

让不懂编程的人爱上iPhone开发(2017秋iOS11+Swift4+Xcode9版)-第3篇

休息好了吗？欢迎回来继续我们的iPhone开发学习之旅。

应用的工作原理

在继续学习之前，让我们来了解一下一个应用究竟是如何工作的？

一个应用通常是由对象组成的，而这些对象之间可以相互发送消息。在我们的这款应用中，很多对象是由iOS提供的，比如按钮(一个UIButton对象)，还有弹出对话框(UIAlertController)。当然，部分对象需要我们自己来实现，比如视图控制器。

这些对象之间通过发送消息来相互交流。比如在我们这款应用中，当玩家触碰应用中的按钮时，UIButton对象会向视图控制器发送消息，而视图控制器则可能向更多的其它对象发送消息。

在iOS中，应用是事件驱动的。也就是说这些对象会等待某些特定的事件发生，然后进行处理。或许你会觉得很奇怪，一个应用大多数的时间都在。。。无所事事。它只是端上一杯茶，坐在那里等待事情的发生。当玩家触碰屏幕的时候，应用会花上几个微妙的时间来处理这个事情。然后呢？然后当然是接着休息，直到下一个事件的到来。

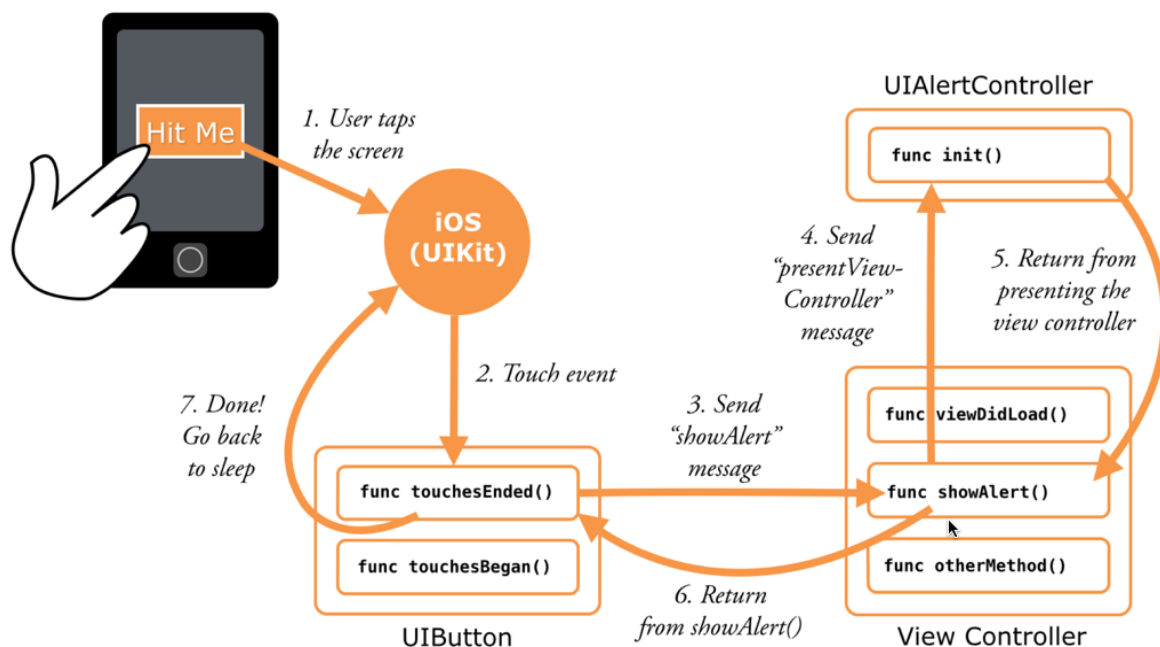
而你作为一个程序猿扮演的角色是什么呢？是编写一些代码，当你的对象接收到事件发生的消息后对它们进行处理。

在我们这款应用中，按钮的Touch Up Inside事件和视图控制器的showAlert动作关联在一起。当按钮发现自己正在被触碰的时候，就会向视图控制器发送showAlert消息。

而在showAlert中，视图控制器向UIAlertController对象发送另一条消息addAction。二为了显示弹出警告提示框，试图控制将发送消息presentViewController。

你的整个应用都是由使用类似方式来相互交流的对象组成的。

当然，你可能之前在自己的网站开发中使用过PHP或者Ruby脚本。上面的这种事件驱动模型和PHP脚本的工作方式是不一样的。PHP代码会从头到尾顺次执行，直到抵达代码的尾部，然后退出。



而应用则不同，除非玩家强制关闭（或者自己崩溃了），它是不会主动退出的。一个应用会将自己的大部分宝贵时间消磨在静静的等待上，它们会等待输入事件的发生，然后如获至宝的去处理这些事件，完成后重新回家休息。

在移动设备上，玩家的输入通常是触摸事件，这一类事件是应用最重要的事件源，但除此之外还有其它类型的事件。比如操作系统会通知应用有来电，或者界面要重新绘制，或者计时器在不断的倒计时，等等。

不管是哪种类型的事件，你需要记住一点，应用所做的每一件事都是由某种事件来驱动的。

继续处理我们的to-do list清单

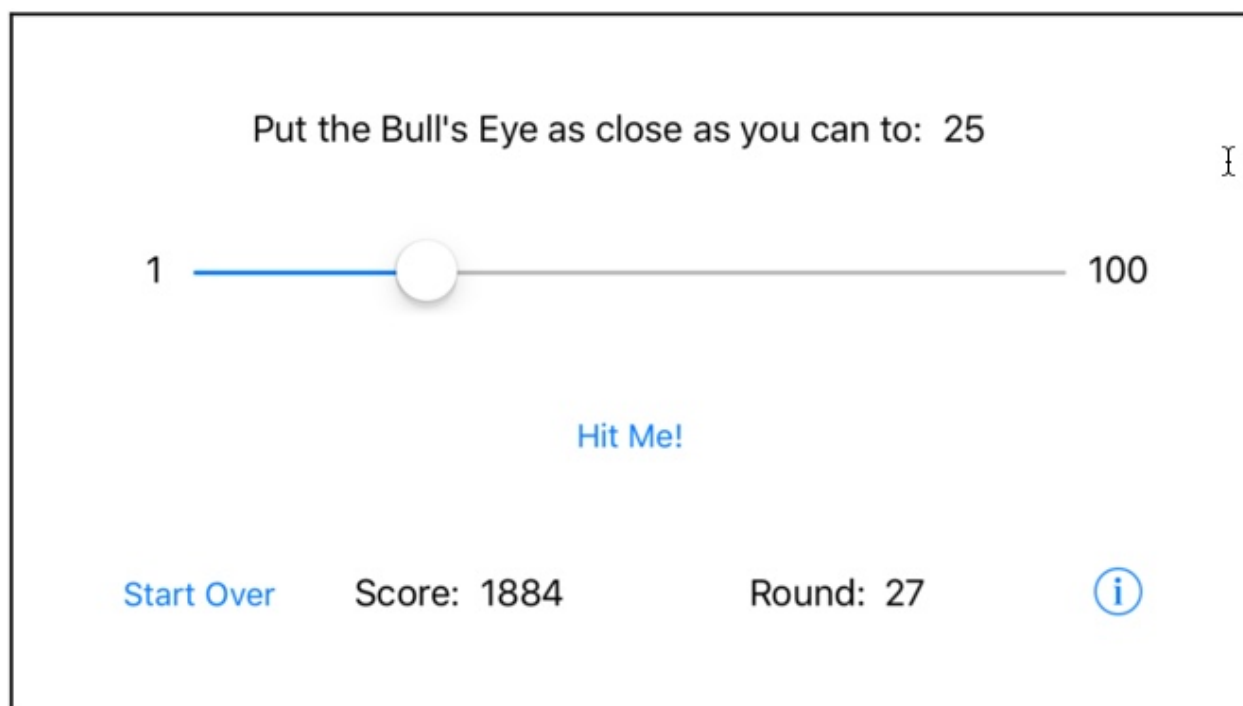
理论知识过后，又到了继续完成to-do list清单的时候了。

在之前的学习中，我们已经成功的把一个按钮放置在界面中，同时让它可以在玩家触碰的时候弹出一个提示对话框。接下来我们需要继续处理清单中的其它事项。

虽然说是一个清单，其实我们没有必要按照特定的顺序依次执行，除非某一个事项必须在另一个事项已完成的情况下才能进行。比如当我们还没有滑动条的时候就不可能读取滑动条上的数值。

好了，现在让我们在界面上添加一些其它的控件-滑动条和文本标签，把这个应用升级为一款真正的游戏！

当我们完成这些任务后，游戏的界面会是这样的：



不过，似乎有点不对劲？这个界面和最开始给你展示的界面看上去不太一样。

这是因为这里所使用的都是标准UIKit控件。

对于常规的应用来说，这种界面差不多也就可以了。不过考虑到对一款游戏来说这样的界面实在是有点无法忍受，我们会在后面的内容里面把它变得漂亮一点。

UIKit和其它框架(frameworks)

iOS中提供了很多建筑程序“大厦”用的基础材料，我们将其称之为frameworks(框架)，或者说“kits(组件)”。UIKit这个框架是最基本也是最重要的一种，其中提供了用户界面的一些控件，比如按钮，标签，导航栏等。使用它可以管理视图控制器，同时处理一切和应用的界面相关的事情。（UI的全称就是User Interface）

如果你必须从零开始给所有的东西编码，恐怕会浪费大量的时间。与之相反，大多数情况下我们可以在系统提供的框架之上来创建应用，从而充分利用苹果工程师提前为你准备好的大礼。

在应用中所有以UI为前缀的对象，比如UIButton都来自UIKit。

当你开发一款iOS应用的时候，可以说大部分的时间都在和UIKit这个框架打交道。

其它的框架也各自有各自的用途。比如Foundation框架提供了构建APP的很多基本要素。Core Graphics框架用于在界面上绘制基本形状，比如直线，矩阵，渐变和图形等。AVFoundation框架用于播放声音。当然还有其它更多的框架。iOS的完整框架又被称之为Cocoa Touch。

说到Cocoa Touch，这里再多扯几句。Cocoa Touch是在Mac的Cocoa开发框架基础上诞生的，同时也为移动设备的特点做了优化。为了了解Cocoa Touch，我们不妨看看它在Mac上的前身-Cocoa。Cocoa是Mac OS X上著名的五大API之一，其它四个分别是Carbon,POSIX,X11和Java。Cocoa起源于1989年乔布斯在NEXT公司搞的NeXTSTEP 1.0,当时没有Foundation框架，只有动态运行库，也即Kit，最重要的就是AppKit了。后来NeXT硬件卖的很糟糕，帮主就把NeXTSTEP 3.1移植到了Intel和HP平台，同时加入了Foundation框架。1996年苹果收购了NEXT，帮主从此开始了王者归来的14年登神之路。

这里放一张乔帮主当年在NEXT时的NB照片，虽在1985年被驱逐出苹果但生就一副高富帅的样子。正如周星星所言，“你以为躲起来就找不到你了吗？没有用的！象你这样拉风的男人，无论在什么地方，都像漆黑中的萤火虫一样，那样的鲜明，那样的出众。。。”



苹果收购NEXT后，Cocoa在Mac 系统上开始独放异彩。在2014年WWDC苹果发布Swift之前，我们都会用Xcode（前身是Project Builder）和Interface Builder上用Objective-C开发Cocoa应用程序。

Cocoa的设计满足最严格的MVC(Model模型-View视图-Controller控制器)原则。所以在斯坦福大学iOS开发教程中会花上一节课的时间专门讲解MVC的概念。这里先不具体说MVC，后面再详细介绍，免得把你给撑着了。

在面向桌面级应用的Cocoa架构中，使用AppKit来提供图形用户界面交互所需要的基本控件。它基于Foundation框架，使用NS前缀。

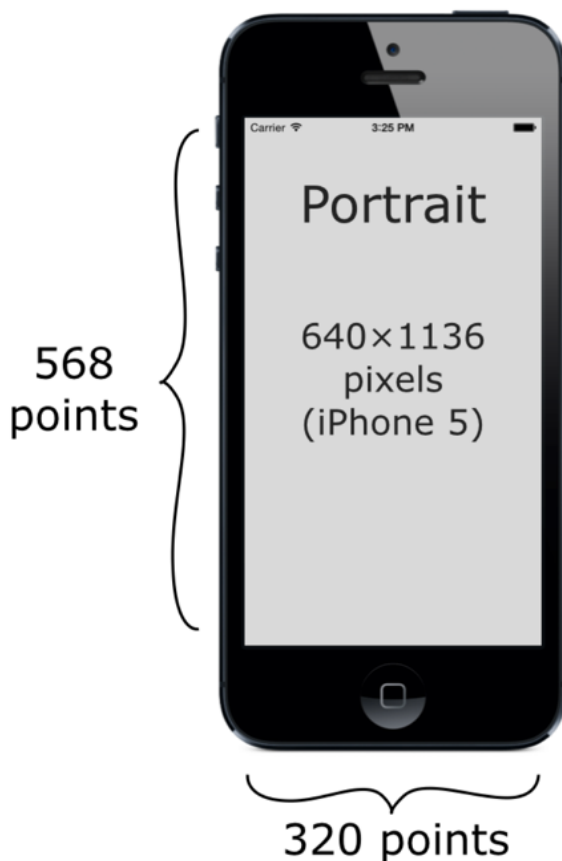
当2008年苹果向第三方开发者开放iOS SDK时，使用UIKit替代AppKit,作为iOS设备的图形用户界面工具包，使用UI前缀。

好了，上面这些知识先大概了解下，感兴趣的童鞋可以自己去探索更深入的内容。

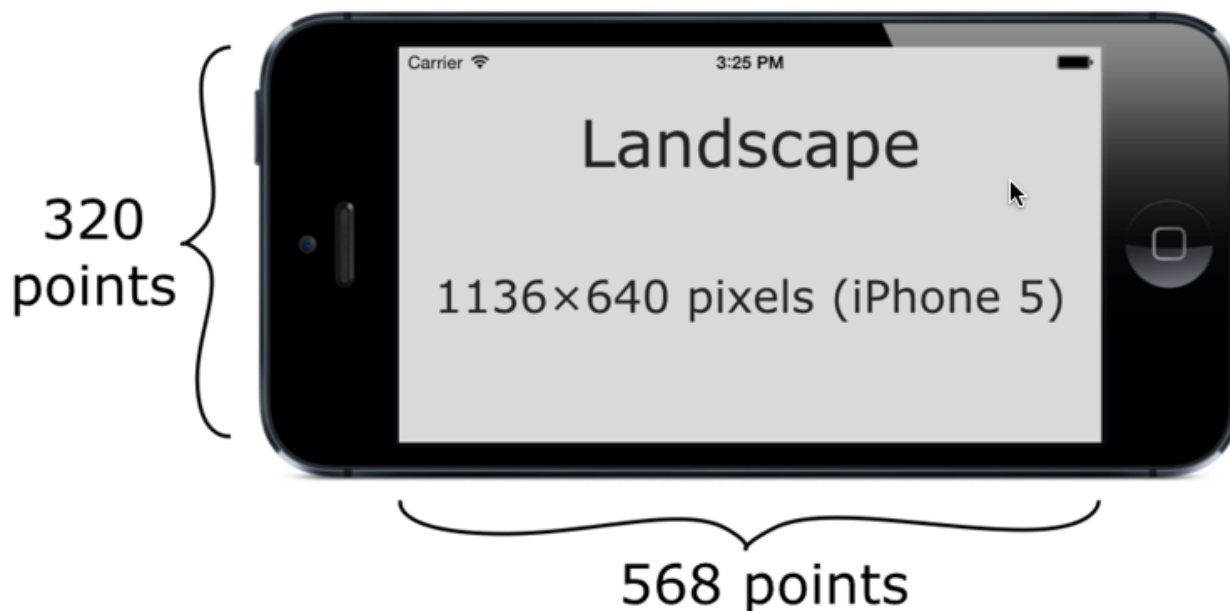
切换设备显示模式（竖屏VS横屏）

什么是横屏，什么是竖屏？宽度比高度要大，就是横屏，反之就是竖屏。这里不多废话，看看下面这两个图就一目了然了。

竖屏是这样滴：



横屏是这样滴：



在竖屏显示模式下，iPhone SE的界面在垂直方向上有586个点。横屏方向则正好颠倒过来。

那么什么是point(点)?

对于iPhone 3GS和之前的老设备，包括对应的Ipod touch设备，还有iPad1,2,ipad mini，一个点就对应一个像素。这样说很明白吧。

那么什么是pixel(像素)? 简单点说，它是任何一个屏幕（包括电脑）的最小显示单位。电子设备的屏幕是由像素矩阵组成的（对于高清屏幕需要在放大镜和显微镜下面才看得到矩阵）。矩阵上的每一个点都有自己的色彩和亮度。当我们在屏幕上显示图片时，可以通过更改这些像素的色彩数值来形成一副图片。

不过对于具备Retina(视网膜显示，或者说高清显示)设备的屏幕上来说，一个点对应垂直和水平方向上的各两个像素，或者说总共对应4个像素。对于老的iOS设备，或者说“标清”，“低清”设备，320*480点就对应320*480像素。

但是对于Retina设备来说，这个数字就变成了640*960像素（iPhone5之后的设备是640*1136）像素。实际上你要显示之前4倍的像素。

而iPhone 6 Plus和iPhone 7 Plus就更疯狂了，它具有3x的分辨率，因此每个点对应9个像素！是不是有点疯狂？如果你不具备刺客一样的鹰眼技能，那么几乎看不出来这种Retina HD显示上的单个像素。同时我们很难用肉眼分辨率一个像素从那里结束，临近的像素从哪里开始。

随着iPhone的不断演化，不仅仅是像素数量有了明显的不同，同时屏幕尺寸也发生了明显的变化。从最初的3.5英寸到如今iPhone 6 Plus的5.5英寸，Cook你老人家忘了乔帮主当年说的话了吗？'Who need a large screen?'

不过作为开发者，还是接受这一事实吧。下面这个表列出了屏幕尺寸和相应的分辨率（以point点为单位）：

设备类型	屏幕尺寸	屏幕分辨率（point点为单位）
iPhone 4S和之前的机型	3.5	320*480
iPhone 5,5c和5s,SE	4	320*568
iPhone 6, 6s,7,8	4.7	375*667
iPhone 6 ,6s ,7,8 Plus	5.5	414*736
iPhone X	5.8	375*812
iPad	7.9,9.7	768*1024
iPad Pro	12.9	1024*1366

在早期开发iOS应用的时候，只有一个屏幕尺寸，很长一段时间里面曾经是👉Android开发的原因之一。但如今一个尺寸打天下的好日子已经一去不复返了，iOS开发者也需要开始应付一系列的屏幕尺寸。

iOS开发者需要记住一点的是，UIKit使用的单位是point，而不是pixel（像素），因此我们只需要考虑以point为单位的不同屏幕尺寸。真实的像素大小对于UI设计师很重要，因为图形图像仍然是以像素为单位的。

所以，开发者以point为单位，设计师以pixel为单位。

point和pixel的区别是不是让你稍微有点糊涂了？不过如果看到现在你只对这一个问题感到迷惑，那就说明我的努力没有白费~

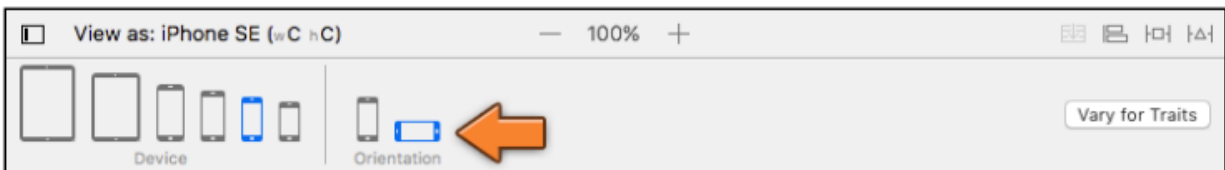
在这篇教程中，我们首先需要使用iPhone SE的320*568 point。而在随后的内容中，我们会学习如何让这款游戏适配不同大小的iPhone。

让应用在横屏模式下工作

为了让应用从竖屏模式切换为横屏模式，我们得完成三件事情：

1. 让Main.storyboard中的视图使用横屏显示
2. 更改设备的”Supported Device Orientations”设置。

在Xcode的Interface Builder中打开Main.storyboard，在View as: iPhone SE面板中，将Orientation 更改为landscape：



接下来将按钮移动回视图的中央位置，



现在我们来点击Run按钮，就可以看到下面的情况。



显然这不是我们想看到的结果，不但屏幕不是横屏显示，而且按钮也不是处在中央位置。

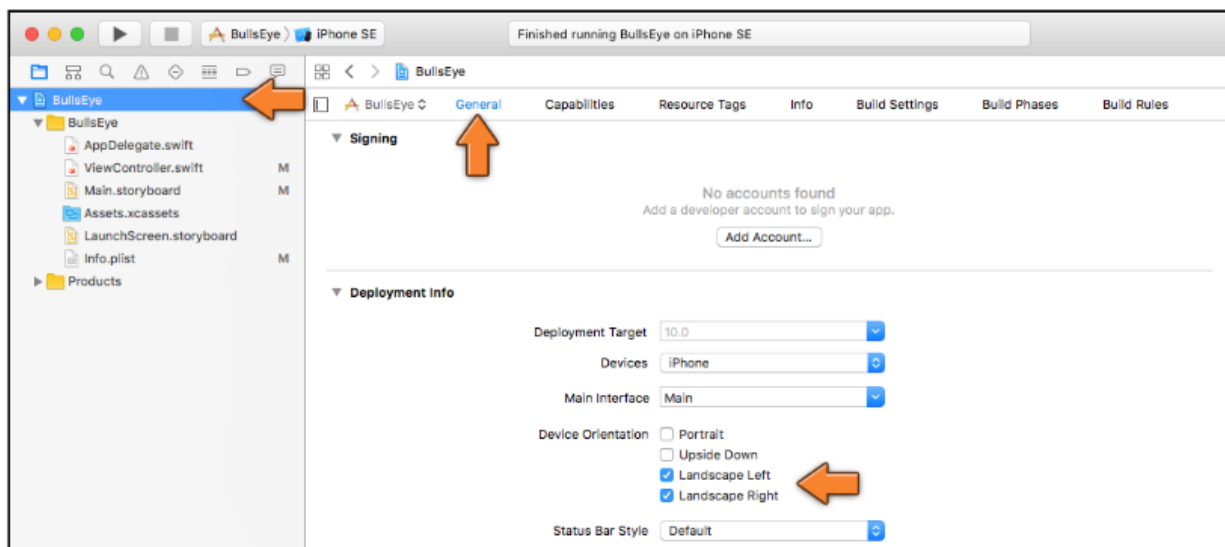
不过，如果我们手动把Simulator旋转到Landscape，那么一切会看起来正常。

方法很简单，从iOS Simulator的菜单栏上选择Hardware-Rotate Left或是Rotate Right，也可以按住command键，然后按左右箭头切换模拟器朝向。

注意到在横屏显示状态下界面不会显示iPhone的状态栏（电池电量，通讯信号，时间。。。），这样可以让应用充分利用空间。

看起来我们还需要做一件事，而不是用上面的方式来手动调节。实际上有一个配置选项让iOS知道应用的显示方向。一般情况下，从模板新建的应用同时支持Portrait和Landscape布局。

在Xcode中点击Project Navigator（项目导航）顶部的BullsEye项目图标，此时Xcode窗口的主面板会显示关于项目的一些设置。确保选中右侧的General选项卡。



在Deployment Info下面有一个 Device Orientations。上面默认选中了三个，我们要取消选中Portrait这个选项。

现在再点Run按钮，一切应该正常了。

注意：因为此前对模拟器的设备进行过旋转操作，所以如果这个时候模拟器属于垂直状态，需要手动将其旋转到水平状态。这样后面就完全正常了。

好了，忙了半天，再来点福利吧。



One more thing... 关于对象，消息和方法

我们这个教程的目的就是，不求最快，但求最好。这不，你才写了一段代码，现在又该进入理论知识充电时间了。

接下来讲点编程的理论知识，希望你别犯困，待会儿有福利的。

Swift属于所谓的“面向对象”的编程语言，也就是说你要做的大多数事情都和某种类型的对象有关。之前我也提到过，一个应用就是由彼此可以发送消息的对象组成的。

当你开发一款iOS应用的时候，通常会用到系统提供给你的一些对象。比如UIKit框架里面的UIButton对象。同时你也会制造属于自己的对象，比如视图控制器。

那么对象(object)究竟是个什么东西？如果把你要开发的产品比作一座大厦，那么对象就是修建这座大厦的砖块。程序猿们喜欢把有一定关联的功能放到对象里面去。这个对象可以解析一个RSS feed，那个对象可以在屏幕上绘制一个图像，还有个对象可以执行复杂的计算。每个对象都负责程序中的某个特定部分。在一个完整的应用中存在着多种不同的对象（几十上百，甚至成千上万？）

即便是我们这款小小的应用，也包含了几个不同的对象。最经常用到的当然是ViewController这个视图控制器。

除此之外，按钮也是一个对象，当然还有提示对话框。

我们的项目有一个名为AppDelegate的对象，当然这里不多说它的作用。不过你会发现基本上每个iOS应用里面都有一个xxxAppDelegate的东西在里面。我们在提示对话框里面所放的内容也是对象。在iOS应用里，对象无处不在！

一个对象既有数据(data)，也有功能(functionality)。

- 比如按钮。当我们把按钮拖曳到storyboard上去的时候，实际上它就成了视图控制器的一种数据（data）。数据总是包含着一些东西。比如在这里，视图控制器就包含着按钮。
-
- 至于功能，也就是执行一些具体的操作。比如我们所添加的 showAlert 动作，它就是一个功能。

按钮本身也有自己的数据和功能。比如按钮上的标签文本内容和颜色，它在屏幕上的位置，宽度和高度，等等，都是它的数据。按钮的功能则是：它能发现玩家触碰到它，然后会触发一个动作作为响应。

在iOS开发中，一个对象的功能通常被称为method(方法)。其它的编程语言可能会换个叫法，比如“函数”，“过程”，“子程序”神马的，不过在Swift里面用的术语是方法。方法就是属于某个对象的功能。

我们的showAlert动作就是一个方法（终于可以给它正名了，动作还是多少有点别扭~）在Swift中识别方法很简单，代码前面使用func(function的缩写），而名字后面是大括号。

```
@IBAction func showAlert() {
```

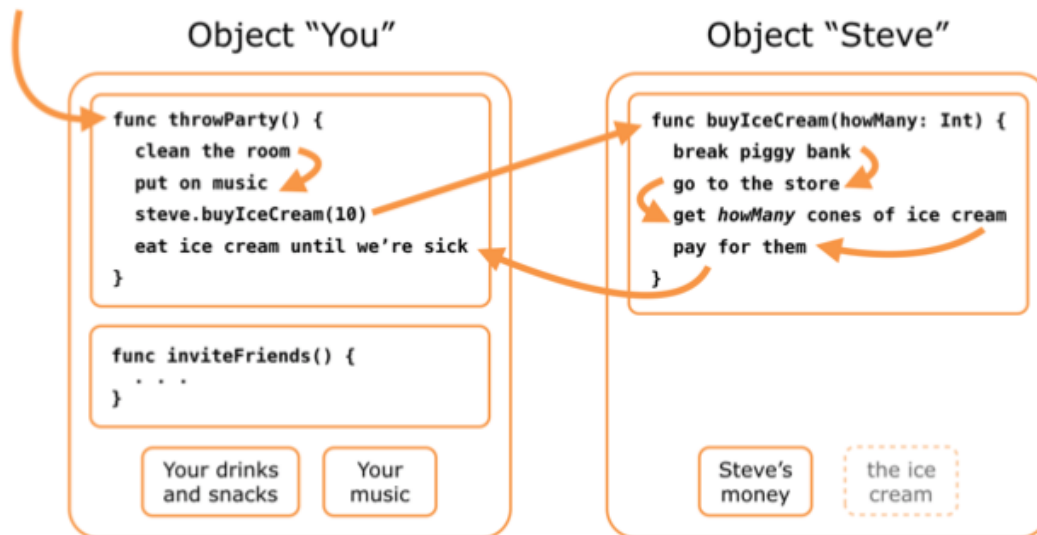


如果你仔细看ViewController.swift中的代码，可以看到几个其它的方法，比如viewDidLoad()和didReceiveMemoryWarning()。

以上这些方法目前并没有做什么实际工作（占位？）Xcode模板只是为了方便开发者所以才会在创建项目时自动帮你生成。这些特殊的方法是由视图控制器使用的，所以我们后面肯定会告诉你究竟它们是干吗用的。别着急，慢慢来。

方法的概念

当然，对于一个从来没写过代码的非程序猿来说，方法的概念肯定有点怪异。下面给了一个例子：



你（或者一个叫“你”的对象）想在周末开个party聚会。但忘了提前准备冰激凌。幸好你认识一个叫Steve的对象（你放心，他不可能是我们伟大的帮主），幸好他家附近有个便利店。如果派对上没有冰激凌肯定让人感觉不爽，所以在你准备派对的过程中你给Steve发了个消息，让他去买点冰激凌。

这个时候系统会切换到叫Steve的对象，然后执行他的buyIceCream()方法，从头到尾依次执行。一旦他的方法完成，系统会返回你的throwParty()方法，继续其它的事情，这样你和你的朋友就可以饱餐Steve带来的冰激凌了。

Steve这个对象也有自己的数据。在他去买冰激凌之前，他知道自己是有钱人。在商店里他和售货员交换了钱的数据，当然更重要的是，用钱的数据的减少换来了冰激凌数据的增加。做好这笔交易之后，他带着冰激凌返回派对（前提是没有半路偷吃光）。

“发送消息”实际上比字面上的意思更复杂。我们可以用它来帮忙理解对象之间是如何交流的，但别指望真有信鸽或者快递员参与到这个过程中。系统只是简单的从throwParty()方法跳转到buyIceCream()方法，然后再跳转回来。

通常来说，“调用方法”和“触发方法”都是一回事：系统会跳转到你要调用的方法，然后在执行完毕其中的代码后返回到之前的方法。

一个很重要的事情是，对象都有自己的方法和数据。对象可以查看其它对象的数据（当然Steve也可能拒绝你查看他的卡上余额），以及让其它对象执行它们的方法。这些事情放在一起，就组成了一个完整的应用。

好了，第三天的内容到此结束。别太贪心，明天再来。

最后的小福利，CNN评出的最令中国人思乡的菜，看看里面有你的菜吗？～





