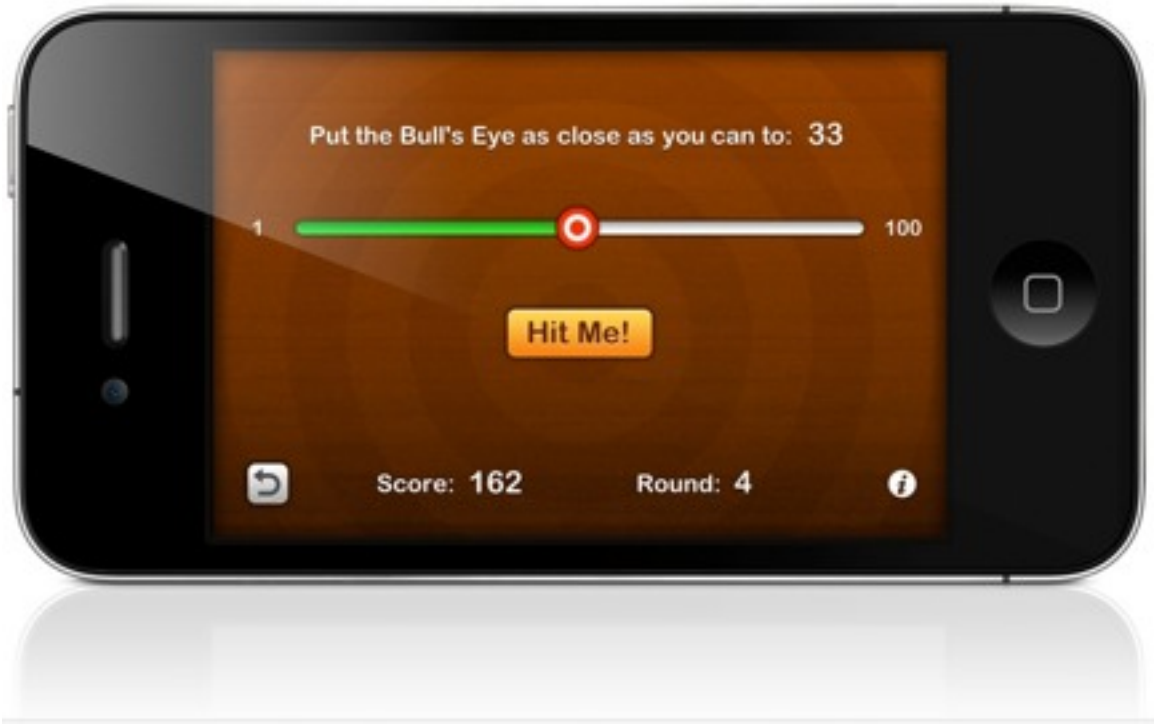
让不懂编程的人爱上iPhone开发(2013秋iOS7版)-第2篇

“拖拖看”小游戏

在我们学习iPhone开发的第一站，将创建一个叫“天天来打靶”的小游戏（好吧，这名字也是与时俱进，跟企鹅学的，天天xxx~）。当游戏最终完成后的效果如下（考虑到iOS7的变化，最终效果会略有差异）：



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33。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。

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最终的视觉效果了，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哪些事情，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。这就是我的编码用事务清单（to-do list）。有了这样的清单，可以把一个产品的设计和功能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。比如制作一款手机网游，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"Hit Me!"
2. 当玩家触碰Hit Me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。因此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。

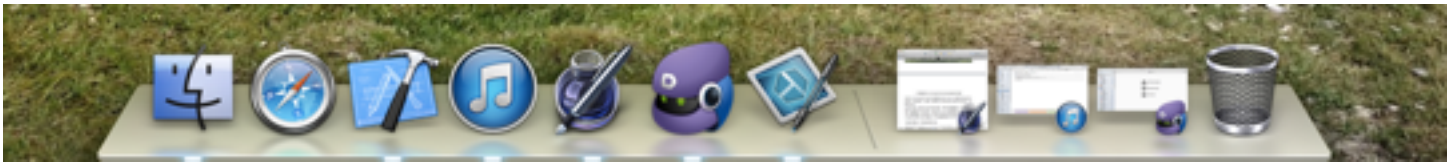
这个小应用的实际运行效果如下：

好了，终于到了真正写代码的时候了！想来你已经有了Moutain Lion操作系统，并且已经下载安装了最新版本的Xcode(目前是5.0)，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。

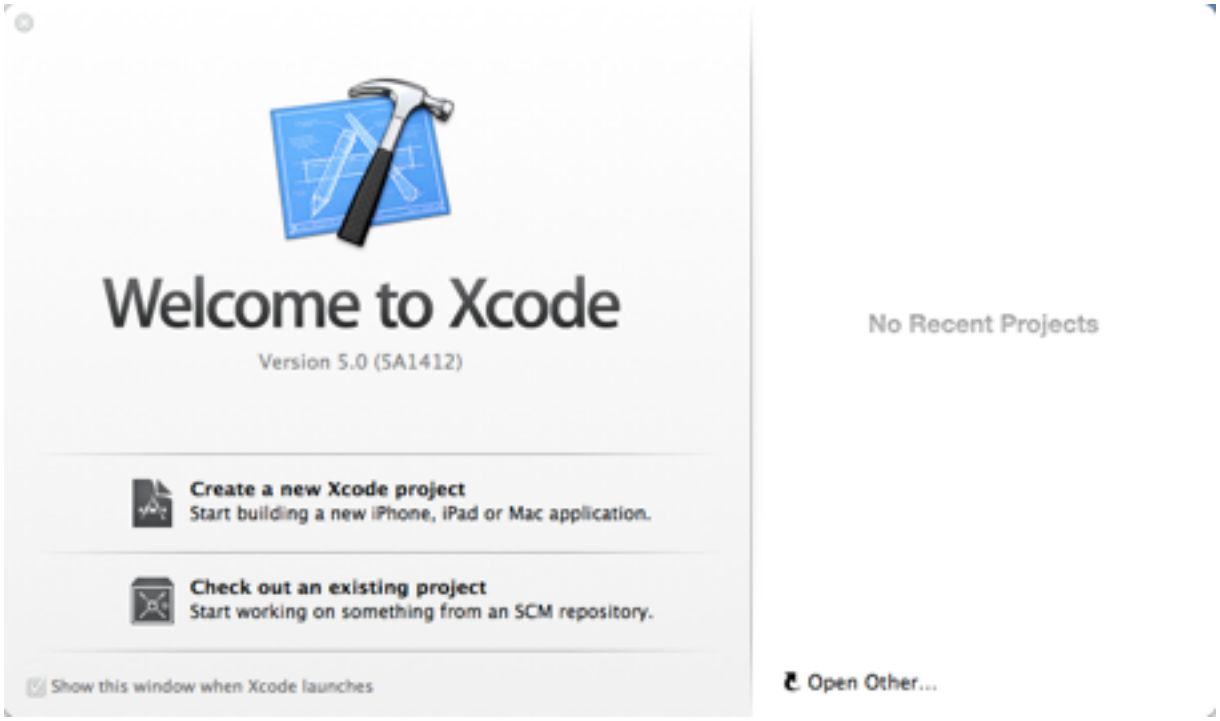
在iOS5和之前的版本间存在着某些巨大的差异，可能有些公司还在用老的Xcode和iOS版本，但作为新手来说，就没必要了，一切从新开始。

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。 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，我就不废话了。
顺便说句题外话，上面的Dock里面有个奇怪的动物图标，叫Dash，对于程序猿很有用，感兴趣可以去了解下。

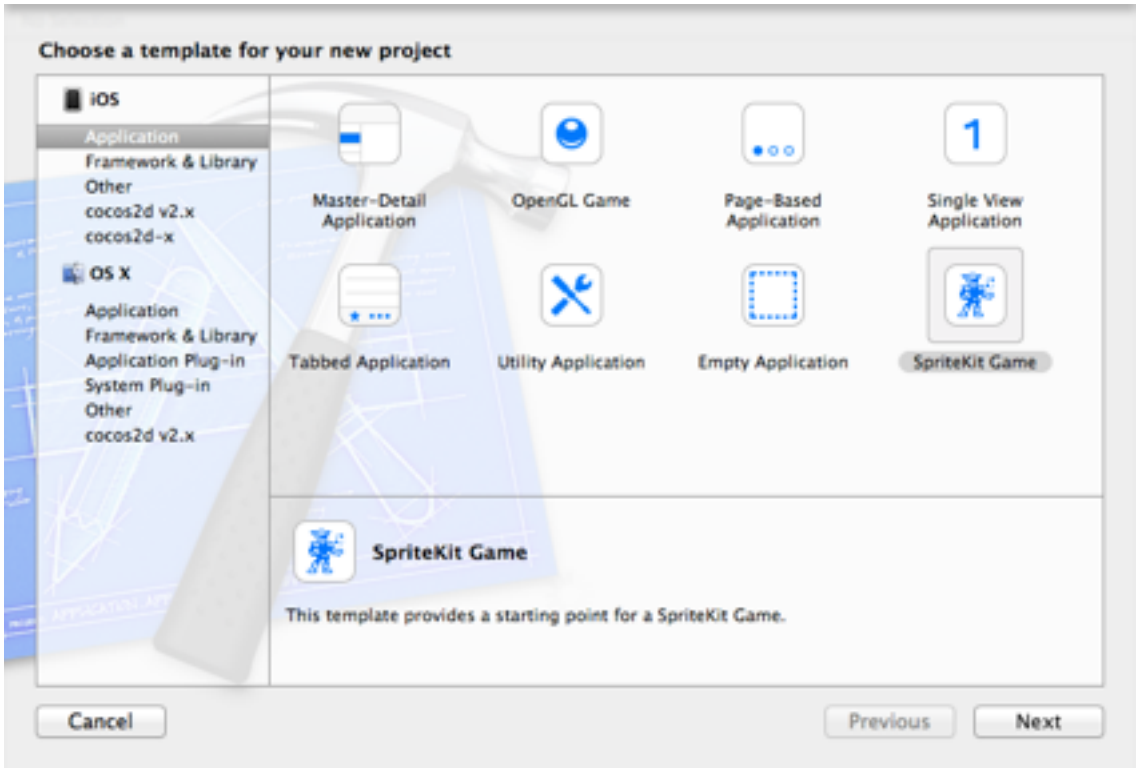


打开Xcode后,首先看到的是欢迎界面，喜迎你的到来。



Step2

这时候点击左侧的Create a new Xcode project，又是一个辅助界面，帮你选择所需要的模板。



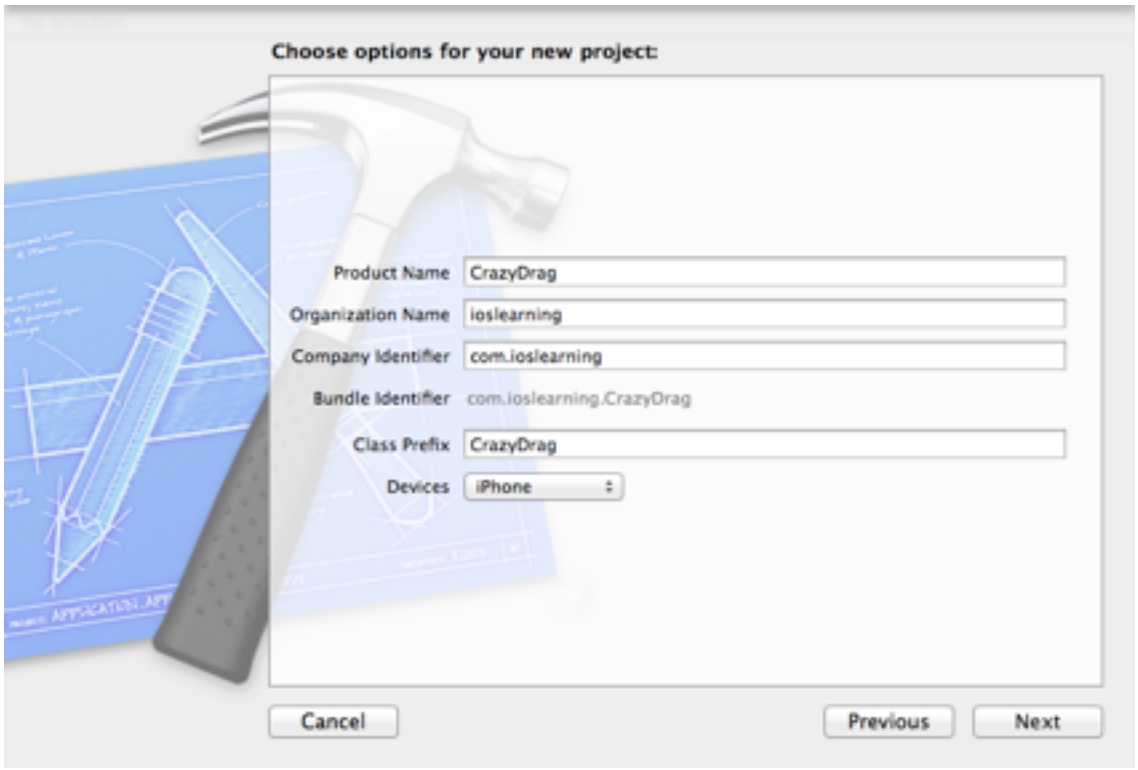
如果你是第一次使用Xcode，可能和我这上面有些差别，因为我还装了cocos2d v2.x和cocos2d-x，这两个是开发2D游戏用的，你暂且不要管它，后面会有相关的介绍。

点击左边的不同文字，你会看到不同的模板，不过默认是Application（应用）所对应的模板。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Xcode5.0，或许这里所显示的模板会有所差异，但至少就“拖拖看”这个游戏来说，所需要的模板在这里是有的。

Step3:

选择Single View Application,然后点击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：



这里说一下各部分的内容该怎么设置吧：

Product Name:产品名称，通常个人习惯用不带空格的纯英文字母组合来写。至于真正发布的产品名称可以在项目中设置。

Organization Name:开发团队名称，随便写，不硬性

Company Identifier: 产品开发者的标识符，如果你已经注册了付费的iOS开发者计划，Xcode会自动帮你填充。如果没有，你可以写一个类似域名的字母数字组合，比如:com.mydomainname（这里的mydomainname用你自己喜欢的字母数字组合来替换）。即便写错了也无所谓，项目中可以重新设置。

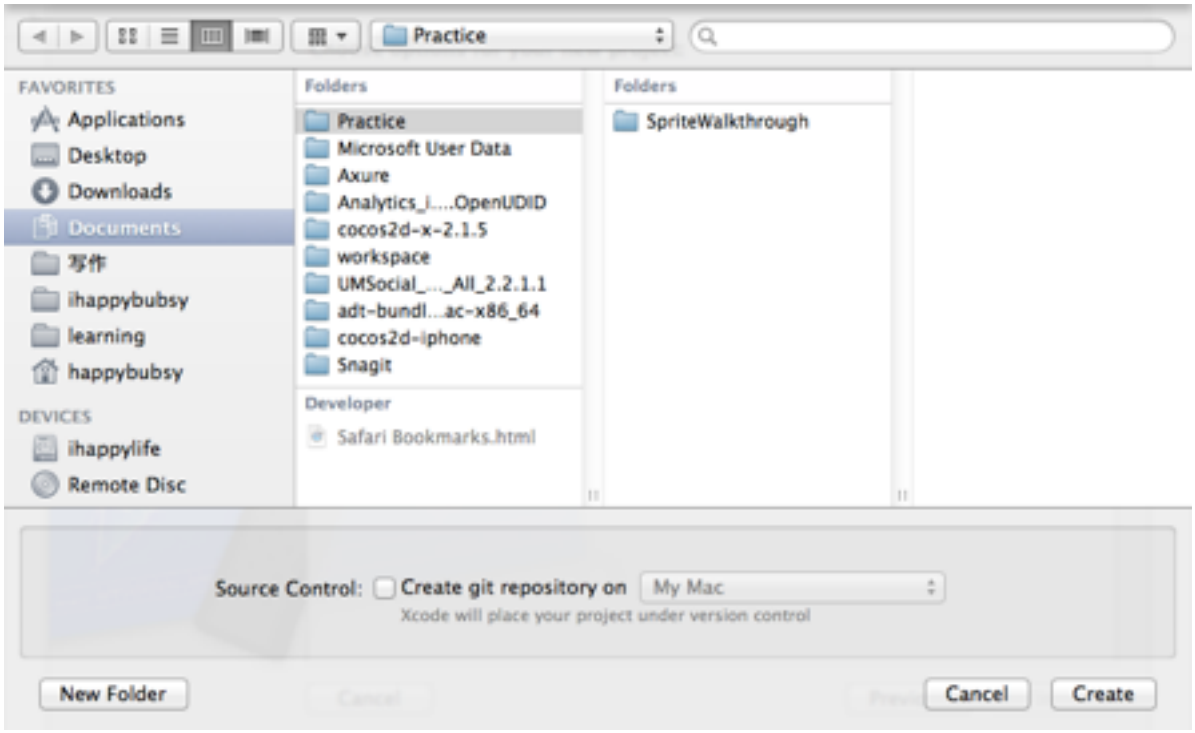
Bundle Identifier:根据Company Identifier和Product Name自动生成。

Class Prefix:类前缀，这里用CrazyDrag，也就是产品名称。

Device:三种选择，iPhone,iPad，Universal，对这个游戏来说就选iPhone。

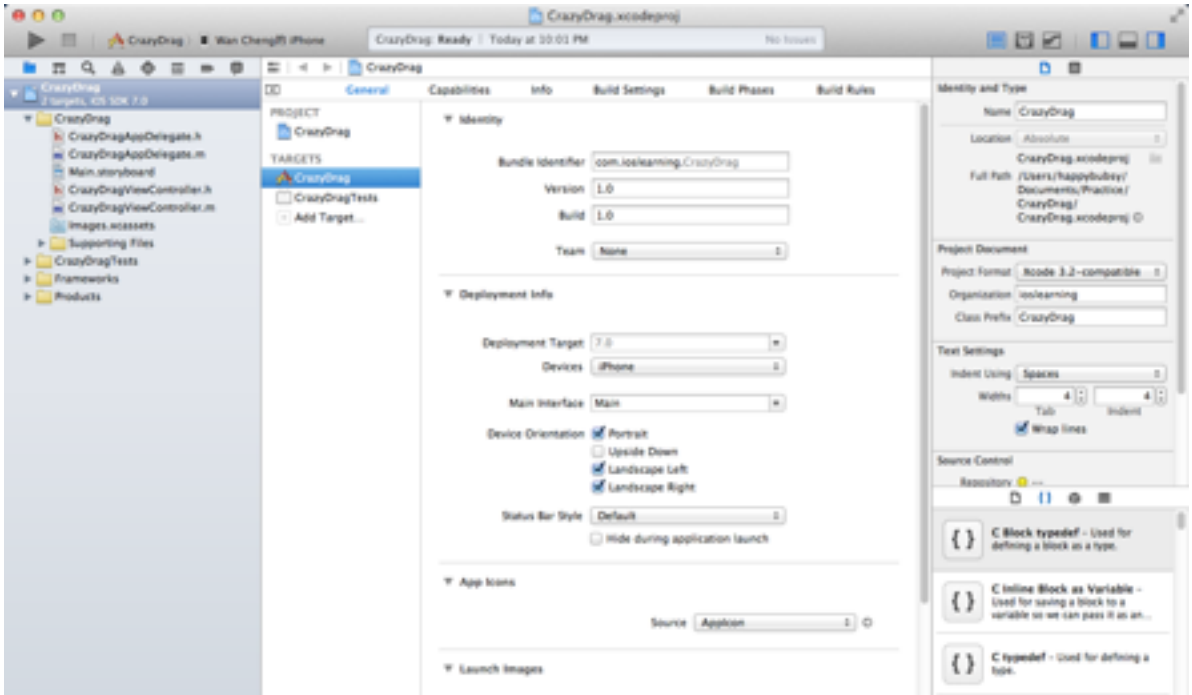
Step 4:

点击Next,选择一个地方保存该项目。



这里不勾选Create local git repository for this project（当然选择了也没错）
你现在不用管它什么意思，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:
点击Create，Xcode会基于Single View Application这个模板创建一个名为CrazyDrag的新项目。完成后的屏幕显示如下：
你会看到有个Deployment Target的地方显示7.0，如果想支持之前的版本，需要手动切换成5.0。



关于iOS的版本，这里还是多废话几句吧。

从iOS1到现在的iOS7，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。比如目前一般的游戏和应用向下最多支持到4.3，95%以上的用户都在使用5.0和以上的系统。因为各个系统版本所支持的API和功能也有所差异，新开发的产品通常会选择支持5.0或5.1及以上。现在7.0出来了，预计很短时间内绝大多数新产品会选择支持6.0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）

iPhone4是乔帮主亲自发布的最后一款也是最成功的一款iPhone手机，而iOS5则是乔帮主亲自发布的最后一款iOS操作系统。

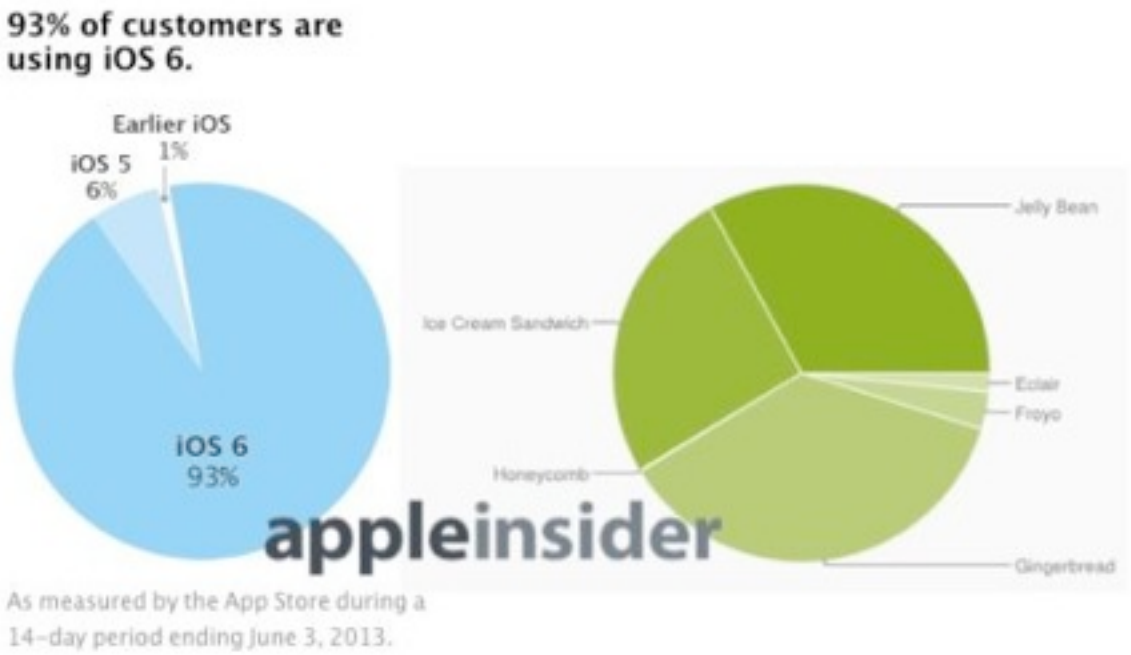
iOS6(2012年）

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。
虽然刚发布时iOS6中的地图功能让人诟病，但如今iOS6已经是最普及的iOS操作系统。

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。

根据苹果的官方数据，目前iOS设备所支持的最主流操作系统是iOS6,大概占了93%，而使用iOS5的iOS设备大概占6%,剩下的只有1%。



也就是说，从开发者的角度来看，完全没有必要支持iOS5以下的系统了，甚至可以只支持iOS6及其以上版本的系统。
支持ios5.0的设备有iphone5,iphone4s,iphone4,iphone3gs,ipad,ipad2,ipad mini,ipad retina(3,4代,含牛排)，ipod touch3,4。更老的设备不支持iOS5.0,但占有率也少的可怜。

这也是为神马新入门的开发者最好直接从iOS5或iOS6开始学起的原因。

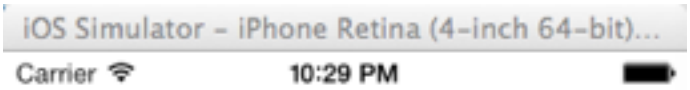
再说下设备分辨率的问题吧，目前iOS小屏设备中支持retina显示的设备有：
iphone5s,iphone5c,iphone5(1136*640),iphone4s(960*640),iphone4(960*640),ipod touch 4,5(960*640),ipod touch6(1136*640)

目前来说，支持iOS5.0的分retina分辨率设备中唯有iPhone3gs和ipod touch3代。
之所以说这个，是因为之前美术作图都要做标清和高清两个版本，但随着retina设备的高覆盖，相信以后就没这个必要了。

Step6:
选择Simulator的设备类型

这里选择iPhone Retina(4-inch 64-bit)，当然，你也可以选择其它设备。这个64-bit的设备当然就是iPhone5S了，虽然有点贵买不起，但在模拟器中体验一下还是没问题的吧~
另外还有四种设备的模拟器可选，感兴趣可以分别试验下。

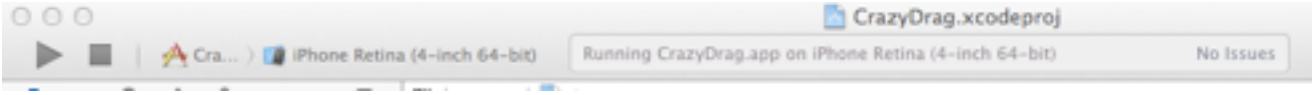
Step7:
点击Xcode最左上的Run
Xcode会打开iOS Simulator(iOS设备模拟器)，从而在模拟器中运行我们的新应用。
当然，实在没什么看头，不过是一个灰白的界面而已，放在白色的文字背景上甚至有点区分不出来，而且你也没法和它交互，但不管怎样，这是我们一个真正的里程碑。



为了让你能够区分界面和真正的空白，只好在这里敲几行字，这就是iOS7让人不爽的一个地方，苍白无力啊。

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。

但在你按下stop之前，Xcode中会显示”Running CrazyDrag.app on iPhone Simulator“。



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Objective-C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，它可以帮助你将Objective-C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，xib文件等等，并把这些东西打包到一个叫“应用程序束”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。

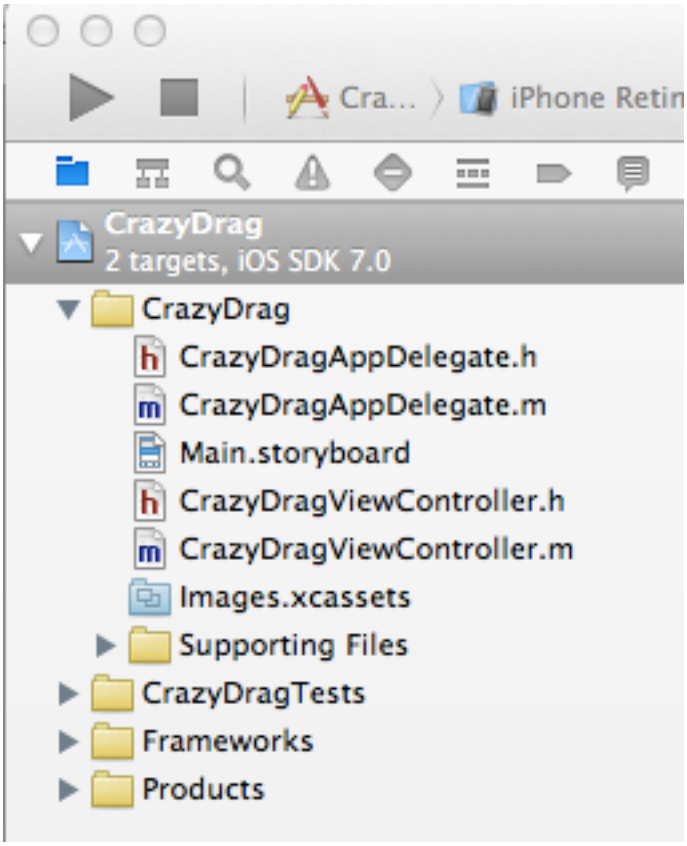
添加按钮

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。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

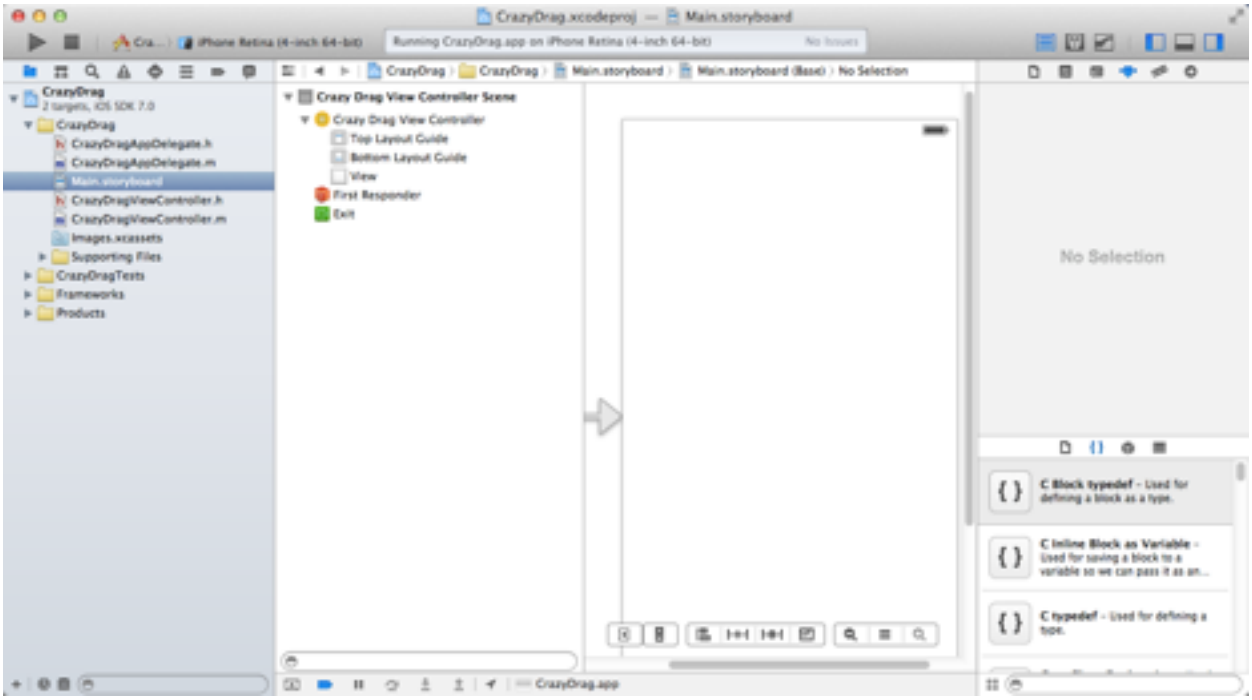
在当前的项目导航中，找到一个名为”Main.storyboard”的文件，点击选中它。



在一瞬间，你刚刚开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。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：

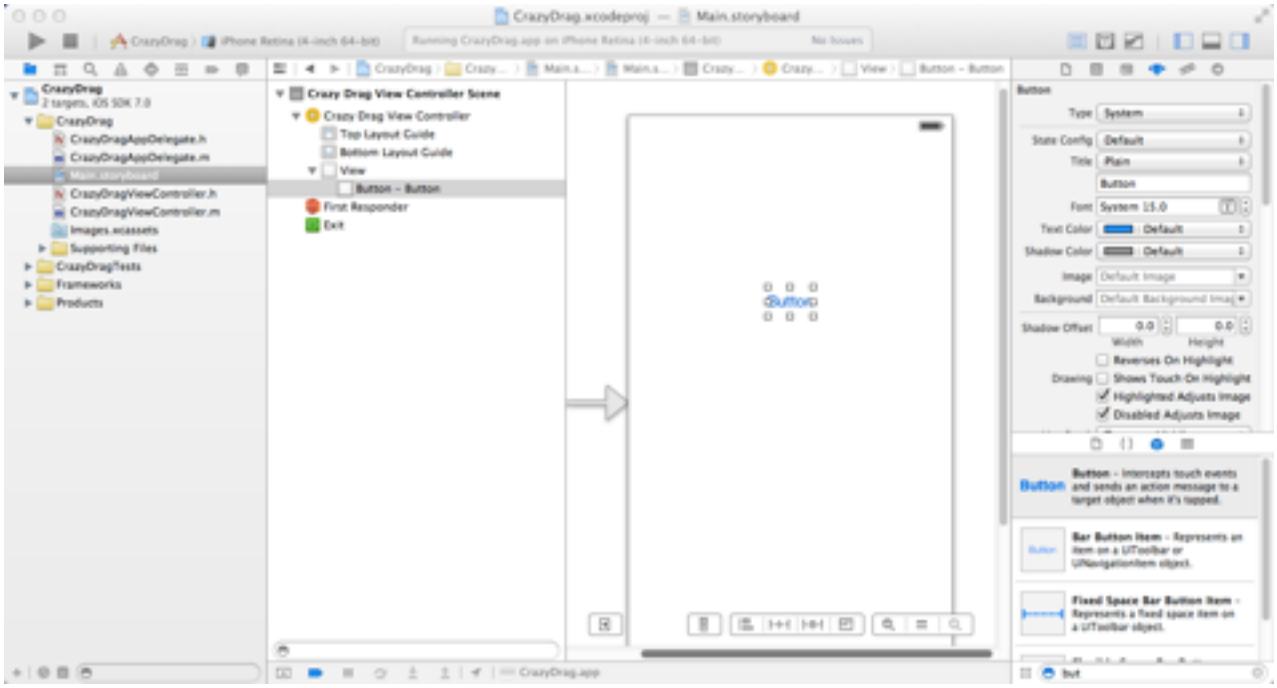


Step 3在Object Library中找到按钮控件

在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。

Step4 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

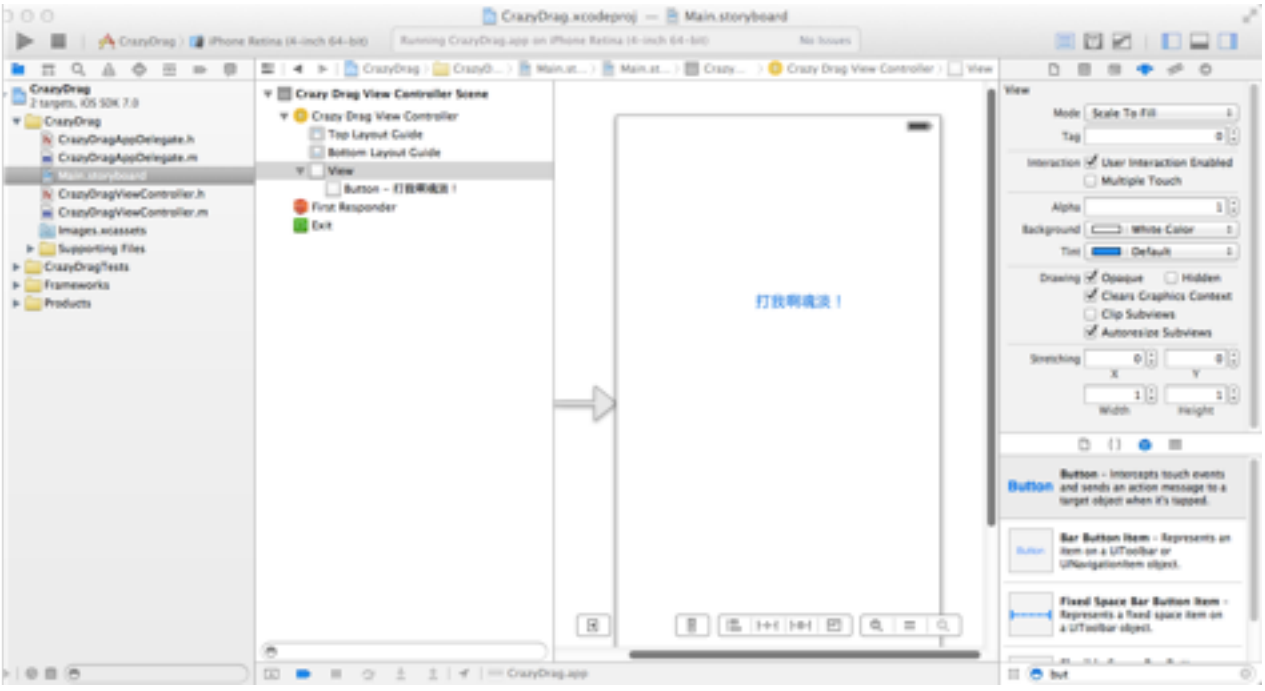
可视化编程是随着面向对象编程概念的兴起而变得普及的。对于旧石器时代的程序猿来说，开发工具远没有今天的各种花哨SDK好用，部分程序猿甚至习惯于在文本编辑器里面直接写就代码。当然，这样做的好处是让这些旧石器时代开发者奠定了非常牢固的代码基础。直到现在仍然有很多程序猿偏好于在非常简单的界面中写代码。但这种类似写文章的编码方式也有很多不足之处，首先它要求程序猿对编程语言，各种类库,API非常的熟悉，不说可以达到左右手互博的程度，起码写10行代码不会轻易出一个错。如若不然，放到编译工具里面调试的时候会让人头大。随着面向对象语言的兴起，以及开发工具的进步，各平台的代码编辑器都开始变得智能起来，可以第一时间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。

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，什么乱八七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，你要做的事情也和在Photoshop上设计界面没有区别。

Step7. 双击按钮，编辑其中的内容为“打我啊魂淡！”之类的，随便你



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魂淡！

在这里才是界面的边界~

在学习iOS编程的时候，也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。然后就是要在设备上多测试。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。

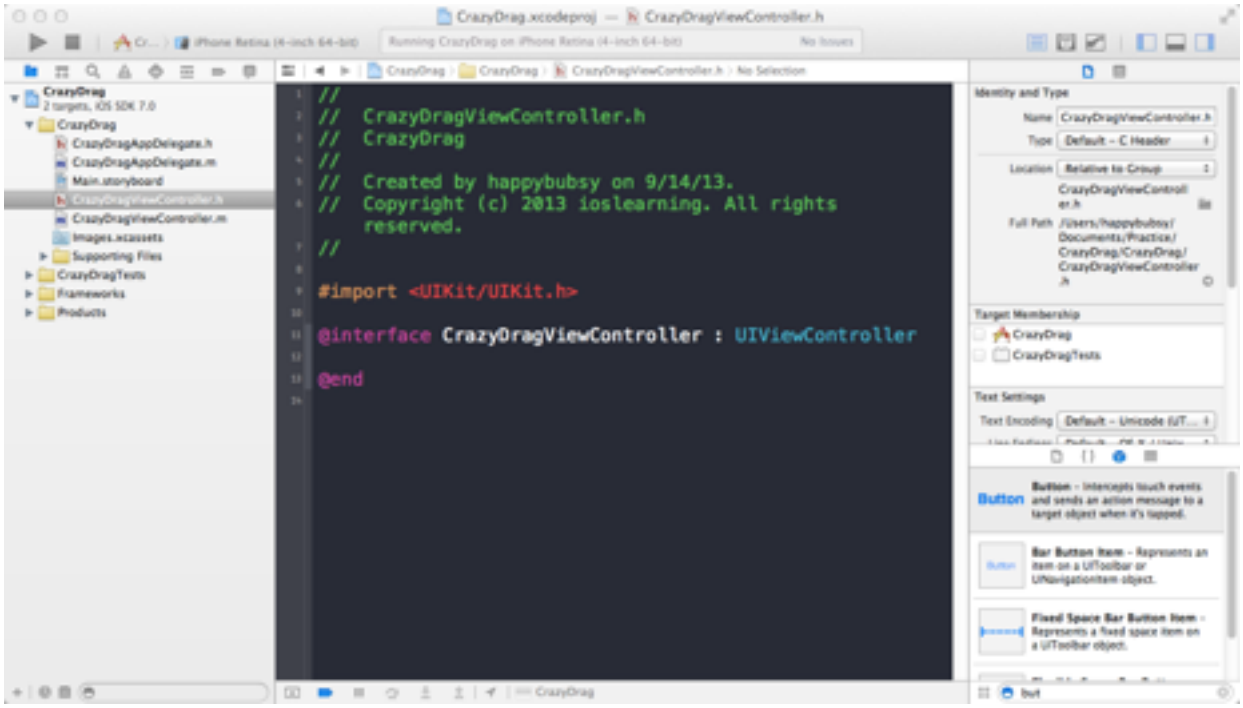
源代码编辑器

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，不过这里我们来个俗点的，只显示一行文本“你好，苍老师！”

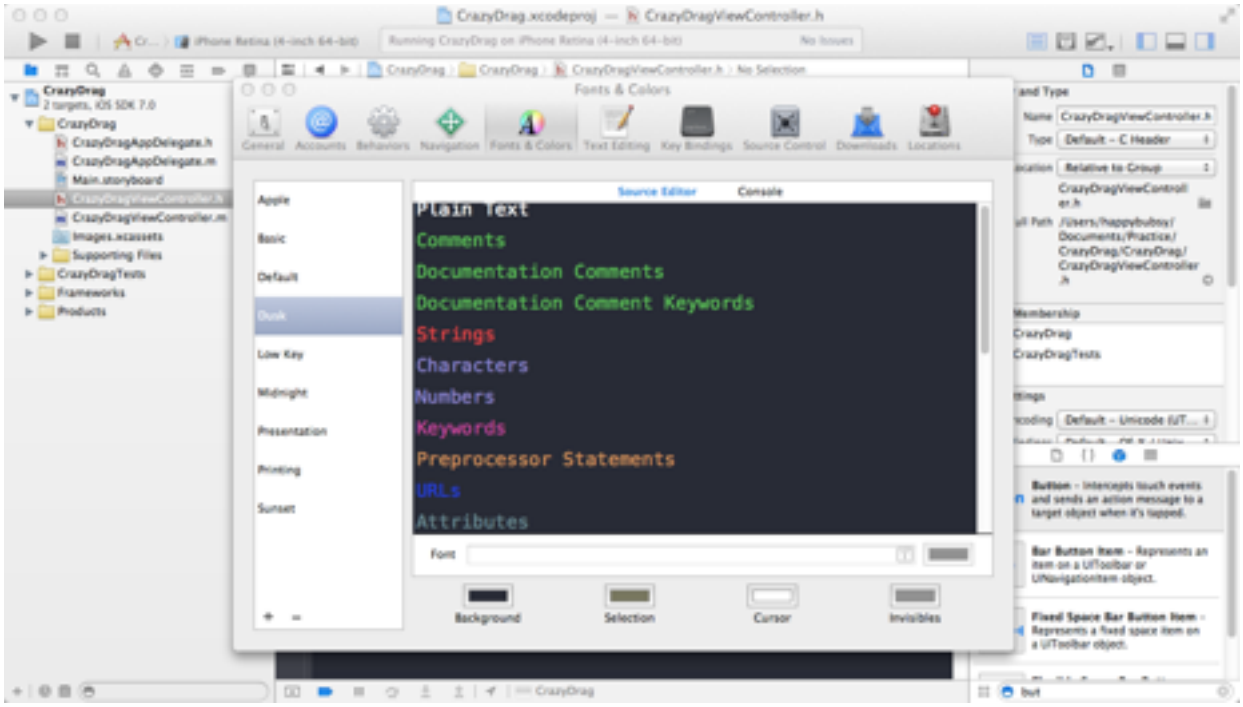
再废话一句，第一次看这个教程的时候新手可能有一些不理解的地方，但别钻牛角尖，先看完，对一些重要的概念我会重复展示，直到你熟悉为止。别没学会走就想成飞猪。你要做的就是，跟着学。信春哥，得永生。

Step 7在Xcode左侧的项目导航（也就是文件列表）中找到CrazyDragViewController.h，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Objective-C源代码。

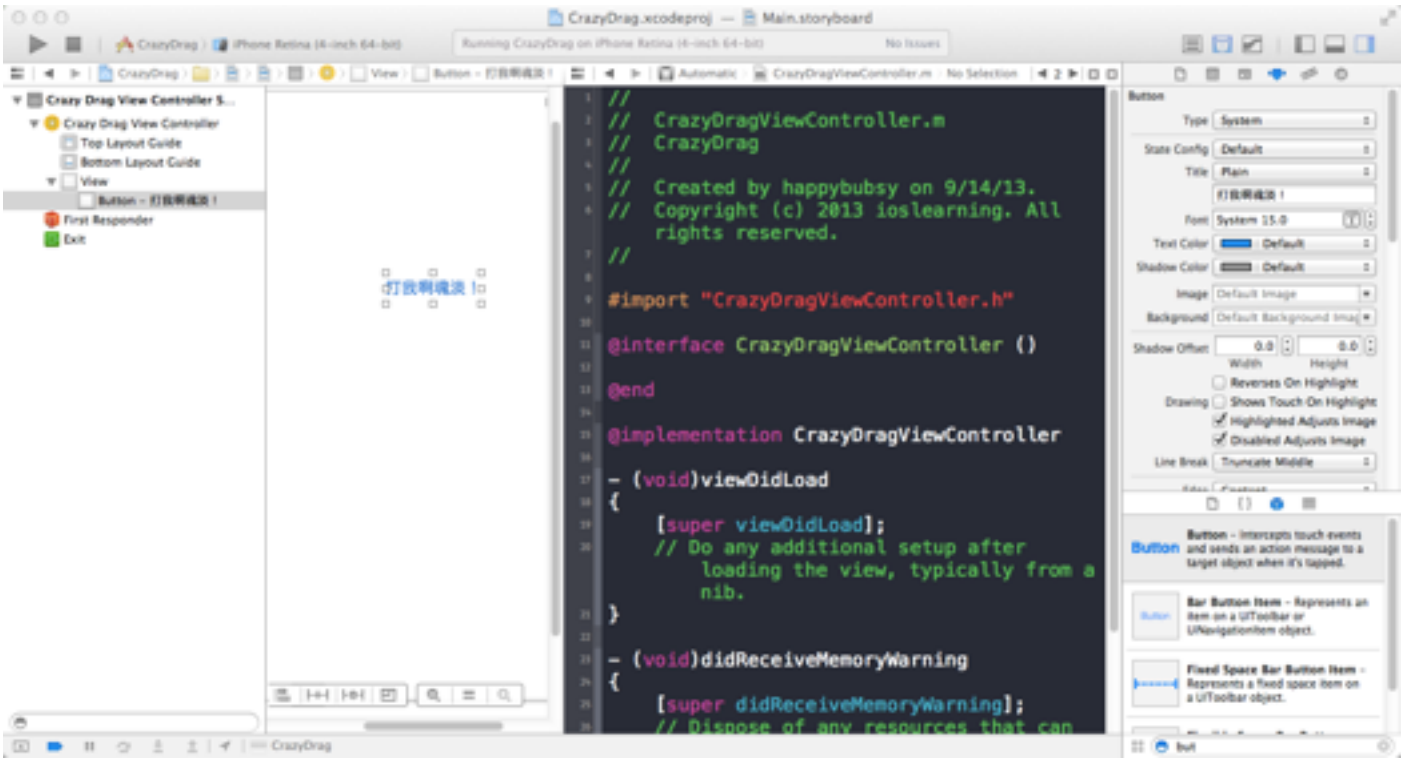


慢着，你可能会问，为毛我的编辑器背景是黑色的呢？这个也是个小小的技巧，毕竟程序猿一天到晚对着电脑看代码很伤眼睛，所以可以自己设置自己喜欢的背景来调节。点击Xcode-->Preferences...，然后切换到Fonts&Colors，就可以选择自己喜欢的色彩搭配了。我这里选的是Dusk，你看着办。



你还可以切换到CrazyDragViewController.m，会看到更多的代码。

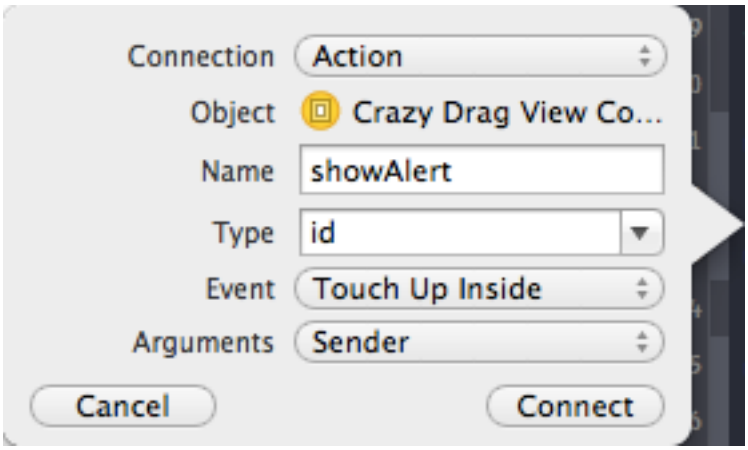
在你彻底被代码吓晕过去之前，我们先回到Main.storyboard，单击选中“打我啊魂淡”这个Button。单击Xcode最右上的6个按钮中的第2个（如果把鼠标悬浮在上面会显示：Show the Assistant Editor，再单击Hide or Show the Navigator按钮（倒数第3个），这时Xcode就成了下面的样子：



让**界面变得可交互**

按住Control键不放，用鼠标从“打我啊魂淡”这个按钮拖到@inerface ....和@end之间的某个位置。

这时会弹出一个小对话框，在里面输入showAlert，其它设置参考图片中的设置，比如把Connection后的选项改为Action，



完了点击Connect，代码编辑区(CrazyDragViewController.m)的内容变成：

```
//  CrazyDragViewController.m
//  CrazyDrag
//
//  Created by happybubsy on 9/14/13.
//  Copyright (c) 2013 ioslearning. All rights reserved.
//

#import "CrazyDragViewController.h"

@interface CrazyDragViewController ()

- (IBAction)showAlert:(id)sender;
@end

@implementation CrazyDragViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically
        from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)showAlert:(id)sender {
}
@end
```

你对这段Objective-C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CrazyDragViewController.h里面添加了人生的第一行处女代码（虽然是由Xcode自动帮你添加的，而且我们完全不懂是什么意思）。你很可能在猜，这两个前缀名相同的文件之间究竟是什么关系？别忘了还有一个前缀名完全一样的CrazyDragViewController.m文件。
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画面的工具。
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Recipse list是菜谱清单，其中列出了几种不同的菜谱名称。而右边的Recipse 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（灰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CrazyDragViewController。

在Xcode开发团队推出Storyboard之前，通常用xib文件进行用户界面的设计，简单点说，后缀分别是.xib,.h和.m，前缀完全相同的这三个文件共同实现了一个view controller(视图控制器)。

再简单点说，xib文件包含了这个视图控制器的用户界面设计，而.h和.m则包含了这个界面的交互功能-也即让用户界面如何运作的逻辑机制，使用Objective-C语言来编写。通常我们使用其代表的视图控制器为这些文件命名，这里是CrazyDragViewController。
当然，视图控制器的用户界面设计并非只有xib文件这一种实现方式，而是有三种。

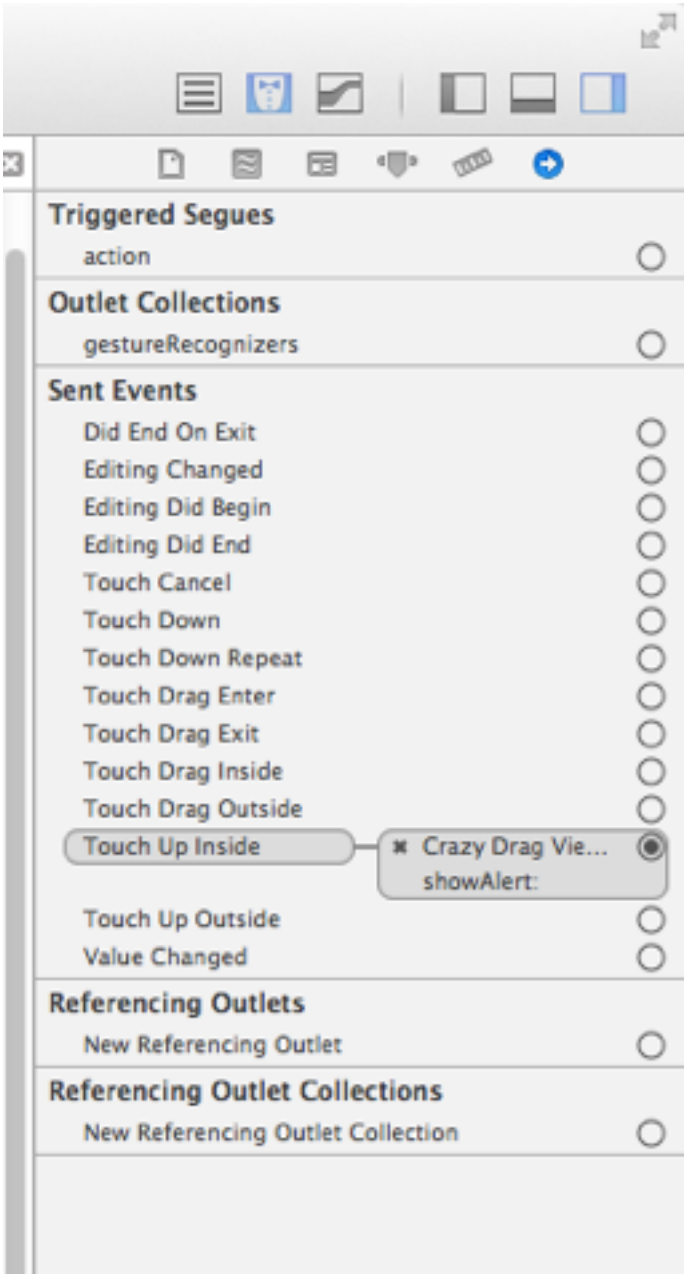
从iOS5开始，苹果推出了storyboard（故事板），也即用故事面板的形式来展示界面间的逻辑关系。storyboard的功能非常强大，可以节省很多开发工作，但同时也有其不好的一面。如果界面关系非常复杂，用storyboard反而是很麻烦的一件事情。
至此，开发者可以选择用三种方式来实现用户界面，分别是手写代码，xib和storyboard。
从代码可移植性的角度来看，纯手写代码 > xib界面文件 > storyboard
从代码复杂度的角度来看，storyboard > xib 界面文件 > 纯手写代码

之前比较大型的项目，如果需要不同团队和个人配合，很多会选择完全手写代码，但是最新的Xcode5提供了对xib非常好的支持。个人的感觉是，大多数情况下用xib，少部分情况下用纯手写代码，一些界面逻辑关系非常简单的可以考虑用storyboard，比如本系列的例子。
此外，使用cocos2d-x等引擎开发游戏的时候，纯手写代码的可移植性和可维护性也能得到很好的保证。

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！）。同时你也可以通过这里直接从某种事件拖曳到File's Owner上。只需按住右侧的小圆圈不放就可以拖曳出蓝线出来。

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CrazyDragViewController.m。

你可能会想.h和.m文件的区别究竟在哪里？.h文件告诉计算机视图控制器要做什么，而.m文件则会告诉它如何来完成这些事情。看不懂？没关系，这里只是提一下，下一篇教程会详细解释这一点。

现在CrazyDragViewController.m中已经有了一大堆代码，不过我们暂且不要看。
在文件的底部有一段代码：

```
- (IBAction)showAlert:(id)sender {

}
```

修改这段代码为：

```
- (IBAction)showAlert:(id)sender {
    [[[UIAlertView alloc]initWithTitle:@"您好，苍老师" message:@"听说您的新贴转发了499次" delegate:nil cancelButtonTitle:@"我来帮转1次，你懂的" otherButtonTitles:nil, nil]show];
}
```

当我们之前通过storyboard拖曳出一个关联时，Xcode会帮我们自动在.h文件中添加-(IBAction)showAlert;这行代码，其实我们是在告诉Interface Builder,"showAlert这个动作

已经准备好了”，但showAlert究竟干什么我们还没提。而这里就具体说明了showAlert会怎么来做。{}之间的代码告诉iPhone要做哪些事情，执行的顺序是从头到尾顺序执行。

在{}的代码里面，创建了一个提示对话框，标题是“你好，苍老师”，消息体是“听说您的新贴转发了499次”，然后一个简单的按钮，上写“我来帮转1次，你懂的”。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。

注意：在具体实现showAlert动作的内容时，我们只需要在.m文件中添加这段代码，而没有在CrazyDragViewController.h中修改。.h文件只是为了说明我们要做什么事情（也就是所谓的interface接口），而实际如何来做则会放在.m文件中（也就是所谓的implementation实现）。

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，至于你转了这一次后会不会查水表，就不是我等需要关注的事情了，自求多福吧。

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求婚体对话框，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。Objective-C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。。。

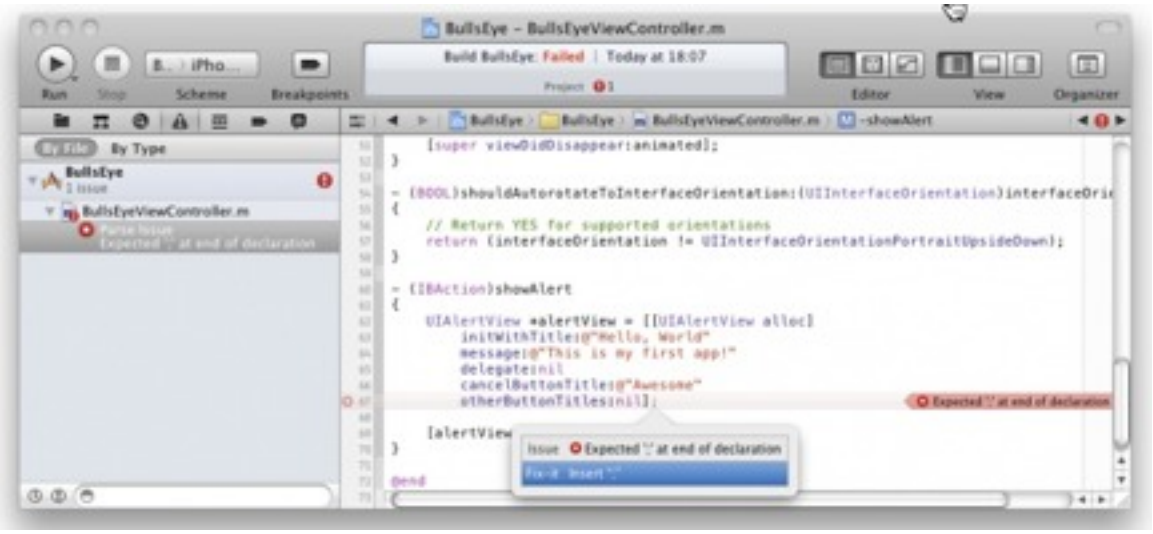
当Xcode告诉你”Parse Issue”或”Expected <something>”的时候，很可能你忘了在每行代码的最后加上一个分号。
是的，忘了在代码后面加上分号是开发中另一个常见的错误。Objective-C语言写的每一行代码之后都必须以分号结束，正如英语中以句号结束一样（什么，可以用感叹号？！你可以瞑目了）。

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。比如在上图中我漏掉了一个分号。

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。

好了，如果没有找到错误，成功的看到求婚体提示，你今天的学习就可以到此结束了。

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。