让不懂编程的人爱上iPhone开发(2013球iOS7版)-第10篇

还是让我们继续iPhone开发的学习之旅吧。
其实在第9篇的课程结束后，我们的游戏机制已经基本上实现，游戏逻辑也没有任何大的漏洞。至少到目前为止，我们没有制造出新的bug。不过这款产品仍然存在着巨大的改进控件。首先，游戏界面太丑了，我们需要尽快来优化。其次，还有一些小细节需要我们来进行调整。
首先，提示玩家游戏表现的对话框标题可以改进一下。我们可以让它的内容和玩家的游戏表现紧密相连。
如果玩家把滑动条正好放在目标位置上，那么提示对话框会说，“完美表现！“，如果滑动条的位置非常接近目标，但还没有百分百对准，那么对话框会说，”太棒了！差一点就到了！“如果滑动条的位置偏的太远，那么对话框会说，”远的没边了！“通过这样的方式，玩家不再只看到冷冰冰的游戏结果数字，而是被这种情感化人性化的反馈所吸引。

小联系：想想怎么来实现上面的想法？我们应该把相关的逻辑放在哪里？以及应该如何编码呢？提示：显然我们会用到一大堆if。
最适合放上述逻辑的地方就是showAlert，因为我们就是在那里创建了UIAlertView。之前我们已经可以动态显示消息中的文字，现在则需要动态修改标题文字。

回到Xcode,点击CrazyDragViewController.m，更改showAlert方法如下：

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

    int difference = abs(currentValue - targetValue);

    int points = 100-difference;

    score += points;

    NSString *title;

    if(difference ==0){
        title = @"土豪你太NB了！";
    }else if(difference <5){
        title = @"土豪太棒了，差一点！";
    }else if(difference <10){
        title = @"好吧，勉强算个土豪";
```

```
    }else{
        title = @"不是土豪少来装！";
    }
    NSString *message = [NSString stringWithFormat:@"恭喜高富帅，您的得分是：
%d",points];
    [[[UIAlertView alloc]initWithTitle:title message:message delegate:nil
cancelButtonTitle:@"朕已知晓，爱卿辛苦了" otherButtonTitles:nil, nil]show];
    [self startNewRound];
    [self updateLabels];
}
```

在上面的代码中，我们创建了一个新的名为title的本地字符串类型变量，用它来保存提示对话框中要显示的文字标题。

为了决定我们该使用哪个标题文字，需要查看滑动条结点所在位置和目标数值之间的差异。如果差值为0，我们将让title的文字内容变成”完美表现！“ 如果差值小于5，我们会使用文字”太棒了，差一点就到了！“ 。如果差值小于10，就会使用文字”不错！“ 不过当差值大于等于10的时候，玩家就需要一点提醒了，”差太远了吧！“

上面的逻辑还能看得懂吗？我们用了一大堆的if条件语句，其中考虑到各种不同的可能性，并选择对应的字符串。当我们创建UIAlertView对象的时候，会把title变量的内容给它，而不是直接提供一个固定的文本内容。

现在点击Run看看效果。

小练习：

游戏中我们常常会使用一些奖励机制。比如在这里，当玩家有完美表现时，我们可以奖励100分。这样玩家就有动力继续去达到完美。否则，98分，95分和100分就没有任何区别了。通过这个小小的奖励，玩家就有了去体验Hard模式的动力。一个完美表现的得分不再仅仅是100分，而是200分。当然，我们还可以给99分的玩家奖励50分。

好了，回到Xcode，修改showAlert方法的代码如下：

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

    int difference = abs(currentValue - targetValue);

    int points = 100-difference;

    score += points;

    NSString *title;

    if(difference ==0){
        title = @"土豪你太NB了！";
        points +=100;

    }else if(difference <5){

        if(difference ==1){
            points +=50;
        }
        title = @"土豪太棒了，差一点！";
    }else if(difference <10){
        title = @"好吧，勉强算个土豪";
    }else{
        title = @"不是土豪少来装！";
    }

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

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

```
}
```

你应该注意到了一个新的不同。在第一个if条件语句中，我们添加了一行新的语句。

```
 if(difference ==0){
     title = @"土豪你太NB了！";
     points +=100;

  }
```

当差值为0的时候，我们不仅让玩家得到”完美表现！“的反馈，还额外奖励100分。
第二个if条件语句的内容也发生了变化：

```
else if(difference <5){

     if(difference ==1){
        points +=50;
     }
     title = @"土豪太棒了，差一点！";
  }
```

这里我们学到了一个新东西，也就是在if条件语句中还可以内嵌if条件语句。如果差值大于0且小于5，就有可能是1（当然也可能不是1）。此时我们会检查差值是否为1，如果是，就额外奖励玩家50分。

最后，我把score = score +points;这行代码移到了if条件语句结束之后。这一点是必须的，因为在if条件语句执行的过程中我们很可能会更新points变量的数值。

点击Run运行一下应用，看看效果吧！

继续**前**进，**等待提示**对话**框消失！**

如果你仔细观察的话，可能会发现一个小小的不爽之处。每次触碰按钮的时候，提示对话框都会显示出了，而滑动条就会迅速回到中央位置，游戏回合数会增加，而目标数值标签已经获取了新的随机数。
也就是说，当我们还在查看上次游戏的结果时，实际上程序已经开启了新一轮的游戏。有时候哥真是困惑不解。

如果我们可以调整一下游戏逻辑机制，也就是当玩家关闭了提示对话框之后才开启新一轮的游戏，岂不是更合理？直到玩家关闭了提示对话框，当前的游戏回合才可以真正结束。

或许你要说，我们不正是这样做的吗？比如在showAlert方法里，我们在提示对话框显示完毕之后才会调用startNewRound方法。

好吧，这里哥需要稍微解释一下原因了。和其它的应用平台不同，在iOS中，提示对话框在显示的时候并没有让当前方法的其它代码停止运行。事实上，　[[[UIAlertView alloc]initWithTitle:title message:message delegate:nil cancelButtonTitle:@"朕已知晓，爱卿辛苦了" otherButtonTitles:nil, nil]show];这行代码只是让对话框显示在屏幕上，然后就迅速返回，此时showAlert方法的其它代码会继续顺利执行下去，而不会等待提示对话框的关闭。

用程序猿的术语，在iOS里面提示对话框的运行是异步的。关于同步和异步我们后面会详细解释，这里你需要明白。在同步运行的情况下，其它的代码必须等待当前的代码完成后才能继续，而在异步运行的情况下，其它的代码无需等待当前的代码完成就可以继续执行。比如某些基于网络的应用需要在后台通过网络上传或发送信息（邮件，照片），考虑到网络传输速度的问题，我们不可能等待这个过程结束后才能进行其它的界面交互，否则用户早就疯掉了。

在这里，事实上我们无法知道提示对话框何时结束，唯一可以确定的是当showAlert方法结束的时候它肯定会结束。

好吧，这样一来我们怎么可能知道alertView什么时候会消失呢，我们怎么来等它关闭呢？答案很简单：使用事件！正如我们已经了解到的，在iOS的开发中，很多代码都需要等待特定的事件发生（按钮被触碰，滑动条被移动，等等）。这里同样如此。

我们可以让alertView提示对话框在自己被关闭的时候发送一条消息。在此之前，我们什么也不做。当玩家触碰了提示对话框上的按钮之后，它就会将自己从屏幕上删除，并发送给我们一条消息。这就是我们的线索。

在《苦逼程序猿的生存指南》中，有一个名词用来描述这种现象，叫监听模式或者观察者模式。我们的视图控制器会监听来自提示对话框的事件。在iOS中，我们把这种监听者称之为delegates（代理），后面将会大量使用到它。在Stanford iOS开发公开课的第一课，对于MVC和观察者模式做了详细的阐述，强烈建议e文好的童鞋去听听。鉴于大家是初学者，这里不打算展开说明什么是观察者模式，感兴趣的可以去自己看看。http://baike.baidu.com/view/1854779.htm

在Xcode中切换到CrazyDragViewController.h，更改@interface这一行的代码如下：

```
@interface CrazyDragViewController : UIViewController<UIAlertViewDelegate>
```

在上面的代码中，通过尖括号<UIAlertViewDelegate>，我们告诉程序，现在视图控制器已经是UIAlertView的代理了。

然后切换到CrazyDragViewController.m，更改showAlert方法最后的部分如下：

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

    int difference = abs(currentValue - targetValue);

    int points = 100-difference;

    score += points;

    NSString *title;

    if(difference ==0){
        title = @"土豪你太NB了！";
        points +=100;

    }else if(difference <5){

        if(difference ==1){
            points +=50;
        }
        title = @"土豪太棒了，差一点！";
    }else if(difference <10){
        title = @"好吧，勉强算个土豪";
    }else{
        title = @"不是土豪少来装！";
    }

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

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

以上代码唯一的改动就是把显示提示信息这一行代码的delegate:nil改为delegate:self，它的意思是告诉提示对话框，当前的视图控制器是它的代理对象了。同时我们还删除了[self startNewRound]和[self updateLabels]方法，因为这两行代码我们希望在提示对话框结束之后再执行。如果还是放在这里，就没法得到我们期望的效果了。

在CrazyDragViewController.m中添加新的方法，在@end这行代码前添加：

```
-(void)alertView:(UIAlertView *)alertView didDismissWithButtonIndex:
(NSInteger)buttonIndex{

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

上面的方法是一个代理方法，当玩家关闭提示对话框时就会调用这个方法。当我们得到这条消息时，其实它在说，"你好，代理对象。因为玩家触碰了某个按钮，所以提示对话框被关闭了。"我们目前所用的提示对话框只用到了一个按钮，如果用到了多个按钮，那么我们需要根据buttonIndex参数来判断哪个按钮被触碰了。
当然，作为对关闭提示对话框的反应，我们启动了新的游戏回合。

点击Run，再次运行一下游戏，现在就好多了。

科普：使用代理对象的两部曲

在iOS 开发的过程中，对代理对象的使用可谓是无所不在。因此就如同创建属性对象需要三部曲一样，我们也应该熟悉使用代理对象的两部曲。

1.宣布自己具备成为代理对象的资格。通常我们通过在@interface这行代码的后面加上类似<UIAlertViewDelegate>的代码来声明。其中尖括号中为具体的代理协议名称。通过这种方式，我们告诉当前的视图控制器（或者其它对象）具备处理此类通知消息的能力。
2.让被代理的对象，比如我们这里的UIAlertView知道谁将成为它的代理。如果你忘了这一点，那么就永远无法得到自己想要的效果。

具体到我们这个例子，

```
   [[[UIAlertView alloc]initWithTitle:title message:message delegate:self
```

cancelButtonTitle:@"朕已知晓，爱卿辛苦了" otherButtonTitles:nil, nil]show];
如果你忘了将delegate:nil修改为delegate:self，那么当提示对话框被关闭后什么也不会发生。

Ok，今天的学习就到此结束了。还是送福利1张。

for 宅男：


琵琶语
信手低眉续弹，续续弹，
弹尽心中无限事。低眉续弹，
续续弹，弹尽心中无限事。

for mm: