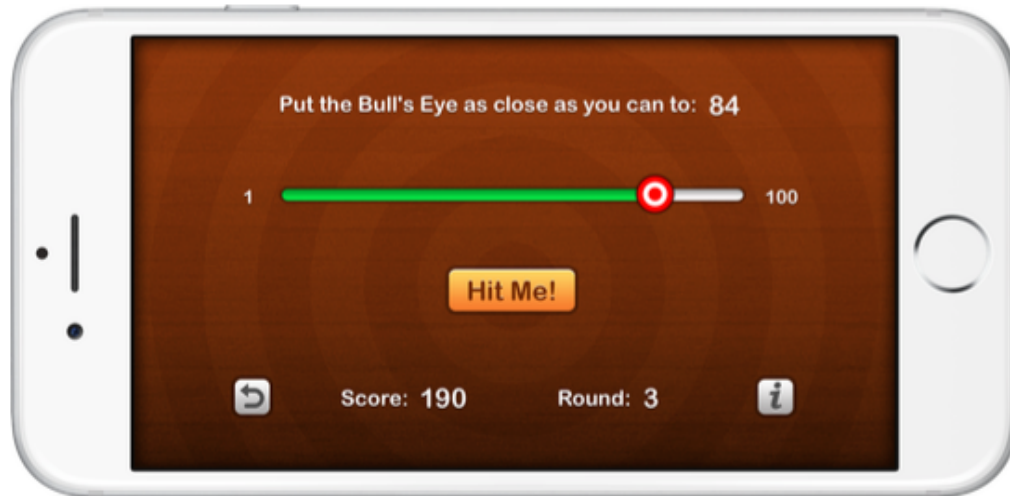


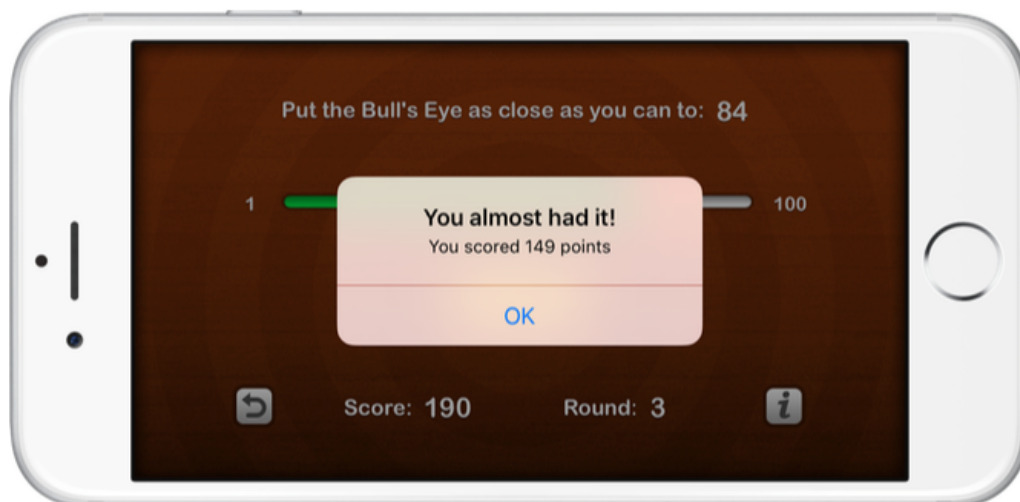
让不懂编程的人爱上iPhone开发(2017秋iOS11+Swift4+Xcode9版)-第2篇

“王者打靶”小游戏

在我们学习iPhone开发的第一站，将创建一个叫“王者打靶”的小游戏。当游戏最终完成后的效果如下：



游戏的规则很简单，你需要拖动滑动条上的红色靶心，让它所在的位置尽可能接近我们设定的目标数字（每次随机生成）。比如在上图中的目标数字是84。因为你没法直接看到滑动条上靶心所在位置的数字，所以就得去猜。当你觉得差不多的时候，可以按下Hit me!按钮，然后就会出现一个对话框告诉你猜的结果如何。



你猜的数字越接近目标数字，你的得分就越高。当你按下OK按钮后就会关闭对话框，然后开始新一轮的游戏。你可以一直玩下去，直到按下“重新来过”按钮（左下角的那个），它会把累积得分重置为0。

虽然开发完这款游戏没法让你成为国民老公，不过即便是亿万富豪也得有个启动资金吧。

帮你整理思路的待完成事务清单

练习：现在你已经大概知道游戏大概的最终视觉效果了，也了解了游戏的基本规则。如果感兴趣的话，可以尝试写下来为了制作这款游戏要做哪些事情。当然，很可能此时你脑中一片空白，但Don't panic。不要恐慌，从零开始没有你想象的那么难。

我还是给你点提示吧：“为了制作这款游戏，我们需要把Hit Me!按钮放到屏幕中，然后当玩家触碰它的时候弹出对话框。。。 ”诸如此类的事情，你可以按这个思路去想，无论想到什么，想写下来。哪怕你现在一行代码都不会写也不要紧。在做任何事情之前，我们首先需要了解的是需要做什么，具体如何去实现反而不是那么重要。

一旦你明白自己该做些什么事情，你就可以真正着手开始去思考如何去实现，不管是请求高手指点，在网上求助，或是自己去查询各种文档来学习。再次强调一点，知道该做什么是最最重要的。很多初学者（包括一些老鸟）在开始写代码之前很少思考，也不会在纸上写写画画，他们甚至不知道自己最终要实现的是个什么样的东西，也难怪很容易就会在中间受阻了。

每当我开始做一个产品的时候，无论是应用还是游戏，首先都会把这个产品要实现的功能详细列出来。这就是我的开发用事务清单（to-do list）。有了这样的清单，可以把一个产品的设计和分解为很多小的模块，从而在具体实现的时候降低了复杂度。

很多时候我们灵光乍现有了非常NB的一个创意，但一旦坐在电脑前开始写代码的时候，就会觉得这个事太难了，简直无法实现。比如制作一款像MineCraft一样NB的独立游戏，想想就令人望而生畏，要做的事情太多了，究竟该从哪里着手开始呢？还是那句话，列一个清单出来，把你要做的事情分解为具体而小的步骤。如果某个步骤让你觉得非常复杂，就继续分解下去，直到你认为马上可以操作为止。然后就从那里去开始执行吧。

毫无疑问，这个练习对于新手来说有点困难，或许依然是无从下手吧。但随着你对软件开发的理解逐渐深入，你会发现如何将一个完整的产品设计分解为具体可执行的模块或组件。

说了这么多，或许你还是找不着北吧。这里我就勉为其难整理一下开发这款游戏要做的事务清单吧：

1. 在屏幕中放一个按钮，在按钮上放一个标签“打我啊笨蛋！”，或者“Hit Me!”
2. 当玩家触碰Hit Me这个按钮的时候，需要让应用弹出一个对话框，告诉玩家他猜的准不准。当然在此之前我们需要计算玩家的得分，并把得分放到对话框的弹出信息中。
3. 在屏幕上放置一些文本标签，比如“Score:”，“Round:”。有些文本标签的内容会随着游戏的进展发生变化，比如玩家的得分，每轮游戏结束后都会增加。
4. 在屏幕上放一个滑动条，把它的数值范围设定在1到100之间。
5. 当玩家触碰Hit Me按钮后读取滑动条上的数值。
6. 在每轮游戏开始的时候生成一个随机数，并把它显示在屏幕中。这个随机数就是目标数值。
7. 比较滑动条上的数值和所生成的随机数，并基于它们之间的差异值来计算玩家的得分。最后把这个分数放到对话框的弹出信息中。
8. 在屏幕上放置一个“重新来过”的按钮。使用它来重置玩家得分和游戏轮数。
9. 把应用设置为横向显示
10. 美化界面 😊

当然，我可能漏掉了某些事情，不过有了这样的一个清单，起码你知道该干吗了。哪怕是如此简单的一款游戏，都需要我们去做这么多的事情。当然，具体如何来实现先不要着急，无论这款游戏是PC版，Mac版，iOS版，还是Android版，我们都需要做以上的这些事情，只是具体的实现方法不同而已。

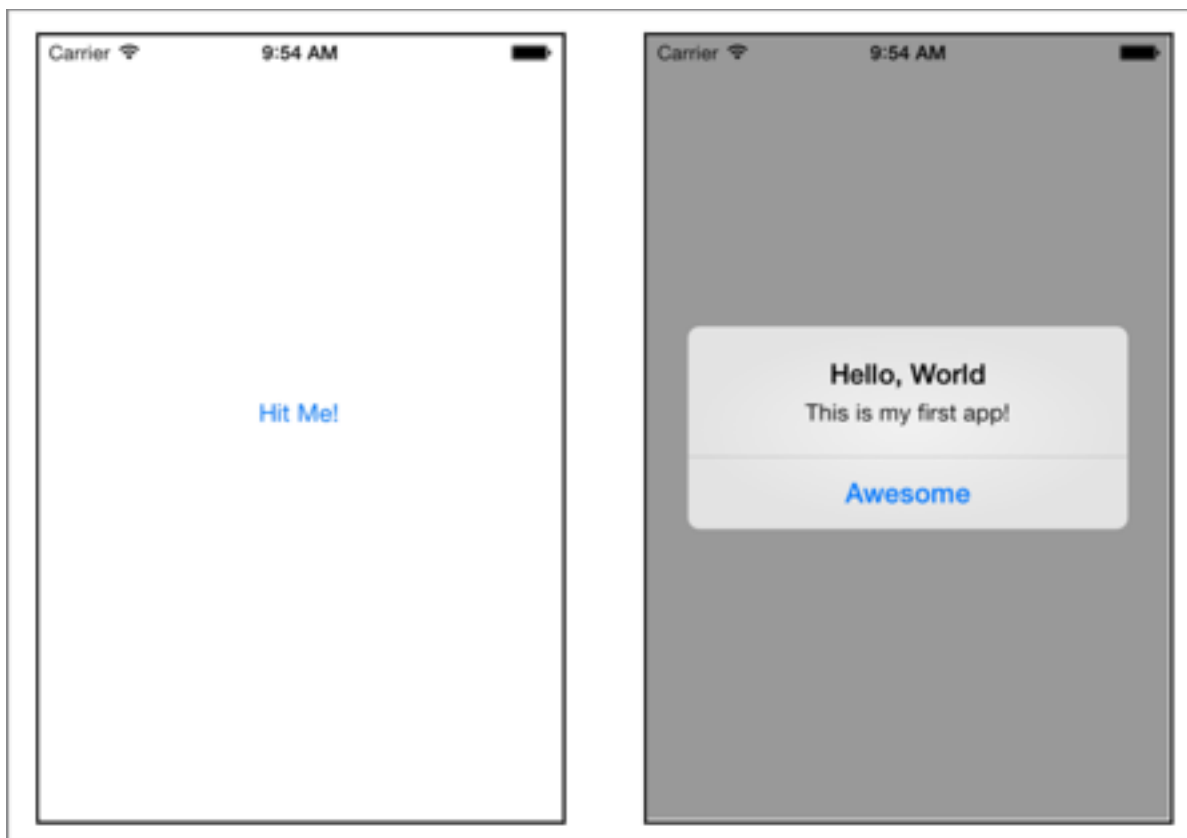
所以，在具体开发产品之前最重要的是明确自己要做哪些事情。

再次重申一下本系列教程的目的，与其说是教你如何做iPhone开发，不如说让你如何具备程序猿的思维方式，从而可以和他们更好的沟通。

给自己一个前进的理由-完成只有一个按钮的应用

我们唠唠叨叨了这么长时间，却没有实现任何一个实际的目标，这样可不行。接下来要给大家一个小小的奖励，用最简单的操作来实现只有一个按钮的应用。也就是to-do list中的第一条。当你触碰按钮的时候，会弹出一个提示信息。就这么简单，但却是整个游戏的基础。

这个小应用的实际运行效果如下：



好了，终于到了真正写代码的时候了！想来你已经有了macOS High Sierra操作系统，并且已经下载安装了最新版本的Xcode(目前是9.0)，这真是极好的。如果你用的是之前的版本，请务必升级到最新版本。如果没有这两样，下面的看也白看。

重要的事情强调三遍：一定要从官方下载！一定要从官方下载！一定要从官方下载！

在iOS11s和之前的版本间存在着某些巨大的差异，可能有些公司还在用老的Xcode和iOS版本，但作为新手来说，就没必要了，一切从新开始。

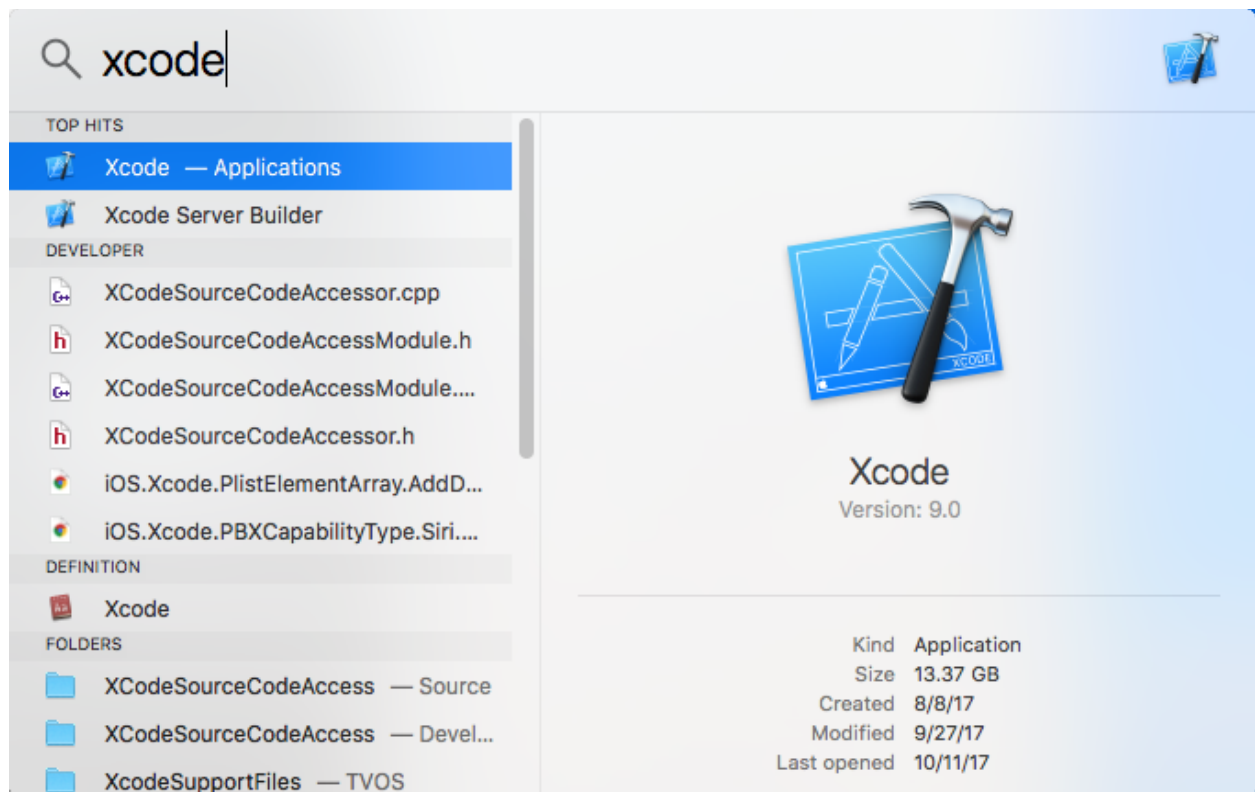
此外，Swift是一门全新的语言，它自身的进化非常快速且巨大，因此如果你的Xcode版本比较陈旧，那么哥可不敢保证这教程里面的代码都能顺顺溜溜的跑下去~同样的，为了保险起见，哥不建议你使用beta版的开发工具，只采用从Mac App Store里面下载的正式版本。

还等什么呢，别害怕，只要你还能看能写，下面的事情就能搞定：

Step1: 打开Xcode。

考虑到你可能第一次接触iOS开发，连Xcode都不知道怎么打开。那么多废话两句。你可以打开Finder（Mac系统的‘我的电脑’），然后进入Applications这个目录，然后找到Xcode，就可以打开了。

如果你是Mac老用户，最简单的方法就是点Mac最右上角的Spotlight（有点象放大镜的一个小图标），然后输入Xcode，回车，就可以了。



当然，如果你直接把Xcode的图标放在下方的Dock里面是最好。方法很简单，右键单击Dock上Xcode的图标，然后选择Options-Keep in Dock，这样Xcode以后就直接显示在Dock中了。

打开Xcode后,首先看到的是欢迎界面，喜迎你的到来。

Step2



Welcome to Xcode

Version 9.0 (9A235)



Get started with a playground

Explore new ideas quickly and easily.



Create a new Xcode project

Create an app for iPhone, iPad, Mac, Apple Watch or Apple TV.



Clone an existing project

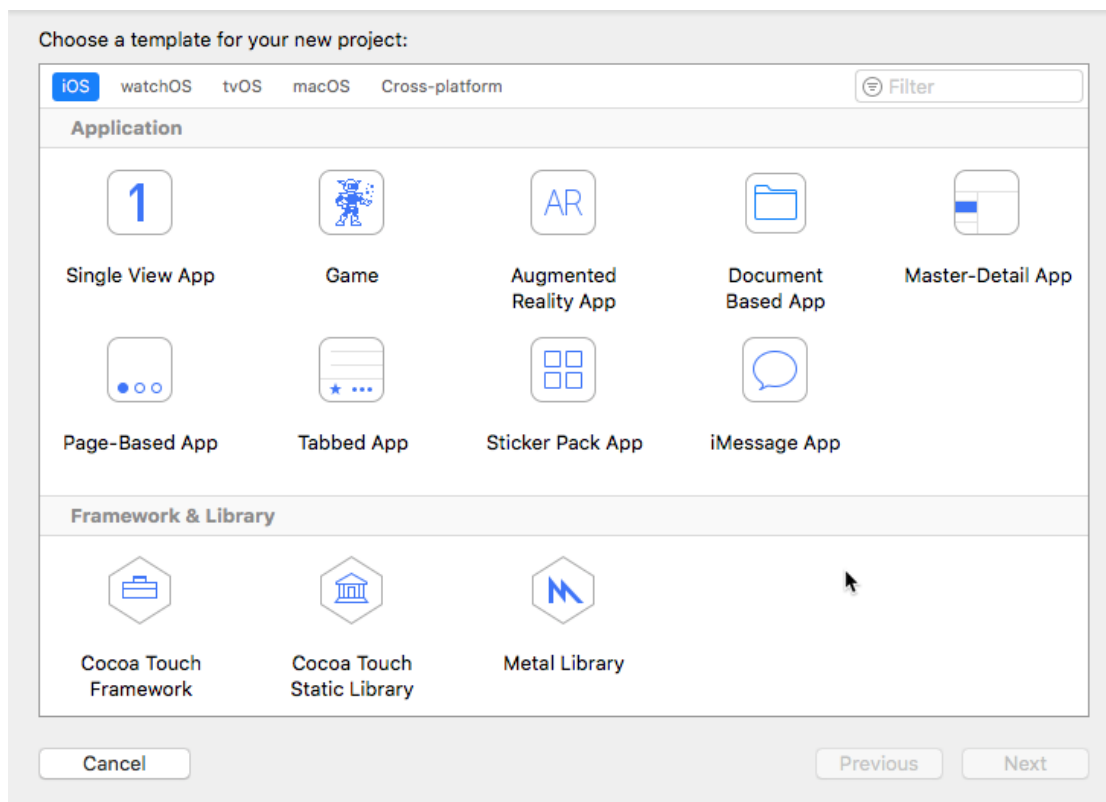
Start working on something from an SCM repository.

No Recent Projects

Open another project...

这时候点击左侧的Create a new Xcode project，会打开一个辅助界面，帮你选择所需要的模板。

因为我们这个系列的教程是iPhone开发（我的专栏里面还有Watch开发~），所以要手动切换到iOS，然后在Application下面选择Single View App。



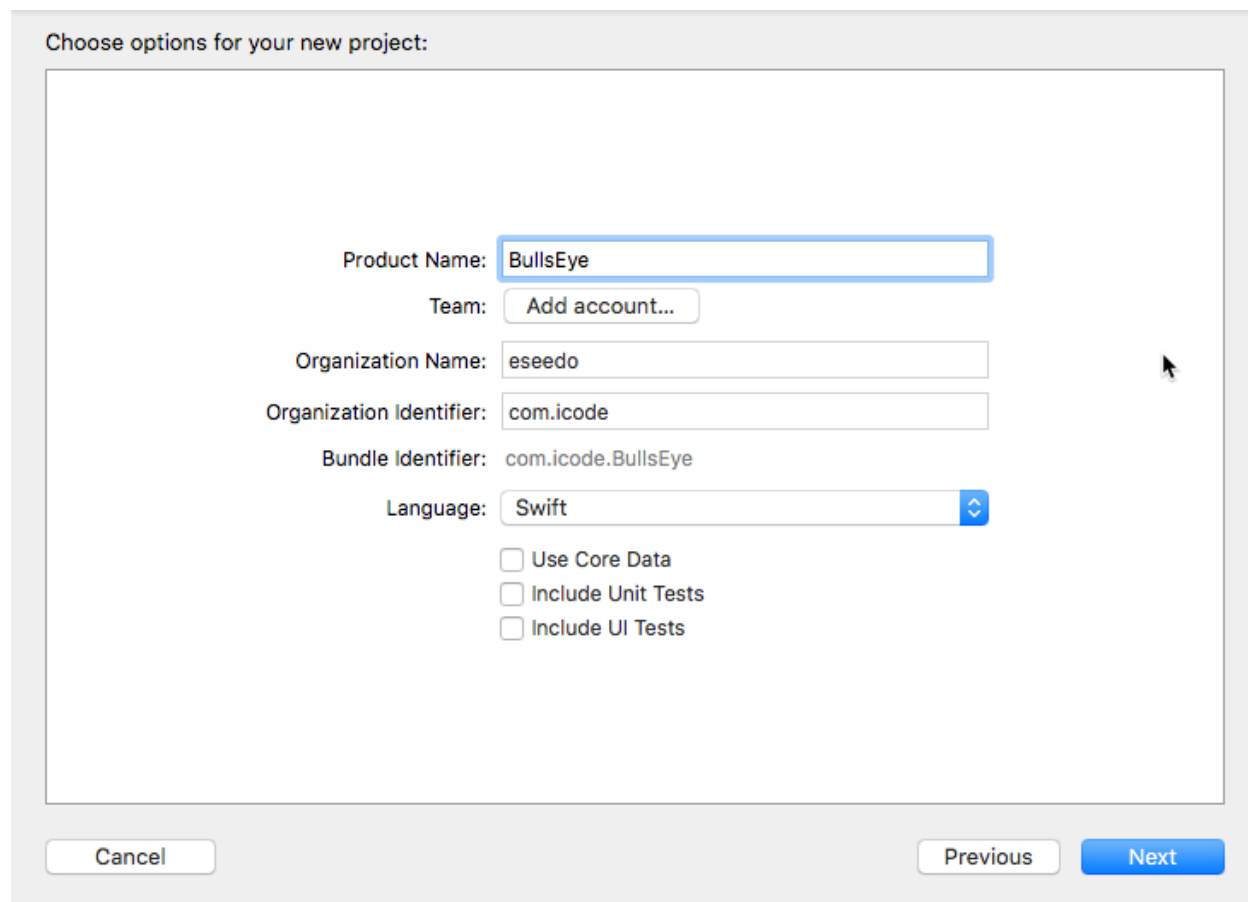
点击左边的不同图标和文字，你会看到不同的模板。所谓的模板是Xcode为了帮开发者节省时间而预先配置好的一些项目。每个从模板中创建的新项目都默认包含了开发应用所需要的很多资源文件。模板的好处就是让你省了很多敲代码的工作。如果你用的不是Xcode9.0，或许这里所显示的模板会有所差异，但至少就“王者打靶”这个游戏来说，所需要的模板在这里是有的。

那么问题来了，为毛我们这里要选择Single View App。其实很简单，我们现在只需要一个最基本的模板，也就是所谓的单界面应用模板，至于其它复杂的东西一律都用不到。对于其它各种类型的模板，随着学习和实际开发工作的进行，你会逐渐接触到的，这里就不浪费时间了。

Step3:

点击Next，你会看到一个新的提示界面，里面需要输入和选择一些信息，你可以照着下面的来：

注意在Language这里要选择Swift!!!这是因为我们的新系列教程将采用Swift编程语言。



The image shows the 'Choose options for your new project' dialog box in Xcode. The dialog has a title bar and a main content area with several input fields and checkboxes. The 'Product Name' field is highlighted with a blue border and contains the text 'BullsEye'. The 'Team' field has a button labeled 'Add account...'. The 'Organization Name' field contains 'eseedo'. The 'Organization Identifier' field contains 'com.icode'. The 'Bundle Identifier' field contains 'com.icode.BullsEye'. The 'Language' field is a dropdown menu set to 'Swift'. Below these fields are three unchecked checkboxes: 'Use Core Data', 'Include Unit Tests', and 'Include UI Tests'. At the bottom of the dialog are three buttons: 'Cancel', 'Previous', and 'Next'.

Choose options for your new project:

Product Name: BullsEye

Team: Add account...

Organization Name: eseedo

Organization Identifier: com.icode

Bundle Identifier: com.icode.BullsEye

Language: Swift

☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

Cancel Previous Next

1.Product Name:

这里填BullsEye, 当然你也可以取个其它的名字, 不过尽量避免在项目名称中使用空格或者特殊符号。

2.Team:

如果你已经加入了苹果开发者计划, 那么这里就会显示你的团队名称。这里我们先暂时不进行相关设置, 在随后的教程中会对此进行处理。

3.Organization Name:

你的梦想团队名称, 或者你自己的名字。这里我填的是eseedo, 你可以随便填上其它的。

4.Organization Identifier:

这里是com.icode, 这个标识符是我用来识别自己的APP, 通常是反转域名, 只要符合com.xxxx.xxxx这种命名习惯就行。。

不要过分担心这里填的信息对不对, 因为这些都是可以修改的。

5.Language:

这里填Swift, 不过如果你想尝试Objective-C, 也可以自己去体验。

6.Use Core Data:

这里不需要勾选, Core Data是iOS的一个类库, 目前还不需要涉及到。

7.Include Unit Tests:

不需要勾选, 目前我们还不会涉及到单元测试

8.Include UI Tests:

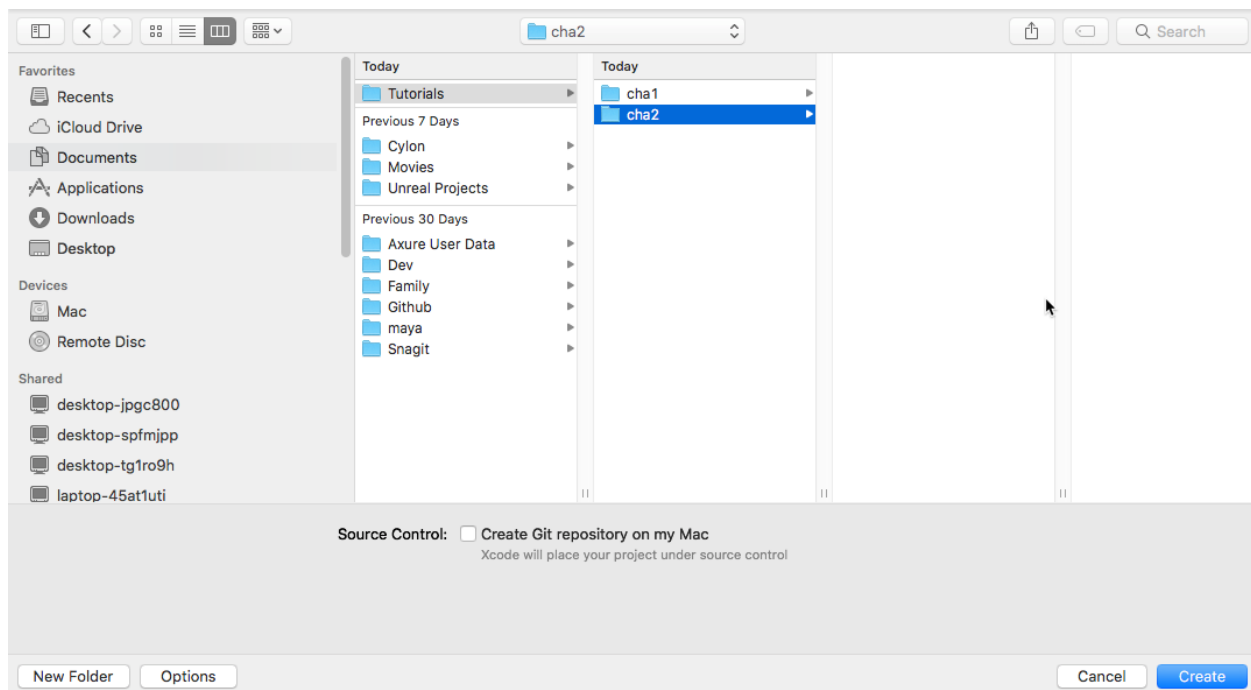
不需要勾选, 目前我们暂时还不涉及到。

Step4:

信息完成后点击Next, 此时Xcode会要求你提供保存项目的位置。

选择一个位置保存项目文件, 具体在哪里取决于你的个人喜好。

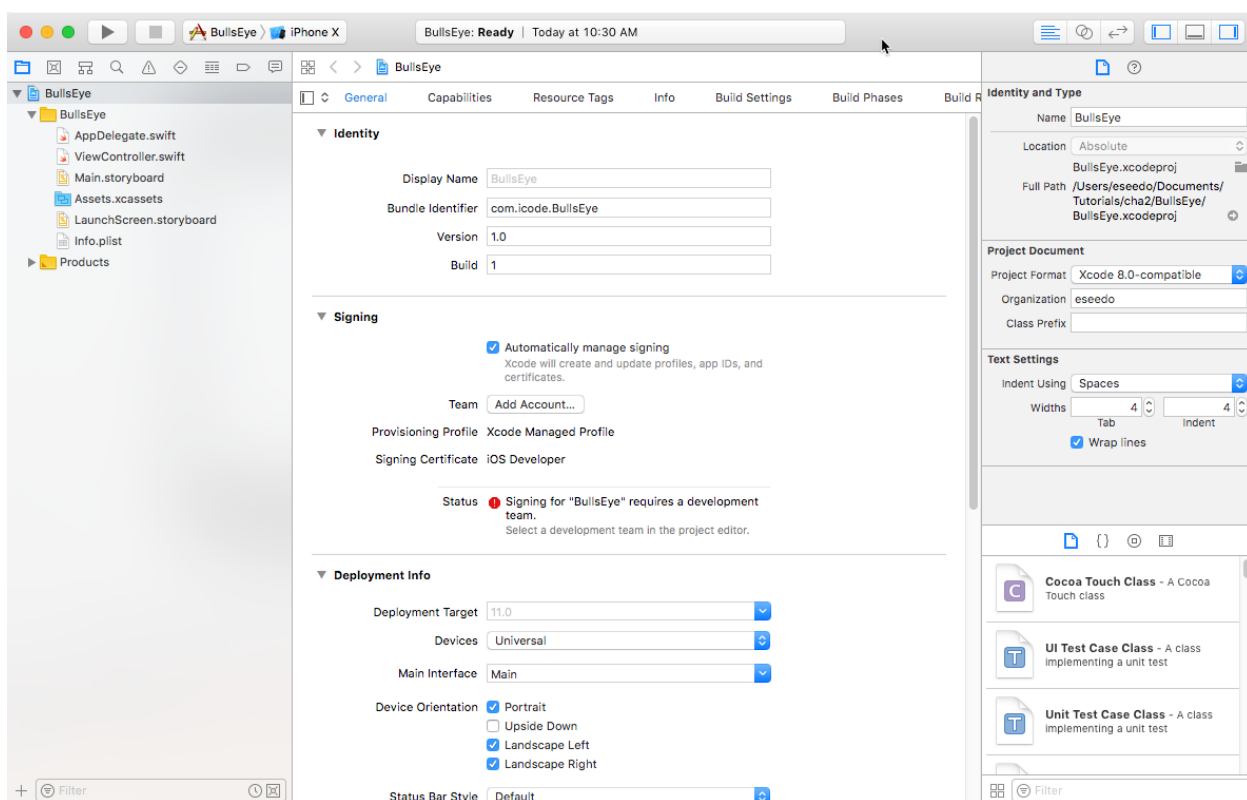
选择完后, Xcode会使用Product Name为项目自动创建一个新的文件夹。



在底部有一个提示框询问你‘Create Git repository on My Mac’。现在你先不用管它，它是用来进行Git版本控制的。关于Git版本控制系统，我们在后面会慢慢接触，它可以帮助开发者更好的管理项目源代码。

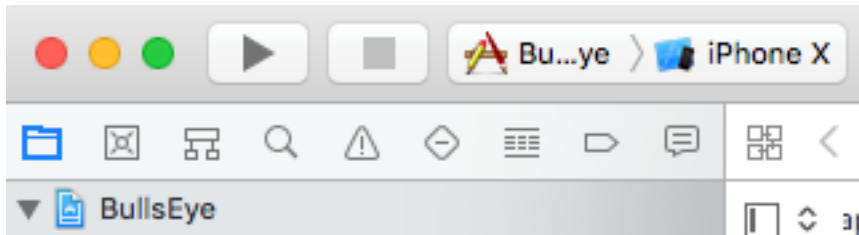
Step5: 一切准备就绪，点击Create完成创建。

点击Create，Xcode会基于Single View Application这个模板创建一个名为BullsEye的新项目。完成后的屏幕显示如下：

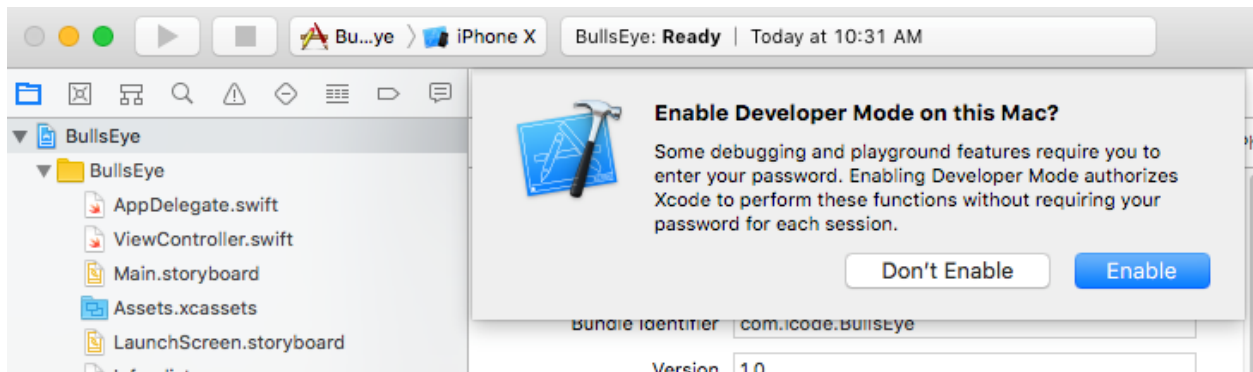


如果你的Xcode版本不同，那么这里的一些现实细节会有所差异。不过Don't panic，这些小小的差异并不影响到我们的学习。

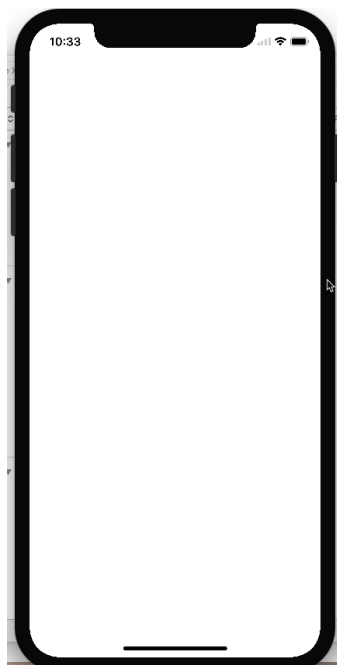
Step6.此时点击最左上角的Run按钮（红绿灯右边的三角）



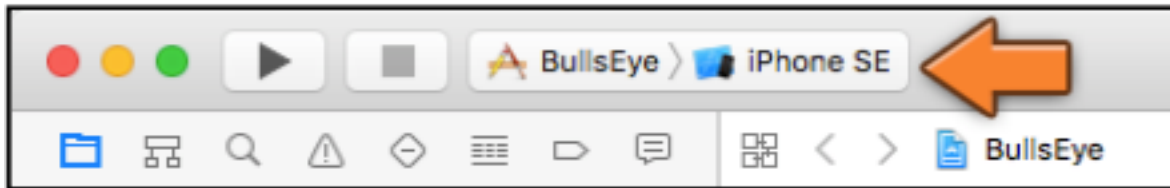
注意：如果你首次使用Xcode，那么会提示你enable developer mode。只需要点击Enable,然后输入密码让Xcode做相应改变就好了~



Xcode会开始努力的工作，然后在iOS Simulator模拟器上运行这个全新的应用。当然，首次启动的时候会看到下面的画面。



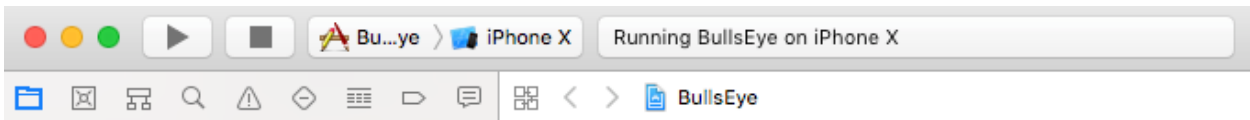
当然，实在没什么看头，不过是一个灰白的界面而已，放在白色的文字背景上甚至有点区分不出来，而且你也没法和它交互，但不管怎样，这是我们一个真正的里程碑。如果Xcode提示Build Failed或者Xcode cannot run using the selected device，那么要检查下左上角是否选中了某个具体的iOS模拟器，而不是iOS Device：



如果此时此刻你的iPhone正使用USB线连接到Mac上，Xcode可能会尝试直接在你的iPhone上运行，不过此时多半会无法顺利进行，因为还需要一些额外的设置。在本教程结束的时候我会教你怎么在自己的iPhone上进行试。现在先别管这些吧~

Step8 停止运行

在Run按钮的旁边是Stop，当你看烦了的时候就可以点它停止应用。你可能会尝试在Simulator选Home键的方式来退出应用（选择菜单栏的Hardware-Home），但这样做其实不能真正的退出应用。它只是从Simulator的屏幕上消失了，但应用仍然悬停在Simulator的内存中，正如在真机上一样。在你按下stop之前，Xcode中会显示“Running BullsEye on iPhone X”。



当然，你也可以不停止运行应用，直接返回Xcode来修改源代码。但直到你再次按下Run按钮前这些修改都不会实际生效。

问：当我们按下Run按钮时发生了什么？

首先，Xcode会将项目的源代码从Swift语言编译（通俗点可以叫翻译）成iPhone（或模拟器）可以理解的机器语言。虽然说开发iPhone应用的编程语言是Swift或是Objective-C，但iPhone自己可理解不了这种语言。因此需要一个翻译的过程。

编译器是Xcode的一部分，它可以帮助你將Swift的源代码转换成可执行的二进制代码。同时它还会整理收集所有组成应用的其它资源-资源文件，图片，storyboard文件等等，并把这些东西打包到一个叫“应用程序束(application bundle)”的东西里面。

以上的整个过程又称之为building(编译)项目。如果在这个过程中有任何错误（比如拼写错误），build会失败。如果一切正常，application bundle(应用程序束)会被拷贝到模拟器或设备中，同时会运行该应用。所有的这一切都是由强大的Run按钮来搞定的。

如果你看不懂上面这段话，也不要着急，对于目前来说这并不重要。甚至对很多编程老手来说这些东西也不是那么的重要。

在继续我们的iOS开发探索之旅之前，先多废话几句iOS的进化史。

从iOS1到现在的iOS11，iOS系统版本经过了多次更新。和android不同，通常新的iOS系统版本覆盖率要占绝大多数。现在11.0出来了，预计很短时间内大多数新产品会选择支持11.0及以上。

iPhone OS 1.0（2007年，这个年代还没有iOS哦，iOS是后来改的名字）

iPhone OS 2.0（2008年）

iPhone OS 3.0（2009年）

iOS4.0(2010年，首次将iPhone OS更名为iOS)

iOS4伴随iPhone4一起出现，打造了历史上最受人欢迎的一代苹果手机，同时也是乔帮主亲自发布的最后一款手机。

iOS5（2011年）

iOS5的发布伴随着iPhone4S，而就在这一年乔帮主去世了，在iPhone4S发布的第二天早上。。。

iOS6(2012年)

此时帮主已经往生西方极乐世界，厨子Cook接过了苹果的海盗旗，然后从良成了海军。就在这一年，iOS之父Scott Forstall被Cook驱逐出了苹果。

从Cook时代开始，苹果对中国市场的重视与日俱增。



iOS7(2013)

2013年WWDC大会上，Cook宣布iOS7的问世，由乔帮主在苹果的灵魂伴侣Jony爵士一手打造。iOS7的设计风格变化堪称革命，彻底抛弃了乔帮主和Scott钟爱的拟物化设计风格，转为微软力导的扁平化设计，让很多开发者一时难以接受。

iOS8(2014)

这是一次完全属于Cook的发布会。三年的转型，外界评价纷纭，不管怎样，苹果已经彻底Cook化了。

iOS9(2015)

从2014年10月30日Tim Cook出柜以来，苹果的产品定位发生了彻底的变更，那就是开始跟时尚结合。Apple Watch是表现最明显的一款产品，而对于iOS的各种设备来说，用Rosegold玫瑰金来吸引中国的万千女性也只有懂时尚的人才想得到吧~

iOS10(2016)

对于iOS10来说，最大的亮点莫过于开放了SiriKit，以及对电话功能、地图功能和iMessage功能更强大的支持。

iOS11(2017)

不用多说，iOS11最大的特色就是增加了ARKit和Core ML，从而直接支持增强现实和人工智能的相关特性。对很多消费者来说，目前可能还没有感觉到ARKit和Core ML的强大威力，但个人觉得iOS11所支持的这两大特性其实也是未来十到二十年信息产业的根基。

再说下设备分辨率的问题吧

目前支持iOS 10的主流iOS设备的分辨率：

2732*2048:iPad Pro 12.9-inch
2436*1125: iPhone X

2048*1536: iPad Pro 9.7-inch, iPad 4,iPad Air, iPad Mini 2,3,4

1920*1080: iPhone 8 Plus, iPhone 7 Plus, iPhone 6s Plus, iPhone 6 Plus

1334*750: iPhone 8,iPhone7, iPhone 6s, iPhone 6

1136*640: iPhone SE, iPhone 5s, iPhone 5c, iPhone 5, iPod touch 6

关于分辨率的问题，我们在后面的学习过程中还会详细讲解。

好了，稍事休息之后，让我们继续iOS开发的学习吧。

添加按钮

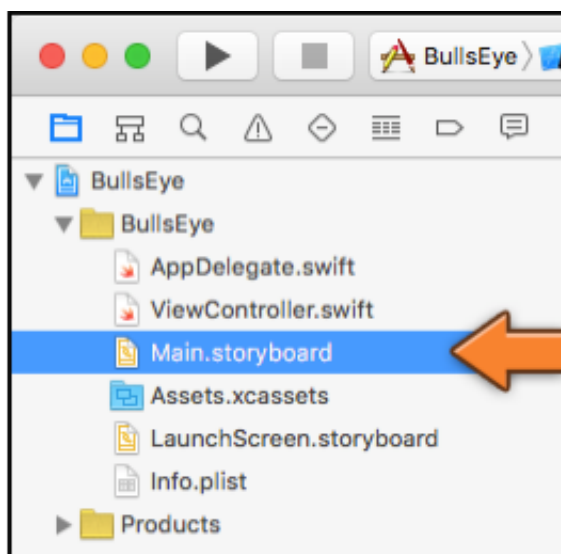
现在这个界面实在是惨不忍睹，比凤姐都难以入目。接下来让我们添加一个按钮上去。

首先回到Xcode中来。在Xcode窗口的左侧是所谓的导航区域(Navigator area)。顶部的小图标决定了哪个导航器是可见的，当前的导航器是Project Navigator（项目导航），也就是说它展示了项目中的所有文件列表。你也可以点其它图标看看，但为了简单起见，这里先不解释那么多。

在项目导航中，文件的组织形式和实际硬盘里的项目结构可能有关，也可能不同。你可以根据个人喜好随意拖动它们的位置。关于项目中各个文件的作用，你暂且不去管它，后面再说。

Step1

在当前的项目导航中，找到一个名为”Main.storyboard”的文件，点击选中它。

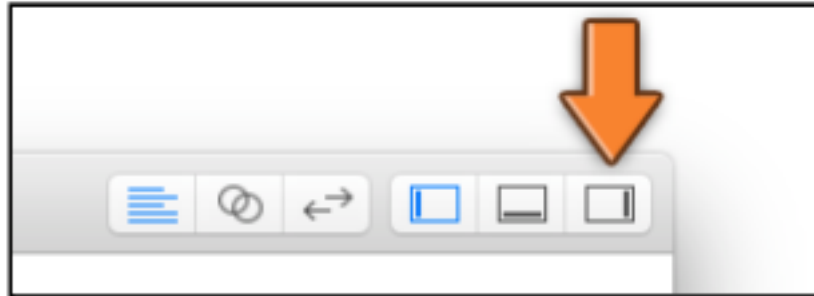


在一瞬间，你刚刚看到的界面就会切换到Interface Builder了。在旧版本的Xcode中，Interface Builder和Xcode是分开的，各自独立的工具，从Xcode4开始，Interface Builder被整合到Xcode里面去。因此这里的所谓切换到Interface Builder并不是说打开了新的工具，而是说切换到了一个新的开发界面。而这个开发界面的主要作用是放置我们所制作产品的交互界面中的可视化元素。曾经有一些牛人和熟手拒绝使用Interface Builder,xib和storyboard之类的可视化编程工具，而习惯于手写所有的视觉控件。这没什么，但通常来说，如果是专门针对iOS开发的应用，使用可视化编程工具可以大大提高开发的效率，尤其适合在代码层级不是那么熟练的新手。

注意，从iOS7开始，用模板创建的项目默认采用storyboard，而非之前的xib文件。关于storyboard和xib的区别这里不多说，以后慢慢接触吧。

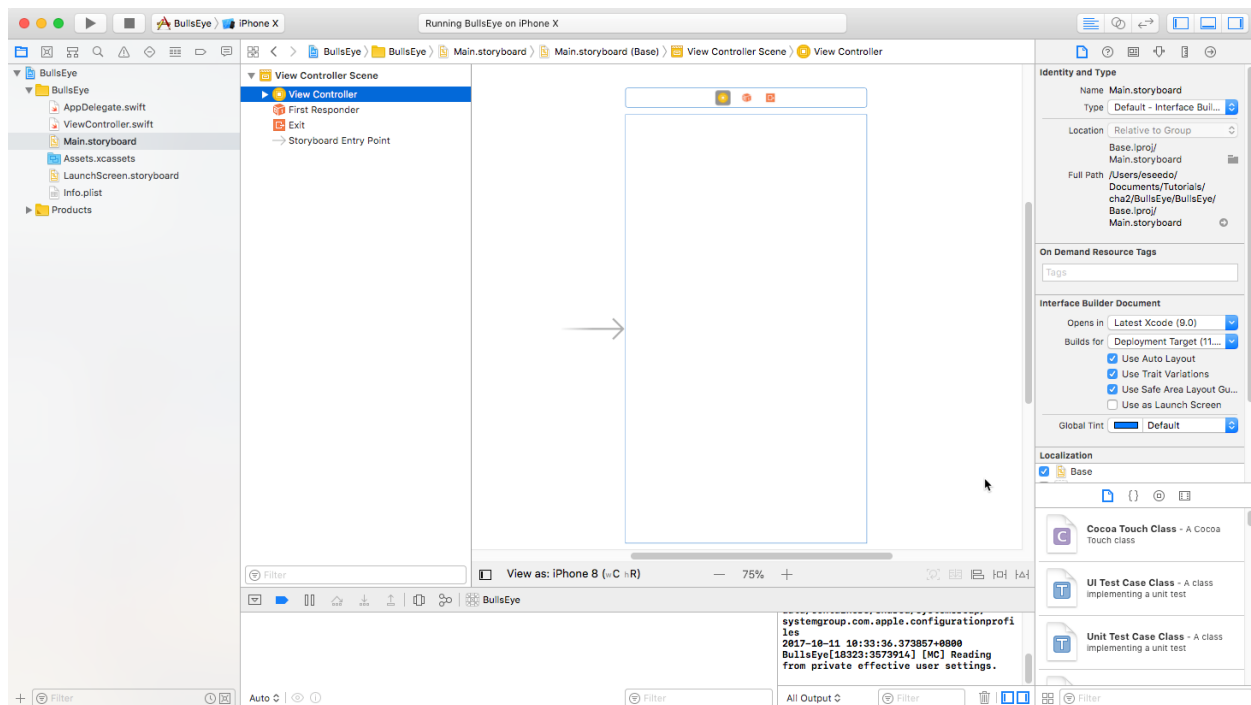
Step2

此时你可以点击Xcode右上角的”Hide or show utilities”按钮，如图所示：



还可以点击其它按钮，通过这些按钮可以更改Xcode的界面布局。而这个”Hide or show utilities”按钮可以显示或隐藏Xcode右侧的新面板。

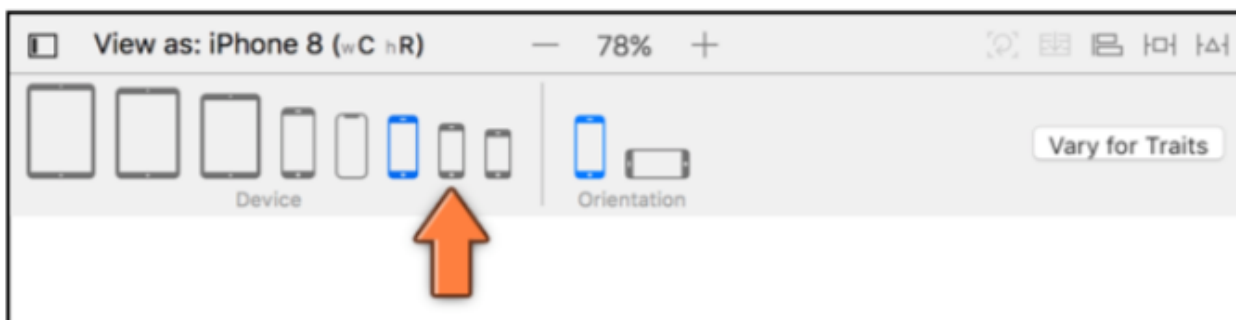
此时界面如下：



这里就是应用的storyboard。storyboard（故事板）里面包含了应用中所有界面的设计，并通过箭头显示界面之间的跳转关系。当然目前来说这个storyboard里面只包含了一个单一界面。

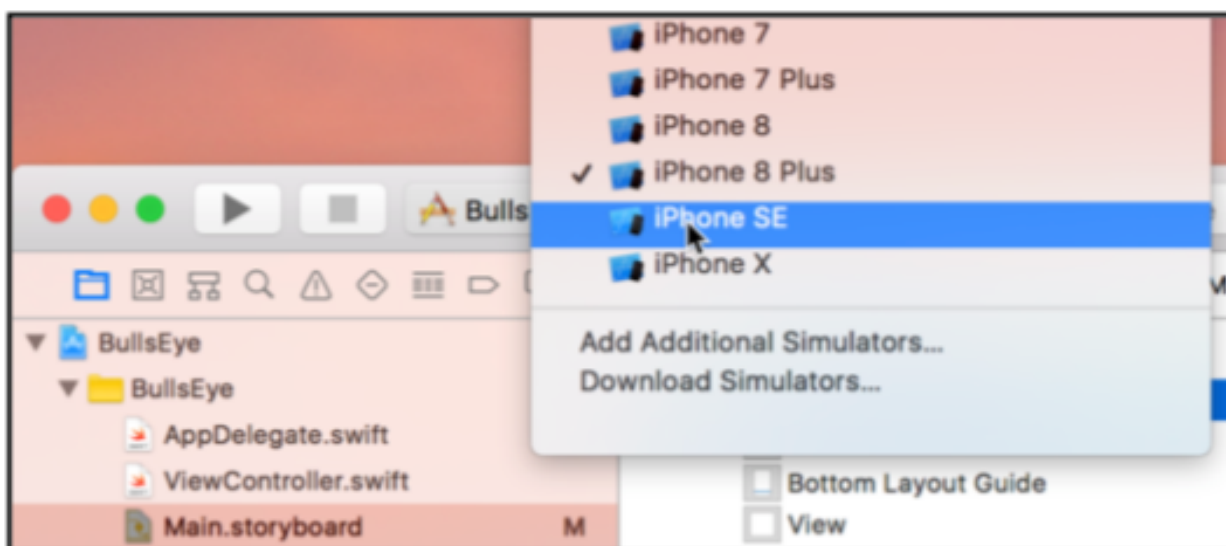
不过这里的设备尺寸是iPhone8的大小。为了尽可能简化操作，第一个应用的设计是针对iPhone SE的。后面我们会学习如何让应用适配到iPhone 8 Plus，甚至是iPhone X。

在Interface Builder窗口的底部点击“View as: iPhone 8”以打开如下所示的面板：



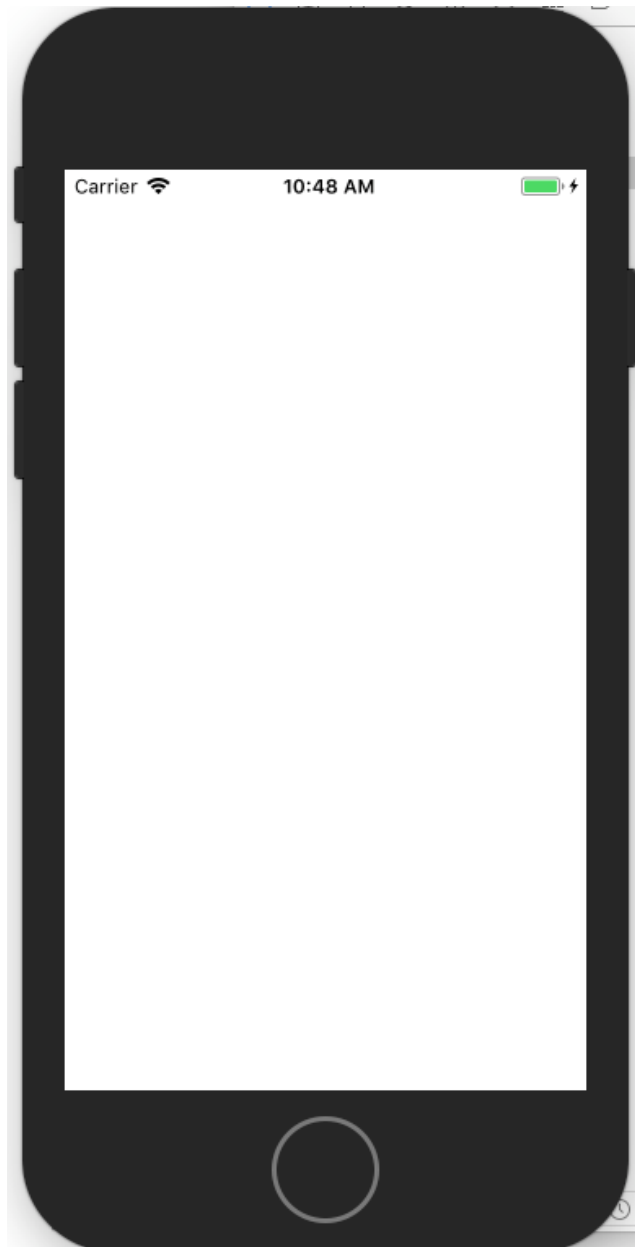
选择iPhone SE,也就是倒数第二小的iPhone。此时界面中的矩形会变得稍小一些。

接下来在Xcode的工具栏中，在Stop按钮的附近选择BullsEye > iPhone SE。



现在再点击Run按钮跑跑看。

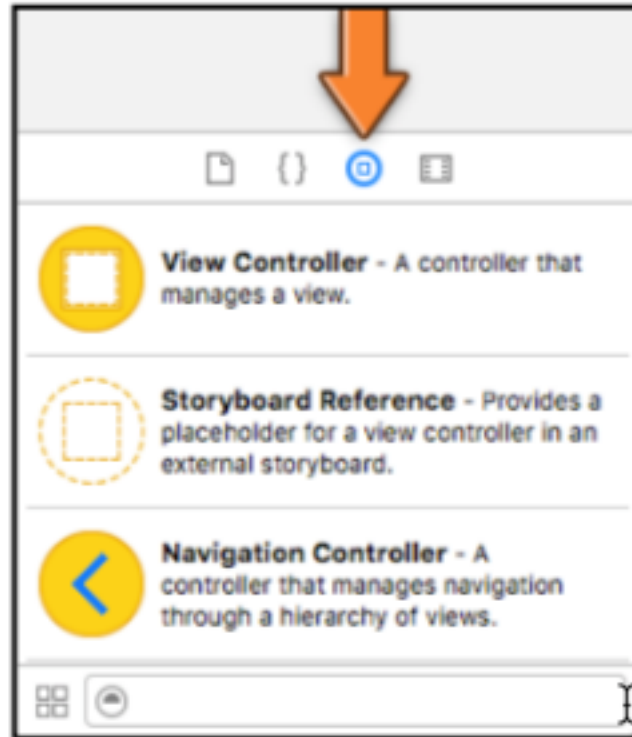
然而并没有什么卵用~ 只不过模拟器的尺寸变成了iPhone SE的而已。



Step 3在Object Library中找到按钮控件

在Xcode右侧面板的下方你会看到Object Library(也即对象库，只需选中下图中的第三个小按钮即可)：

上下滚动，可以找到一个名为Button的项目。也可以直接在最下面的搜索栏里面输入button来搜索。

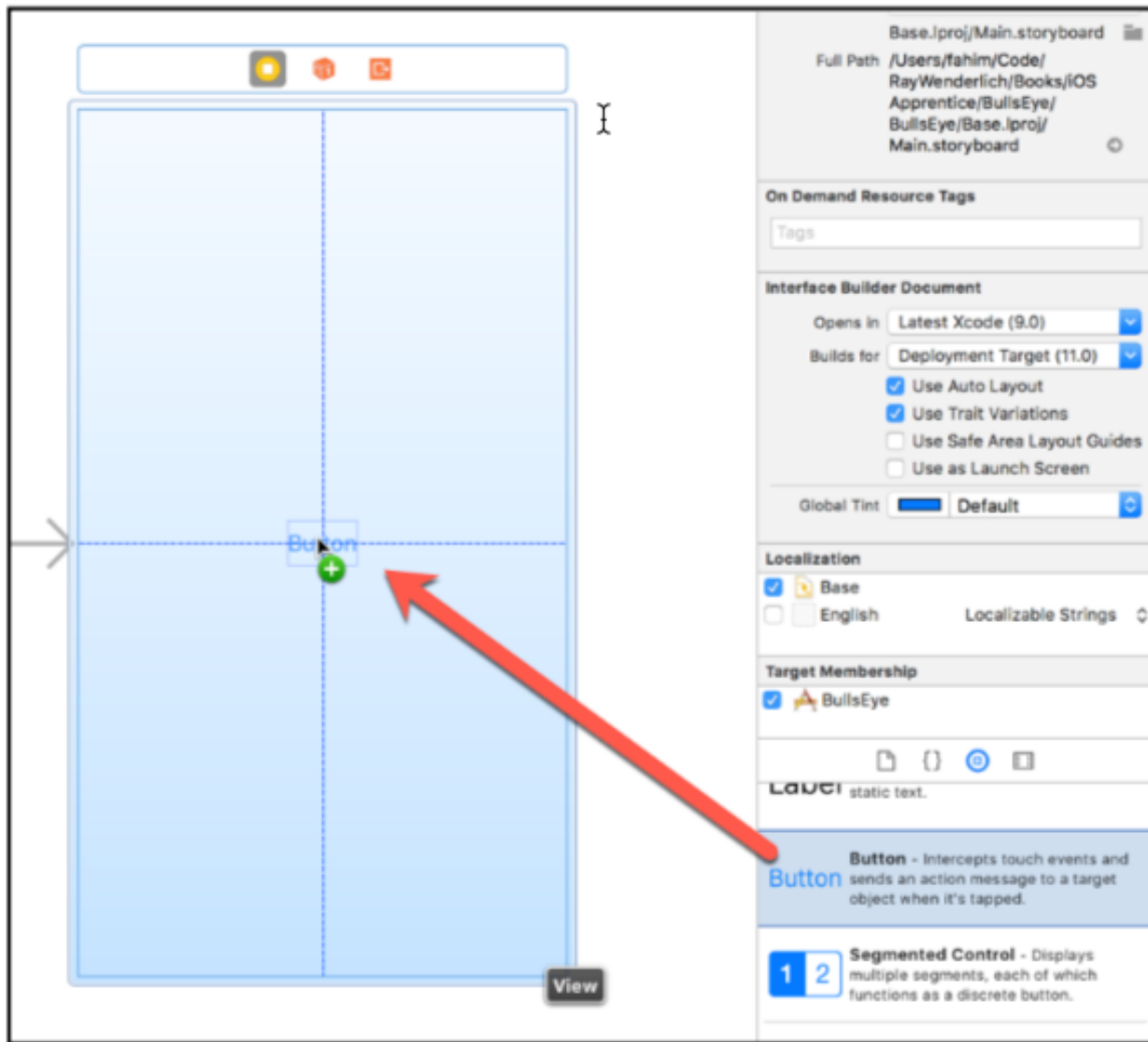


Step 4 点击Button,把它拖曳到工作区的白色视图上。

这样我们就添加了新的按钮，而且非常简单，只是拖曳放下而已。对于iOS应用中的所有其它用户界面元素，都可以采取类似的方式操作。后续我们还将大量重复这样的操作，很快你就会对此非常习惯。这就是传说中的可视化编程。

关于可视化编程的几句废话

可视化编程是随着面向对象编程概念的兴起而变得普及的。对于旧石器时代的程序猿来说，开发工具远没有今天的各种花哨SDK好用，部分程序猿甚至习惯于在文本编辑器里面直接写就代码。当然，这样做的好处是让这些旧石器时代开发者奠定了非常牢固的代码基础。直到现在仍然有很多程序猿偏好于在非常简单的界面中写代码。但这种类似写文章的编码方式也有很多不足之处，首先它要求程序猿对编程语言，各种类库,API非常的熟



悉，不说可以达到左右手互博的程度，起码写10行代码不会轻易出一个错。如若不然，放到编译工具里面调试的时候会让人头大。随着面向对象语言的兴起，以及开发工具的进步，各平台的代码编辑器都开始变得智能起来，可以第一时间发现代码的低级语法错误和拼写错误，大大提高了程序猿的效率，也解救了大量的经常丢三落四宅男的时间。但仅仅对代码级别的敏感还不足够，人们逐渐发现很多的标准控件是没有必要每次都重复去编写的，特别是一些基本的按钮，视图，和用户交互元素都可以直接重用。也因此现在基本上所有的大型编程工具都提供了简单易用的可视化编程环境。比如Visual Studio, Xcode, Netbeans, JBuilder等等，包括Eclipse也可以通过插件来实现可视化编程。当然，还有Adobe全系列的产品也是如此。

目前的主流游戏开发引擎Unity，Unreal更是如此。

补充一点，对于非程序猿出身的产品和设计人员来说，可视化编程其实也是非常有利的原型工具。很多人问iOS应用开发最好的原型工具是什么？什么样的回答都有，比如Axure，Pop等等专业的原型设计工具，但大部分人都不会提到Xcode。

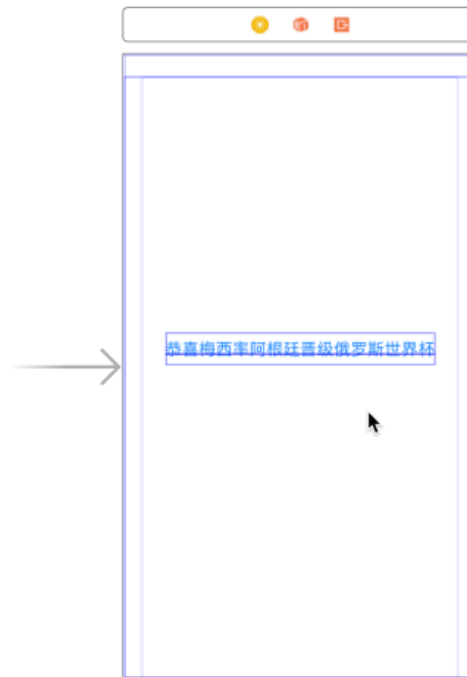
其实最好的iOS应用开发原型工具是Xcode这个可视化开发工具。你只需要了解一点点Xcode的使用方法，甚至不需要学会一行代码，就可以直接做一个带有视觉效果的单一界面应用，放在设备上实际查看展示效果。如果你再稍微懂一些编程的东西，那么可以在很短的时间里实现多界面的应用视觉元素。至于应用本身的各种复杂功能，在制作原型的时候可以直接无视。一个懂Xcode的iOS产品经理才能更好的跟开发人员沟通。一个懂Xcode的UI和设计人员简直如虎添翼，可以第一时间来精细调整界面布局，在设备上查看交互效果，而不是停留在纸上谈兵的阶段。

Step5. 放入其它控件，体验下可视化编程的快感

比如标签，滑动条，切换开关按钮，什么乱七八糟的都拖到视图上吧。

这一刻你不再是一个程序猿，而是一个画家，你要做的事情也和Photoshop上设计界面没有区别。

Step6. 双击按钮，编辑其中的内容为“恭喜梅西率阿根廷晋级俄罗斯世界杯”之类的，随便你



当然，有时候你可能会看到在按钮的周围有明显的边框，其实完全不用担心。对于处女座的同学，可以手动关闭。在Xcode的菜单栏中选择Editor-Canvas-Show Bounds Rectangles即可开启或关闭这种边框。

完成之后，点击Xcode左上的Run按钮。现在应用会在模拟器中运行，并显示一个按钮。当然，这个时候你去碰它的话它还是不会理你的！



在继续之前，先唠叨几句废话~

在学习iOS编程的时候，要适当的了解一个事实，那就是一些新的系统版本中的功能在运行旧系统版本的模拟器或设备上不一定能跑起来。同样，新的系统版本也会抛弃或者禁用旧系统版本的一些功能，因此在运行新系统版本的模拟器或设备上也不一定能跑起来。肿么办？首先新版本被删除或禁用的功能苹果都会重点说明，尽量避免使用。然后就是要设备上多测试。模拟器上的测试基本上都能找到此类问题。而设备上的测试则能发现一些与硬件特征相关的问题，比如内存不足，硬件特性调用等等。

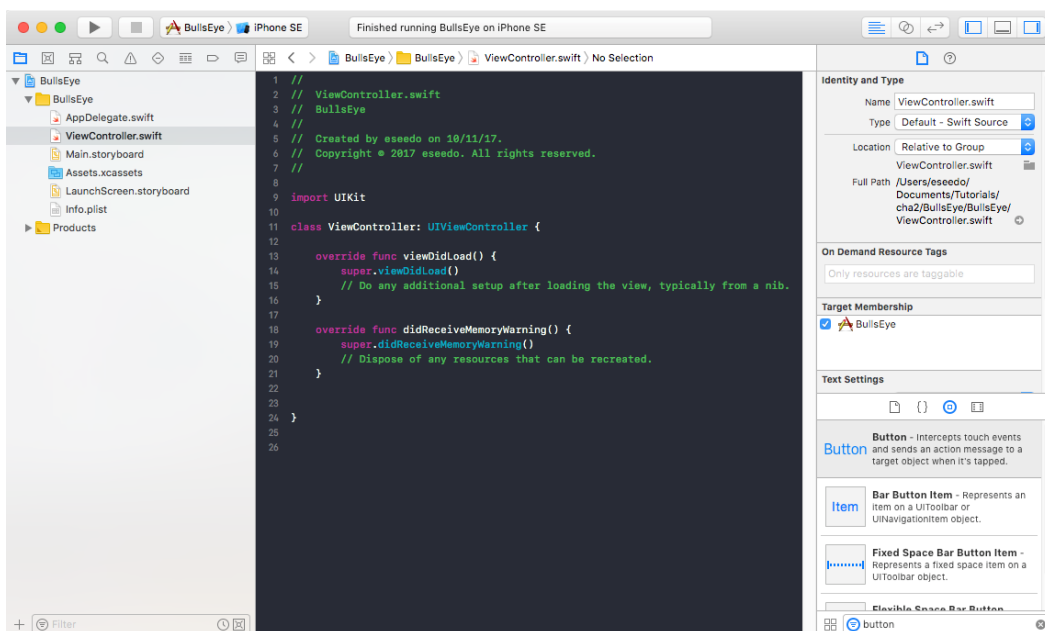
关于自动保存

通常来说Xcode会帮你自动保存，但考虑到它本身的脆弱性（特别是iOS之父Scott Forstall的黯然离去），还有停电外星生物干扰猫爬上桌之类的不可抗力事件，最好还是时不时按下Command + S组合键。基本上我会每半分钟不自觉就按一次Command + s,无论之前在做什么。

源代码编辑器

现在界面上的这个按钮还不能和玩家产生交互，因此让我们来实现一个弹出式对话框。在最终完成的游戏版本中，对话框会显示玩家的当前得分，不过这里我们来个比较随意的，比如只显示一行文本“梅西上演帽子戏法拯救阿根廷”

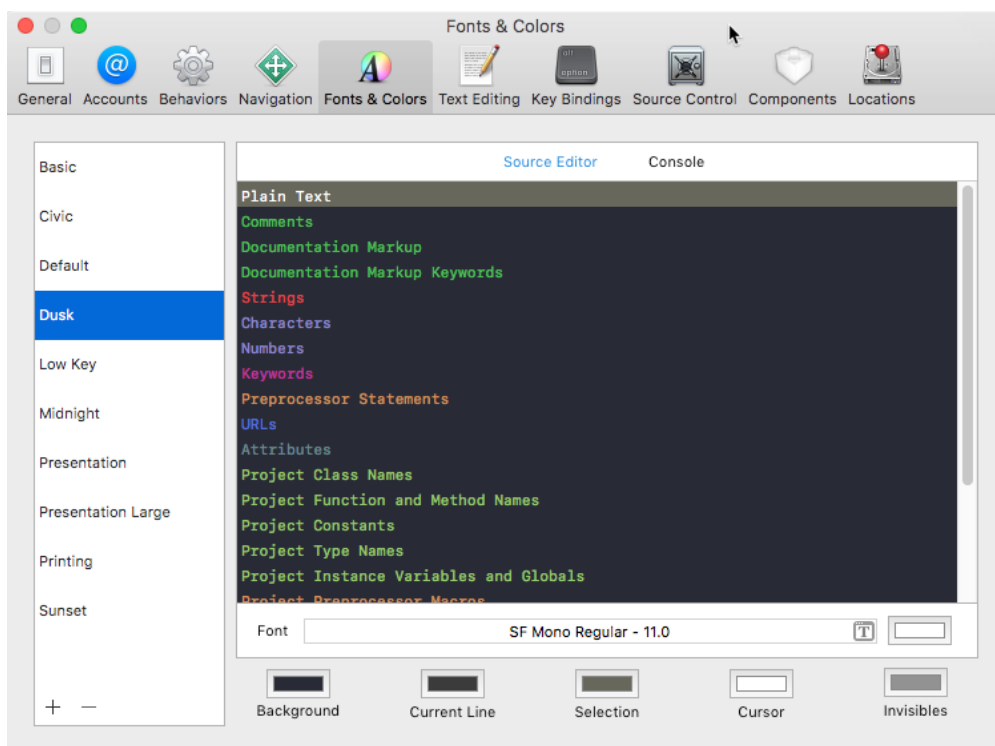
再废话一句，第一次看这个教程的时候新手可能有一些不理解的地方，但别钻牛角尖，先看完，对一些重要的概念我会重复展示，直到你熟悉为止。别没学会走就想成飞猪。你要做的就是，跟着学。



Step 7在Xcode左侧的项目导航（也就是文件列表）中找到ViewController.swift，鼠标点击它。

然后刚才的Interface Builder（界面编辑器）就消失了，出现在你眼前的是世界上最恐怖的东西-代码。是的，这些花花绿绿的字母和数字组合就是我们这个应用的Swift源代码。如果你的Xcode编辑器窗口中没有在每行代码前显示行数，那么同样可以从菜单栏中进行设置。从Xcode顶部菜单栏中选择Xcode-Preferences...-Text Editing，切换到Editing选修课，然后在Line Numbers前面勾选上即可。

慢着，你可能会问，为毛你的编辑器背景是黑色的呢？这个也是个小小的技巧，毕竟程序猿一天到晚对着电脑看代码很伤眼睛，所以可以自己设置自己喜欢的背景来调节。点击Xcode—>Preferences...，然后切换到Fonts&Colors，就可以选择自己喜欢的色彩搭配了。我这里选的是Dusk，你看着办，我觉得Midnight也不错~



回到我们的代码。

在ViewController.swift中，在最后一个}花括号前面添加以下代码：

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

此时ViewController.swift中的完整代码如下：

```
//  
// ViewController.swift  
// BullsEye  
//  
// Created by eseedo on 10/11/17.  
// Copyright © 2017 eseedo. All rights reserved.  
//  
  
import UIKit  
  
class ViewController: UIViewController {  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view, typically from a nib.  
    }  
  
    override func didReceiveMemoryWarning() {  
        super.didReceiveMemoryWarning()  
        // Dispose of any resources that can be recreated.  
    }  
  
    @IBAction func showAlert(){  
  
    }  
  
}
```

你对这段Swift代码感觉如何？估计什么感觉都没有。在我告诉你答案之前，有必要介绍一下view controller（视图控制器）的概念。

什么是view controller（视图控制器）

之前我们打开过Main.storyboard文件，然后在Xcode内置的Interface Builder里面用拖曳的方式添加了一些控件。很简单，不过是灰白背景上有一个孤零零的按钮而已，或许你还加了点别的什么，不过这好歹也算是用户界面。然后刚才我们还在ViewController.swift里面添加了人生的第一行处女代码。

这两个文件，Main.storyboard和ViewController.swift共同组成了一个view controller（视图控制器）的设计和实现。

在开发iOS应用的过程中，我们要大量用到视图控制器。视图控制器，顾名思义，就是控制一个视图的工具，或者说管理一个单一界面的工具。

让我们来举个例子说明吧。比如说我们有个简单的菜谱应用。当你打开这个应用的时候，它的主界面上会列出所有的菜谱。当你触碰某个菜谱的时候，就会打开一个新的界面，里面显示了详细的文字介绍，做法，还有令人垂涎欲滴的美味食品照片。主界面和具体的菜谱界面分别由各自的视图控制器来管理。



这两个界面的作用各不相同。左边的Recipe list是菜谱清单，其中列出了几种不同的菜谱名称。而右边的Recipe Details界面则会显示某种菜谱的具体内容。这样你就需要至少两个视图控制器，其中一个知道如何显示清单，而另一个则需要知道如何显示图片和菜谱说明。现在可以提前透露一点，iOS应用开发的一个原则是，应用中的所有界面都必须有自己的视图控制器。

现在我们的这个应用只有一个界面（白色背景加无厘头的按钮），因此只需要一个视图控制器。而这个视图控制器的名称是ViewController。Main.storyboard和ViewController.swift文件共同实现了该视图控制器。

简单来说，Main.storyboard文件包含了视图控制器中的界面设计，而ViewController.swift则包含了视图控制器的功能-也就是让用户界面工作的逻辑，使用Swift语言来编写。

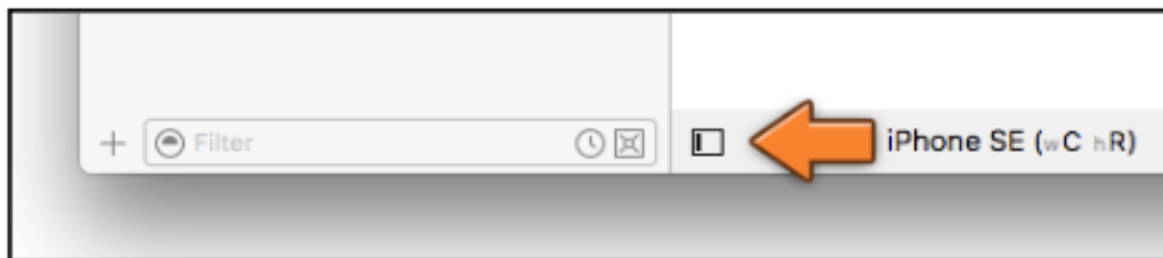
因为我们在创建项目时选择了Single View App模板，Xcode会自动为我们创建一个视图控制器。之后我们还将添加一个新的界面，并创建一个属于自己的视图控制器。

Step8 创建关联

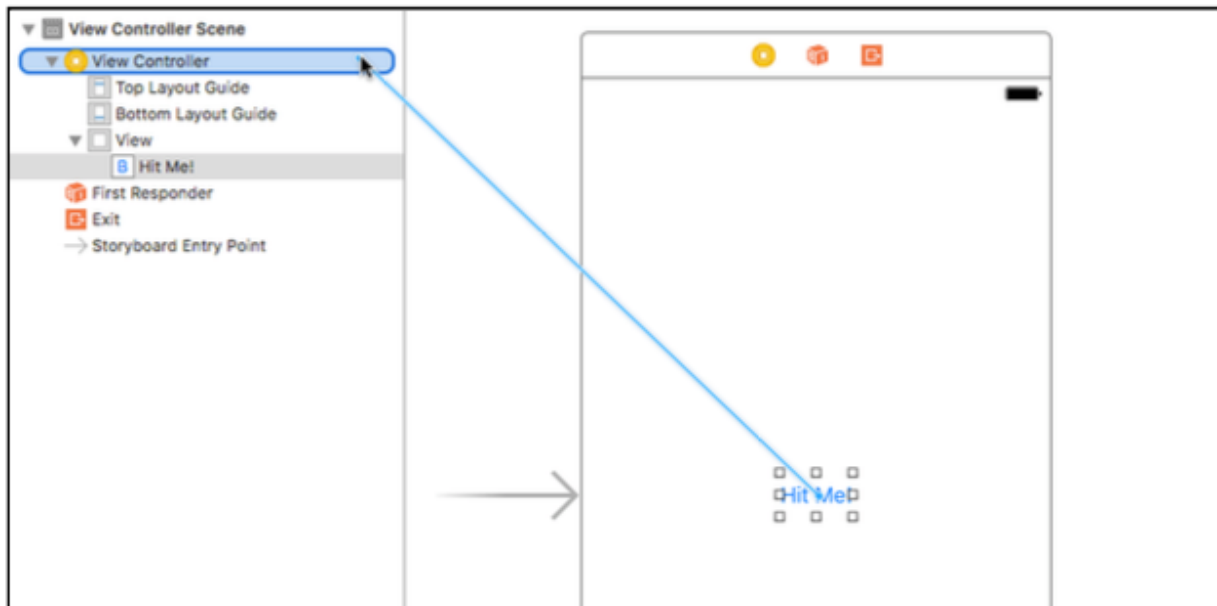
我们刚刚添加到ViewController.swift中的代码可以通知Interface Builder，当前的视图控制器有一个'showAlert'动作，触发该动作将显示一个弹出警告框。现在我们需要将按钮和动作关联在一起。

回到Xcode，点击Main.storyboard以回到Interface Builder。

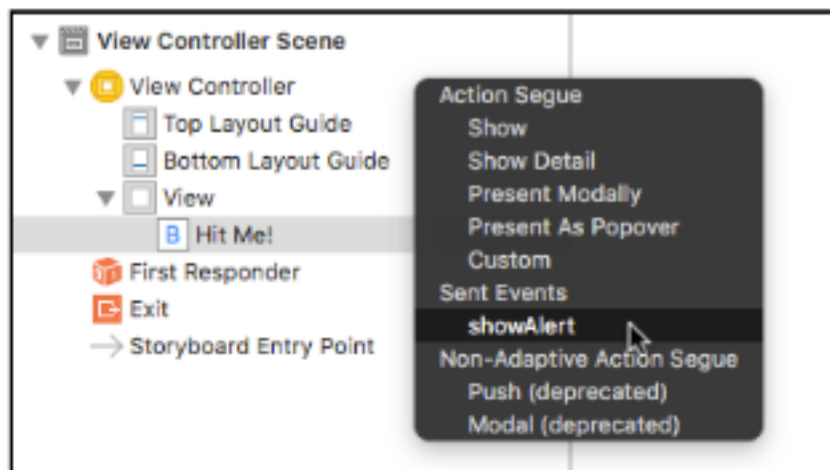
此时在Interface Builder的左侧应该显示一个Outline pane（轮廓面板），其中将列出storyboard中的所有视觉元素。如果看不到该面板，可以点击Interface Builder画布左下角的小切换按钮。



点击刚才的按钮'恭喜梅西率阿根廷晋级俄罗斯世界杯'（或者其它内容比如Hit Me），按住Ctrl键，从按钮到View Controller上拖出一条线，你可以看到一个蓝色的线条从按钮一直到View Controller。（如果不按Ctrl键，也可以点击右键拖动的方式，但是记住不要在中途松开鼠标）



当线条拖到View Controller时，松开鼠标键，然后会显示一个小的弹出菜单。

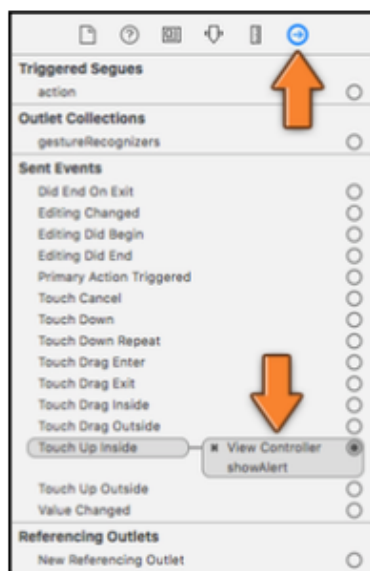


这里包含了两个部分，一个是'Action Segue'，另一个是“Sent Events”，每个部分下面又有几个选项。这里我们关心的是Sent Events下面的showAlert选项。该选项也是我们在ViewController.swift源代码中所添加的动作名称。

点击showAlert选中它。此时Interface Builder将会在界面的按钮和代码@IBAction func showAlert()之间创建关联。

通过刚才简单的拖曳操作，这样做就可以让Interface Builder帮你在按钮和showAlert动作之间创建一种关联。从现在开始，这样你触碰这个按钮，就会执行showAlert动作。这就是你如何让按钮和其它控件真正产生交互的方式：在Interface Builder中通过拖曳和某个控件创建一个关联。

我们现在可以查看已经创建的这个关联，选中该按钮，在Xcode右侧Utilities面板的Connections Inspector选项卡下可以看到。如图，点击小箭头形状的按钮就可以切换到Connections Inspector:

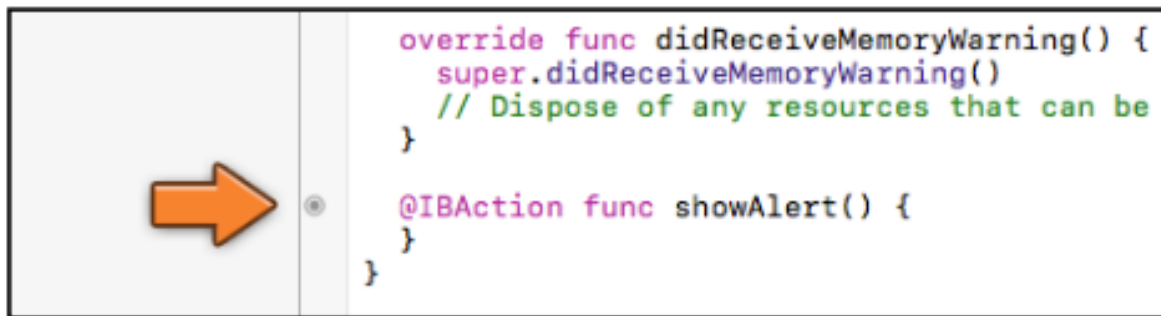


在上图中左侧的Sent Events部分，Touch Up Inside事件和showAlert动作关联了起来。如果你要删除这种关联，可以点击小x图标（现在别这么做！）。

此时我们也可以在Swift文件中查看这种关联。

在Xcode中选择ViewController.swift。

注意到在@IBAction func showAlert() {这行代码的左边有一个实心圆。



点击这个实心圆就可以看到代码所关联的动作。

让按钮知道自己做什么

现在我们已经有了一个带按钮的界面，同时把它和showAlert这个动作关联起来。从而当玩家触碰按钮的时候会执行这个动作。不过我们还没有告诉应用，这个动作具体的内容是什么。

在Xcode中选择ViewController.swift。

在@IBAction func showAlert()的花括号之间添加代码如下：

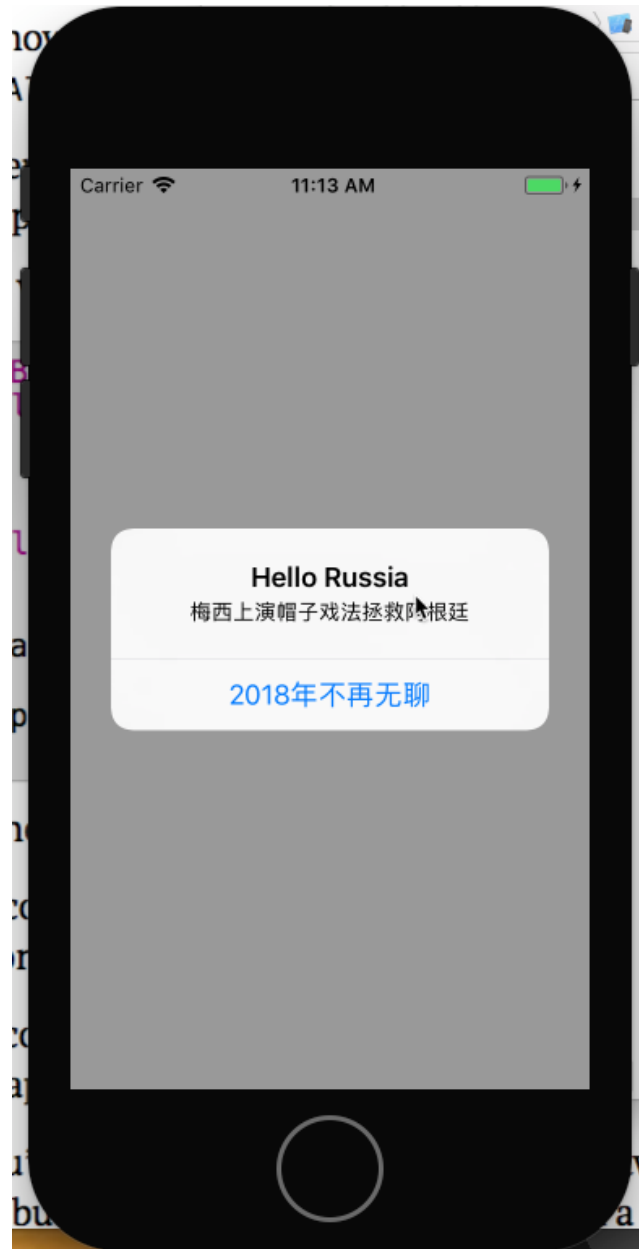
```
@IBAction func showAlert(){  
  
    let alert = UIAlertController(title:"Hello Russia",  
                                message:"梅西上演帽子戏法拯救阿根廷",  
                                preferredStyle: .alert)  
    let action = UIAlertAction(title:"2018年不再无聊",style: .default,handler: nil)  
    alert.addAction(action)  
  
    present(alert, animated: true, completion: nil)  
}
```

在{}的代码里面，创建了一个提示对话框，标题是“Hello Russia”，消息体是“梅西上演帽子戏法拯救阿根廷”，然后一个简单的按钮，上写“2018年不再无聊”。那么标题和消息体的区别何在？两个都会显示文本，不过标题通常字体大点，而且是粗体显示。

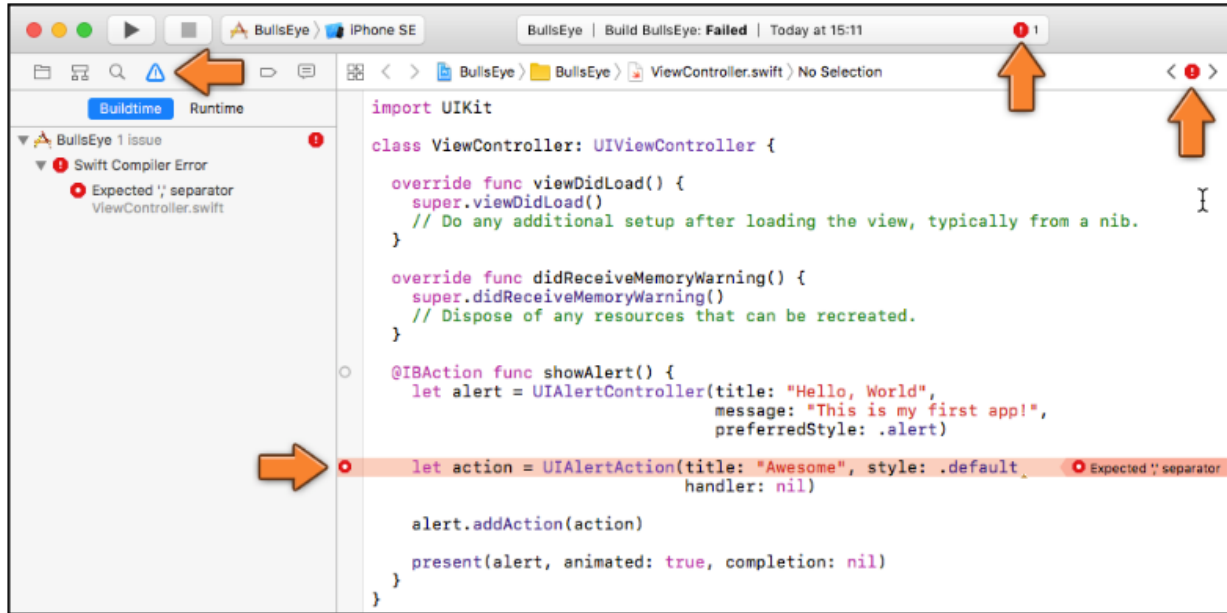
现在点击Xcode左上侧的Run按钮。

如果你没打错字的话，应用会在模拟器中运行修改后的应用。而且当你触碰按钮的时候会弹出一个提示对话框。

恭喜你，你的第一个iOS应用大功告成了，祝你早日卖肾成功。



或许这其中有很多东西你还是不懂，没关系，不管怎样，我们搞定了。



到目前为止，我们已经完成了清单上的前两项：在屏幕上放一个按钮，当玩家触碰按钮的时候显示一个提示对话框。好了，你现在可以泡上一壶茶好好休息一下了。别一下子吃太饱，慢慢来。

什么？出问题了？没搞定？

如果你没能成功的看到上面的对话框，反而看到红色的Build Failed提示，那么确保你没敲错代码。即便是最小的错误也可能让Xcode困惑不已。它所谓的智能感应毕竟还是有限的。在程序猿的人生中，看到红色的错误提示，然后如疯狗般的查找错误的来源消磨了有为青年的大部分美好青春。

作为产品经理或设计人员的你，应该可以体会到程序猿的痛苦了吧？

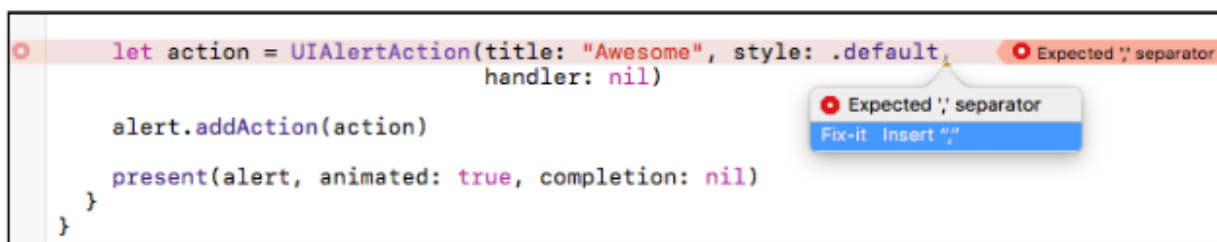
在编写代码的过程中，最常见的错误就是大小写问题。Swift编程语言是对大小写敏感的，也就是说AlertView和alertView对它来说是完全不同的东西。Xcode看到此类事物会给你一个<something>undeclared或是'Use of unresolved identifier'的红色提示。于是你崩溃了。。。据bug星球长老会的小道消息，80%以上的代码错误来自于大小写和误拼。。。

当Xcode告诉你"Parse Issue"或"Expected <something>"的时候，很可能你忘了在每行代码的最后加上一个花括号或是)。

在编码的过程中，诸如此类的小细节常常让人困扰不已。大小写，误拼，标点符号构成了代码错误的主力军。有些代码错误在调试的时候就会被发现，虽然让程序猿很头大，但至少还不至于祸害众生。但有些代码错误是在产品交付之后才被发现，代价就不用说了。幸运的是，一般如大小写错误，误拼，标点符号之类的错误很容易在编译的过程中被发现。

当Xcode发现某个错误的时候，会将左侧的项目导航面板切换成Issue Navigator麻烦导航栏（你可以通过上面的小图标切换回去）。这里列出了Xcode所发现的所有错误和警告信息。

比如在下图中我漏掉了一个逗号。



点击左侧的错误信息，Xcode会带你到相关的代码行，甚至还会提示你如何进行修改：

别指望每次出现错误的时候你都能享受到这种五星级服务，实际上大部分的代码错误都需要你手动寻找，或是使用一些技巧让Xcode帮你来定位。

关于错误和提示

Xcode使用红色来标示error（错误），用黄色来标示warning（警告）。错误是致命的（死神恩赐），如果出现编译错误，那么应用是无法运行的。警告则是提供了参考信息。Xcode的意思是，“你或许不想这么做，不过如果你实在想要这么做，那就做吧，只是后果自负哦。”

个人认为对待warning的态度应该和error一样严肃。尽量做到让自己的应用实现0 warning,0 error。即便不能，也要做好备注，提醒自己或别人。warning所在的地方以及可能导致的后果。

好了，如果没有找到错误，成功的看到提示，你今天的学习就可以到此结束了。
最后庆祝今天煤球王帽子戏法拯救了阿根廷队！！
哦，顺便忘了说，如果你输入网址messi.ai，也可以访问icode.ai~



最后送韩国mm福利一张。



你也是MM? 好吧，送你一张有创意的照片，权作安慰吧。



休息，休息一下吧。