

## 让不懂编程的人爱上iPhone开发(2017秋iOS11+Swift4+Xcode7版)-第16篇

恭喜你，你已经来到了本系列课程的最后一部分，看完最后的这篇教程后，我们的入门系列教程就算是结束了。

不知不觉，我们已经在学习iOS开发的道路上前进了这么多。如果你仔细读过所有的内容，无论是唠叨，YY实践，还是科普，那么你对iOS产品的开发应该有了一些基本的了解。

不过学无止境，而且在学习iOS开发的过程中，编码不是最重要的，设计和开发出满足用户需求的产品才是王道。永远记住这一点！无论是程序猿，产品经理或是设计师，都需要把这一点铭记在心。科技与艺术的结合才能打造真正优秀的产品，帮主早就为我们指明了前行的道路，至于如何做就看我们自己的了。

还是建议大家把英文水平提高下，多去看官方示例文档，多去stack overflow,多看github上的源代码。无它，唯手熟尔。当然，只追求手熟远远不够，这只能让你成为一个码农。在代码世界中，除了坚持写代码，更重要的还是要多思考，多学习。

在最后一章的内容中，我们将会涵盖以下内容：

- 1.支持更多设备分辨率
- 2.添加一些动态特效
- 3.设置应用的图标
- 4.设置应用的名称
- 5.在  上进行测试

接下来我们将一一进行讲解。

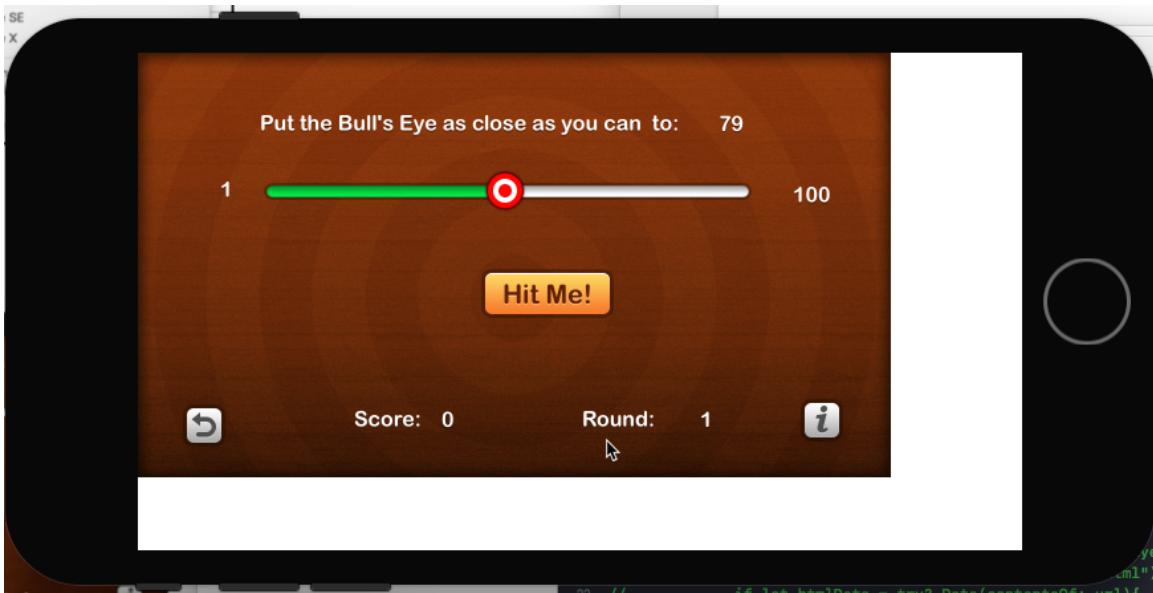
### 支持更多设备分辨率

在目前的设计中，我们是为iPhone SE设备准备的，那么对于更大屏幕的设备支持如何呢？）。

首先将Simulator从目前的iPhone SE更换为iPhone8或者iPhone8 Plus，甚至是iPhone X。

编译运行，你会发现呈现出来的效果不是我们想要的~

如果你觉得模拟器太大，可以用⌘-1, ⌘-2, ⌘-3来调整模拟器大小，看看效果。

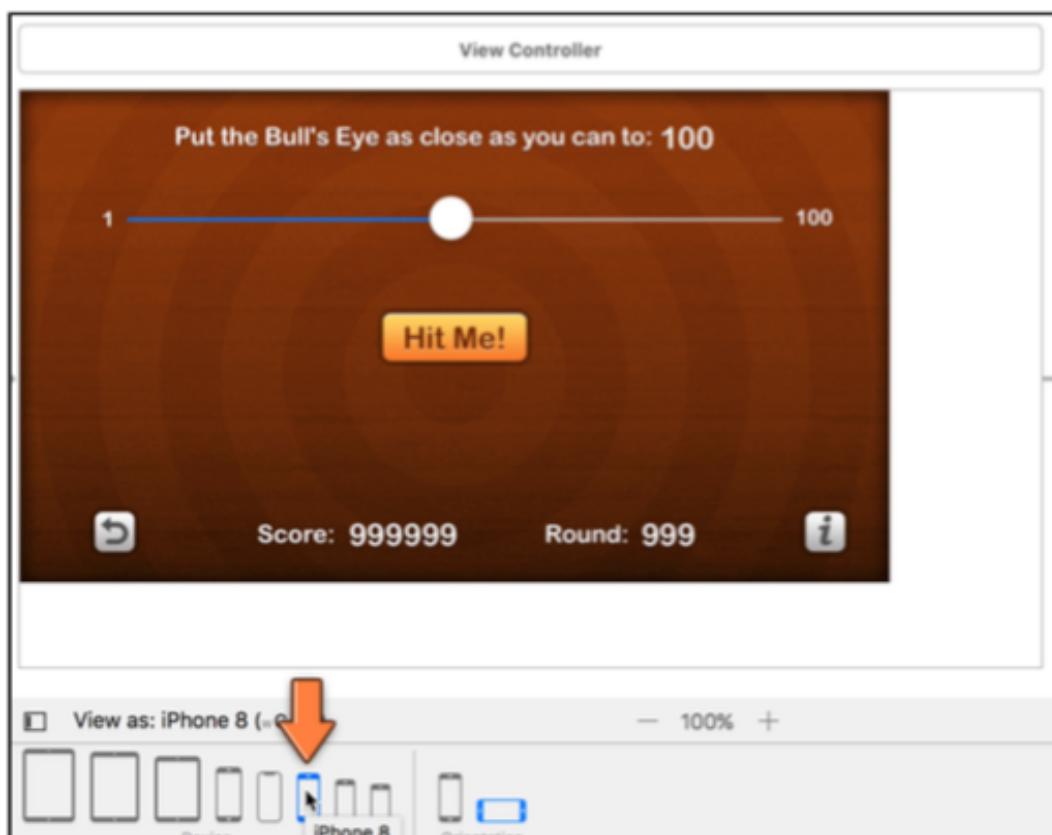


好吧，怎么说呢，看起来效果还不是很糟糕，当然也不是特别棒。

最大的问题就是，貌似留下了巨大的空白。如果你在真机上测试，那么也是一个德行~

是时候介绍iOS开发的一大利器-Auto Layout了！Auto Layout(自动布局) 是UIKit的一项核心科技，可以让我们的应用轻松适配多种设备分辨率，包括4.7寸，5.5寸，5.8寸的iPhone，以及iPad。

在Xcode中打开Main.storyboard，打开底部的View as: 面板，然后选择iPhone 8设备（别忘了把Orientation调整回landscape）。



首先让我们修复背景图片的显示问题。很明显，目前的背景图片无法填满更大的屏幕。

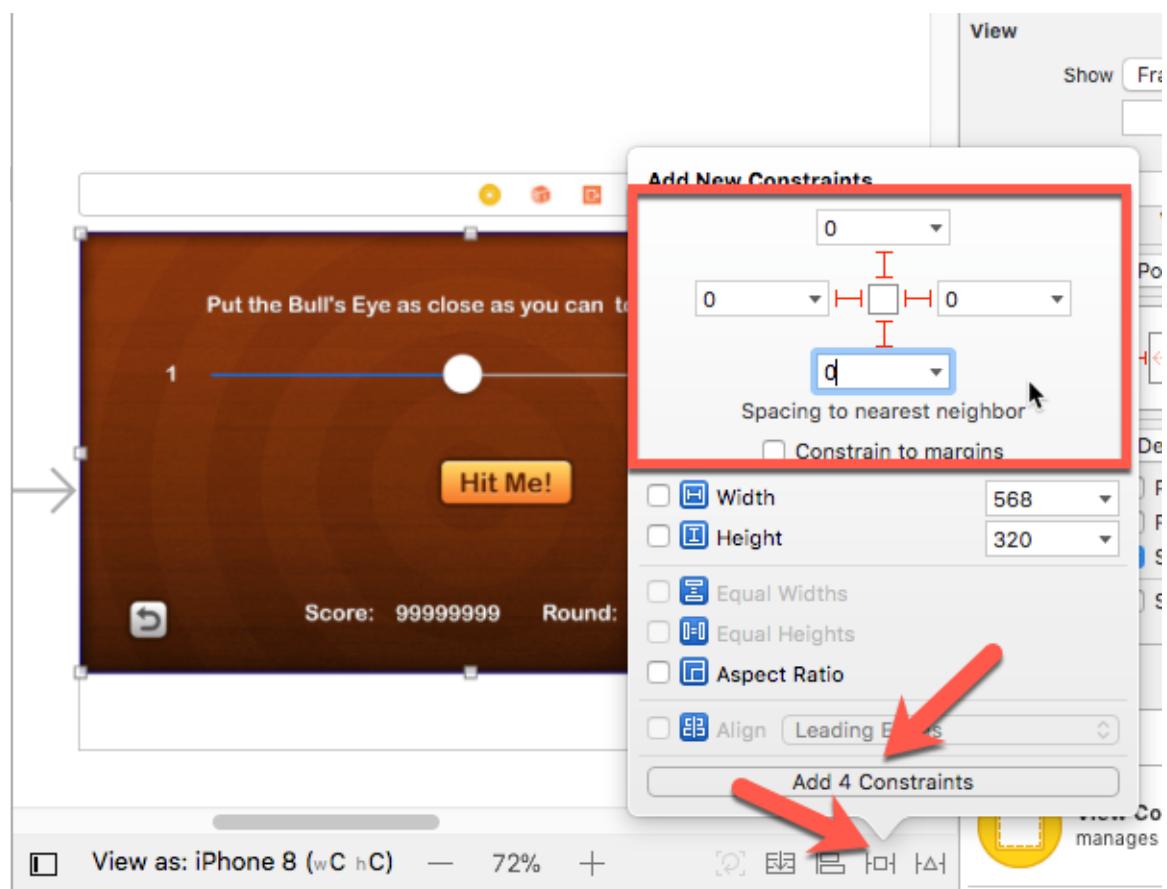
这个时候Auto Layout就可以发挥作用了。

在storyboard中，选择View Controller下的Background image view，然后点击Xcode底部的Add New Constraints。

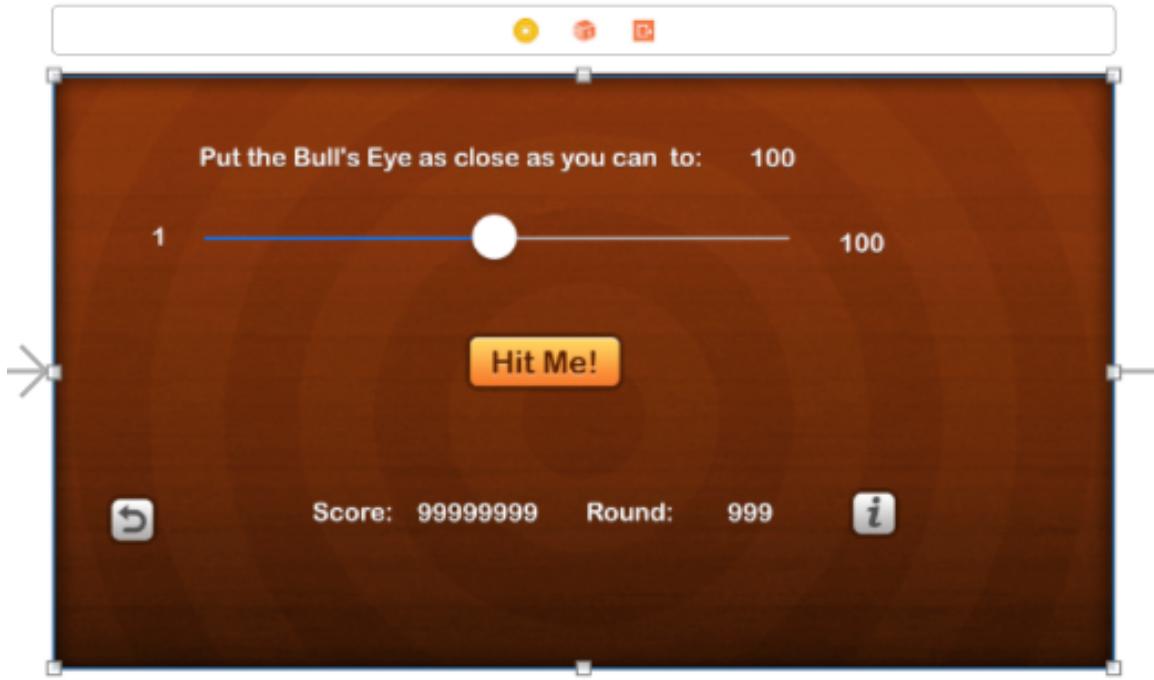
使用该按钮可以定义名为constraints的关系，它会影响界面中所选视觉元素和其它视觉元素之间的显示关系。当我们运行应用时，UIKit将评估这些constraints，然后计算视觉元素的最终呈现方式。听起来好像有点抽象，不过实际试一下就知道了。

为了让背景图拉伸到屏幕的边缘，那么背景图的左、上、右、下四个角都必须和屏幕边缘平齐。那么使用Auto Layout，只需创建两个对其的constraints即可，一个水平的，以及一个垂直的。

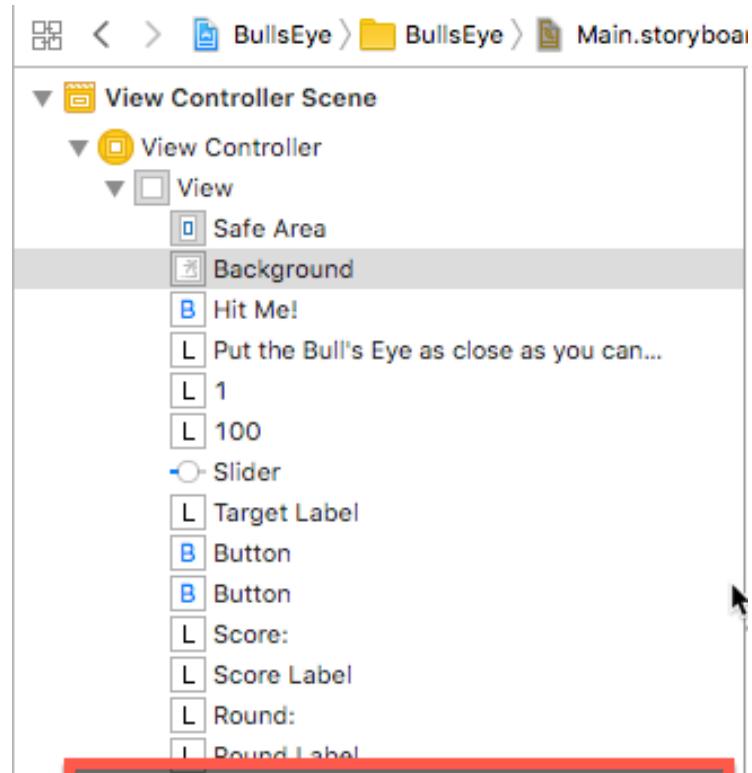
在弹出的Add New Constraints 菜单中，将left,top,right和bottom的spacing都设置为0，然后确保每个红色I形条都被启用。（红色的I形条用来指定在添加新的constraints哪些应被启用。）设置完成后点击Add 4 Constraints即可。



此时，你会看到背景图填满了整个视图。如果你没有看清楚变化，那么可以使用撤销和重复操作来看看具体的差别。



此时，在Document Outline下面可以看到新添加的Constraints:



如果设置正确的话，此处应该显示了四个constraints，其中每个代表了图片的一个角。

选择iPhone 8 Simulator 编译运行项目，看看此时的显示是否正常。当然，其它视觉元素还没有完美显示，但是至少背景图片已经OK了~

现在让我们来设置关于界面。

在storyboard中选择About view controller，然后选择背景图，使用类似的方式Add New Constraints。具体的操作方式参考刚才的讲解，这里就不再废话了。

好了，现在关于界面的背景图也显示正常了，但是Close按钮和Web view的显示还有点问题。

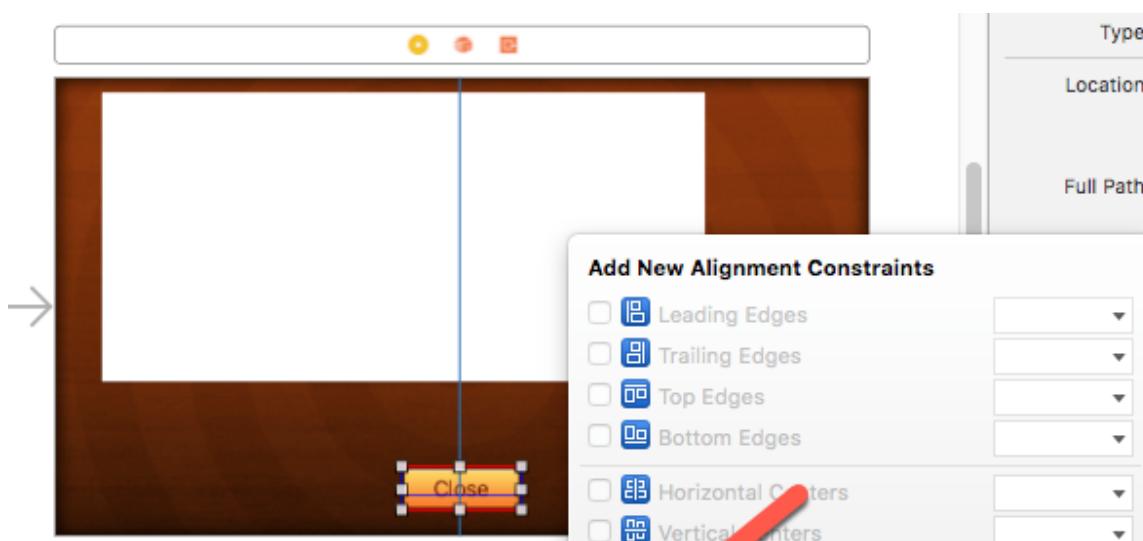
在storyboard中，拖曳Close按钮，使其贴近视图和bottom guide的中心。

Interface Builder提供了非常方便的蓝色点虚线，可以帮助我们来手动对齐视觉元素。



为了让Close始终位于界面的中间位置（而无论屏幕的宽度是多少），我们需要创建一个centering constraint。

点击Close按钮以选中它，从Align菜单（在Add New Constraints按钮的旁边）中选择Horizontally in Container，然后点击Add 1 Constraint。



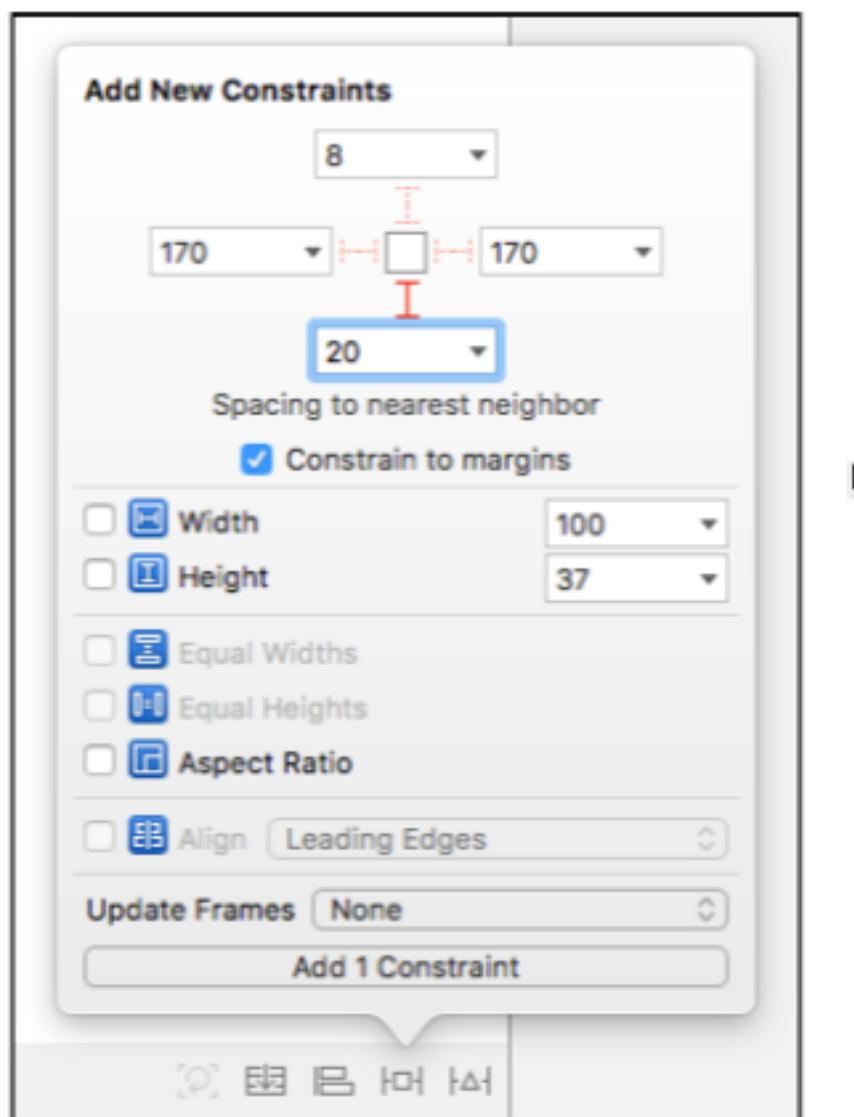
此时Interface Builder回执了一个红色的bar来代表constraint，然后在按钮的周围出现了一个红色的方框。

不过这里就出现问题了：这些bar应该是蓝色的，而不是红色的。如果是红色的，就代表这些constraints有些问题，通常是提醒我们constraints不够。

需要记住的是：对每个视觉元素来说，都需要有足够的constraints来定义其位置和大小。目前Close按钮已经知道自己的大小了-我们在Size inspector中已经指定了。但是对它的位置来说，目前只有X轴上的一个constraint，这还不够，我们需要在Y轴上也添加一个constraint。

如你所见，constraints有几种不同的类型，比如用于对齐的constraints，和spacing constraints，等等。

选中Close按钮，点击Add New Constraints按钮，在Spacing to nearest neighbor部分，将bottom spacing设置为20，并确保选中对应的I形条。  
最后点击Add 1 Constraint以完成设置。



现在红色的constraints变成了蓝色的，代表一切OK了。

我们可以选择不同的Simulator，来查看最终的效果。

### 小提示：

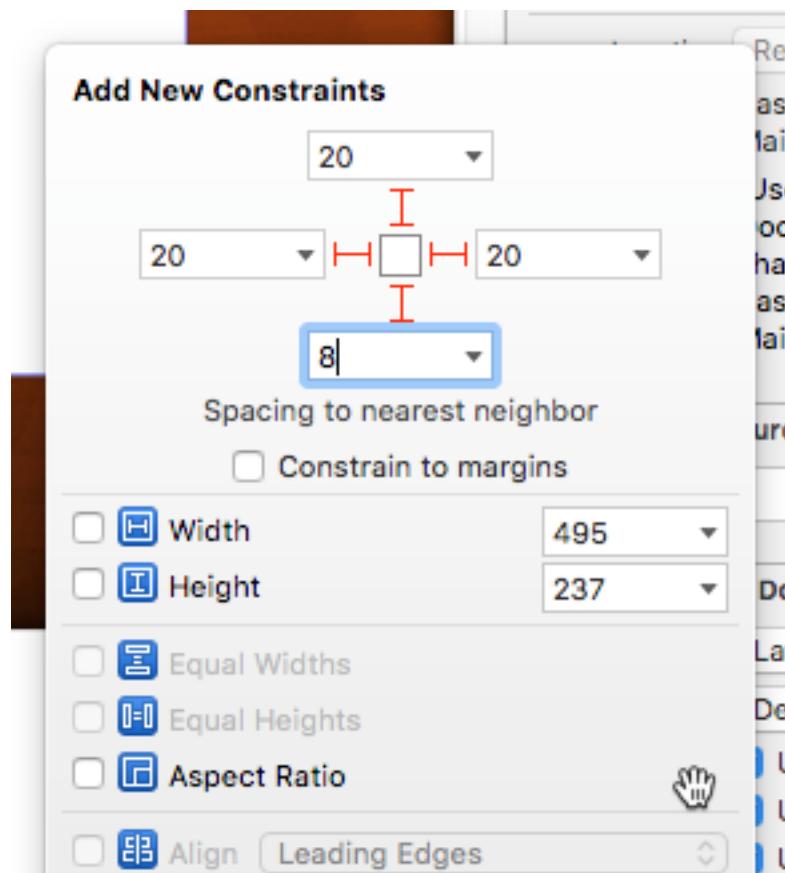
如果我们没有在视图中添加任何constraints会怎样？事实上，Xcode将会在编译应用时自动添加constraints。这也是为什么最开始我们什么也没做，依然能够顺利运行应用的原因。

不过遗憾的是，这些默认的constraints往往不能满足我们的要求。比如，默认的constraints统筹无法自动适配视图到不同分辨率的屏幕上。毕竟，Auto Layout还没有引入AI技术~

还有一点要注意的是，一旦我们在视图中添加了一个自定义的constraints，Xcode就不会再为视图自动添加任何constraints了。一旦迈出了第一步，我们就要为视图中所有视觉元素的布局和显示负责了~

接下来我们还要设置关于界面。

在storyboard中选择关于界面中的Web view,然后点击Add New Constraints。首先确保取消勾选Constraint to margins。然后点击所有四个I形条，并将spacing都设置为20，除了底部的spacing设置为8:



完成设置后，点击Add 4 Constraints即可。

此时web view就有了四个constraints：



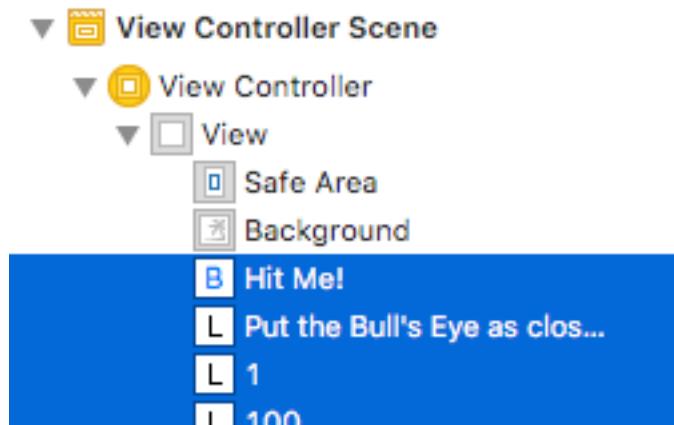
其中的三个将web view和主视图的相对位置确定，而下面的那个则是确定和Close按钮之间的相对位置。

有了以上四个constraints，已经足够确定任何场景下web view的大小和位置了~

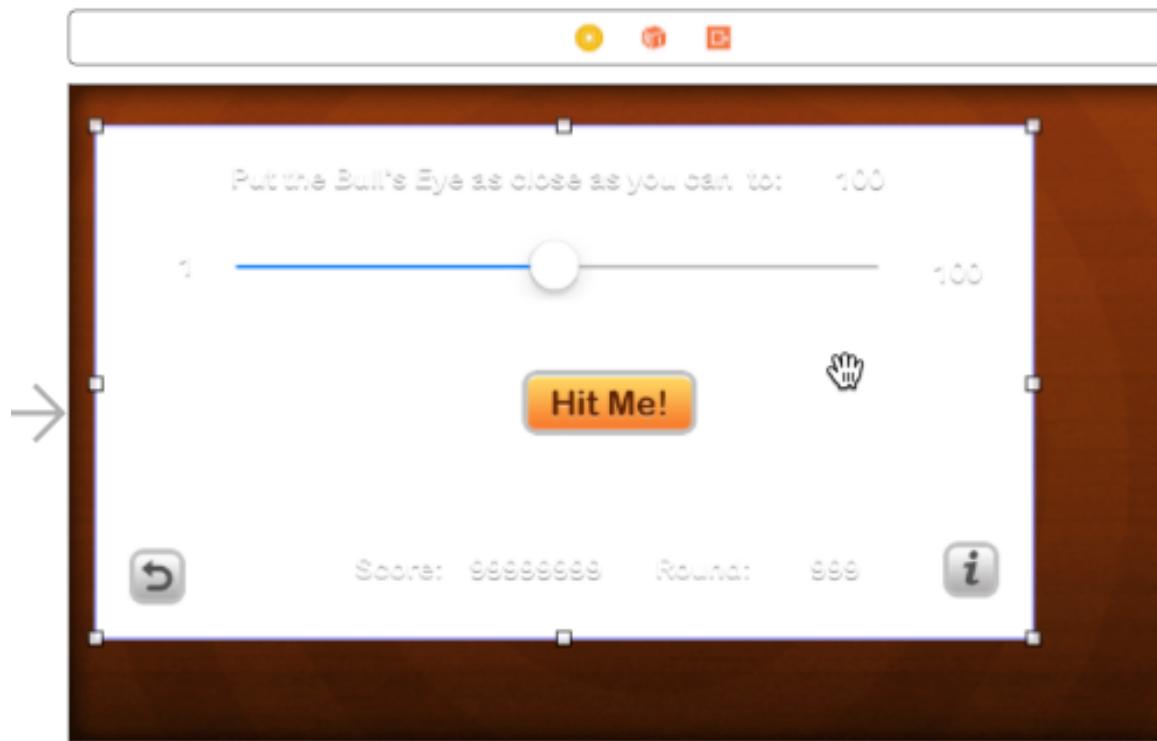
修复主场景中的其它视觉元素

接下来让我们在storyboard中回到主游戏场景。我们需要将所有的label,button和slider放入一个新的container 视图中。这样我们就可以使用Auto Layout将其始终放置在屏幕的中间。

在Document Outline中选中所有的label,button和slider，如图所示：

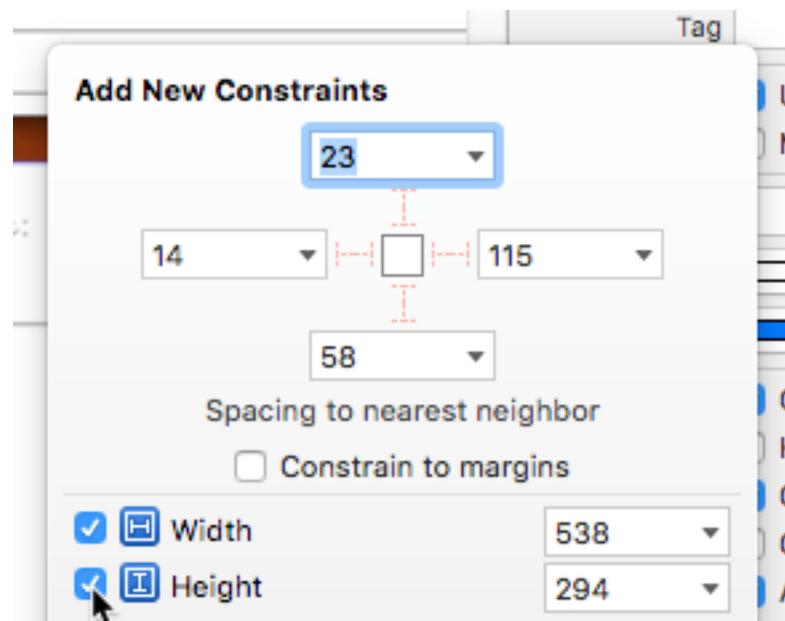


从Xcode的菜单栏中选择Editor- Embed In- View，这样就可以将所选的视觉元素放入一个新的container 视图中：



这个新的视图是全白的，当然最终我们肯定不希望看到它是这个样子的，不过目前先保持这样，以便我们添加constraints.

选中这个新添加的container view，然后点击Add New Constraints，勾选Width和Height，并使用Interface Builder中所指定的数值。然后点击Add 2 Constraints以完成设置。



此时Interface Builder会在这个container view的周围绘制几个条，以代表我们刚刚添加的Width和Height constraint，但是目前它们显示是红色的。Don't panic!这里的红色只是代表没有足够的constraints。我们接下来将添加所缺失的constraints。

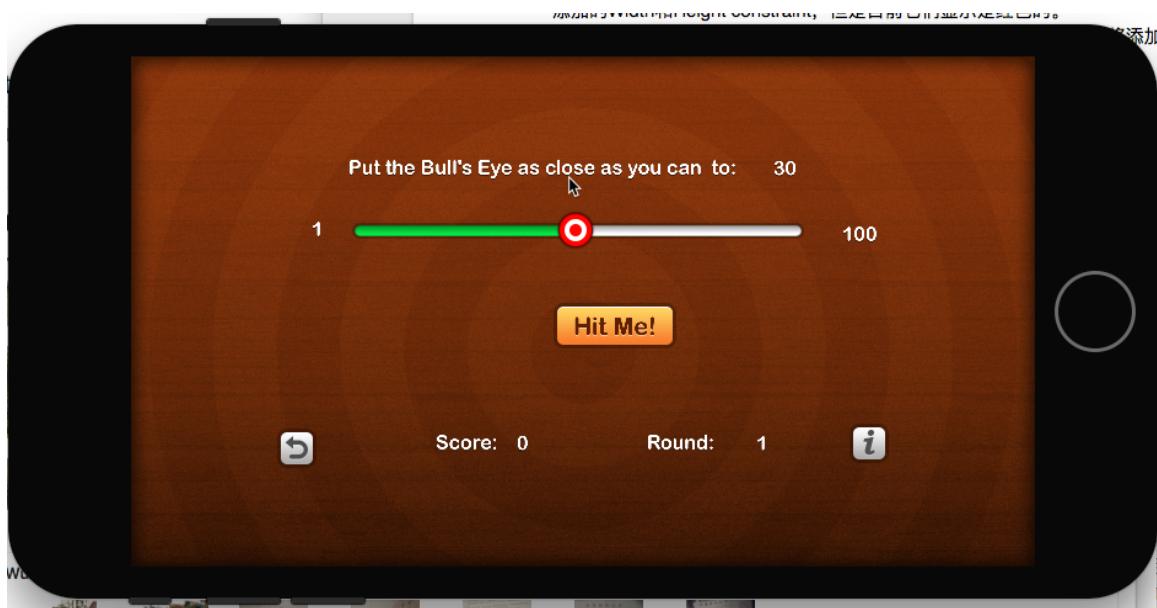
继续选中这个container view，然后点击Add New Constraints旁边的Align图标，勾选Horizontally in Container和Vertically in Container选项，然后点击Add 2 Constraints。

好了，现在所有的Auto Layout 条都会变成蓝色，而视图也会完美的显示在中央位置。

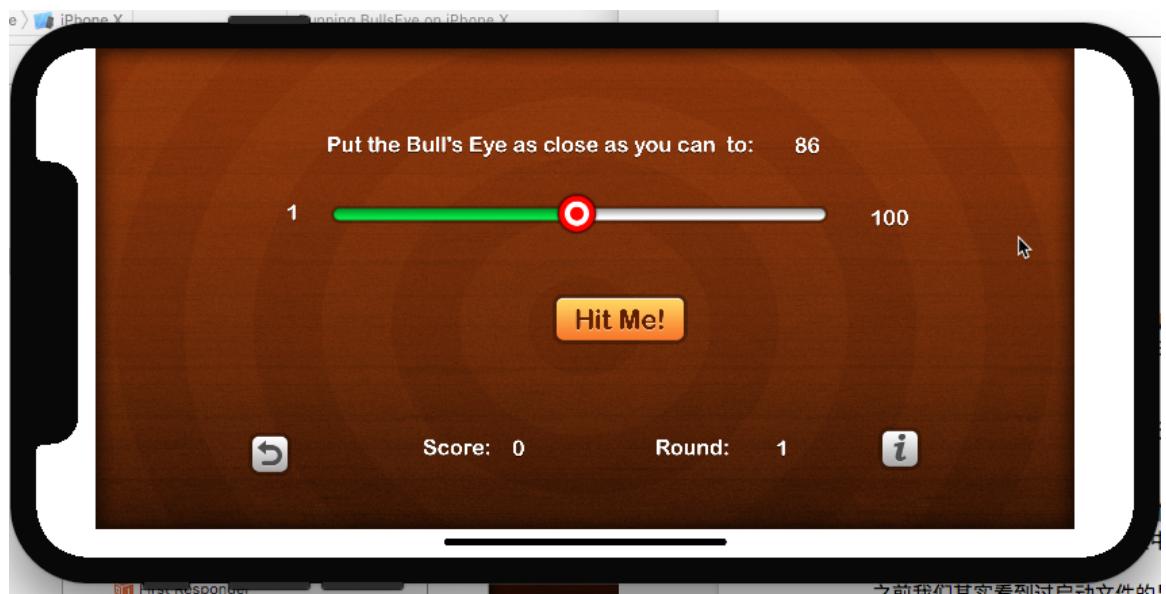
最后，让我们更改container view的Background 色彩为Clear Color(0% opacity,也就是100%透明)。

此时我们可以随意选择任何一种iPhone Simulator进行测试~

比如iPhone 8 Plus:



我也在iPhone X的Simulator上尝试了下，结果是下面这个样子~



不过这就不是我们设置的问题了，这是iPhone X本身的问题，😢

小练习：

如果你足够细心的话，会发现当我们在不同分辨率的设备上进行测试时，虽然视觉元素始终处于界面的中间，但是在大一点的屏幕中，视觉元素并没有填满整个区域，这是因为我们将container view设置为某个特定的大小。如果我们希望界面元素可以根据屏幕的实际可用大小更改位置和大小，那么就必须删掉container view，然后为每个视觉元素分别添加所需的autolayout constraints了。

试试看~

### 添加Crossfade（淡入淡出动态特效）

回顾我们的清单，接下来我们还需要添加Crossfade动态特效。为此，我不得不提一下Core Animation。Core Animation是iOS的系统框架之一，使用它可以给应用添加各种酷炫的动画，而且只需要短短几行代码就可以实现。还是那句话，仅仅让代码跑起来，让应用可以正常运行还不够，我们得从一些微妙的细节出发，让用户体会到使用我们产品或游戏的爽快感觉。

这里将要添加一个简单的淡入淡出效果，也就是当“Start Over”的按钮被触碰后，会使用一个过渡效果让这个过程显得不是那么突兀。

在Xcode中切换到ViewController.swift，在文件顶部添加一行代码如下：

```
import QuartzCore
```

位置可以在导入UIKit的下面。

Core Animation需要用到QuartzCore框架，通过这个重要的import语句，我们告诉编译器将会用到这个框架。

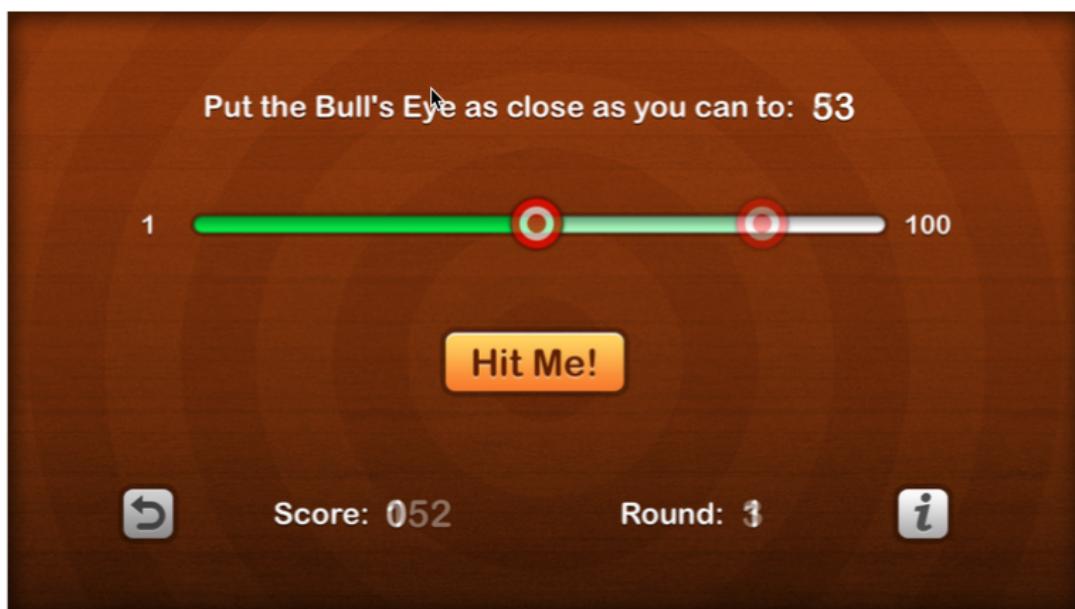
更改startNewGame方法的代码如下：

```
func startNewGame() {  
    score = 0  
    round = 0  
    startNewRound()  
  
    //add crossfade effects  
    let transition = CATransition()  
    transition.type = kCATransitionFade  
    transition.duration = 1  
    transition.timingFunction = CAMediaTimingFunction(name:  
kCAMediaTimingFunctionEaseOut)  
  
    view.layer.add(transition, forKey: nil)  
}
```

简单的来看，我们创建了一个动画，它的类型是淡入淡出，时间长度是1秒。最终的效果是从当前屏幕上所显示的内容切换到更改后的内容。

再次点击Run运行，可以看看效果。

如果你足够仔细，会发现一些微妙的视觉体验差异~



## 快结束了-添加图标

其实这个东西一点都不新，只要你通过手机上或itunes上的app store下载过东西，就会明白对于一款iOS产品来说，图标是多么的重要！

在Xcode中点击左侧导航栏的Assets.xcassets，然后选择AppIcon

现在里面有若干个空槽，分别代表应用所需的不同类型图标。



在Finder中打开本教程的Icon文件夹，把Icon-40.png文件拖到iPhone Notification iOS 7-11 20pt的2x槽中。

你可能会问，为什么要把Icon-40.png拖入为20pt准备好的槽中呢？记住这个槽里面的描述是2x，代表着它是位高清设备准备的，而每个Point代表2个像素。

把Icon-60.png文件拖入旁边的3x槽，这是为3x分辨率准备的。3\*20是60（这个乘法计算应该没什么难度吧？）

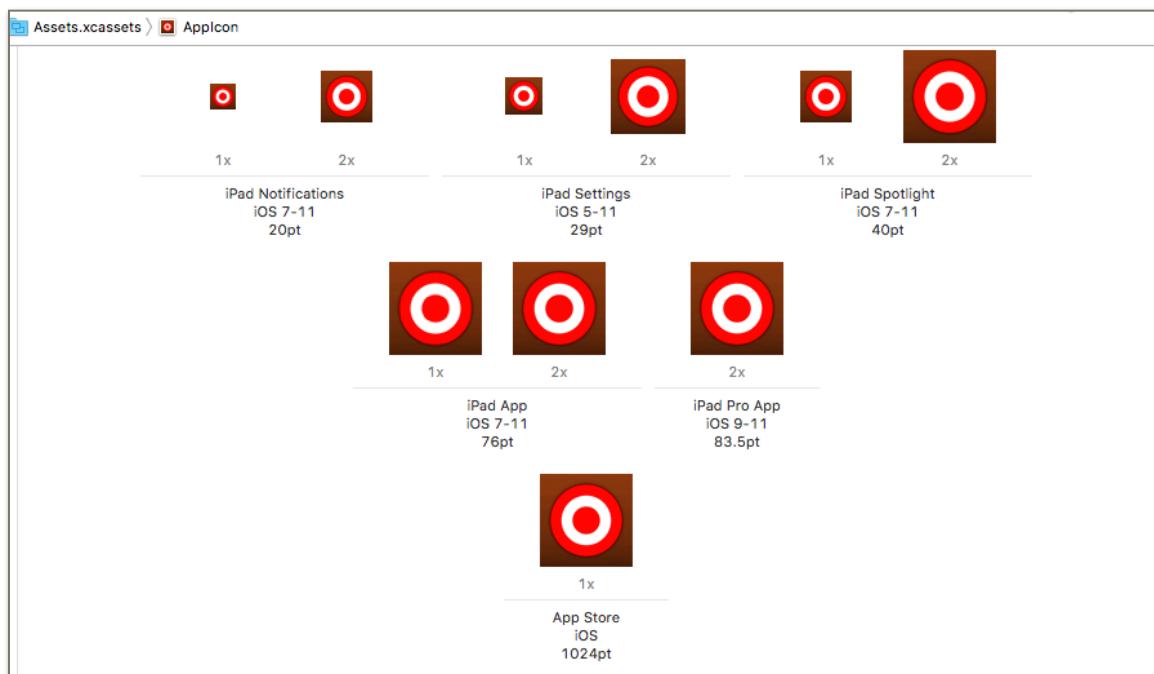
对于其它的几个槽，我们依次类推，这次就不再赘述了。

文件夹中的其它文件是为iPad准备的。虽然我们这个应用是专为iPhone设计的，不过所有的iPad都可以运行为iPhone准备的应用。

注意为iPad准备的图标是1x和2x（而非3x）。

注意Icon-1024.png文件不是为应用内部使用准备的，它是为后面提交到App Store准备的，届时我们将需要1024\*1024像素的图标。

全部完成之后的设置如下图所示：



编译运行应用，然后将其关闭。在Simulator中可以看到我们想看到的图标。

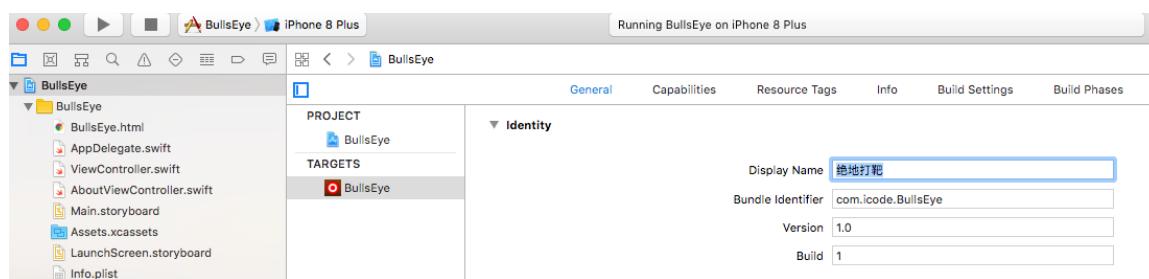


## 显示游戏名称

图标有了，启动画面有了，我们还需要有一个好的产品名称。

之前我们用了简单的英文名称BullsEye，但我希望在Appstore里面显示的是王者打靶，或者绝地打靶，总之和人类的表达习惯更接近。不过需要注意的是，在图标下面的空间非常有限，你的产品名称也不能太长了，否则就无法显示完全。

在xcode中切换到Project Settings界面，然后更改Display Name为绝地打靶即可。



好了，现在点击Run来运行，就可以在模拟器上看到图标和更改后的产品名称了。



**在结束教程之前，我们还要给这个小游戏加点新东西。**

任何一款优秀的游戏都是一款艺术作品，它包含了精美的画面，炫酷的动画，动听的音效。。。且慢，我们这款游戏似乎是默片？你可以接受苍老师的视频教程里面竟然没有声音吗？显然不能，我们这个小游戏也一样，要加点好听的。

首先来个背景音乐怎么样？

在ViewController.swift的文件顶部添加一行代码：

```
import AVFoundation
```

这就引入了一个新的框架，所谓的AVFoundation框架，此AV非苍老师的AV，但这个AVFoundation框架的主要作用就是播放音频视频，所以苍老师也用得着。

接下来在定义其它实例变量的代码下面添加一行代码：

```
var audioPlayer: AVAudioPlayer!
```

这就添加了一个AVAudioPlayer类型的播放器变量。

添加一个新的方法：

```
// play background music

func playBgMusic(){
    let musicPath = Bundle.main.path(forResource: "bgmusic", ofType: "mp3")
    let url = URL.init(fileURLWithPath: musicPath!)

    do{
        audioPlayer = try AVAudioPlayer(contentsOf: url)
    }catch _ {
        audioPlayer = nil
    }

    audioPlayer.numberOfLoops = -1
    audioPlayer.prepareToPlay()
    audioPlayer.play()
```

}

这段代码的主要作用就是指定背景音乐所在的位置，然后创建并初始化一个播放器变量，设置循环播放。如果初始化失败，就提示出错的原因，如果成功，就直接播放背景音乐。

最后在viewDidLoad()方法中添加一行代码：

```
//播放背景音乐  
playBgMusic()
```

我在resources文件夹里面放了这个曲子，你可以把这个bgmusic.mp3文件拖到项目中。

完成后点击Run，就可以在玩游戏的同时欣赏美妙的音乐了。

小练习：

你可以从网上找一些或者自己制作一些短音效，然后在游戏交互的地方添加一些小音效。

科普：iOS常用框架-官方和第三方

iOS系统源自Mac系统，它的结构分为四个层次：



最下面的一层是核心操作系统层(Core OS Layer)，这部分和Mac系统基本上完全相同，当然在电源管理等方面存在着一些细小的差别，后续的Mac系统改进也从

iOS系统中吸取了一些不错的内客。它包括内存管理、文件系统、电源管理以及一些其他的操作系统任务。它可以直接和硬件设备进行交互。核心操作系统层包括以下组件：

OS X Kernel, Mach 3.0, BSD, SOCKETS, Power Mgmt, File System, Keychain, Certificates, Security, Bonjour

倒数第二层是核心服务层(Core Services)，同样和Mac桌面系统基本相同，但增加了移动设备的一些特性，比如Core Location和Map。主要可以通过它来访问iOS的一些服务：

Collections, Address Book, Networking, File Access, SQLite, Core Location, NET Services, Threading, Preferences, URL Utilities

第三层是媒体层(Media Layer)，和Mac系统基本相同，但有一些小的差异。通过Media Layer可以在应用中使用各种多媒体文件，录制音频和视频，绘制图形，或制作动画效果（比如我们这里用到的Core Animation）：

Core Audio, OpenGL ES, Audio Mixing, Audio Recording, Video Playback,  
JPG, PNG, TIFF, PDF,  
Quartz, Core Animation

最上面的一层是界面交互层(Cocoa Touch),这一部分和Mac系统存在着非常大的差异，Mac的交互层称为Cocoa。它主要负责用户在iOS设备上的多点触摸交互操作，包括以下组件：

UIKit (在Mac桌面系统对应的是AppKit) , Multi-Touch Events, Core Motion, Camera, View Hierarchy, Localization, Alerts, Web Views, Image Picker, Multi-Touch Controls

一般来说，开发iOS应用中一大半的时间都是在和Cocoa Touch这个层打交道，比如UIKit框架中的可视化组件，比如Image Picker获取照片库中的照片，通过Core Motion获取重力感应和陀螺仪的支持，以及获取用户通信录等等。

Framework (框架) -实质上就是帮我们实现各种特定功能的API类库。

在iOS产品开发的过程中，我们最经常接触到的就是Cocoa Touch，确切的说是UIKit Framework，所以任何一个iOS应用项目创建后都会自带UIKit Framework和Foundation Framework。当我们在开发产品的过程中需要用到一些特殊功能的时候，首先应该从最顶层寻找所需要的框架，只有顶层中的框架无法解决问题时，才能往下寻找相关的技术。

下面列出了一些常用的苹果官方提供的iOS 框架。

框架名称	功    能
CoreLocation.framework	使用 GPS 和 Wi-Fi 获取位置信息
Foundation.framework	提供 Object-C 的基础类 (像 NSObject)、基本数据类型和操作系统服务等
GameKit.framework	为游戏提供网络功能：点对点互联和游戏中的语音交流
MapKit.framework	为应用程序提供内嵌地图的接口
MediaPlayer.framework	提供播放视频和音频的功能
MessageUI.framework	提供视图控制接口用以处理 E-mail 和短信
OpenGL ES.framework	提供简洁而高效的绘制 2D 和 3D 图形的 OpenGL API 子集
QuartzCore.framework	提供动画特效以及通过硬件进行渲染的能力
StoreKit.framework	为应用程序提供在程序运行中消费的支持
SystemConfiguration.framework	检测当前网络是否可用和硬件设备状态的能力
UIKit.framework	创建和管理应用程序的用户界面

框架名称	功    能
CoreLocation.framework	使用 GPS 和 Wi-Fi 获取位置信息
Foundation.framework	提供 Object-C 的基础类 (像 NSObject)、基本数据类型和操作系统服务等
GameKit.framework	为游戏提供网络功能：点对点互联和游戏中的语音交流
MapKit.framework	为应用程序提供内嵌地图的接口
MediaPlayer.framework	提供播放视频和音频的功能
MessageUI.framework	提供视图控制接口用以处理 E-mail 和短信
OpenGL ES.framework	提供简洁而高效的绘制 2D 和 3D 图形的 OpenGL API 子集
QuartzCore.framework	提供动画特效以及通过硬件进行渲染的能力
StoreKit.framework	为应用程序提供在程序运行中消费的支持
SystemConfiguration.framework	检测当前网络是否可用和硬件设备状态的能力
UIKit.framework	创建和管理应用程序的用户界面

更详细的信息请参考苹果官方文档：

<https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSFrameworks/iPhoneOSFrameworks.html>

除了这些官方提供的框架，一些热心的程序猿（通常是老外，我朝程序猿热心开源项目的不多，开源中国上的还有些热血青年，其它的老油条都只顾自己赚钱去了）还在开发自己的项目之余搞了一些非常有用的iOS类库。

**One more thing...One last thing...**

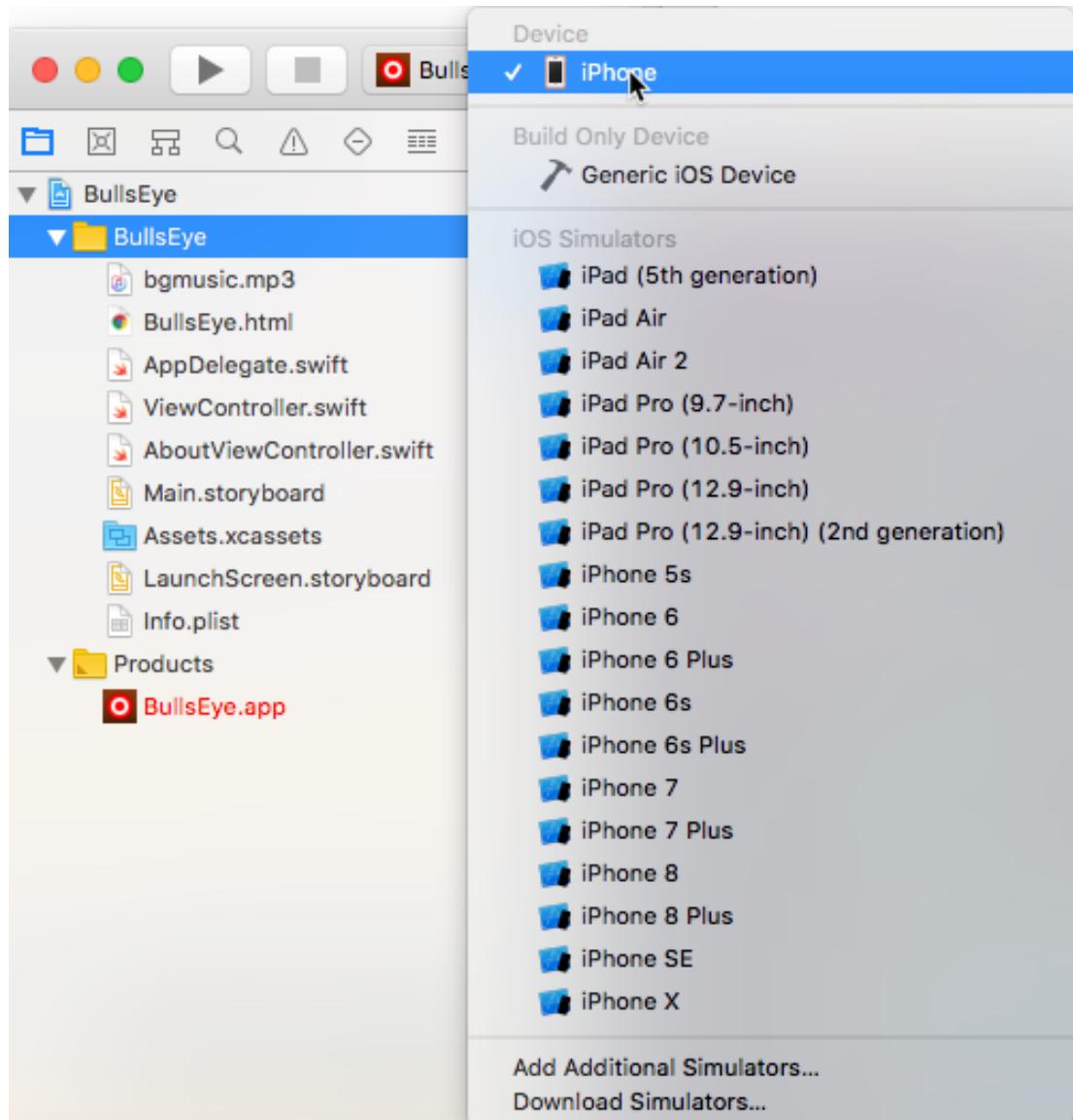
最后的最后，让我们在设备上实际测试这款应用

在Xcode7之前，如果开发者想要在设备上测试是需要付费账号的。不过如今连这一点也可以省掉了，只要你有一个Apple ID就行，而Apple ID是免费注册的哦~

Step 1.把你的测试设备iPhone,iPod touch或iPad用USB线连到mac上。

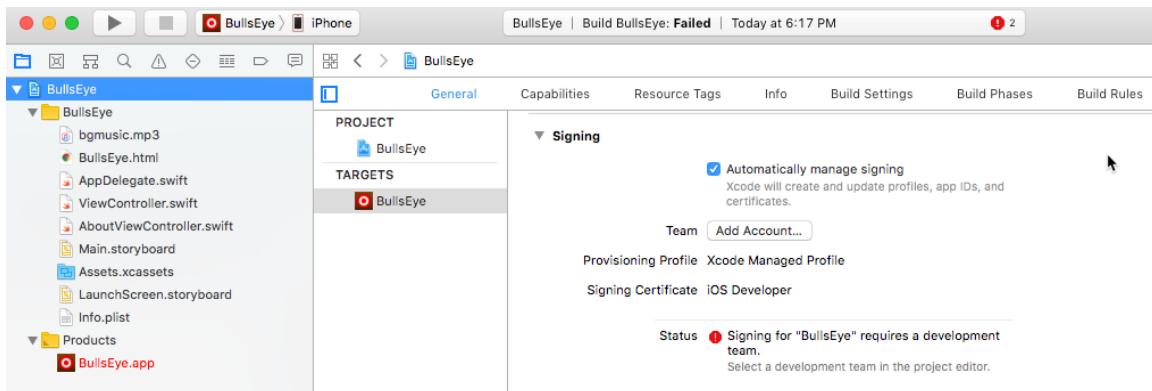
Step 2.从Xcode的工具栏中选择你所对应的设备

我的如下图所示 (iPhone 6s Plus)

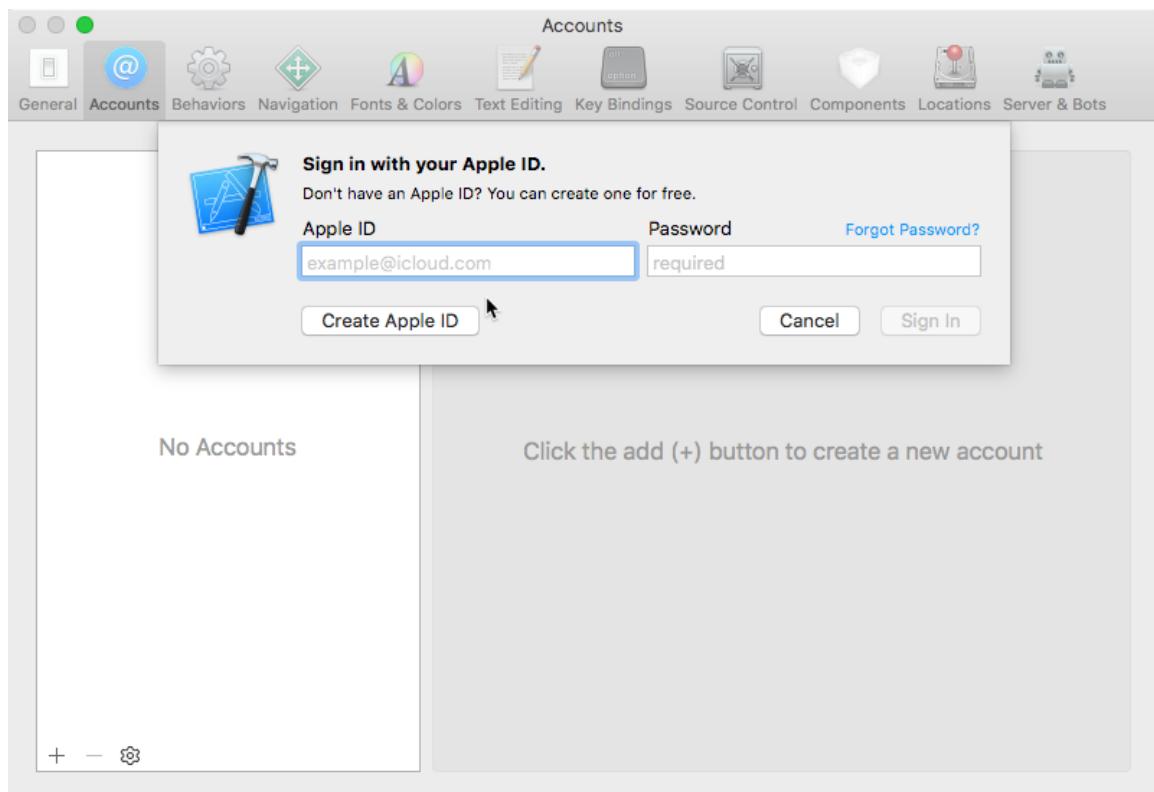


Step 3.设备搞定之后，接下来就是设置你的Apple ID.你可以用自己习惯的iCloud,iTunes账号，不过如果你希望靠iOS开发赚点money，那么建议申请一个新的Apple ID，专用于开发。当然，如果你已经注册了付费的开发者账户，显然就需要使用那个Apple ID了。

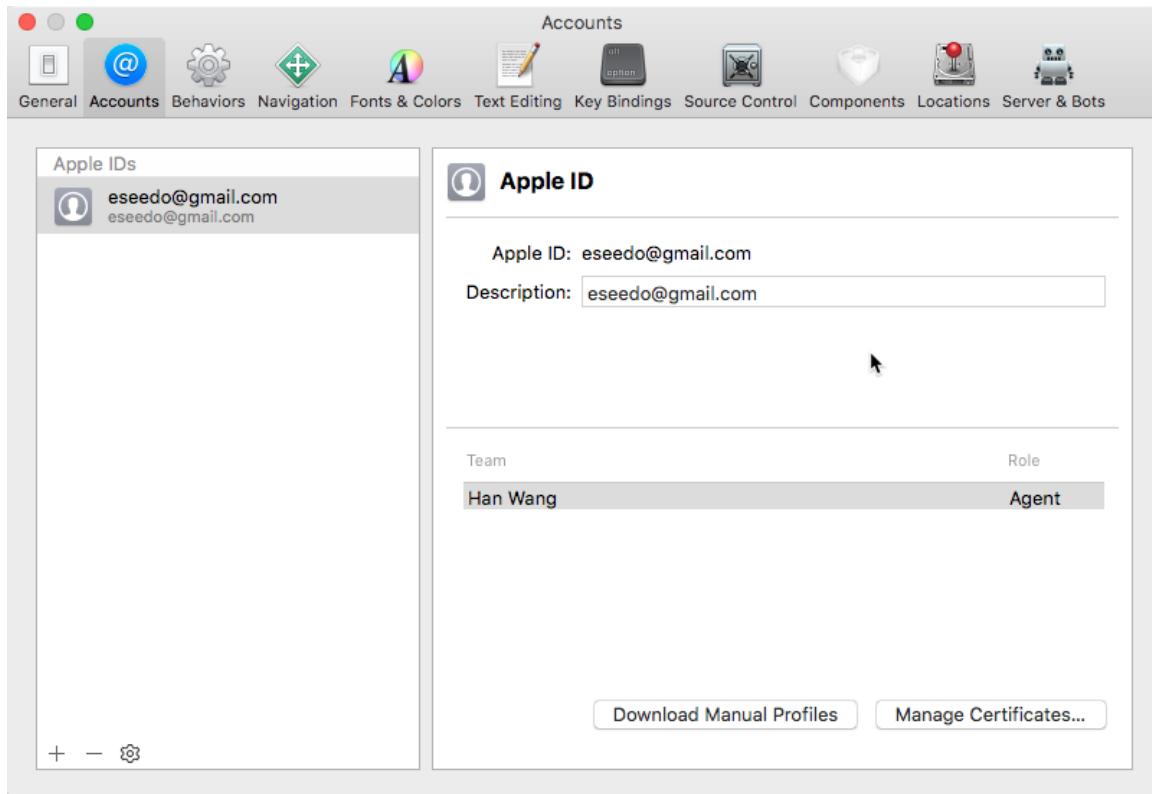
在Xcode的菜单中选择Preferences...，然后切换到Accounts选项卡。此时会看到如下界面。



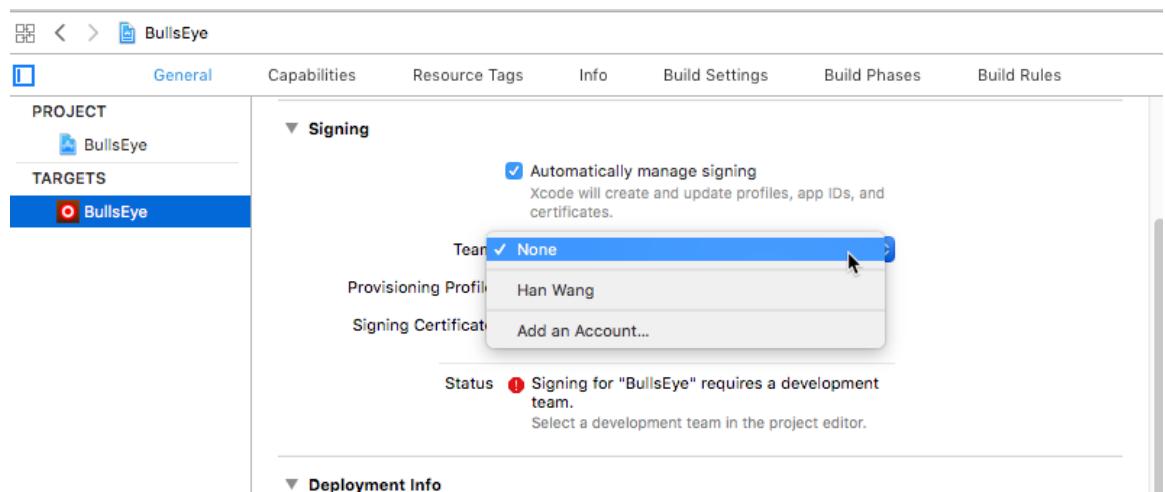
点击Add Account...并添加自己的账号：



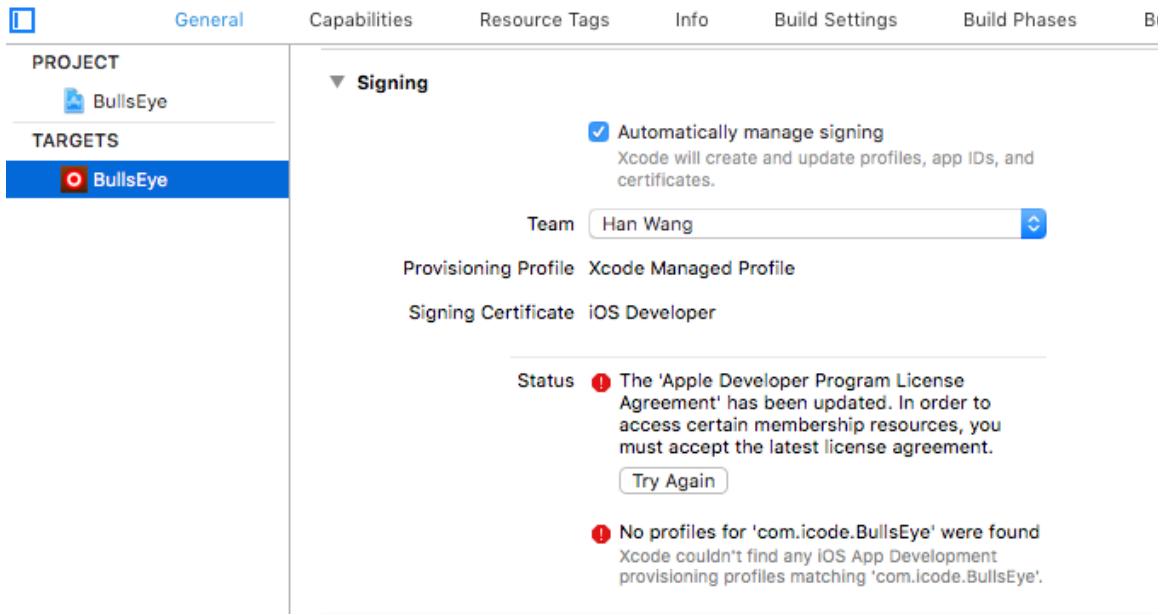
添加成功后如图所示：



然后就可以在刚才的设置界面选择自己的开发者账号了。



此时会看到以下的错误提示：（



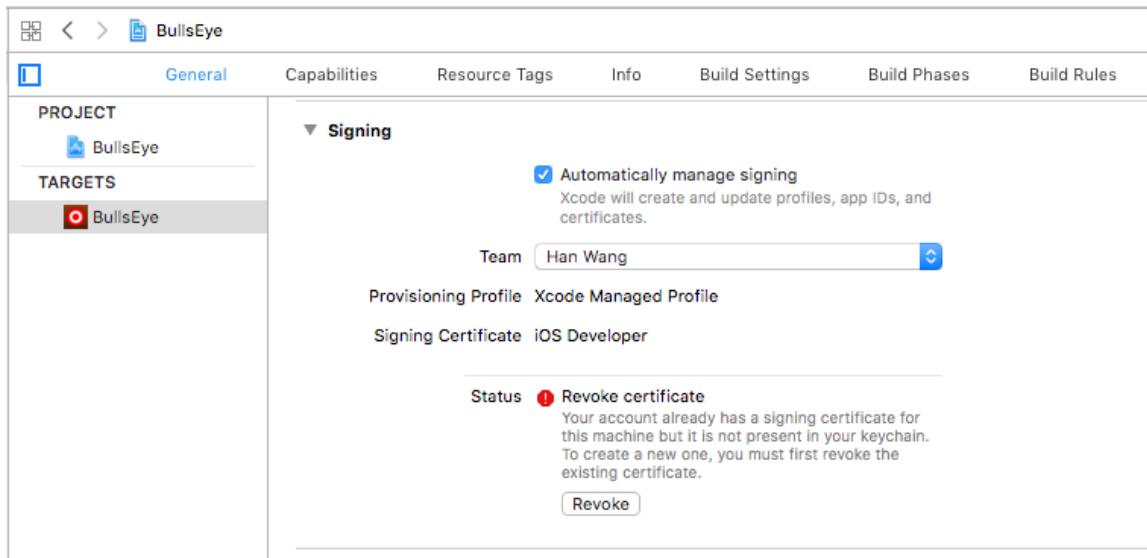
现在我们需要用浏览器打开<http://developer.apple.com>，并登陆自己的账户。

点击Review Agreement

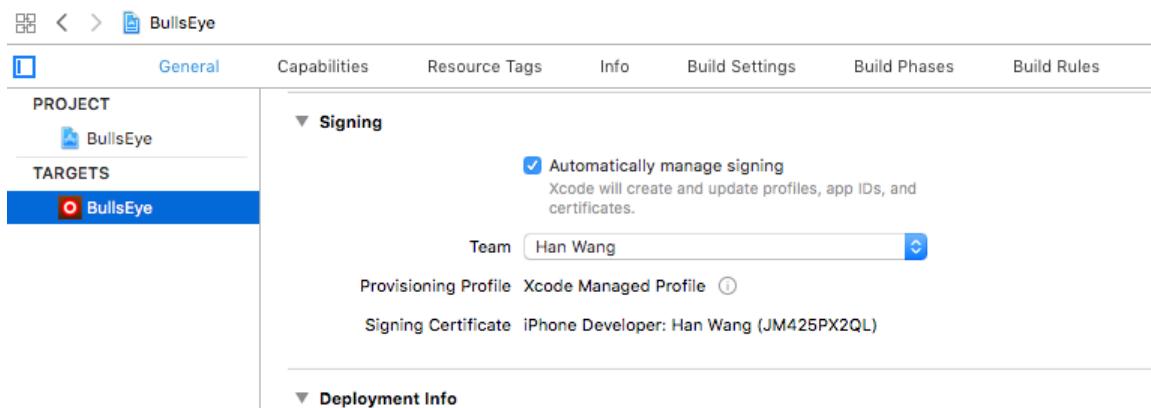
The screenshot shows the Apple Developer Program account page for 'Han Wang'. A red arrow points to a blue 'Review Agreement' button in a prominent red callout box. The box contains the text: 'The Apple Developer Program License Agreement has been updated. In order to access certain membership resources, you must accept the latest license agreement.' Below the button, the user's name 'Han Wang' and 'Apple Developer Program' are displayed. On the left sidebar, there are links for Overview, Membership, Certificates, IDs & Profiles, iTunes Connect, CloudKit Dashboard, and Code-Level Support. At the bottom, there are links for Documentation, Downloads, and Forums. Two main sections are visible on the right: 'Certificates, Identifiers & Profiles' and 'iTunes Connect'.

勾选并点击 I Agree 即可。

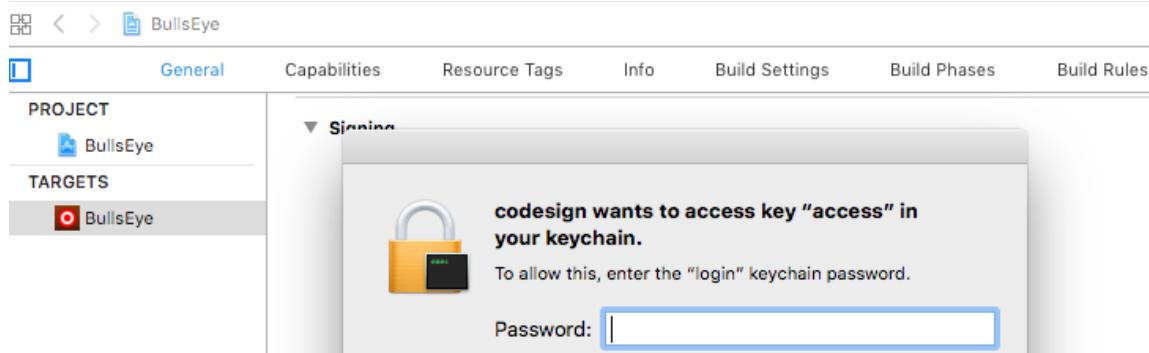
回到 Xcode，会看到以下提示：



点击 Revoke 修复，可以看到这里显示正常了。



点击工具栏上的编译运行按钮来启动应用，会看到如下提示：



输入密码，并点击Always Allow，就可以了。

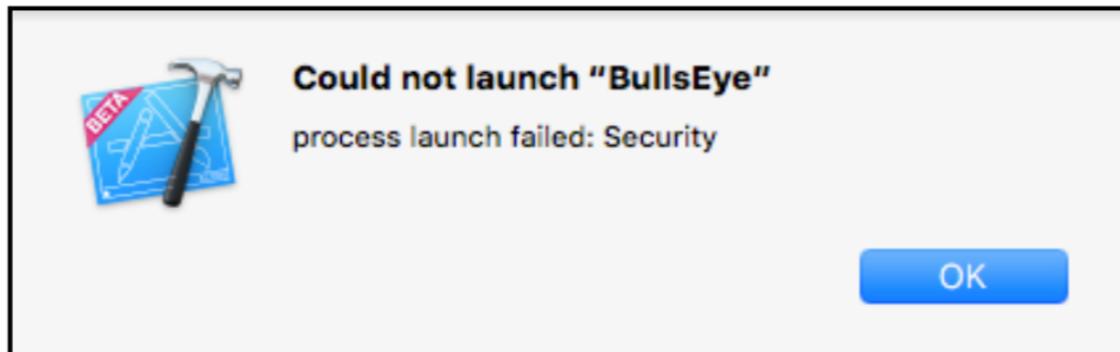
如果你之前没有配置过，那么很可能会看到错误提示。

可能的错误原因1：

设备根本就没有连接，这个通常是接口松动的问题。

可能的错误原因2：

设备没有信任你



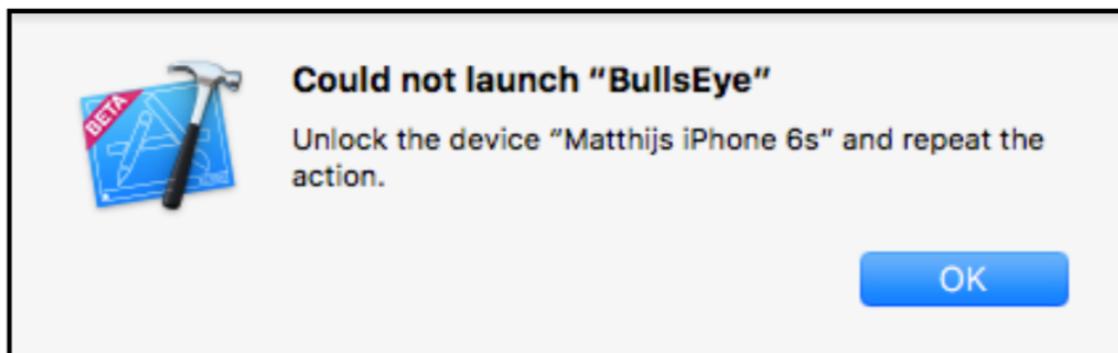
此时在设备上可能会看到提示“Untrusted Developer. Your device management settings do not allow using apps from developer ...”

如果看到此类提示，那么请在手机上进入Setting-General- Profile。点击所列出的Apple ID，然后触碰它和Trust按钮。

可能的错误原因3：

设备被锁定。

如果你的手机已被锁定，那么可能会看到如下提示：



只需要解锁并重新编译运行即可。

可能的错误原因4：

没有添加开发者账号。

此类错误的判断很简单，在项目设置中就看得到。

好了，到了这里，我们的系列教程第一部分已经结束。

希望在我们一起学习iPhone开发的这个过程中，对于iOS产品的开发可以形成一个整体的认识。

在后续的系列教程中，我们也期待着更多收获。如果你懒得看后续的教程，也可以到这里就结束，然后自己去探索iOS产品开发的更多未知秘密。祝你好运！

最后的福利：



素材来源: [raywenderlich.com](http://raywenderlich.com)  
新浪博客: [blog.sina.com.cn/eseedo](http://blog.sina.com.cn/eseedo)  
个人网站: <http://icode.ai>  
VR网站: <http://vr910.com>

所有教程和源代码、素材等均可以在github上找到:

<https://github.com/eseedo/iOSCourse>

因为时间精力有限，无法为大家详细解答教程中的问题。如果大家碰到问题建议多看官方文档，多去cocoachina,泰然网， cocoachina, stack overflow, 苹果官方论坛。另外建议多看github,code4app和苹果官方提供的示例代码。

笨猫学编程讨论QQ群： 375143733