrap_content，表示与内部的内容同宽 |
| layout_height | 指定该视图的高度，取值与 layout_width 相同，其中，match_parent 表示与其父窗体同高 |
| margin | 指定该视图与周围视图之间的间距，通常称为外边距。可以通过 marginTop、marginBottom、marginLeft、marginRight 分别定义该视图的上、下、左、右外边距 |
| padding | 指定该视图边缘与内部内容之间的空白间距，通常称为内边距。可以通过 paddingTop、paddingBottom、paddingLeft、paddingRight 分别指定其上、下、左、右内边距 |
| background | 指定该视图的背景颜色，也可以指定其背景图片 |
| gravity | 用来控制元素在该控件里的显示位置，也就是对齐方式 |
| visibility | 设置控件的可见性 |

### 2.1.2　视图组 ViewGroup

视图组 ViewGroup 是容纳 View 及其派生类的容器，ViewGroup 也是从 View 派生出来的，负责对添加进 ViewGroup 的这些 View 进行布局和管理。一般来说，开发 UI 界面很少直接用 View 和 ViewGroup 来写布局，更多的时候是使用它们的子类控件或容器来构建布局。

ViewGroup 有 3 个方法，这也是所有布局类视图共有的方法。

(1)addView：添加一个视图到布局中；

(2)removeView：从布局中删除指定的视图；

(3)removeAllViews：删除该布局下的所有视图。

### 2.1.3　布局原则

如何正确、高效地使用这些布局方式来组织 UI 控件，是构建优秀 Android App 的主要前提之一。通过一些惯用、有效的布局原则，可以制作出加载效率高且复用性高的 UI。在 Android UI 布局过程中，需要遵守的原则包括如下几点：

(1)掌握 UI 布局属性及其定义的方法；

(2)尽量多使用相对布局、约束布局等，尽量少使用布局嵌套；

(3)将可复用的组件抽取出来并通过＜include/＞标签进行使用；

(4)使用＜ViewStub/＞标签来加载一些不常用的布局；

(5)使用＜merge/＞标签减少布局的嵌套层次；

(6)布局使用的单位，文字的尺寸在律用 sp(抽象像素)，非文字的尺寸一律使用 dp(与像素密度有关的相对单位)，px(像素)只在产生一条细的分割线时使用。

## 2.2 线性布局 LinearLayout

线性布局通常被用于水平方向和垂直方向的组件布局，是一种常用的布局管理器，也会通过布局的嵌套使用来布局一些相对复杂的界面。

### 2.2.1 什么是线性布局

线性布局(LinearLayout)在实际开发中比较常用，主要以水平和垂直方式来控制界面中各组件的显示位置。当组件以水平方式进行排列时，显示顺序依次为从左到右；当组件以垂直方式进行排列时，显示顺序依次为从上到下。

在 Android 中，使用 XML 布局文件对线性布局进行定义，其布局节点定义时需要使用<LinearLayout>对布局进行标记，其基本语法格式为：

```
<LinearLayout
    属性列表
>
</LinearLayout>
```

### 2.2.2 线性布局的常用属性

LinearLayout 除了继承 View/ViewGroup 类的所有属性和方法外，还有其特有的 XML 属性，借助这些属性来实现界面的组织和布局。

**1. orientation 属性**

方向，用于设置布局管理器内组件的排列方式，其值为 horizontal 表示水平布局，所有控件水平排列，顺序为从左到右，依次排列，一行排满后，继续添加的元素将不会被显示出来。值为 vertical 表示垂直布局，顺序为从上到下依次排列，一列排满后，继续添加的元素也不会被显示出来。如果不指定该属性，则系统默认为 horizontal 水平布局。

**2. gravity 属性**

重力，用于指定线性布局内部视图的对齐方式。常用属性值有 top、bottom、left、right 设置内部控件居顶、底、左、右布局；center、center_vertical、center_horizontal 设置控件居于中心、垂直居中和水平居中。这个属性也可以在 RelativeLayout、TableLayout 布局中使用。但这里要与 layout_gravity 区分开，layout_gravity 是用来设置自身相对于父元素的布局。

**3. layout_weight 属性**

权重，用来分配当前控件在剩余空间的大小。使用权重一般要把分配该权重方向的长度设置为零。例如，在水平方向分配权重，就把 width 设置为零；在垂直方向分配权重，就把 height 设置为零。

**4. background 属性**

背景，用于为组件进行背景的设置，可以设置为图片或颜色。设置背景图片时，需要先将背景图片复制到 Android 的图片目录 res/drawable 下，假设图片名称为 beijing. png，设置

背景图片的方法如下：

```
android:background="@drawable/beijing"
```

如果要将颜色指定为背景，需要用六位十六进制数来表示颜色值，并在颜色值前加“#”。例如，将背景颜色设置为红色，代码可写成：

```
android:background="#ff0000"
```

### 5. divider 属性

分割线，用于为 LinearLayout 设置分割线，从而使得界面更加整洁美观。divider 指定分割线图片，通过 showDividers 来设置分割线所在的位置，有 4 个可选值：none（无）、beginning（开始）、end（结束）、middle（每两个组件间），通过 dividerPadding 可以设置分割线左右两端的间距。使用方式如下：

```
android:divider="@drawable/line"      //指定图片 line.png 作为分割线
android:showDividers="middle"         //指定在每两个组件之间加入分割线
android:dividerPadding="10dp"         //指定分割线两端 Padding 为 10dp
```

## 2.2.3　线性布局应用

新建 XML 布局文件，使用线性布局定义根节点，利用 Button（按钮）控件参与页面布局，应用线性布局的相关属性，达到页面布局的效果。

**【例 2-1】** 使用 orientation 属性与 gravity 属性，完成如图 2-1 所示界面布局效果，即水平排列且居中对齐。实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"          //设置当前布局宽度为填充父窗体
    android:layout_height="match_parent"         //设置当前布局高度为填充父窗体
    android:orientation="horizontal"             //设置方向为水平
    android:gravity="center_horizontal">         //设置对齐方式为水平居中
    <Button
        android:layout_width="wrap_content"      //设置该按钮宽度为包裹内容
        android:layout_height="wrap_content"     //设置该按钮高度为包裹内容
        android:text="控件 1"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="控件 2"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="控件 3"/>
</LinearLayout>
```

在上述代码中，修改 orientation="vertical"，则按钮将以垂直方向进行排列，界面布局效果如图 2-2 所示。若修改代码如下，则界面布局效果如图 2-3 所示。

```
android:orientation="vertical"      //设置方向为垂直
android:gravity="center"            //设置对齐方式为中心对齐，即水平与垂直都居中
```

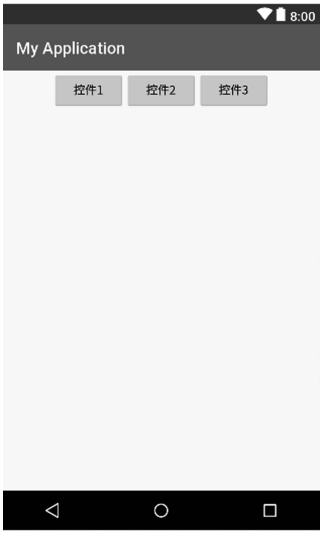


图 2-1　水平排列水平居中

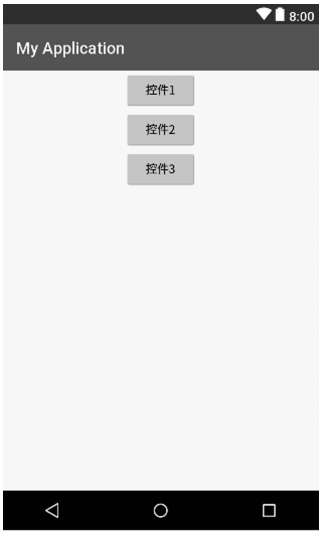


图 2-2　垂直排列水平居中

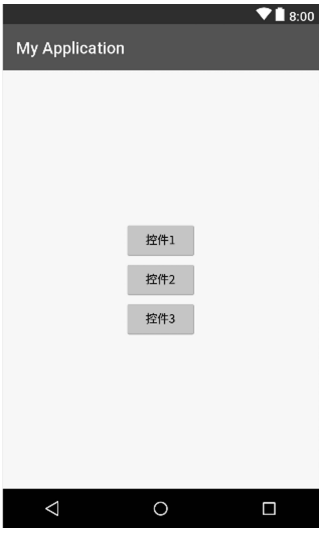


图 2-3　垂直中心

【例 2-2】 使用 weight 属性，控制控件与父窗体宽度或高度的分配比例。制作布局内部控件宽度平均分配的效果，完成界面布局如图 2-4 所示，3 个 Button 控件宽度为平分父窗体的宽度，每个控件各占总宽度的 1/3。实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <Button
        android:layout_width="0dp"          //设置控件宽度为 0dp
        android:layout_height="wrap_content"
        android:text="控件 1"
        android:layout_weight="1"/>         //设置权重为 1
    <Button
        android:layout_width="0dp"          //设置控件宽度为 0dp
        android:layout_height="wrap_content"
        android:text="控件 2"
```

```
        android:layout_weight="1"/>          //设置权重为 1
    <Button
        android:layout_width="0dp"           //设置控件宽度为 0dp
        android:layout_height="wrap_content"
        android:text="控件 3"
        android:layout_weight="1"/>          //设置权重为 1
</LinearLayout>
```

**【例 2-3】** 竖直方向类推，只需设置 android:height 为 0dp，然后设置 weight 属性即可。如图 2-5 所示，分别为 3 个 Button 控件设置 weight 值为 1、1、2，可认为高度总值为 4，其中各控件高度分别占 1/4、1/4 和 1/2。

实现图 2-5 所示效果，程序源码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center_horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"          //设置控件高度为 0dp
        android:text="控件 1"
        android:layout_weight="1"/>          //设置权重为 1
    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"          //设置控件高度为 0dp
        android:text="控件 2"
        android:layout_weight="1"/>          //设置权重为 1
    <Button
        android:layout_width="wrap_content"
        android:layout_height="0dp"          //设置控件高度为 0dp
        android:text="控件 3"
        android:layout_weight="2"/>          //设置权重为 2
</LinearLayout>
```

在上述代码中，“控件 1”的高度被设置为 wrap_content，且不为其设置 weight 属性，则得到的界面效果如图 2-6 所示。

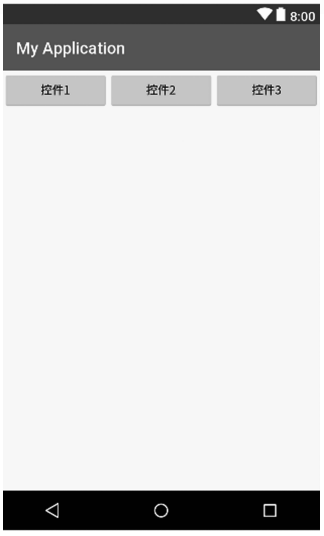


图 2-4　平均分配宽度


图 2-5　按比例分配高度

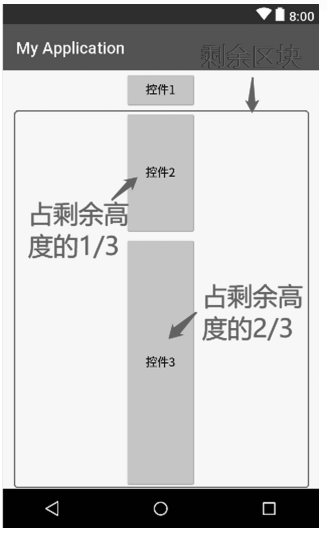


图 2-6　按比例分配剩余高度

## 2.3　相对布局 RelativeLayout

相对布局也是一种常用的布局。与 LinearLayout 的排列规则不同，RelativeLayout 显得更加灵活一些。也正因为如此，RelativeLayout 中的属性非常多，不过这些属性都有规律可循，也容易理解和记忆。

### 2.3.1　什么是相对布局

相对布局(RelativeLayout)是一种通过相对定位的方式进行界面布局的管理器，其定位方式可以是相对于父容器，也可以是相对于兄弟组件。通过一些属性联合应用，可以控制组件灵活地布置在界面的任何位置，实现相对比较复杂的布局。

确定使用相对布局，布局节点定义时需要使用＜RelativeLayout＞对布局进行标记，其基本语法格式为：

```
＜RelativeLayout
    属性列表
＞
＜/RelativeLayout＞
```

### 2.3.2　相对布局的常用属性

RelativeLayout 同样继承了 View/ViewGroup 类的所有属性和方法，同时又具有自身特有的属性。当控件被加入时，默认位于当前布局的左上角，需要借助各种相对布局属性确定其位置。

### 1. 相对父容器的属性

以组件的父容器作为参照对象的属性，当该组件的属性值被设置为 true 时，该组件就会位于指定的位置。相对于父容器的属性，见表 2-2。

表 2-2　相对于父容器的属性

| 属 性 名 称 | 说　　明 |
|---|---|
| alighParentLeft | 位于父容器的左侧，左对齐 |
| alighParentRight | 位于父容器的右侧，右对齐 |
| alighParentTop | 位于父容器的顶部，顶端对齐 |
| alighParentBottom | 位于父容器的底部，底部对齐 |
| centerHorizontal | 相对于父容器水平方向居中，水平居中 |
| centerVertical | 相对于父容器垂直方向居中，垂直居中 |
| centerInParent | 位于父容器的中央位置 |

### 2. 相对兄弟组件的属性

相对兄弟组件的属性就是以组件的其他兄弟组件作为参照对象的属性，所谓兄弟组件就是处于同一层次容器的组件。在设置属性值时，需要明确指定相对于哪个组件，所以应用这些属性的前提是其他兄弟组件要事先设置 ID。相对于兄弟组件的属性，见表 2-3。

表 2-3　相对于兄弟组件的属性

| 属 性 名 称 | 说　　明 |
|---|---|
| toLeftOf | 位于某组件的左侧 |
| toRightOf | 位于某组件的右侧 |
| above | 位于某组件的上方 |
| below | 位于某组件的下方 |
| alignTop | 与指定的组件的上边界对齐 |
| alignBottom | 与指定的组件的下边界对齐 |
| alignLeft | 与指定的组件的左边界对齐 |
| alignRight | 与指定的组件的右边界对齐 |

## 2.3.3　相对布局的应用

新建 XML 布局文件，使用相对布局定义根节点，利用 Button（按钮）控件参与页面布局，应用相对布局的相关属性，达到页面布局的效果。

**【例 2-4】**　使用相对父容器的属性，制作如图 2-7 所示的布局效果。实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="#d9f7a8"                    //设置父窗的背景颜色
    android:padding="10dp">                         //设置父窗体的内边距为 10dp
    <!--组件 1 默认位于父窗体的左上角-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="组件 1" />
    <!--设置组件 2 位于父窗体的右上角-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"      //设置位于父窗体的右侧
        android:text="组件 2" />
    <!--设置组件 3 位于父窗体的左下角-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"     //设置位于父窗体的底部
        android:text="组件 3" />
    <!--设置组件 4 位于父窗体的右下角-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:text="组件 4" />
    <!--设置组件 5 位于父窗体的左侧居中-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"        //设置在父窗体中垂直居中
        android:text="组件 5" />
    <!--设置组件 6 位于父窗体的左侧居中-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"      //设置在父窗体中水平居中
        android:text="组件 6" />
    <!--设置组件 7 位于父窗体的右侧居中-->
    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="组件 7"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"/>
    <!--设置组件 8 位于父窗体的底部居中-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="组件 2" />
    <!--设置组件 9 位于父窗体的中心-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"      //设置在父窗体的中心位置
        android:text="组件 9" />
</RelativeLayout>
```

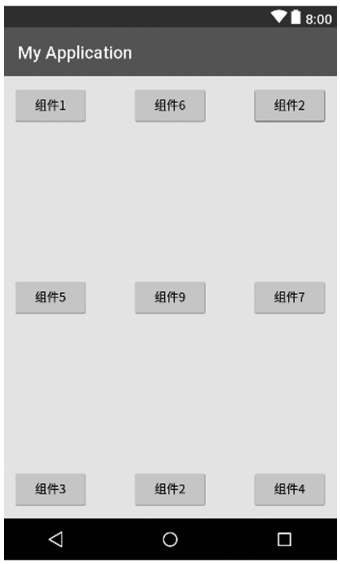


图 2-7　相对父容器的属性

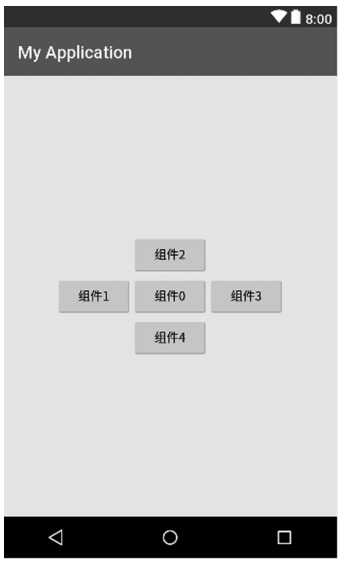


图 2-8　相对兄弟组件的属性

【例 2-5】　使用相对兄弟组件的属性，制作如图 2-8 所示的布局效果。实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#d9f7a8">
```

```
    <!--设置组件 0 位于父窗体的中心-->
    <Button
        android:id="@+id/but"                          //为组件 0 设置 ID 为 but
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="组件 0" />
    <!--设置组件 1 位于组件 0 的左侧-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/but"              //设置当前组件在 but 左侧
        android:layout_alignBottom="@id/but"           //设置当前组件与 but 底部对齐
        android:text="组件 1" />
    <!--设置组件 2 位于组件 0 的上方-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/but"                 //设置当前组件在 but 上方
        android:layout_alignRight="@id/but"            //设置当前组件与 but 右侧对齐
        android:text="组件 2" />
    <!--设置组件 3 位于组件 0 的右侧-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/but"             //设置当前组件在 but 右侧
        android:layout_alignTop="@id/but"              //设置当前组件与 but 顶部对齐
        android:text="组件 3" />
    <!--设置组件 4 位于组件 0 的下方-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/but"                 //设置当前组件在 but 下方
        android:layout_alignLeft="@id/but"             //设置当前组件与 but 左侧对齐
        android:text="组件 4" />
</RelativeLayout>
```

## 2.4 表格布局 TableLayout

表格布局经常被应用在一些规则布局中，在很多的输出操作中，往往会使用表格的形式对显示的数据进行排版。在本节中，我们将详细说明表格布局的属性，及如何运用这些属性。

### 2.4.1　什么是表格布局

表格布局(TableLayout)继承自 LinearLayout。Tablelayout 类以行和列的形式对控件进行管理,通过 TableRow 设置行,有多少个 TableRow 对象,就有多少行。表格的列数是由 TableRow 中的子控件决定的,即所有行中含有的最多子控件数就是表格的列数。例如,第一个 TableRow 含 2 个子控件,第二个 TableRow 含 3 个,第三个 TableRow 含 4 个,那么该 TableLayout 的列数为 4 列,即该表格为 3 行 4 列。如果直接在 TableLayout 中添加子控件,那么子控件会占据整个一行。

确定使用表格布局,布局节点定义时需要使用<TableLayout>对布局进行标记,其基本语法格式为:

```
<TableLayout
    属性列表
>
<TableRow>添加组件并定义组件属性</TableRow>
.....
</TableLayout>
```

### 2.4.2　表格布局的属性

TableLayout 表格并没有边框,由多个 TableRow 对象组成,每个 TableRow 可以有 0 个或多个单元格,每个单元格就是一个 View。这些 TableRow 单元格不能设置 layout_width,宽度默认为 fill_parent,只有高度 layout_height 可以自定义,默认是 wrap_content。TableLayout 可设置的属性包括全局属性及单元格属性。

#### 1. TableLayout 全局属性

TableLayout 全局属性,即列属性,是用在<TableLayout>标签中的,用于控制 TableLayout 中指定的列,共有 3 个常用属性,列号都是从 0 开始索引编号,同时设置多列时,用逗号隔开指定的列号,如果是所有列都生效,则用“*”号表示,见表 2-4。

表 2-4　TableLayout 全局属性

| 属性名称 | 说　明 |
| --- | --- |
| stretchColumns | 设置可伸展的列。该列可以向行方向伸展,最多可占据一整行,如 android:stretchColumns="0"指定第 0 列可伸展 |
| shrinkColumns | 设置可收缩的列。当该列子控件的内容太多,已经挤满所在行时,该子控件的内容将往列方向显示。例如,android:shrinkColumns="2,3"指定第 2、3 列都可以收缩 |
| collapseColumns | 设置要隐藏的列,如 android:collapseColumns="*" 隐藏所有列 |

#### 2. TableLayout 单元格属性

TableLayout 单元格属性用于设置单元格属性,分别是跳格子及合并单元格,是应用在

<TableRow>标签上的属性，共有 2 个属性，见表 2-5。

表 2-5　TableLayout 单元格属性

| 属 性 名 称 | 说　　明 |
| --- | --- |
| layout_column | 指定该单元格在第几列显示。例如，android:layout_column="2"，该控件显示在第 3 列 |
| layout_span | 指定该单元格占据的列数，未指定时，为 1。例如，android:layout_span="2"，该控件占据 2 列，合并两个列 |

### 2.4.3　TableLayout 的使用

【例 2-6】 TableLayout 常用属性练习。新建 XML 布局文件，使用 TableLayout 定义根节点，利用 Button(按钮)控件参与页面布局，应用表格布局的相关属性，实现页面布局的效果，如图 2-9 所示。实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#77b2f5"                    //设置背景颜色
    android:stretchColumns="0,1,2,3">               //设置 4 列全部拉伸
    <TableRow>                                      //定义行的头标记，行内有以下 4 个按钮
        <Button android:text="第 0 列" />
        <Button android:text="第 1 列" />
        <Button android:text="第 2 列" />
        <Button android:text="第 3 列" />
    </TableRow>                                     //定义行的结束标记
    <TableRow>                                      //继续定义下一行
        <Button
            android:layout_column="1"               //定义当前按钮跳过第 0 列，占据第 1 列
            android:text="第 1 列" />
        <Button
            android:layout_column="3"               //定义按钮位于索引位置第 3 列
            android:text="第 3 列" />
    </TableRow>
    <TableRow>
        <Button
            android:text="跨前 2 列"
            android:layout_span="2"/>               //设置合并 2 列
        <Button
            android:text="跨后 2 列"
```

```
            android:layout_span="2"/>              //设置合并 2 列
    </TableRow>
    <TableRow>
        <Button
            android:text="横跨 4 列"
            android:layout_span="4"/>              //设置合并 4 列
    </TableRow>
    <TableRow>
        <Button
            android:text="占中间 2 列"
            android:layout_span="2"                //设置合并 2 列
            android:layout_column="1"/>            //设置位于第 1 列
    </TableRow>
</TableLayout>
```

【例 2-7】 仿制计算器界面。利用表格布局的相关属性，实现如图 2-10 所示的计算器界面，实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#77b2f5"
    android:stretchColumns="0,1,2,3">
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="90dp"      //设置按钮高度为固定值 90dp
            android:layout_span="4"           //设置表格当前行 4 列合并
            android:text=" " />
    </TableRow>
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="C" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:layout_span="2"
            android:text="DEL" />
        <Button
            android:layout_width="wrap_content"
```

```
            android:layout_height="80dp"
            android:text="/" />
    </TableRow>
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="7" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="8" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="9" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="*" />
    </TableRow>
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="4" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="5" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="6" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="+" />
```

```
        </TableRow>
        <TableRow>
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="1" />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="2" />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="3" />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="-" />
        </TableRow>
        <TableRow>
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:layout_span="2"
                android:text="0" />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="." />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="80dp"
                android:text="=" />
        </TableRow>
    </TableLayout>
```

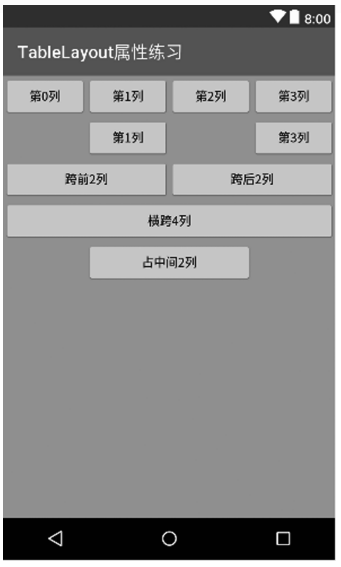


图 2-9　TableLayout 基本属性应用


图 2-10　仿制计算器界面

## 2.5　网格布局 GridLayout

GridLayout 布局是 Android 4.0 以后引入的新布局，和 TableLayout(表格布局)非常相似，但它的功能更多，也更加好用，实现了控件的交错显示，能够避免因布局嵌套对设备性能的影响，更利于自由布局的开发。

### 2.5.1　什么是网格布局

网格布局(GridLayout)使用虚细线将布局划分为行、列和单元格，使容器中的各组件呈 $M$ 行×$N$ 列的网格状分布，也支持一个组件在行、列上都有交错的排列。GridLayout 的布局策略简单分为以下三个部分：

首先，它与 LinearLayout 布局一样，也分为水平和垂直两种方式，默认是水平布局。容器内的组件一个接着一个，依次从左到右或从上到下排列，通过指定 columnCount 属性设置列数，控制自动换行进行排列。另一方面，对于 GridLayout 布局中的子控件，默认按照 wrap_content 的方式设置其显示，这只需在 GridLayout 布局中进行显示声明即可。

其次，若要指定某控件显示在固定的行或列，只需设置该子控件的 layout_row 和 layout_column 属性。需要注意的是，android:layout_row="0"表示从第一行开始，android:layout_column="0"表示从第一列开始，这与编程语言中一维数组的赋值情况类似。

最后，可以设置某控件跨越多行或多列，需要 rowSpan 或者 columnSpan 属性的参与，再设置其 layout_gravity 属性为 fill 即可，前一个设置表明该控件跨越的行数或列数，后一个设置表明该控件填满所跨越的整行或整列。

使用网格布局时，节点定义需要使用<GridLayout>对布局进行标记，其基本语法格式与前面的布局基本相同，这里不再赘述。

## 2.5.2　网格布局的属性

GridLayout 继承了 LinearLayout 的排列对齐属性，由 orientation 属性来设置组件的排列方式，由 gravity 属性来指定组件的对齐方式。除此之外，GridLayout 还有设置行列数的属性、设置组件所在行列的属性及设置组件横跨几行（几列）的属性。通过这些属性可以轻松地实现相对复杂的界面布局。

### 1. GridLayout 属性

作用于 GridLayout 自身的属性，用于对网格整体进行设置，可定义网格最大行数或者最大列数，见表 2-6。

表 2-6　GridLayout 自身属性

| 属性名称 | 说　明 |
| --- | --- |
| rowCount | 设置网格布局的最大行数。例如，android:rowCount="3"，设置网格最多有 3 行 |
| columnCount | 设置网格布局的最大列数。例如，android:columnCount="3"，设置网格最多有 3 列 |
| orientation | 网格中控件的排列方向，取值可以是 horizontal（水平）和 vertical（垂直） |

### 2. GridLayout 子控件属性

作用于 GridLayout 子控件的属性，用于设置当前控件位于哪一行、哪一列，或者跨越行列等，见表 2-7。

表 2-7　GridLayout 子控件属性

| 属性名称 | 说　明 |
| --- | --- |
| layout_row | 设置当前组件位于网格的哪一行。例如，android:layout_row="2"，该组件实际位于网格的第 3 行（索引编号从 0 开始） |
| layout_column | 设置当前组件位于网格的哪一列。例如，android:layout_column="3"，该组件实际位于网格的第 4 列（索引编号从 0 开始） |
| layout_rowSpan | 纵向跨越的行数。例如，android:layout_rowSpan="2"，跨 2 行。需要同时设置 layout_gravity="fill_vertical"，自动充满所跨行 |
| layout_columnSpan | 横向跨越的列数。例如，android:layout_columnSpan="3"，跨 3 列。需要同时设置 layout_gravity="fill_horizontal"，自动充满所跨列 |
| columnWeight | 该控件的列权重，与线性布局中 android:layout_weight 类似，使用方法如：android:layout_columnWeight="1" |
| rowWeight | 该控件的行权重，原理同 android:layout_columnWeight |

## 2.5.3　网格布局的应用

利用网格布局 GridLayout 来实现一个简单的计算器界面布局，效果如图 2-11 所示。可以实现在表格布局下无法实现的效果，设计源码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4">                          //指定网格最大列数为 4
    <Button
        android:text=" "
        android:layout_columnSpan="4"                 //当前 Button 跨越 4 列
        android:layout_gravity="fill_horizontal"      //水平填充所跨越的 4 列
        android:layout_columnWeight="4"               //列权重为 4
        android:layout_rowWeight="1"/>                //行权重为 1
    <Button
        android:text="="
        android:layout_columnWeight="1"               //列权重为 1
        android:layout_rowWeight="1"/>                //行权重为 1
    <Button
        android:text="DEL"
        android:layout_columnSpan="2"                 //当前 Button 跨越 2 列
        android:layout_gravity="fill_horizontal"      //水平填充所跨越的 2 列
        android:layout_columnWeight="2"               //列权重为 2
        android:layout_rowWeight="1"/>
    <Button
        android:text="/"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="7"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="8"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="9"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="*"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
```

```
        android:text="4"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="5"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="6"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="+"
        android:layout_rowSpan="2"                    //纵向跨越 2 行
        android:layout_gravity="fill_vertical"        //垂直填充所跨越的 2 行
        android:layout_columnWeight="1"               //列权重为 1
        android:layout_rowWeight="2"/>                //行权重为 2

    <Button
        android:text="1"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="2"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="3"
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="0"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"
        android:layout_columnWeight="2"
        android:layout_rowWeight="1"/>
    <Button
        android:text="."
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
    <Button
        android:text="—"
```

```
        android:layout_columnWeight="1"
        android:layout_rowWeight="1"/>
</GridLayout>
```


图 2-11　GridLayout 实现计算器效果

## 2.6　帧布局 FrameLayout

帧布局(FrameLayout)可以说是六大布局中最简单的一个布局，它的使用范围相对来说也比较小，应用场景也不多。但帧布局在游戏开发方面用得比较多，会使用 FrameLayout 来动态加载碎片(fragment)，完成一些特定的效果。

### 2.6.1　什么是帧布局

帧布局(FrameLayout)直接继承 ViewGroup 组件，每加入一个组件帧布局容器就创建一个空白区域，每个组件占据一帧，添加的组件是一个一个叠在一起的，所有子控件都会按照创建的先后顺序在屏幕的左上角重叠显示，后面元素都直接覆盖在前面元素之上。

帧布局使用<FrameLayout>标签对布局进行标记，其基本语法格式与前面布局基本相同，这里不再赘述。

### 2.6.2　帧布局的属性

FrameLayout 同样继承 LinerLayout 类，所以拥有 LinerLayout 的 XML 属性。同时它也有自己独特的属性，见表 2-8。

表 2-8　FrameLayout 属性

| 属 性 名 称 | 说　　明 |
| --- | --- |
| android:foreground | 设置该帧布局容器的前景图像。所谓前景图像是永远处于帧布局最顶的、直接面对用户不会被覆盖的图像 |
| android:foregroundGravity | 指定前景图像的对齐方式，取值与 gravity 属性相同 |

### 2.6.3　帧布局的应用

FrameLayout 属性比较简单，我们通过一个简单的示例来说明 FrameLayout 及其属性的基本用法。本例中引入 TextView 控件，并且需要先将使用到的图片复制到 res/drawable 文件夹下。完成效果如图 2-12 所示，实例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
//根节点定义为 FrameLayout 帧布局
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foreground="@drawable/qq2"              //设置该帧布局容器的前景图像
    android:foregroundGravity="center"              //前景图像位于布局中心位置
    android:background="#e9f9fa">                   //设置 FrameLayout 背景颜色
    <TextView
        android:layout_width="380dp"
        android:layout_height="380dp"
        android:layout_gravity="center"             //当前 TextView 控件位于布局中心
        android:background="#44ffff" />             //设置 TextView 背景颜色
    <TextView
        android:layout_width="320dp"
        android:layout_height="320dp"
        android:layout_gravity="center"
        android:background="#44ccfff" />
    <TextView
        android:layout_width="260dp"
        android:layout_height="260dp"
        android:layout_gravity="center"
        android:background="#4499ff" />
    <TextView
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:layout_gravity="center"
        android:background="#4455ff" />
    <TextView
        android:layout_width="120dp"
```

```
        android:layout_height="120dp"
        android:layout_gravity="center"
        android:background="#0033aa" />
</FrameLayout>
```

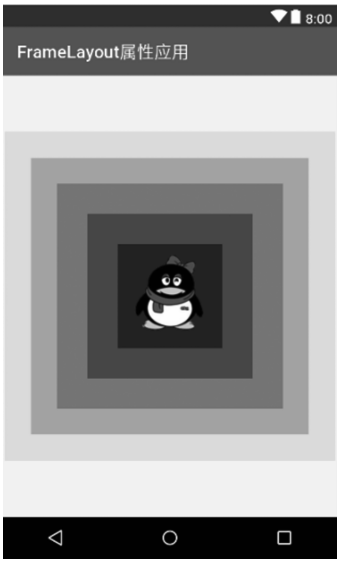


图 2-12　FrameLayout 属性应用

## 2.7　约束布局 ConstraintLayout

约束布局(ConstraintLayout)是一个 ViewGroup,可以在 API 9 以上的 Android 系统中使用。它的出现主要是为了解决布局嵌套过多的问题,以灵活的方式定位和调整小部件。从 Android Studio 2.3 起,官方的模板默认使用 ConstraintLayout。

### 2.7.1　什么是约束布局

约束布局(ConstraintLayout)是使用约束的方式来指定各个控件的位置和关系的一种布局模式。ConstraintLayout 有些类似于 RelativeLayout,但远比 RelativeLayout 强大,它可以按照比例约束控件位置和尺寸,能够更好地适配屏幕大小及不同的机型。

另外,ConstraintLayout 非常适合使用可视化的方式来编写界面,它的用法是直接对控件进行拖曳,通过指定控件之间或控件与父容器之间的位置关系来进行定位,由 Android Studio 自动生成 XML 代码。

约束布局使用<android.support.constraint.ConstraintLayout>标签对布局进行标记,其基本语法格式与前面布局基本相同,这里不再赘述。

### 2.7.2　约束布局的属性

ConstraintLayout 属性很多,但大部分与 RelativeLayout 属性比较相近。在界面绘制过

程中，这些属性的灵活运用可以让 ConstraintLayout 性能优于 RelativeLayout 达 40%，层次更扁平，构建复杂布局更容易。

ConstraintLayout 非常适合使用可视化的方式来编写界面，并不太适合使用 XML 的方式来进行编写。但可视化操作仍然是使用 XML 代码来实现的，Android Studio 会根据操作结果自动生成相应的代码。为了便于记忆和掌握约束布局的属性，下面按类型分别进行介绍。

### 1. 尺寸约束(Dimensions constraints)属性

当 ContraintLayout 尺寸设置为 wrap_content 时，默认使用最小尺寸；当设置为 0dp 时，相当于 match_constraint。用于定义约束布局的最小、最大尺寸的属性有以下 4 个：

(1)android_maxHeight：控制 View 最大高度。

(2)android_maxWidth：控制 View 最大宽度。

(3)android_minHeight：控制 View 最小高度。

(4)android_minWidth：控制 View 最小宽度。

### 2. 引导线(Guideline)的属性

作用在 Guideline 上，用来辅助定位的一个不可见的元素，包括以下属性：

(1)android_orientation：控制 Guideline 是横向的还是纵向的。

(2)layout_constraintGuide_begin：控制 Guideline 距离父容器开始的距离。

(3)layout_constraintGuide_end：控制 Guideline 距离父容器末尾的距离。

(4)layout_constraintGuide_percent：距离 left 或 top 的比例，float 类型。

### 3. 相对定位(Relative positioning)属性

用来控制子 View 相对位置，这些属性和 RelativeLayout 的布局属性非常类似，用来控制子 View 的某一个属性相对于另外一个 View 或者父容器的位置，包括以下属性：

(1)constraintLeft_toLeftOf：该控件的左边相对于某控件或父布局的左边对齐。

(2)constraintLeft_toRightOf：该控件的左边相对于某控件或父布局的右边对齐。

(3)constraintRight_toLeftOf：该控件的右边相对于某控件或父布局的左边对齐。

(4)constraintRight_toRightOf：该控件的右边相对于某控件或父布局的右边对齐。

(5)constraintTop_toTopOf：该控件的顶边相对于某控件或父布局的顶边对齐。

(6)constraintTop_toBottomOf：该控件的顶边相对于某控件或父布局的底边对齐。

(7)constraintBottom_toTopOf：该控件的底边相对于某控件或父布局的顶边对齐。

(8)constraintBottom_toBottomOf：该控件的底边相对于某控件或父布局的底边对齐。

(9)constraintBaseline_toBaselineOf：该控件水平基准线相对于某控件或父布局的水平基准线对齐。

(10)constraintStart_toStartOf：该控件开始位置相对于某控件或父布局开始位置对齐。

(11)constraintStart_toEndOf：该控件开始位置相对于某控件或父布局的结束位置对齐。

(12)constraintEnd_toStartOf：该控件结束位置相对于某控件或父布局的开始位置对齐。

(13)constraintEnd_toEndOf：该控件的结束位置相对于某控件或父布局的结束位置对齐。

#### 4. 边距(Margin)

Margin 是很常见的属性,前面也有介绍,约束布局同样适用。这里介绍的是当 View 状态为 GONE(不可见,且不保留位置)时,所应用的边距属性。例如,A 控件左边相对于父容器左边对齐,B 控件左边相对于 A 控件右边对齐。如果 A 控件的状态为 GONE,则 B 控件就相对于父容器左边对齐了。当 A 控件不可见时,如果 B 控件和父容器左边需要有一个 20dp 的边距,就需要使用 goneMarginLeft 或者 goneMarginStart 属性了,如果设置了这个属性,当 B 控件所参考的 A 控件可见时,这个边距属性不起作用;当 A 控件不可见(GONE)时,这个边距就在 B 控件上起作用了。

(1)goneMarginBottom:底部外边距。

(2)goneMarginEnd:某控件结束后的外边距。

(3)goneMarginLeft:左外边距。

(4)goneMarginRight:右外边距。

(5)goneMarginStart:某控件开始位置的外边距。

(6)goneMarginTop:顶部外边距。

#### 5. 角度定位(Circular positioning)

角度定位指的是以两个控件的中心为准,用一个角度和一个距离来约束两个控件的位置关系,包括 3 个属性:

(1)constraintCircle:指定参照控件的 ID。

(2)constraintCircleAngle:指定该控件中心到其参照控件中心的距离。

(3)constraintCircleRadius:与参照控件的角度,取值为 0～360°,0 为正上方,90°为正右方,180°为正下方,270°为正左方。

#### 6. 比率与偏移率

比率用于定义一个控件相对于另一个控件的尺寸比例,通过浮点值(float)或宽高比例(width:height)来设置。百分比布局意义也非常重要,常用来解决碎片化问题。

(1)constraintVertical_bias:垂直偏移率。

(2)constraintHorizontal_bias:水平偏移率。

(3)constraintHeight_percent:高度百分比,占父类高度的百分比。

(4)constraintWidth_percent:宽度百分比,占父类宽度的百分比。

(5)constraintDimensionRatio:设置宽高比。

### 2.7.3 约束布局的应用

约束布局的可视化操作为界面编写提供了很大的便利条件,在实际开发过程中,设计人员通常利用可视化操作和 XML 代码编写的方式,来完成界面的编排。为了更好地熟悉 ConstraintLayout 属性,在这里使用代码的方式来实现示例效果。

**【例 2-8】** 以角度和半径距离约束控件位置。如图 2-13 所示,按钮 A 位于父容器中心,其他按钮在以按钮 A 为中心,半径 100dp,角度为 45°以及 45°的倍数的位置。

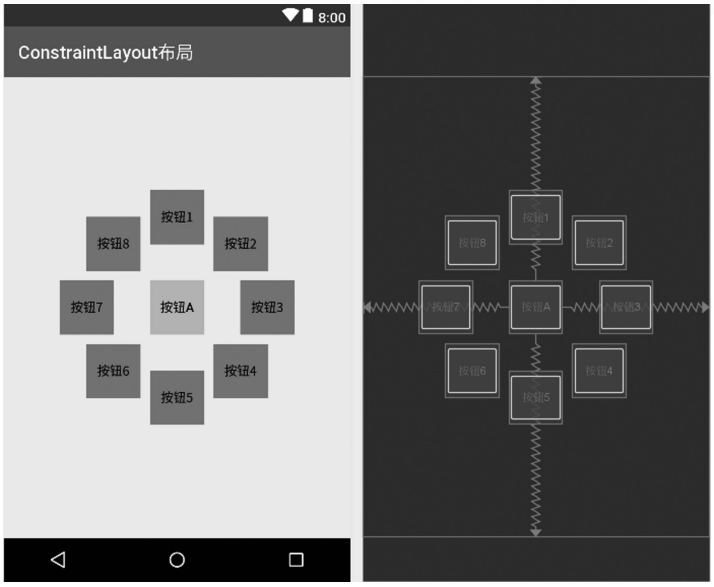


图 2-13　ConstraintLayout 属性应用

```
    <?xml version="1.0" encoding="utf-8"?>
    //根节点设置为约束布局,工程创建后直接默认为该布局
    <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/
apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#ccffaa">
        <Button
            android:id="@+id/buttonA"
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:background="#ffbb00"
            android:text="按钮 A"
    //设置当前按钮上、下、左、右分别与父容器上、下、左、右对齐,是设置中心对齐的一种方法
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"/>
        <Button
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:background="#0099ff"
            android:text="按钮 1"
```

```
        app:layout_constraintCircle="@id/buttonA"          //参照 buttonA 的中心点
        app:layout_constraintCircleAngle="0"
        app:layout_constraintCircleRadius="100dp" />       //与 buttonA 的距离为 100dp
    <Button
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 2"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="45"              //与 buttonA 的角度为 45°
        app:layout_constraintCircleRadius="100dp" />
    <Button
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 3"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="90"              //与 buttonA 的角度为 90°
        app:layout_constraintCircleRadius="100dp"/>
    <Button
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 4"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="135"             //与 buttonA 的角度为 135°
        app:layout_constraintCircleRadius="100dp" />
    <Button
        android:id="@+id/button4"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 5"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="180"             //与 buttonA 的角度为 180°
        app:layout_constraintCircleRadius="100dp"/>
    <Button
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 6"
        app:layout_constraintCircle="@id/buttonA"
```

```
        app:layout_constraintCircleAngle="225"              //与 buttonA 的角度为 225°
        app:layout_constraintCircleRadius="100dp" />
    <Button
        android:id="@+id/button3"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 7"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="270"              //与 buttonA 的角度为 270°
        app:layout_constraintCircleRadius="100dp" />
    <Button
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="#0099ff"
        android:text="按钮 8"
        app:layout_constraintCircle="@id/buttonA"
        app:layout_constraintCircleAngle="315"              //与 buttonA 的角度为 315°
        app:layout_constraintCircleRadius="100dp" />
</android.support.constraint.ConstraintLayout>              //约束布局结束标记
```

**【例 2-9】** 按钮 A 和按钮 B 分别在屏幕水平方向一半区域的中间，在屏幕垂直方向顶部 30%区域的中间，效果如图 2-14 所示。本例主要应用 Guideline 及其相关属性，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/
apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <android.support.constraint.Guideline                   //添加引导线 Guideline
        android:id="@+id/guideline"                         //为 Guideline 设置 id 为 guideline
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"                      //设置 Guideline 方向为垂直
        app:layout_constraintGuide_percent="0.5"/>          //位于父容器 50％位置
    <android.support.constraint.Guideline                   //添加第 2 个引导线
        android:id="@+id/guideline1"                        //设置 id 为 guideline1
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"                    //设置 Guideline 方向为水平
        app:layout_constraintGuide_percent="0.3" />         //位于父容器 30％位置
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 A"
        android:background="#22ccdd"
        //设置该按钮与父容器左侧对齐
        app:layout_constraintLeft_toLeftOf="parent"
        //设置该按钮与引导线 guideline 右侧对齐
        app:layout_constraintRight_toRightOf="@id/guideline"
        //设置该按钮与父容器顶部对齐
        app:layout_constraintTop_toTopOf="parent"
        //设置该按钮与引导线 guideline1 底部对齐
        app:layout_constraintBottom_toBottomOf="@id/guideline1"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 B"
        android:background="#22ccdd"
        app:layout_constraintLeft_toLeftOf="@id/guideline"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="@id/guideline1"/>
</android.support.constraint.ConstraintLayout>
```

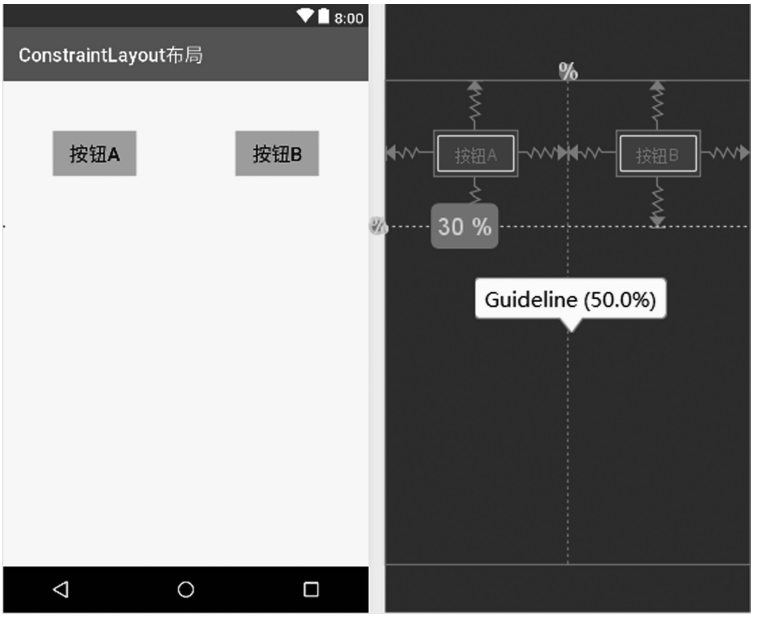


图 2-14　ConstraintLayout 属性应用

## 实战演练——成语小秀才游戏界面

在 UI 布局中，总会有一种不好的现象，就是滥用 LinearLayout，特别是初学者。这将会导致 View 树中的 View 数量激增，布局管理器添加到应用程序里都会带来一定的消耗。例如，我们嵌套了一些 LinearLayout，并使用了 weight 参数，这会导致子元素要计算两次，嵌套布局的花销尤其昂贵。

所以在做设计之前要先分析界面结构，之后再确定设计方案，选择合适的布局方式完成设计要求。也就是说，Android 的几种布局方式各有千秋，不论使用哪种布局方式，最后都是追求以最少的步骤完成设计图的要求，这应该是每个 Android 开发程序员的宗旨。

下面是一个小游戏界面的效果图，可以通过多种手段来完成它的布局。在本例中我们利用 Button 控件配合 shape 文件，以及给定的素材，以约束布局的方式，完成如图 2-15 所示的效果。

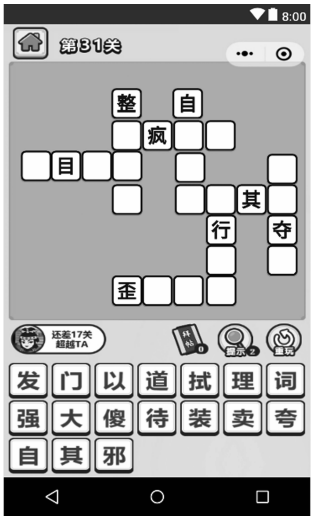


图 2-15　成语小秀才游戏界面

【步骤 1】　创建布局文件。新建工程，确定 layout 布局文件根节点为默认的约束布局。

【步骤 2】　修改主题风格。打开 app/res/values/styles. xml 文件，设置 App 的标题栏为 NoActionBar 风格主题，如图 2-16 所示。


图 2-16　设置 App 主题风格

【步骤 3】 编写 shape 形状文件。shape 文件是用来定义任意形状和颜色的资源文件，常常用来做背景样式等。以 shape 属性指定基本形状，因为矩形最为常用，所以 shape 文件创建后默认形状为矩形。

①在 res 上右击，在弹出的快捷菜单中选择“New”→“Android Resource File”选项，在打开的对话框中定义 shape 文件的文件名、资源类型及根节点标签等，如图 2-17 所示。


图 2-17 创建 shape 文件

②使用 shape 节点的默认配置项，直接在＜shape＞＜/shape＞标记之间定义 shape 的样式，如图 2-18 所示。


图 2-18 定义 shape 样式

③了解 shape 相关属性，见表 2-9。

表 2-9 shape 相关属性

| 属性名称 | 说　明 |
| --- | --- |
| android:shape | 在 shape 头标记中定义 shape 形状，取值有：rectangle(矩形)、oval(椭圆)、line(线条)和 ring(环)。例如，android:shape="oval"，设置形状为椭圆 |
| stroke | 描边，定义描边的 width(宽度)、color(颜色)、虚实线(dashWidth)，值为整数表示虚线的宽度，值为 0 时表示实线，dashGap 表示虚线的间隔 |

续表

| 属 性 名 称 | 说　　明 |
| --- | --- |
| solid | 指定形状内部填充色，只有一个属性 color |
| corners | 定义四个角的圆角半径，四个角也可以分别单独定义 |
| gradient | 定义渐变色，可以定义两色渐变和三色渐变，以及渐变样式，属性有：type(设置渐变类型)，取值 linear 为线性渐变(默认)，radial 为放射渐变，sweep 为扫描式渐变；startColor(渐变开始点的颜色)；centerColor(渐变中间点的颜色)，在开始与结束点之间；endColor(渐变结束点的颜色)；gradientRadius(渐变的半径)，只有当渐变类型为 radial 时才能使用 |

**【步骤 4】** 为主布局设置背景图像。首先将素材文件中的图片 back01.jpg 复制到 res/drawable 文件夹下，然后在布局根节点起始标记中使用 background 载入背景图像。

**【步骤 5】** 如图 2-19 所示，拖动 Button 控件到布局中，在右侧的属性面板设置所需要的属性。为按钮设置背景样式，即将 shape 文件以图片文件的形式载入。

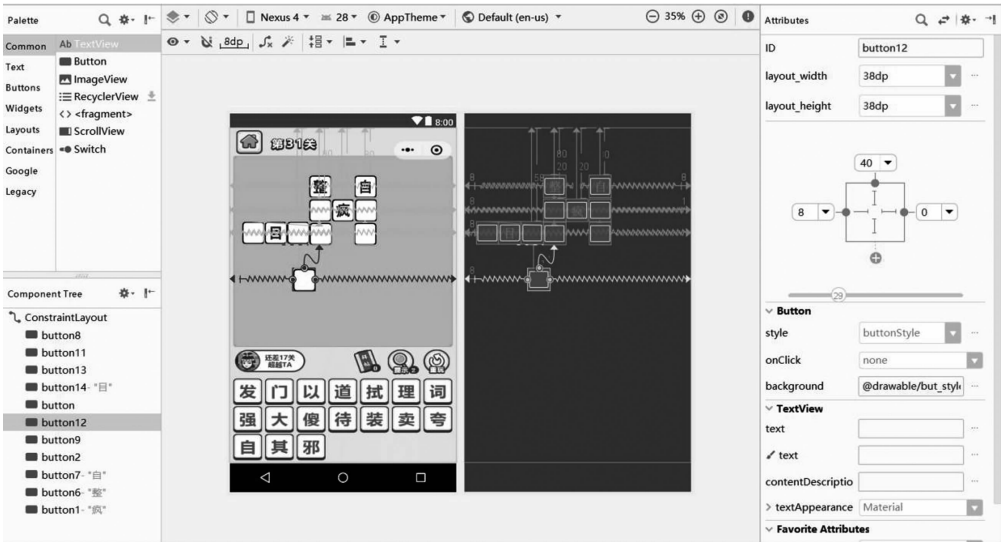


图 2-19　界面布局可视化操作

在利用可视化操作的同时，也可以使用代码模式，二者相结合。通过代码形式对可视化操作自动生成的代码进行修改、加工，使用布局更加精准。个别按钮代码如下：

```
<Button
        android:id="@+id/button1"
        android:layout_width="38dp"
        android:layout_height="38dp"
        android:layout_marginTop="120dp"
        android:background="@drawable/but_style"  //载入背景样式文件 but_style
        android:text="疯"
        android:textSize="24sp"
```

```
android:textStyle="bold"                                    //设置文本样式为粗体
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

以次类推，添加 Button 控件，设置其大小、位置、样式等属性，最终完成效果如图 2-15 所示。

## 课后提升

**1. 判断题**

(1)在 Android 开发中 UI 布局常用的一种设计方式是利用可视化工具来进行，允许通过拖曳控件来进行布局。（　　）

(2)用户所能见到的所有界面布局，都是使用 View 和 ViewGroup 对象建立的。（　　）

(3)指定该视图与周围视图之间的间距通常称为内边距。（　　）

(4)View 是从 ViewGroup 派生出来的，负责对添加进 ViewGroup 的 View 进行布局和管理。（　　）

(5)相对布局(RelativeLayout)是一种通过相对定位的方式进行界面布局的管理器。（　　）

**2. 选择题**

(1)(　　)是用于设置布局管理器内组件的排列方式，取值为 horizontal 或 vertical 的属性。

A. orientation　　B. gravity

C. weight　　D. background

(2)以下选项(　　)是网格布局。

A. LinearLayout　　B. RelativeLayout

C. FrameLayout　　D. GridLayout

(3)在相对布局中使用(　　)属性可让控件位于父容器的中央位置。

A. alighParentBottom　　B. centerHorizontal

C. centerVertical　　D. centerInParent

(4)TableLayout 表格布局使用(　　)属性来定义可伸展的列。

A. stretchColumns　　B. TableRow

C. stretchColumns　　D. layout_span

(5)下列选项(　　)是设置网格布局的最大行数的属性。

A. columnCount　　B. rowCount

C. layout_rowSpan　　D. columnWeight

**3. 实操题**

创建一个工程，通过执行 res/layout/New/XML/layout XML File 创建布局文件，分别使用线性布局、相对布局、表格布格、网格布局、约束布局等，对计算器界面、成语小游戏界面进行重新设计制作。