让不懂编程的人爱上iPhone开发(2013秋iOS7版)-第9篇

欢迎回来，让我们继续iPhone开发之旅。
在上一篇的内容里面我们首次接触了算法，我也提到过还有其它方法来计算滑动条当前数值和目标数值之间的差值。之前所使用的算法虽然可以用，但它包含了八行代码。其实我们可以用更简单的方法来实现。

这里是我构想的新算法：
“首先用滑动条的当前数值减去目标数值。
如果这个结果小于0，就用它乘以-1。”
就这么简单，我们成功的把负数转换成正数，确保最后的差值总是正数或0.

小练习：
现在你可以尝试一下把上面的算法用代码来实现。

最后的结果是这样的：

```
int difference = currentValue - targetValue;
if (difference < 0) {
   difference = difference * -1;
}
```

新的算法用代码体现出来就是这样的。首先我们用currentValue变量的数值减去targetValue变量的数值，然后把结果保存在difference变量中。结果使用if判断语句来判断它的值是否为负（小于0）。如果是，就让它乘以-1，然后保存这个新的结果到difference变量中。
当你用下面这行代码时：

```
difference = difference * -1;
```

程序首先用difference的数值乘以-1，然后再把计算的结果重新保存到difference中去。实际的效果是，新计算的数值覆盖了difference之前的内容。

当然我们还有一种更简单的方法：

```
difference *= -1;
```

*=操作符把*和=放到一个单独的操作里。最后的结果是相同的：变量中保存的旧数值被计算结果所替代。

科普：相同的符号，不同的意义

在上面的代码中，*的意思是乘以，不过只有在数学运算的时候*才代表这个意思。其实之前我们也看到过*这个符号，比如UISlider *slider，这里的*号告诉我们正在和对象打交道，而不是乘法操作符。在代码世界里，有些符号具备多种不同的含义，具体则根据上下文而定。

同样，我们还可以用下面的代码来实现这个算法：

```
int difference = currentValue - targetValue;
if (difference < 0) {
   difference = -difference;
}
```

这里我们不是用-1乘以difference变量的数值，而是直接用取负操作符。这是因为负数的负数必然就是正数了。如果你不相信我，可以回小学请教下你的老师~

好了，让我们试试新的算法吧。在Xcode中点击CrazyDragViewController.m，找到showAlert方法，用下面的代码替代：

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

   int difference = currentValue - targetValue;

   if(difference<0){
      difference = difference*-1;
   }

   NSString *message = [NSString stringWithFormat:@"滑动条的当前数值是:%d，我们的目标数值是：%d，之间的差距是：%d",currentValue,targetValue,difference];

   [[[UIAlertView alloc]initWithTitle:@"您好，苍老师" message:message delegate:nil cancelButtonTitle:@"本老师知道了" otherButtonTitles:nil, nil]show];
   [self startNewRound];
   [self updateLabels];
}
```

现在来点击Run运行游戏，你会发现结果和之前没有任何变化，只不过我们换了一种实现方式而已。

除了上面的方法，我们还有第三种方式，也就是用函数来处理这个问题。之前我们已经接

触了两个函数，分别是用来取整的lroundf(),以及用来生成随机数的arc4random()。为了确保一个数字永远是正数，我们可以用abs()这个函数，它的作用是获取当前数字的绝对值。
只要你成功从小学毕业了，应该知道绝对值是什么意思吧，哥就不用介绍了吧。
使用这个函数，可以让我们的算法实现代码进一步缩减为一行代码：

```
int difference = abs(targetValue - currentValue);
```

这个时候showAlert方法的代码就会更加精悍：

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

    int difference = abs(currentValue - targetValue);

    NSString *message = [NSString stringWithFormat:@"滑动条的当前数值是:%d，我们的目标数值是：%d，之间的差距是：%d",currentValue,targetValue,difference];

    [[[UIAlertView alloc]initWithTitle:@"您好，苍老师" message:message delegate:nil cancelButtonTitle:@"本老师知道了" otherButtonTitles:nil, nil]show];
    [self startNewRound];
    [self updateLabels];
}
```

abs这个函数很有用，在实际的编码中也会经常碰到。

好了，既然我们已经有了差值，就可以根据它来计算玩家的得分。

再次修改showAlert方法的代码如下：

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

    int difference = abs(currentValue - targetValue);

    int points = 100-difference;

    NSString *message = [NSString stringWithFormat:@"恭喜高富帅，您的得分是：%d",points];

    [[[UIAlertView alloc]initWithTitle:@"您好，土豪" message:message delegate:nil cancelButtonTitle:@"朕已知晓，爱卿辛苦了" otherButtonTitles:nil, nil]show];
```

```
    [self startNewRound];
    [self updateLabels];
}
```

玩家的最高得分是100，也就说你猜的数值和目标数值完全相同。最少得分是1，除非你点背的不行或是严重眼花（目标数值是100，结果你直接拖到最左边，或者目标数值是1，你直接拖到最右边）。

继续前进-保存玩家的总得分

在to-do 清单中，我们希望把玩家的总得分显示在屏幕上。在每一轮游戏结束后，我们都会把当前回合的得分添加到总得分中，然后更新得分标签。因为我们需要长期保存总的得分，所以需要把它保存在实例变量中。

让我们回到Xcode，在CrazyDragViewController.m中，在@interface部分添加变量声明：

```
@interface CrazyDragViewController (){

    int currentValue;
    int targetValue;
    int score;

}
```

现在我们可以来修正showAlert方法，从而更新score得分标签。

更改showAlert方法的代码如下：

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

    int difference = abs(currentValue - targetValue);

    int points = 100-difference;

    score += points;

    NSString *message = [NSString stringWithFormat:@"恭喜高富帅，您的得分是：
%d",points];

    [[[UIAlertView alloc]initWithTitle:@"您好，土豪" message:message delegate:nil
```

```
cancelButtonTitle:@"朕已知晓，爱卿辛苦了" otherButtonTitles:nil, nil]show];
    [self startNewRound];
    [self updateLabels];
}
```

上面的代码没有什么好奇怪的，我们只是添加了一行代码：

```
  score += points;
```

它的作用是把玩家在本轮游戏的得分添加到总得分里面去。其实这行代码完全可以这样来写：

```
score = score + points;
```

作用完全相同，不过我更喜欢精简一点的写法。

好吧，这里我要再次说明下本地变量和实例变量(ivar)之间的区别。现在你应该已经很清楚，本地变量只会在定义它的方法中生存，而实例变量则会与拥有它的视图控制器同寿。

在showAlert方法中，我们用到了四个本地变量和三个实例变量：

```
int difference = abs(targetValue - currentValue);
int points = 100 - difference;
score += points;
NSString *message = . . . ;
UIAlertView *alertView = . . . ;
```

小练习：猜猜看其中哪些是本地变量，哪些是实例变量？

本地变量很容易识别，因为当我们在某个方法中首次使用它时必须预先用某个数据类型来定义，比如int ,NSString和UIAlertView这几个就属于本地变量。

```
int difference = . . . ;
int points = . . . ;
NSString *message = . . . ;
UIAlertView *alertView = . . . ;
```

上面的代码形式就是对新变量的定义。因为变量在方法内创建，所以只能在方法内使用。一旦方法执行完毕，本地变量的生命也就走到了终点。不过下次再调用这个方法时，又会创建新的本地变量。

实例变量则不同，是在文件的顶部非任何方法之内定义的：

```
@interface CrazyDragViewController (){
   int currentValue;
   int targetValue;
   int score;
}

. . .

@end
```

因此我们可以在任何一个方法里面使用实例变量，而无需再次声明。

如果我们把刚才的代码修改为：

```
- (IBAction)showAlert:(id)sender
{
   int difference = abs(targetValue - currentValue);
   int points = 100 - difference;
   int score = score + points;

   . . .
}
```

那么你又会重新看到黄色的警告。因为当我们在某个变量前面添加数据类型时，其实就是创建了一个新的本地变量，而同名的本地变量会覆盖实例变量。这个时候，我们并没有把分数添加给实例变量score，而是添加给一个同名的本地变量。显然这是我们应该避免的。

说说实例变量名称的下划线

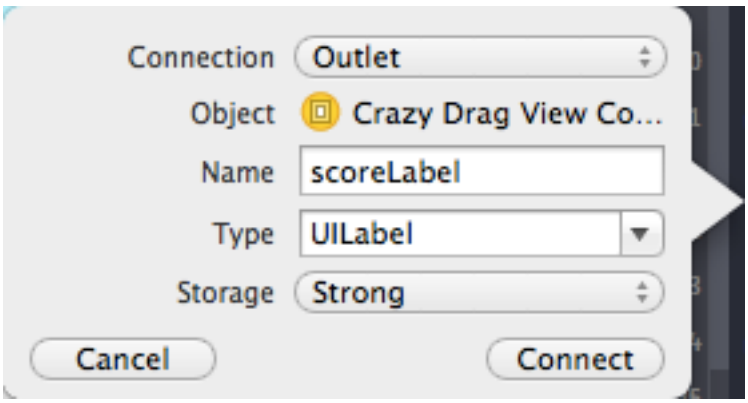
如果你之前看过其它的教程，可能会看到在实例变量（ivar)的名称前面有一个下划线。比如_score,_currentValue之类的。有些程序猿喜欢这样的命名方式，因为一眼就知道一个变量是否是实例变量（有下划线），还是本地变量（没有下划线）。
不过下划线在Objective-C语言里面并没有任何特殊的含义。为了区分实例变量和本地变量，有人用m或f作为前缀。我个人的喜好是不加下划线，原因很简单，敲入下划线要多花1秒钟时间，而且让代码显得很难看。作为帮主的粉丝，哥一向秉持美的东西都是由内到外的。不过如果你愿意加个前缀神马的也没关系，只要自己和别人都能看懂。

好了，继续前进。
现在该是在屏幕上显示玩家得分的时候了。

我们要做的事情其实很简单，和之前的目标数值标签一样，我们需要把得分标签关联到一个属性上，然后在标签的文本中显示得分。

小练习：试着看自己来操作一下。
第一步是在storyboard界面里面创建关联，第二步是在.m里面添加@synthesize语句来合成属性。

现在让我们一步步来，首先在Xcode中打开storyboard界面文件，然后按住ctrl键，从代表分数的标签拖出一条线到Assistant Editor的@interface和@end之间，最好在之前的@property变量代码附近。在弹出的提示框中输入以下信息，将Name改为scoreLabel，其它不变。点击Connect创建关联。



最后让我们进入CrazyDragViewController.m文件，在其它的@synthesize语句之后添加下面的语句：

@synthesize scoreLabel;

现在我们已经创建好了scoreLabel这个属性，并且让它和界面文件中的得分标签关联在一起。接下来让我们把得分放到标签的文本中。
那么这部分代码应该放在哪里呢？显然是updateLabels方法。

在CrazyDragViewController.m文件中找到updateLabels方法，更改它的代码如下：

```
-(void)updateLabels{

    self.targetLabel.text = [NSString stringWithFormat:@"%d",targetValue];
    self.scoreLabel.text = [NSString stringWithFormat:@"%d",score];
```

}

上面的代码也没有新东西。我们把int类型的score变量转换成一个字符串，然后把它赋予标签的text属性。实际执行的时候，标签会使用新的得分更新自己的文本内容。

点击Run运行游戏，然后看看总得分是不是对的。

接下来我们还得在每次开启新一轮游戏的时候更新回合数。

小练习：试着自己来尝试一下，看看之前学的东西管不管用。

聪明如你应该已经想到我们还需要定义一个新的实例变量。让我们在.m文件中的@interface后面的花括号里面添加一行代码：

```
@interface CrazyDragViewController (){

    int currentValue;
    int targetValue;
    int score;
    int round;

}
```

然后切换到Main.storyboard文件，创建一个关联，最后在.m文件里面使用@synthesize语句来合成。自己来动手完成这两步吧。
我想这两步不再需要哥手把手教你了吧，如果需要，说明哥之前教的还不够好啊。

提醒：别忘了三部曲的第二部

很多新手都会忘了在Interface Builder里面创建关联，这是非常常见的错误。很多时候我创建了一个按钮属性，然后也写了当玩家触碰按钮时应该如何处理的代码。结果当游戏跑起来的时候，发现按钮没用。通常这要浪费哥几分钟时间来冥思苦想，到底哪里出了错。实在是浪费生命啊。
记住，三部曲一步都不能少。

然后让我们更改updateLabels方法的代码如下：

```
-(void)updateLabels{

    self.targetLabel.text = [NSString stringWithFormat:@"%d",targetValue];
```

```
    self.scoreLabel.text = [NSString stringWithFormat:@"%d",score];
    self.roundLabel.text = [NSString stringWithFormat:@"%d",round];
}
```

好吧，还有一件事别忘了，我们需要让round变量的数值加1。在哪里加呢？我认为startNewRound方法是个不错的地方。毕竟每次开启新一轮游戏的时候我们都会调用这个方法。
在.m中找到startNewRound方法，更改其中的代码如下：

```
- (void)startNewRound
{
    round += 1;

    targetValue = 1 + (arc4random() % 100);

    currentValue = 50;
    self.slider.value = currentValue;
}
```

有一件事哥忘了告诉你。实例变量的默认数值是0。也就是说，当应用开始跑起来的时候，回合数变量的初始值是0.而当我们调用startNewRound方法的时候，都会让回合数变量的值加1，所以第一回合就会显示1。

点击Run按钮来运行游戏，现在就会看到游戏回合数每次都会自动增加。

到了这一步，基本的游戏逻辑都已经实现了。我们可以先小小的庆祝一下顺利，休息一下。

老规矩，上图！
这次献给大家的是AKB48，宅男最爱的秋叶原超级组合。

送给mm的：


blog.sina.com.cn/qcyy386837368