

好了，继续我们上次的课程。

在上一篇的内容中我们花了很多功夫进行一些科普和理论知识的介绍，现在又到了动手时间了。

首先回到CrazyDragViewController.m，更改sliderMoved动作的内容如下：

```
- (IBAction)sliderMoved:(id)sender {
    UISlider *slider = (UISlider*)sender;

    currentValue = lroundf(slider.value);
}
```

这里我们删除了NSLog()这行代码，然后增加了下面的这行代码替。

```
currentValue = lroundf(slider.value);
```

那么这行代码是什么意思呢？之前我们曾看到过slider.value，它代表当前时刻滑动条的数值。该数值的范围在1到100之间，当然后面还有小数点几位。currentValue是我们之前所创建的变量名称。

当我们要把一个新的数值放到一个变量里，只需要这样来写：

```
variable = the new value;
```

这就是术语党们号称的“赋值”。我们把一个新的值“赋予”变量。这里我们把滑动条的数值赋予currentValue这个变量。

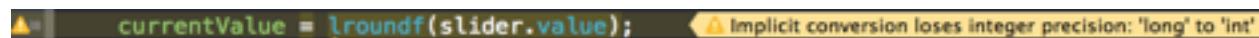
这个应该比较好理解，但lroundf又是怎么回事？刚才我们说过，滑动条的数值不是一个整数，而是小数点后面还有几位(在我的Xcode里面小数点后面有6位)。

刚才使用NSLog()的时候，只要拖动滑动条，就可以在调试区看到这个数值。

当然，这样一来游戏的难度也大大增加了。你不可能猜得那么准，甚至连小数点后面6位都猜到，除非你是神。为了让游戏难度不至于这么变态，我们需要把数值转换成一个最接近的整数。这就是为神马我们把currentValue定义为一个int整数值，因为int类型的变量里面会保存整数(integers)。

这里的lroundf()是一个函数，它可以把一个带小数点的数值四舍五入到最接近的整数，然后我们会把转换后的整数保存到currentValue这个变量里。

不过如果你足够细心的话，会发现在这行新添加代码的旁边有个黄色的警告。

A screenshot of an Xcode warning message. On the left, a line of code is shown: `currentValue = lroundf(slider.value);`. To the right of the code, a yellow warning icon is followed by the text: "Implicit conversion loses integer precision: 'long' to 'int'".

警告的内容是：Implicit conversion loses integer precision: 'long' to 'int'

这是什么意思呢？如果你英文过得去的话，看字面意思就是，隐式变换导致integer的精度丢失。我们所定义的currentValue是int类型的变量，而lroundf这个函数返回的是long int类型的变量，所以Xcode提醒我们可能会造成精度丢失。

在计算机的世界里，即便是简单的整数也是分为多种类型的，每种类型可以保存的数值范围也有所不同。比如标准int类型的取值范围是-32768-32767，而long int的取值范围是-2147483648-2147483647。此外还有无符号的整数（也就是正数或0），比如unsigned int。除了整数类型，还有带小数的数值类型，比如float, double等等。那么我们应该选择使用哪种类型来保存一个基本的数值呢？

首先一点，Do not panic。不要恐慌，你无需像记住九九乘法表一样记住每种变量的取值范围（虽然你可以这样做，而且熟悉了之后不需要刻意去记也大概知道），可以从书或者网络上查找。此外，在一般的程序开发中，整数类型用int就可以了，小数类型用float也差不多。但是如果涉及到复杂的数学运算，可能就要改用long int和double。

在我们这个例子中，可以确定的是滑动条的数值在1-100之间，所以int是足够的。但这个黄色的三角警告总是让人看着有点不爽，所以我们可以把这行代码改为：

```
currentValue = (int)lroundf(slider.value);
```

这里用了所谓的类型强制转换，虽然lroundf(slider.value)返回了一个long int类型的整数，但因为我们知道这个数值的范围肯定不会超出int类型的整数范围，所以在前面加个(int)把它强制转换为int类型的整数。这样一来警告就消除了。

不过需要提醒的是，强制类型转换在编程中其实是一大忌讳，除非我们有百分百的把握，不要轻易使用这种方法。

什么是函数？函数和方法有什么区别？

我们已经知道了方法会提供功能，而函数则是为应用添加功能的另一种方式。Objective-C基于老而弥坚的C语言（三十多年过去了，至今仍在编程语言排行榜的Top3中）。而C语

言中使用函数把多行代码整合到一个连贯的整体中。在Objective-C中我们使用方法，而C语言中我们则使用函数。

在开发iOS应用的时候，我们很少会直接编写函数，大多数使用Objective-C开发的应用会使用对象，然后让对象调用方法。但函数和方法在本质上是很类似的。

它们唯一也是本质的区别是，函数是代码世界里独立的生命体，它不依赖于某个对象而生存。但方法不同，任何一个方法都是某个对象的附属生命（寄生体？），同时也只有它的宿主对象才能调用这个方法。

当然，它们的样子也很不一样：

// 下面的这个是方法:

```
[someObject methodName:parameter];
```

// 下面的这个是函数:

```
SomeFunction(parameter);
```

幸运的是，iOS的应用中同样支持C语言编写的代码，也同时支持C语言所提供的大量有用的内置函数库。Lroundf()这个函数就是其中的一个。后面我们还会用到其它的函数。顺便提一下，NSLog()也是一个函数。

区分函数和方法其实很简单，函数的名称后面带着括号，括号里面有一个或多个参数，而方法的调用则是用方括号，任何时候你都不会看错的。

现在让我们更改 showAlert 方法如下：

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

```
    NSString *message = [NSString stringWithFormat:@"滑动条的当前数值是: %d",currentValue];
```

```
    [[[UIAlertView alloc]initWithTitle:@"您好，苍老师" message:message delegate:nil cancelButtonTitle:@"一师是个好学校" otherButtonTitles:nil, nil]show];
```

```
}
```

和之前一样，我们创建并显示一个提示对话框，不过这一次的内容变了，我们使用 message 代替之前的文字内容，而 message 的内容则使用 NSString 来定义。

NSString是iPhone的字符串对象。在Objective-C的世界里，对象无处不在。字符串同样属于对象。

通过下面的这行代码，

```
NSString *message = [NSString stringWithFormat:@"滑动条的当前数值是：
%d",currentValue];
```

我们使用stringWithFormat方法创建了一个名为message的新字符串对象，该方法带有两个参数：一个带格式的字符串，还有一个用来替代占位符的数值。

后面的部分我们曾经看到过。在NSLog()函数里面用的是类似的字符串，只不过用%d替代了%f。%d和%f的区别是，%d代表一个整数，而%f则代表小数（用术语党的行话又叫“浮点数”，至于为神马这么叫，你只需要知道和计算机的CPU硬件有关，就不多做解释了，只当是学外语的习惯用法了）。因为currentValue这个变量里面保存的是整数，所以需要使用%d。

顺便提一下，程序猿术语党的口中经常会冒出很多稀奇古怪的术语，让我等吓得屁滚尿流奉若神明。其实完全不用恐慌，想想我们是如何学汉字和英语的。你学某个汉字会追溯到繁体字，甲骨文，甚至用说文解字来透彻分析每个字的起源吗？不会，即便是莫言莫大师恐怕也没这份功力，但是影响他写出拿诺奖的作品吗？不影响。所以，每当看到或听到这些术语，最多到wikipedia或百度百科里面看看大概的意思就好了，不必深究。语言只是一个工具，计算机语言也是如此，除非你要发明自己的语言，否则最重要的事情还是用它来做出一款好的产品，其它都是浮云。大道至简，凡是喜欢用术语忽悠人的，大可以鄙夷之。若干年后，这些热门术语自会烟消云散。

假定currentValue是34，这就意味着滑动条大概在靠左边1/3的位置。那么我们上面的代码就会把字符串转化为：“滑动条的当前数值是：34”，然后把这个字符串放到名为message的NSString对象中。NSLog()之前做了类似的事情，只不过它是直接把结果打印在调试面板中。这里我们则使用提示对话框，因为玩家看不到调试面板的内容，但可以看到提示对话框。

点击Run运行应用，拖动滑动条，然后触碰按钮。现在就会弹出一个提示对话框告诉你滑动条的当前数值。



不错，又前进了一步。虽然我们前进每一步都很慢，但每一步都很坚实。这里我们使用一个变量来保存滑动条当前位置四舍五入后的数值，这样应用中的其它地方就可以用到变量里面的值。比如这里在提示对话框中就用到了。如果我们不移动滑动条就触碰按钮，那么对话框中的数值也会保持不变。变量会始终记住保存在它里面的数值，直到我们放入新的数值。

本地变量 vs 实例变量

让我们回过头来再看看showAlert这个动作的内容：

```
- (IBAction)showAlert:(id)sender {  
  
    NSString *message = [NSString stringWithFormat:@"滑动条的当前数值是:  
%d",currentValue];  
  
    [[[UIAlertView alloc] initWithTitle:@"您好, 苍老师" message:message delegate:nil  
cancelButtonTitle:@"一师是个好学校" otherButtonTitles:nil, nil]show];  
}
```

```
}
```

这里面的message也是一个变量，它的作用是保存新创建的字符串。它的数据类型是NSString。Oops,这里有一个*星号，又是干吗用的呢？*星号是必不可少的，因为NSString是一个对象，而在Objective-C中，创建对象的时候必须在其名称前面加上一个星号。

话说回来，alertView这个提示对话框也是一个变量，不过它保存的是UIAlertView对象。

message和alertView这两个变量属于同一类型，而currentValue则是另外一种类型。这是因为message和alertView变量作为代码世界的生命体，其生命周期非常短暂。它们被称为本地变量，因为它们的声明开启于showAlert动作开始执行的时候，而当showAlert这个动作完成后，它们的生命就走到了终点。一旦showAlert方法执行完毕，CrazyDragViewController就会负责销毁message和alertView这两个变量，同时清理掉它们所占用的存储空间。

currentValue变量则不同，它会活的久一点。只要CrazyDragViewController还活着（玩家没有退出应用），它就会一直活着。这个类型的变量又被称为实例变量（术语党给了一个可恶的缩写是ivar），因为它的生命周期和它所属的对象实例的生命周期是相同的。如果我们要让某个数值从一个事件传递到另一个事件，就必须使用实例变量。

好吧？糊涂了吗？别太担心这一点，在教程的过程中我们还将重复一些重要的概念。很快对变量的理解就会变成你的本能反应。

捉“虫子”时间

在代码的世界里面，bug（虫子）很可恶，它的作用是让玩家所看到和感受到的东西和你所设计的东西南辕北辙。大多数程序猿终其一生（好吧，是在他们的职业生命周期里）都在和虫子打交道。无虫子，不编程。

虽然这个游戏还没多少东西，里面已经出现了bug。现在让我们看看这个bug的表现：点击Xcode上的Stop完全关闭应用。然后点击Run重新运行，先不要拖动滑动条，直接按按钮。

这时对话框提示，“滑动条的当前数值是：0”。但显然滑动条的拖动手柄在中间，这个数值应该是50才对啊魂淡！可恶的虫子！



小练习：想想看为神马这个数值会是0.

答案：线索在这里，只有当我们不拖动滑动条就按按钮的时候才会出现这个bug.显然，当我们还没有拖动滑动条的时候，`sliderMoved`这个消息根本没机会发送，我们也不可能读取滑动条的数值，把它保存到`currentValue`变量中。而在Objective-C里面，实例变量的默认值是0，所以我们就看到了这个bug。

为了消灭这个虫子，我们需要更改一个方法。在Xcode中点击 `CrazyDragViewController.m`，然后找到`viewDidLoad`这个方法，现在里面的代码如下：

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

当我们使用标准模板创建这个项目的时候，Xcode自动加入了viewDidLoad这个方法。现在我们需要加点代码进去。

更改viewDidLoad方法如下：

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    currentValue = 50;
    // Do any additional setup after loading the view, typically from a nib.
}
```

当我们打开应用的时候，一旦视图控制器从nib文件加载它的用户界面，就会通过UIKit发送viewDidLoad消息。这时视图控制器仍然不是可见的，所以此时设定高一些初始数值非常合适。在这里，我们将currentValue的初始值设置为50，也就是滑动条的默认初始位置的数值。

点击Run运行应用，然后看看虫子被消灭没有。



科普-什么是注释

你可能已经多次看到了用//开始的绿线了吧？这些被称为注释。你可以在这个符号后面写任何文字（包括吐槽，情诗，政治宣言，入党申请书，笑话，YY小说，财务计划。。。）反正编译器看到//就会自动无视后面的东西。

// 我是注释行。窗外闪电雷鸣，我虎躯一震，发现哥竟然穿越了，有木有！

除了//之外，我们还可以使用/*和*/。区别在于//只能注释一行文字，而/*和*/则可以注释它们中间的多行文字。

通常使用/*和*/来注释掉一整块代码。

不过在实际的使用中，哥推荐//。在Xcode中，把光标放在要注释的那行代码上，然后按快捷键Command 和/，就可以了。或者选中一整块代码，用这个快捷键，也是一样。

注释行的最佳用处是解释你的代码的作用。虽然说好的源代码自己可以说明自己，但打点广告，添加点说明和解释是很有必要的。向谁解释呢？很多时候向你自己，不过是一个月后的自己，半年后的自己，甚至是几年后的自己。除非你的记忆如同计算机一样的精确，而且还不死机，那么注释就很有必要。

你可能看到了，Xcode在你的项目文件中自动为你加了这样的一段注释：

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

这个其实没什么用，看看就好，不爽了可以删掉。

好了，今天的学习到此结束，老习惯送福利1P，苍老师的教师节感言。

