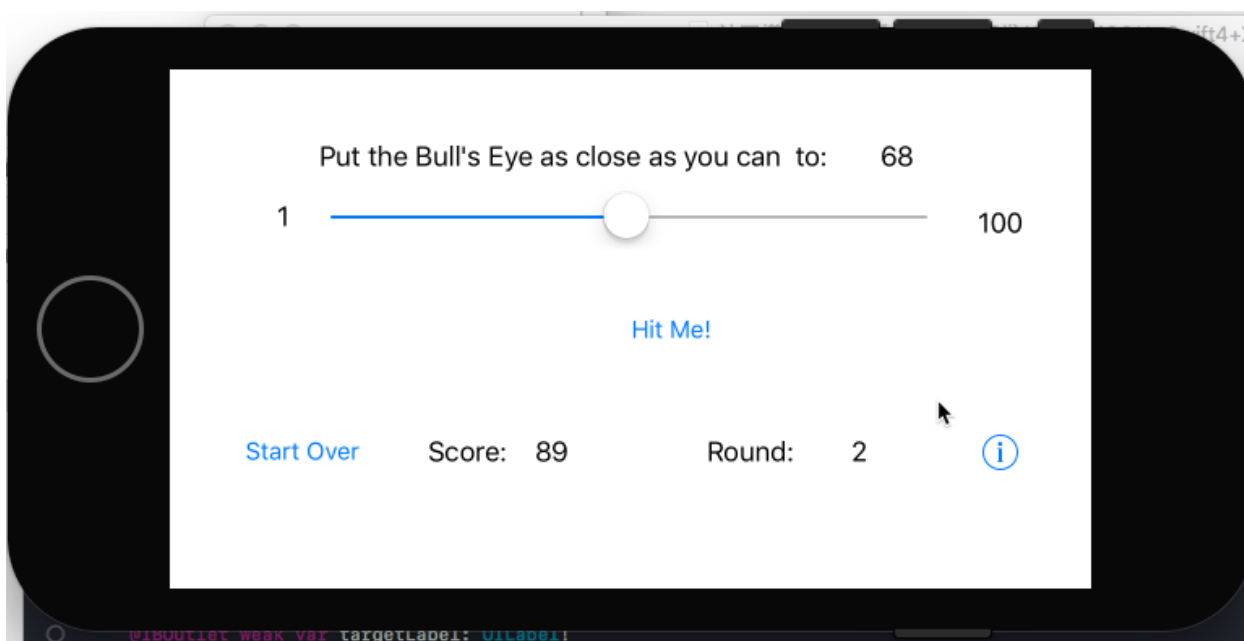


让不懂编程的人爱上iPhone开发(2017秋iOS11+Swift4+Xcode9版)-第13篇

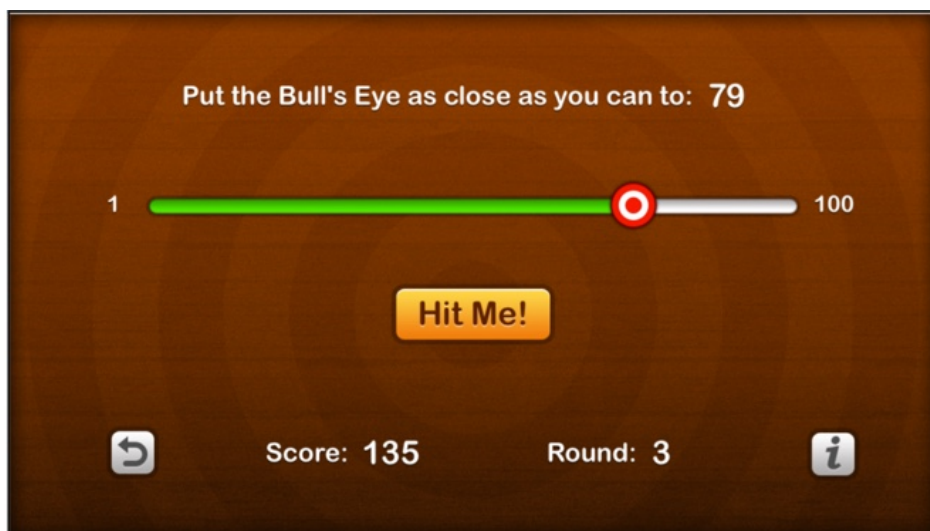
继续我们的iPhone开发学习。这里我们还将继续美化游戏的界面。

去掉状态栏只是万里长征第一步，我们还需要让界面变得更漂亮。

当前我们的界面看起来是这样的：



我们期望中的界面应该是这样的：



实际的控件和动作并没有发生变化，不过我们将要用图片来美化界面，同时调整一下一些颜色，字体设计（正是产品和设计人员的拿手好戏）。

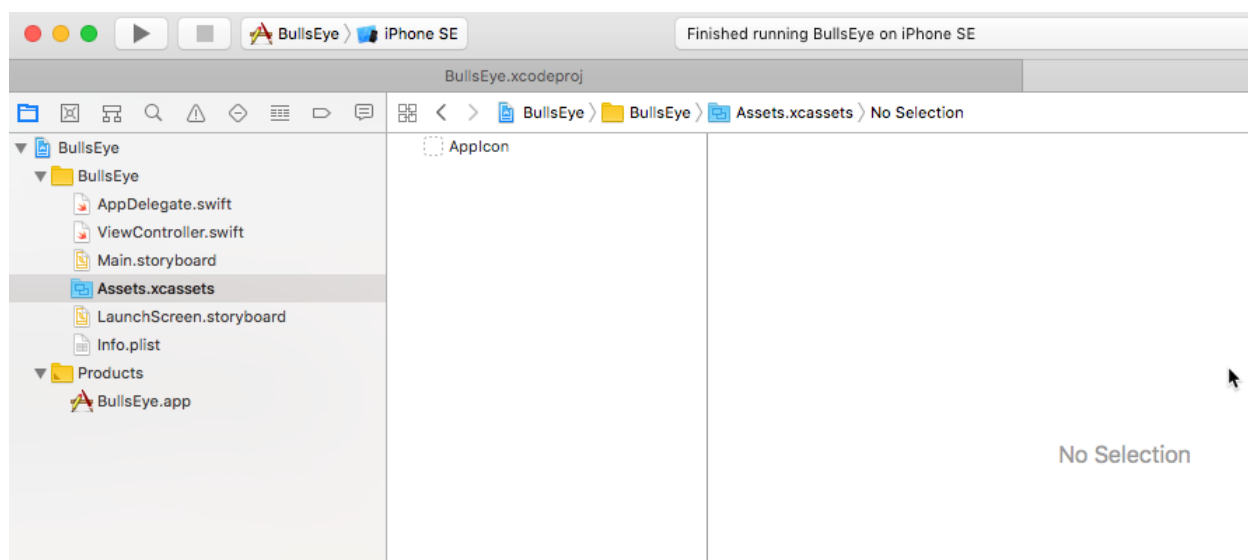
我们可以在背景中使用图片，在按钮上使用图片，甚至在滑动条上使用图片。

需要注意的是，在iOS产品开发中，我们应尽可能使用PNG格式的图片。你可以完全使用自己的图片，我这里也准备了几个备用的。

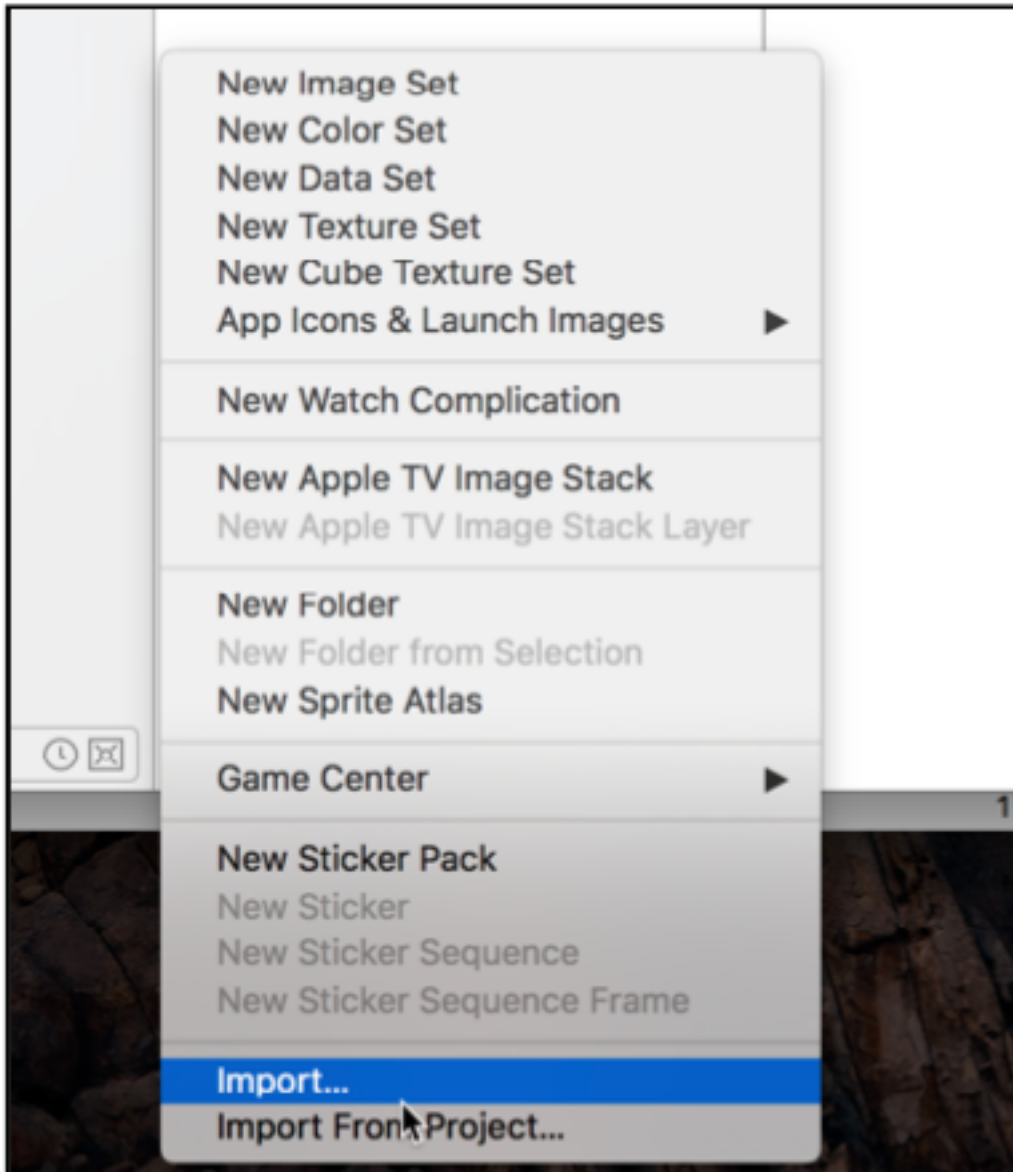
如果你是个Photoshop达人，那么完全可以设计一套自己的美术素材。

如果你对设计一窍不通，Don't panic~ 本教程附带的Resources文件夹中有一个Images子文件夹，你首先需要把它导入Xcode项目中。

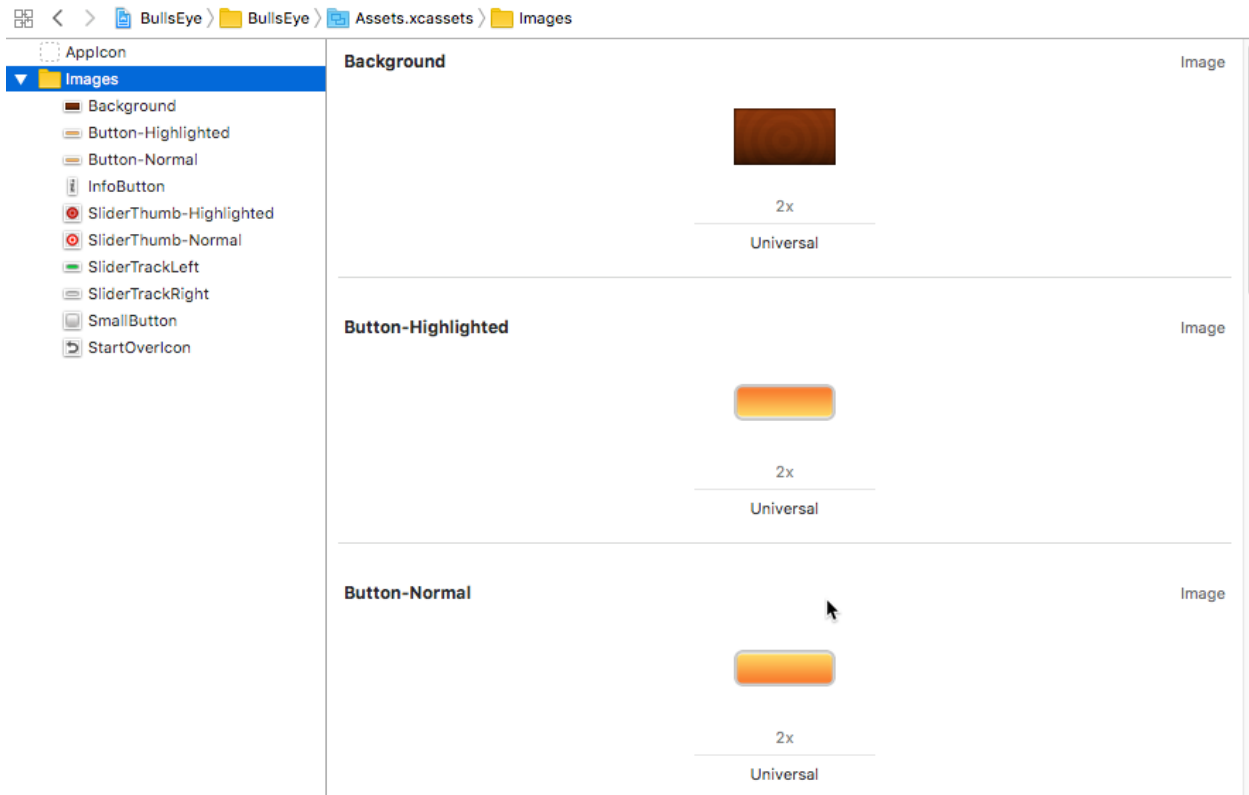
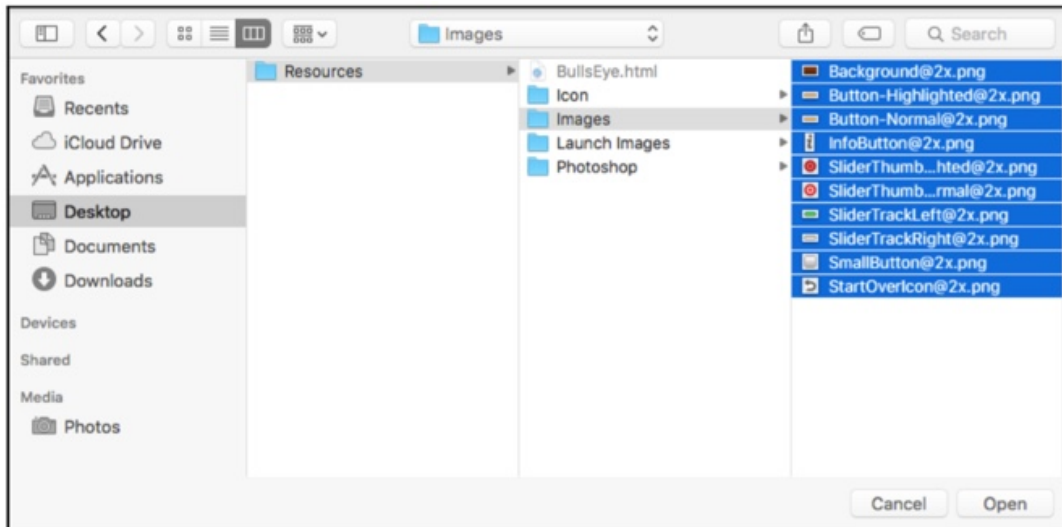
在Xcode的项目导航部分找到Assets.xcassets，点击它。



点击中间这个面板下面的+号，会出现弹出式菜单：
选择这里的Import...



Xcode会提供一个文件选择器，选择该教程对应的资源文件夹下面的Images文件夹,用⌘+A的方式来选中Images文件夹中的所有文件。最后点Open就好了。



注意，如果Xcode添加了一个名为Images的文件夹，而非单个的图片文件，请重新尝试，并确保选中了文件夹中的图片，然后再点击Open。

小技巧：

除了使用Import...菜单命令，你还可以直接从Finder中将所需的文件拖到Xcode asset中。

科普- 1x,2x和3x显示

注意到在asset catalog的分类中有四个空格区：1x,2x,Retina 4 2x和3x。通过提供相同图片的不同版本，可以更好的适应iPhone和iPad的屏幕尺寸。

1x代表低分辨率屏幕，目前所有低分辨率的iPhone设备都不再被iOS11支持。因此，除非你希望应用可以支持iOS9的设备，那么可以提供1x的图片。

2x代表高清Retina分辨率，包含主流的iPhone和iPad设备。高清Retina图片是低清的2倍大小，因此用2x代表。我们刚才导入的只是2x图片。

3x是iPhone Plus设备的专属。如果我们需要为土豪机型提供最好的体验，可以把3x的图片放到这里。

我们的“王者打靶”游戏并非一个universal应用，因此1x图片可以免了。不过仅仅是提醒你，你也可以在iPad上运行这个游戏。iPad的一大好处是可以在特殊的仿真模式下运行所有的iPhone应用。在这种情况下应用会使用2x图片。

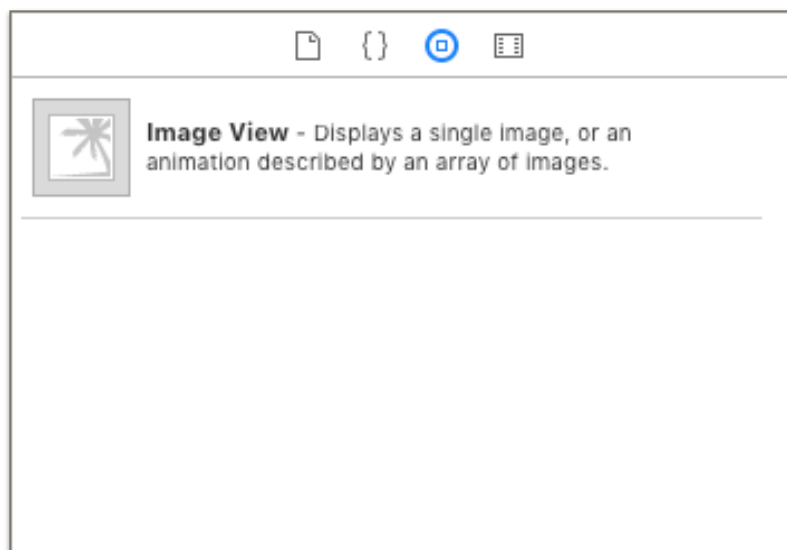
还有一个特殊的命名习惯。如果图片资源的名称后面有@2x或@3x，那么就是说它是为Retina或者Retina HD版本量身定做的。低清分辨率图片不需要加任何后缀（不需要@1x）。

更换背景图片

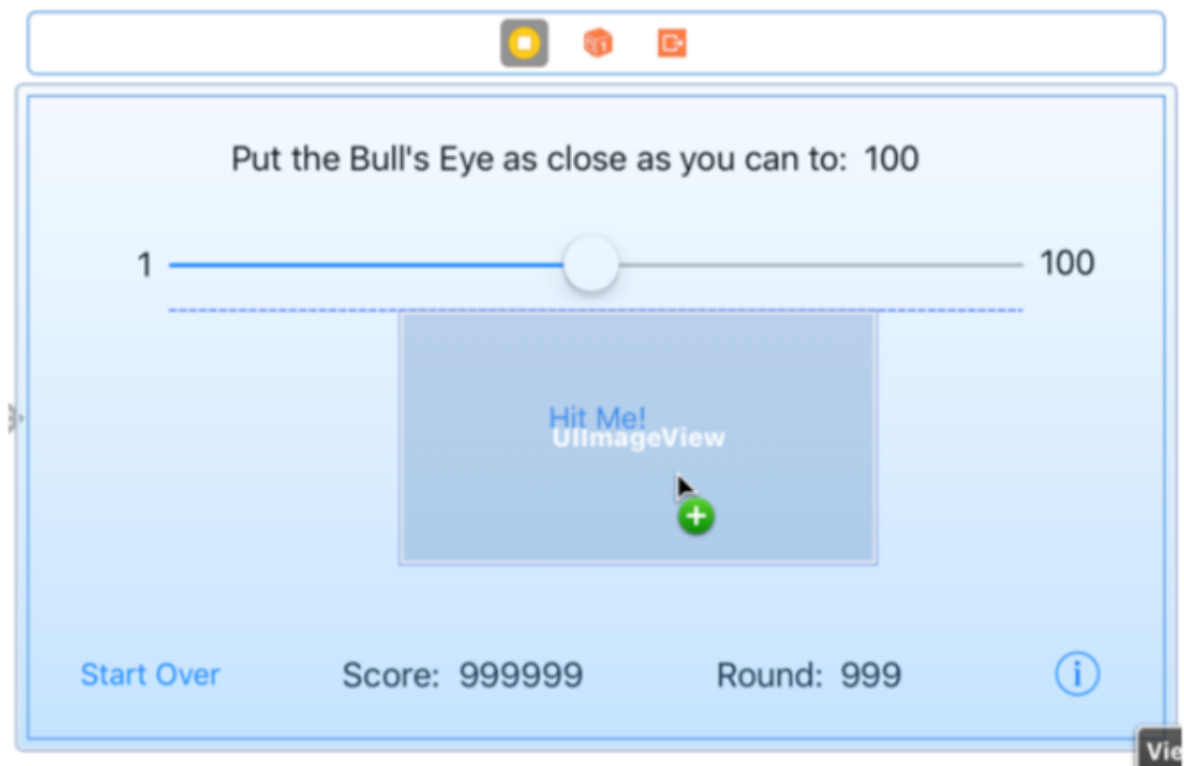
准备就绪了，让我们首先更换背景图片吧。

在Xcode中打开Main.storyboard，在Xcode右侧面板的Object Library中找到Image View。

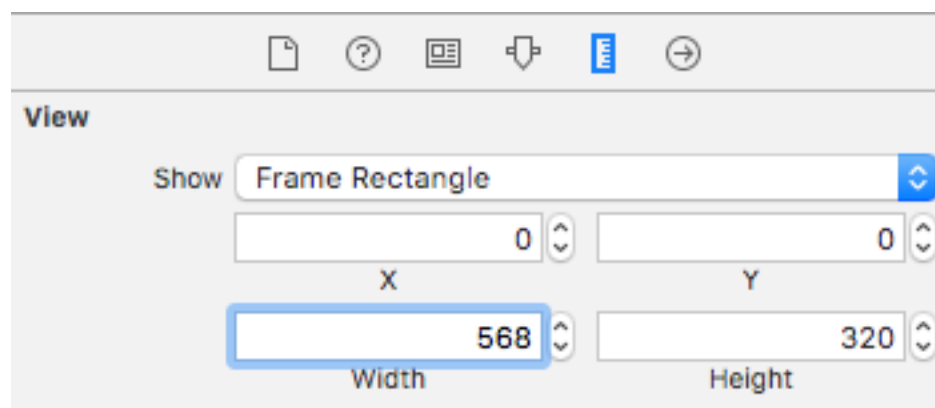
小技巧：在Object Library底部的搜索栏中输入image，就可以快速过滤掉其它的视图。



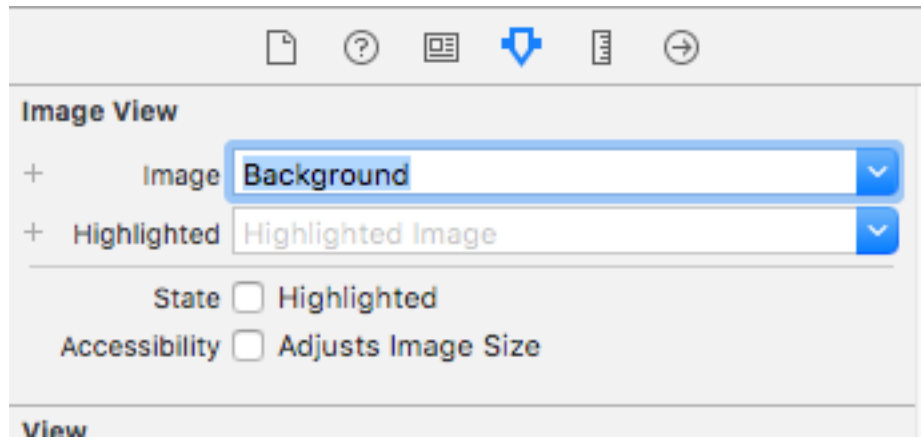
把这个Image View拖到我们已有的用户界面上，先别管放在哪儿，待会儿我们会调整位置。



选中Image View，在右侧的面板中切换到Size Inspector，，设置X和Y为0，Width为568，Height为320。这样图片就会覆盖整个屏幕。

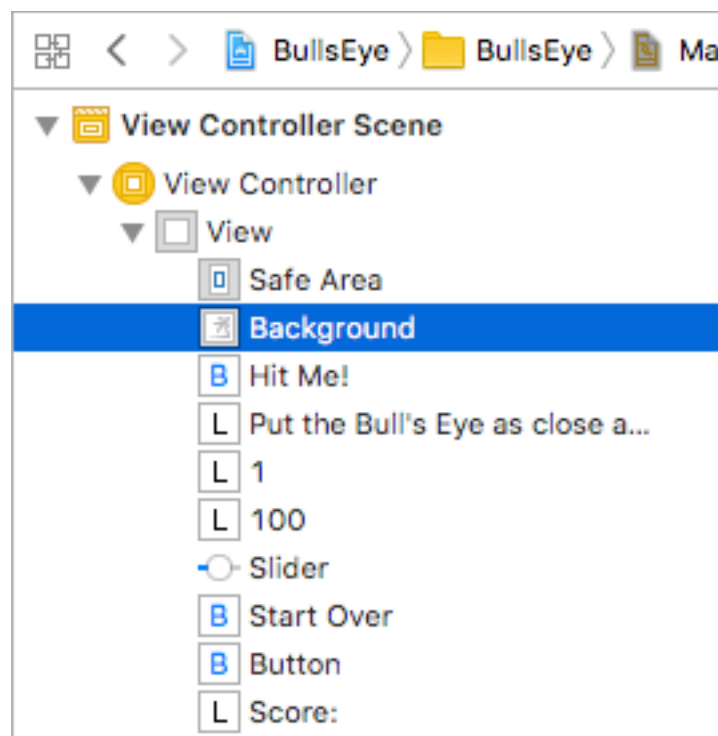


然后继续选中Image View，在右侧的面板中切换到Attributes Inspector，按照下图的提示将Image部分选择为Background。

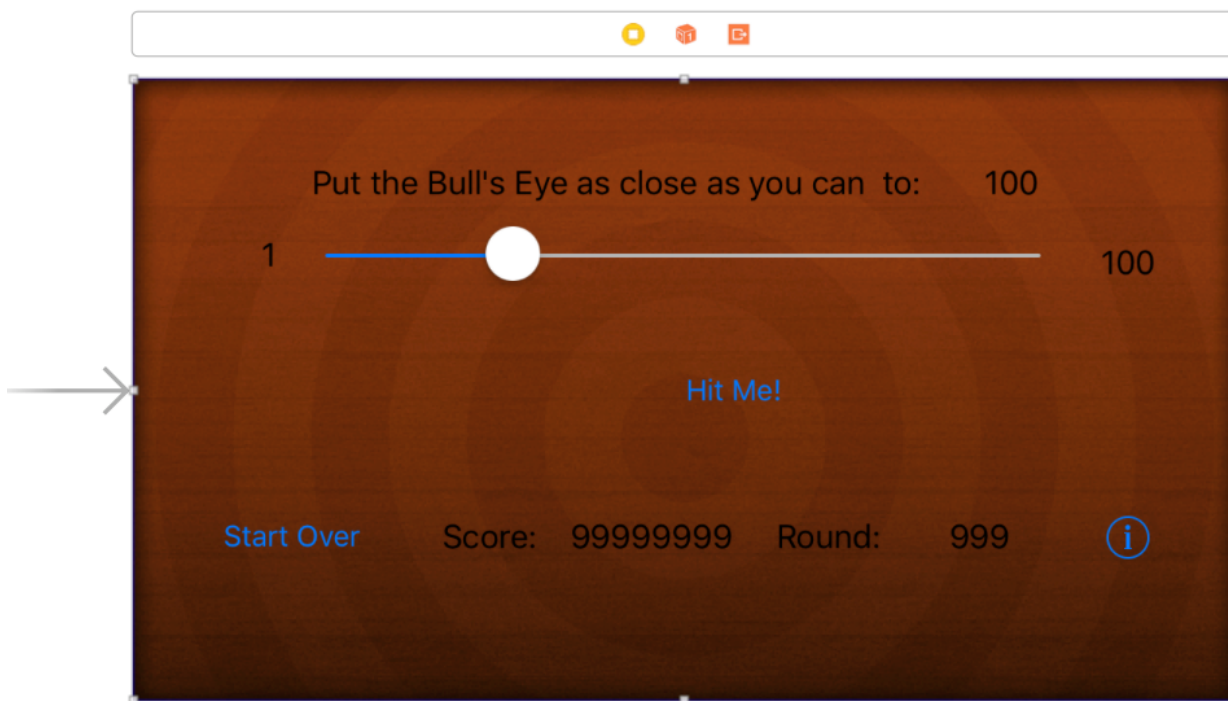


当然，现在背景图片覆盖了其它控件。我们需要让其它控件的显示在背景之上。这一点很简单。

在Xcode的菜单栏中选中Editor,选择Arrangement- Send to Back，就好了。不过有时候可能你会发现这个选项是灰色的，那么有一种可替代的更直观的方式（我个人倾向使用），在Interface Builder的布局面板中把Image View拖到最上面，就可以了。



此时你的界面看起来应该是类似下面的：



点击工具栏上的编译运行按钮，来测试下效果。

优化标签

让我们继续前进，接下来我们将优化标签的显示。

因为背景图片比较暗，所以需要让标签中文本的色彩变亮。

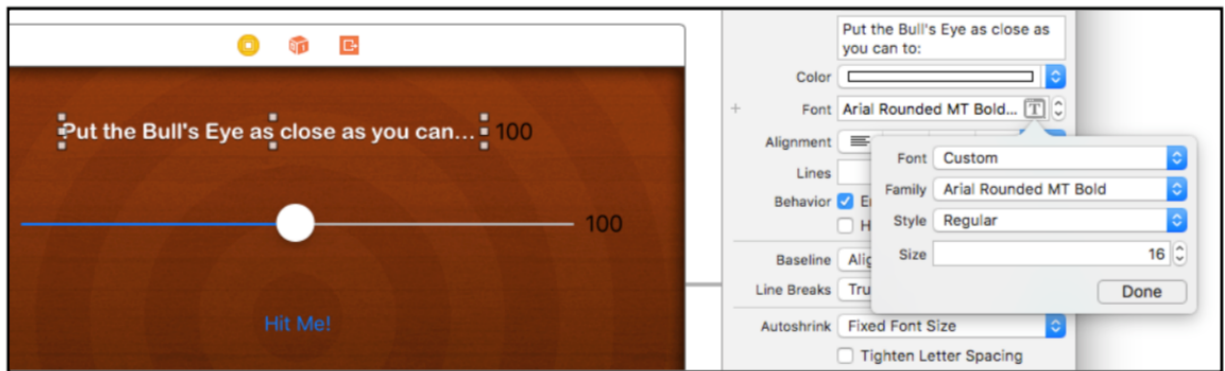
在storyboard中选择顶部的标签，在Xcode的右侧面板中切换到Attributes Inspector，然后点击Color（注意具体的位置跟Xcode的版本有关。自从Scott Forstall被赶出苹果后，Xcode在这些细节上是一塌糊涂），要选择实际的色块，而不是旁边的文字选项。

在色彩拾取器里面选择RGB Sliders，然后选择R:255,G:255,B:255,透明度100%，也就是纯白色，会用PS的都知道。

然后点击Shadow，将其设置为R:0,G:0,B:0,透明度50%。

然后把Shadow Offset设置为Horizontal:0,Vertical :1，这样阴影会显示在标签下面。

接着点击Font属性的[T]小图标，这里就可以设置字体了。你可以设置自己喜欢的字体，也可以直接参考下图的设置。



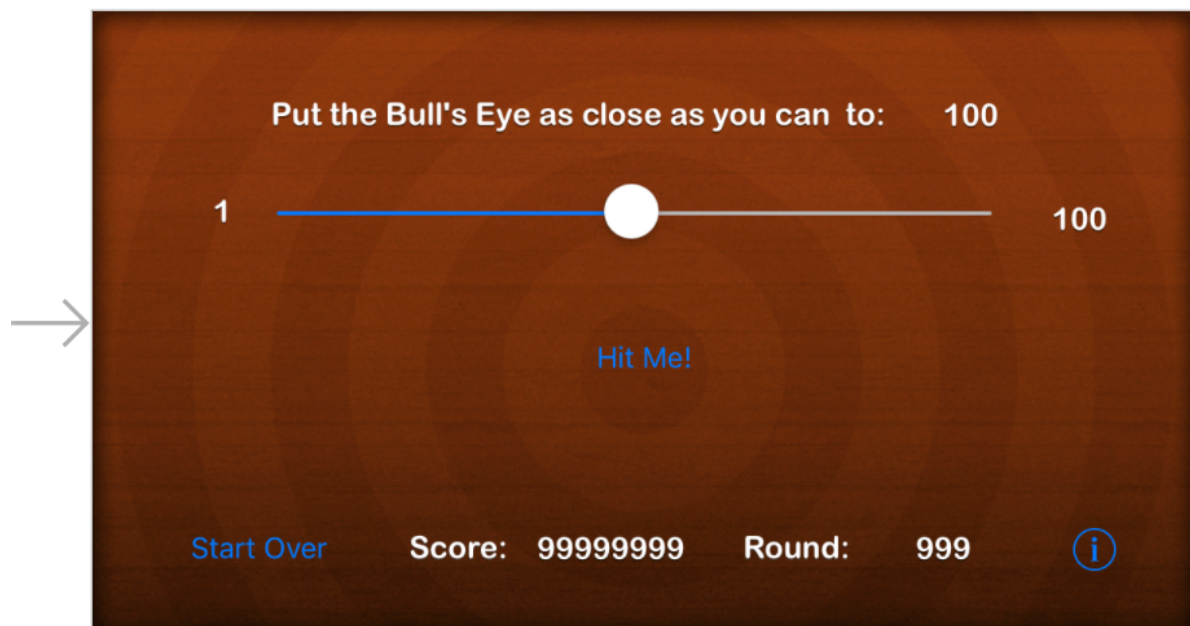
然后确保Autoshrink选项为Fixed font size。

此时当文本比标签大时，Autoshrink功能将会自动调整字体大小以适应标签大小。

接下来选择Xcode顶部菜单中的Editor的Size to Fit Content。（有时菜单栏上会显示灰色，这是Xcode的bug，就不必为这个操心了），快捷键command =

对于其它标签哥就不废话了，不然就没完没了了。具体怎么设置您说了算。

最终的效果类似下面：

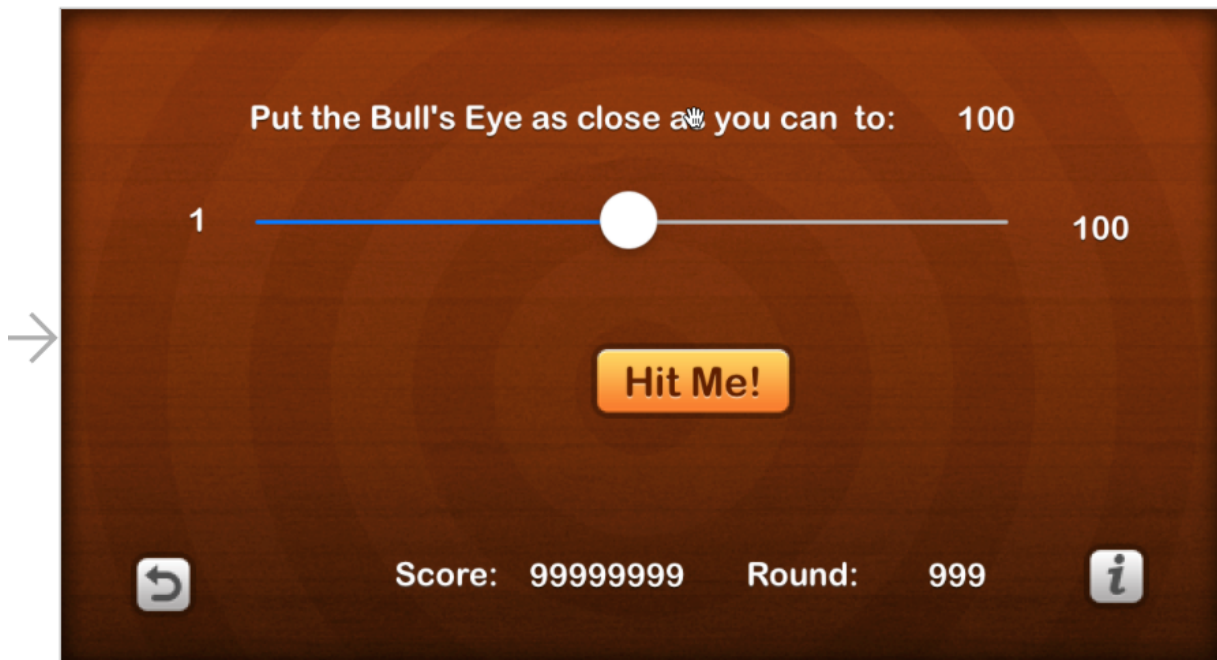


科普：关于iOS开发中所使用的字体

Interface Builder中所显示的字体是你的Mac电脑上有的字体，但不能确保在iPhone上也有。实际上iPhone所支持的字体要远远少于Mac上的字体。怎么办呢？建议你到这个网站来看看iOS中可以用的字体：

<http://iosfonts.com>

接下来我们用类似的方式来美化按钮。我们可以给按钮添加背景图片，设置字体等等。具体如何操作就不再赘述了，你完全可以发挥自己的想象力，而最终的效果可能是类似下面的：



美化滑动条

对滑动条的美化要稍微复杂一些，事实上如果单纯使用Interface Builder的话我们只能稍微修改滑动条的外观。

这个时候，手写控件的灵活性就会体现出来了。

比如，为了设置按钮的色彩，我们既可以在Interface Builder中直接设置，也可以使用代码setTitleColor()来实现。

当然，通过可视化的设计方式（所见即所得）可以大大提高开发效率。不过对于某些特殊的情况，比如这里的滑动条，我们也不得不直接手动控件来修改其属性。谁让苹果开发团队偷懒没给它提供在Interface Builder中直接修改的方式呢。

在Xcode中切换到ViewController.swift，在viewDidLoad()方法中添加几行代码：

```
override func viewDidLoad() {
```

```
    super.viewDidLoad()  
    startNewGame()
```

```
    //设置滑动条的外观
```

```
    let thumbImageNormal = UIImage(named: "SliderThumb-Normal")!  
    slider.setThumbImage(thumbImageNormal, for: .normal)
```

```
    let thumbImageHighlighted = UIImage(named: "SliderThumb-Highlighted")!  
    slider.setThumbImage(thumbImageHighlighted, for: .highlighted)
```

```
    let insets = UIEdgeInsets(top: 0, left: 14, bottom: 0, right: 14)
```

```
    let trackLeftImage = UIImage(named: "SliderTrackLeft")!  
    let trackLeftResizable = trackLeftImage.resizableImage(withCapInsets: insets)
```

```
    slider.setMinimumTrackImage(trackLeftResizable, for: .normal)
```

```
    let trackRightImage = UIImage(named: "SliderTrackRight")!  
    let trackRightResizable = trackRightImage.resizableImage(withCapInsets: insets)
```

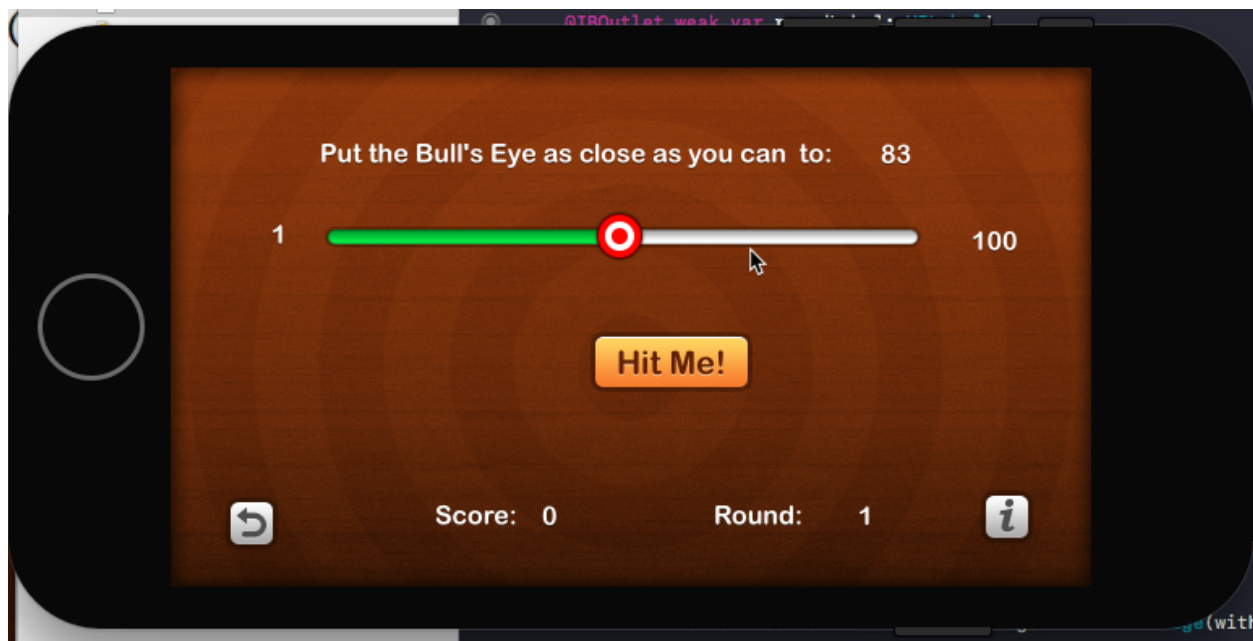
```
    slider.setMaximumTrackImage(trackRightResizable, for: .normal)
```

```
}
```

在上面的代码中，我们为滑动条准备了四种图片：两个结点图片，两个滑动背景图片。结点图片和按钮类似，因此有一个正常状态，还有一个高亮状态。同时滑动条对于结点两边的滑动背景也采用不同的图片。左边的是绿色，右边的是灰色。

如果你看不懂上面的代码怎么办？按住option键，点自己需要看的方法 imageNamed,或者 setThumbImage，就可以查看详细的帮助说明。

点击Run运行游戏，现在我们的游戏界面看起来就比较顺眼了！



小提示：关于文件名名称

你可以还记得我们导入图片资源库（asset catalog）中的文件名是类似SliderThumb-Normal@2x.png。

当我们创建一个UIImage对象时，并不会使用原始的文件名，而是在asset catalog中的文件名，比如SliderThumb-Normal。这就意味着我们无需添加@2x后缀。

小技巧：

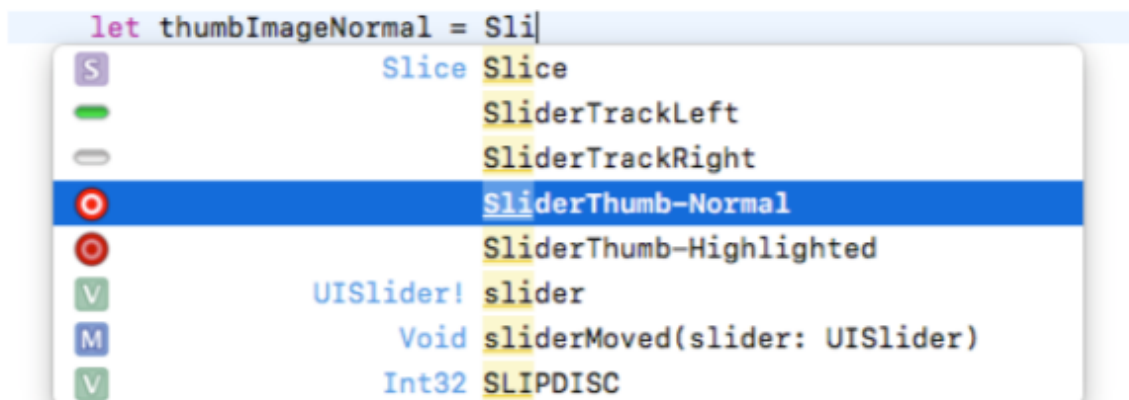
Xcode现在提供了一个很方便的新特性，可以轻松在代码中添加图片。
比如上面的代码：

```
let thumbImageNormal = UIImage(named: "SliderThumb-Normal")!
```

我们可以输入：


Let thumbImageNormal = Sli


此时，Xcode的自动完成功能就会给出关于Sli的提示，其中包含了所有名字中有以上字幕的图片：




从列表中选择SliderThumb-Normal，此时Xcode会添加一个小的图标到代码中，这个小图标就是所谓的image literal。重复这种操作，那么刚才添加的代码就变成了下面的样子：

这种小图标一下子让整个代码变得生动起来，是不是很好玩呢~

```
// Customize slider
let thumbImageNormal = 
slider.setThumbImage(thumbImageNormal, for: .normal)

let thumbImageHighlighted = 
slider.setThumbImage(thumbImageHighlighted, for: .highlighted)

let insets = UIEdgeInsets(top: 0, left: 14, bottom: 0, right: 14)

let trackLeftImage = 
let trackLeftResizable =
    trackLeftImage.resizableImage(withCapInsets: insets)
slider.setMinimumTrackImage(trackLeftResizable, for: .normal)

let trackRightImage = 
let trackRightResizable =
    trackRightImage.resizableImage(withCapInsets: insets)
slider.setMaximumTrackImage(trackRightResizable, for: .normal)
```

再次点击工具栏上的编译运行按钮，看看是否还能够正常运行，只要你没输错，结果当然是令人满意的~

好了，这一课就到这里结束了，让我们享受下福利吧~