# 14.1 Flutter UI系统

在本书的前面章节中，我们多次提到“UI系统”这个概念，本书中所指的UI系统特指：基于一个平台，在此平台上实现GUI的一个系统，这里的平台特指操作系统，如Android、iOS或者Windows、macOS。我们说过各个平台UI系统的原理是相通的，也就是说无论是Android还是iOS，他们将一个用户界面展示到屏幕的流程是相似的，所以，在介绍Flutter UI系统之前，我们先看看UI系统的基本原理，这样可以帮助读者对操作系统和系统底层UI逻辑有一个清晰的认识。

### 硬件绘图基本原理

提到原理，我们要从屏幕显示图像的基本原理谈起。我们知道显示器（屏幕）是由一个个物理显示单元组成，每一个单元我们可以称之为一个物理像素点，而每一个像素点可以发出多种颜色，显示器成相的原理就是在不同的物理像素点上显示不同的颜色，最终构成完整的图像。

一个像素点能发出的所有颜色总数是显示器的一个重要指标，比如我们所说的1600万色的屏幕就是指一个像素点可以显示出1600万种颜色，而显示器颜色是有RGB三基色组成，所以1600万即2的24次方，即每个基本色（R、G、B）深度扩展至8 bit(位)，颜色深度越深，所能显示的色彩更加丰富靓丽。

为了更新显示画面，显示器是以固定的频率刷新（从GPU取数据），比如有一部手机屏幕的刷新频率是 60Hz。当一帧图像绘制完毕后准备绘制下一帧时，显示器会发出一个垂直同步信号（如VSync）， 60Hz的屏幕就会一秒内发出 60次这样的信号。而这个信号主要是用于同步CPU、GPU和显示器的。一般地来说，计算机系统中，CPU、GPU和显示器以一种特定的方式协作：CPU将计算好的显示内容提交给 GPU，GPU渲染后放入帧缓冲区，然后视频控制器按照同步信号从帧缓冲区取帧数据传递给显示器显示。

CPU和GPU的任务是各有偏重的，CPU主要用于基本数学和逻辑计算，而GPU主要执行和图形处理相关的复杂的数学，如矩阵变化和几何计算，GPU的主要作用就是确定最终输送给显示器的各个像素点的色值。

### 操作系统绘制API的封装

由于最终的图形计算和绘制都是由相应的硬件来完成，而直接操作硬件的指令通常都会有操作系统屏蔽，应用开发者通常不会直接面对硬件，操作系统屏蔽了这些底层硬件操作后会提供一些封装后的API供操作系统之上的应用调用，但是对于应用开发者来说，直接调用这些操作系统提供的API是比较复杂和低效的，因为操作系统提供的API往往比较基础，直接调用需要了解API的很多细节。正是因为这个原因，几乎所有用于开发GUI程序的编程语言都会在操作系统之上再封装一层，将操作系统原生API封装在一个编程框架和模型中，然后定义一种简单的开发规则来开发GUI应用程序，而这一层抽象，正是我们所说的“UI”系统，如Android SDK正是封装了Android操作系统API，提供了一个“UI描述文件XML+Java操作DOM”的UI系统，而iOS的UIKit 对View的抽象也是一样的，他们都将操作系统API抽象成一个基础对象（如用于2D图形绘制的Canvas），然后再定义一套规则来描述UI，如UI树结构，UI操作的单线程原则等。

### Flutter UI系统

我们可以看到，无论是Android SDK还是iOS的UIKit 的职责都是相同的，它们只是语言载体和底层的系统不同而已。那么可不可以实现这么一个UI系统：可以使用同一种编程语言开发，然后针对不同操作系统API抽象一个对上接口一致，对下适配不同操作系统的的中间层，然后在打包编译时再使用相应的中间层代码？如果可以做到，那么我们就可以使用同一套代码编写跨平台的应用了。而Flutter的原理正是如此，它提供了一套Dart API，然后在底层通过OpenGL这种跨平台的绘制库（内部会调用操作系统API）实现了一套代码跨多端。由于Dart API也是调用操作系统API，所以它的性能接近原生。

注意，虽然Dart是先调用了OpenGL，OpenGL才会调用操作系统API，但是这仍然是原生渲染，因为OpenGL只是操作系统API的一个封装库，它并不像WebView渲染那样需要JavaScript运行环境和CSS渲染器，所以不会有性能损失。

至此，我们已经介绍了Flutter UI系统和操作系统交互的这一部分原理，现在需要说一些它对应用开发者定义的开发标准。其实在前面的章节中，我们已经对这个标准非常熟悉了, 简单概括就是：组合和响应式。我们要开发一个UI界面，需要通过组合其它Widget来实现，Flutter中，一切都是Widget，当UI要发生变化时，我们不去直接修改DOM，而是通过更新状态，让Flutter UI系统来根据新的状态来重新构建UI。

讲到这里，读者可能发现Flutter UI系统和Flutter Framework的概念是差不多的，的确如此，之所以用“UI系统”，是因为其他平台中可能不这么叫，我们只是为了概念统一，便于描述，读者不必纠结于概念本身。

在接下来的小节中，我们先详细介绍一下Element、RenderObject，它们是组成Flutter UI系统的基石。最后我们在分析一下Flutter应用启动和运行的整体过程。