iOS 5 Programming Cookbook

第七章
实现手势识别

版本 1.0

翻译时间：2012-06-25

DevDiv 翻译： kyelup cloudhsu 耐心摩卡
wangli2003j3 xiebaochun dymx101
DevDiv 校对： laigb kyelup
DevDiv 编辑： BeyondVincent

写在前面

目前，移动开发被广大的开发者们看好，并大量的加入移动领域的开发。

鉴于以下原因：

- 国内的相关中文资料缺乏
- 许多开发者对 E 文很是感冒
- 电子版的文档利于技术传播和交流

DevDiv.com 移动开发论坛特此成立了翻译组，翻译组成员具有丰富的移动开发经验和英语翻译水平。组员们利用业余时间，把一些好的相关英文资料翻译成中文，为广大移动开发者尽一点绵薄之力，希望能对读者有些许作用，在此也感谢组员们的辛勤付出。

关于 DevDiv

DevDiv 已成长为国内最具人气的综合性移动开发社区

更多相关信息请访问 DevDiv 移动开发论坛。

技术支持

首先 DevDiv 翻译组对您能够阅读本文以及关注 DevDiv 表示由衷的感谢。

在您学习和开发过程中，或多或少会遇到一些问题。DevDiv 论坛集结了一流的移动专家，我们很乐意与您一起探讨移动开发。如果您有什么问题和技术需要支持的话，请访问 DevDiv 移动开发论坛 或者发送邮件到 BeyondVincent@DevDiv.com，我们将尽力所能及的帮助你。

关于本文的翻译

感谢 kyelup、cloudhsu、耐心摩卡、wangli2003j3、xiebaochun 和 dymx101 对本文的翻译，同时非常感谢 laigb 和 kyelup 在百忙中抽出时间对翻译初稿的认真校验，指出了文章中的错误。才使本文与读者尽快见面。由于书稿内容多，我们的知识有限，尽管我们进行了细心的检查，但是还是会存在错误，这里恳请广大读者批评指正，并发送邮件至 BeyondVincent@devdiv.com，在此我们表示衷心的感谢。

DevDiv翻译

读者查看下面的帖子可以持续关注章节翻译更新情况

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译各章节汇总

DevDiv翻译

目录

DevDiv翻译

DevDiv翻译

前言

参考帖子

[DevDiv翻译]iOS 5 Programming Cookbook翻译_前言

DevDiv翻译

第 1 章 基础入门

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第一章_基础入门

DevDiv翻译

第 2 章 使用控制器和视图

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译 第二章 使用控制器和视图(上)

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译 第二章 使用控制器和视图(下)

DevDiv翻译

第 3 章　构造和使用 Table View

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第三章_构造和使用 Table View

DevDiv翻译

第 4 章 Storyboards

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第四章 _Storyboards

DevDiv翻译

第 5 章　并发

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第五章_并发

DevDiv翻译

第 6 章 定位核心与地图

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第六章_核心定位与地图

DevDiv翻译

第 7 章 实现手势识别的功能

7.0 介绍

手势其实是一组触摸事件的组合，一个最明显的手势例子就是 iOS 的图片浏览功能，用户可以通过手势来对图片进行放大缩小通过两个手指的触摸。一些基础的手势触摸事件已经添加到 iOS SDK 中来了。这些相关的类可以用来检测点击，拖 拉，双击等手势事件。

手势事件识别必须要添加到一个 UIView 这个类里面去，一个单独存在的视图可以添加多个手势识别器。一旦这个界面捕获到了一些手势动作，这个视图将会把这个手势动作传递给其他的手势识别器。

一些触摸事件需要手机系统的支持，如下是 iOS SDK5 提供的 6 个手势识别器.

- Swipe
- Rotation
- Pinch
- Pan
- Long press
- Tap

最基础的框架为了能够处理手势的动作必须要按照如下步骤来进行操作。

1.创建一个合适的手势识别器的对象。

2.把这个手势识别器的对象绑定到一个视图上。

3.添加一些捕获手势事件发生的方法。

- 这个方法必须返回类型为空
- 这个方法要么是无参数类型的，要么只能接受一个 UIGestureRecognizer 类型的参数。

你可以参考如下代码.

```
- (void) tapRecognizer:(UITapGestureRecognizer *)paramSender{
   /* */
}
- (void) tapRecognizer{
   /* */
}
```

手势识别器一般可以分为两个大类，一个是单独的一个手势，一个是连贯的手势组合。单独的顾名思义，就是一个手势之后就会有一个监听捕获的事件，然后来做相应的操作。连贯的就是一组手势动作，然后在进行监听捕获事件进行相关的处理。

例如，双击事件，其实是一个单独的事件。虽然是有两个点击事件组成的，但是这个系统的手势识别器还是把它当做一个事件来处理的。双击事件发生之后会调用添加的双击事件的捕获方法。

一个很好的连续手势组的例子就是旋转，手势动作将会当用户开始旋转之后就开始了，当用户的手离开了屏幕之后就停止了。这种类型的手势监听方法将会多次的运行。

UIGestureRecognizer 类有一个 state 属性的变量，这个变量代表了不同状态的手势以便

DevDiv翻译

手势识别器能够很好的进行辨别，单独的手势和连贯的手势识别器都有一组不同的状态值。
单独的手势识别器可以传递如下状态值。

1. UIGestureRecognizerStatePossible
2. UIGestureRecognizerStateRecognized
3. UIGestureRecognizerStateFailed

一组连贯的手势组可以传递如下的状态值。

1. UIGestureRecognizerStatePossible
2. UIGestureRecognizerStateBegan
3. UIGestureRecognizerStateChanged
4. UIGestureRecognizerStateEnded
5. UIGestureRecognizerStateFailed

7.1. 捕获点击划屏的手势

7.1.1. 问题

你想让你的程序能够检测到用户点击的手势，然后可以弹出一个图片的展示窗体

7.1.2. 方案

创建一个 UISwipeGestureRecognizer 类型的对象，然后把它添加到你对应的视图中。
代码类似如下

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

    /* Instantiate our object */
    self.swipeGestureRecognizer = [[UISwipeGestureRecognizer alloc]
                                   initWithTarget:self
                                   action:@selector(handleSwipes:)];

    /* Swipes that are performed from right to
      left are to be detected */
    self.swipeGestureRecognizer.direction =
        UISwipeGestureRecognizerDirectionLeft;

    /* Just one finger needed */
    self.swipeGestureRecognizer.numberOfTouchesRequired = 1;

    /* Add it to the view */
    [self.view addGestureRecognizer:self.swipeGestureRecognizer];

}
- (void) viewDidUnload{
    [super viewDidUnload];
    self.swipeGestureRecognizer = nil;
}
```

如上代码创建了一个手势识别器，由于我们只是在一个视图中使用我们就采用了这种单

独添加的方式。我们把他添加到视图控制器的手势识别器中。如上代码中我们添加了一个处理事件的方法 handleSwipes 用来监听处理点击事件。

7.1.3. 讨论

点击事件是最简单的也是最经常使用的手势动作。我们经常也使用一个手指或者多个手指来进行类似翻页的功能，从一个视图到其他的视图上。我们可以使用 UISwipGestureRecognizer 这个识别器，这个识别器是从 UIGestureRecognizer 类衍生出来的。然后又添加了一些特有的方法。例如添加了一组属性允许我们来设置我们手势事件的方向。或者多多少个手指来进行了一个手势的动作，不过这些手势动作都是分开的，没联系的手势动作。

这个 handleSwipes 方法我们可以按照如下方法来实现。

```
- (void) handleSwipes:(UISwipeGestureRecognizer *)paramSender{

   if (paramSender.direction & UISwipeGestureRecognizerDirectionDown){
      NSLog(@"Swiped Down.");
   }
   if (paramSender.direction & UISwipeGestureRecognizerDirectionLeft){
      NSLog(@"Swiped Left.");
   }
   if (paramSender.direction & UISwipeGestureRecognizerDirectionRight){
      NSLog(@"Swiped Right.");
   }
   if (paramSender.direction & UISwipeGestureRecognizerDirectionUp){
      NSLog(@"Swiped Up.");
   }

}
```

你可以添加多个方位的手势识别动作，通过实现 UISwipeGestureRecognizer 这个类。我们可以使用或的判断来进行操作，Objective-C 中的或可以用“|”来表示。例如，如果你想捕获到从底部向左上屏幕的一个手势动作，你可以使用 UISwipeGestureRecognizerDirectionDown 这个值来进行操作。

虽然敲击的手势我们通常是使用一个手指的，但是并不代表我们不能使用多个手指，我们只需要根据我们的需要，然后在 UISwipeGestureRecognizer 这个类中 numberOfTouchesRequired 这个属性中进行申明。

7.2. 捕获旋转的手势

7.2.1. 问题

你想捕获用户利用手指在屏幕上做旋转的手势

7.2.2. 方案

创建一个 UIRotationGestureRecognizer 的监视器来监听这个旋转的手势，然后在绑定到

你的视图中。代码如下：

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

    self.view.backgroundColor = [UIColor whiteColor];

    self.helloWorldLabel = [[UILabel alloc] initWithFrame:CGRectZero];
    self.helloWorldLabel.text = @"Hello, World!";
    self.helloWorldLabel.font = [UIFont systemFontOfSize:16.0f];
    [self.helloWorldLabel sizeToFit];
    self.helloWorldLabel.center = self.view.center;
    [self.view addSubview:self.helloWorldLabel];

    self.rotationGestureRecognizer = [[UIRotationGestureRecognizer alloc]
                                      initWithTarget:self
                                      action:@selector(handleRotations:)];

    [self.view addGestureRecognizer:self.rotationGestureRecognizer];

}
- (void) viewDidUnload{
    [super viewDidUnload];
    self.helloWorldLabel = nil;
    self.rotationGestureRecognizer = nil;
}
```

7.2.3. 讨论

UIRotationGestureRecognizer 手势识别器，就像名称一样，这个类能用来监听和捕获旋转的手势，比如一种场景，当你的应用中有一个展示图片的视图，你可以添加一个旋转的手势识别器，这样用户就可以顺利的使用手势来调整图片的位置。

UIRotationGestureRecognizer 这个类有一个 rotation 的属性，这个属性可以用来设置旋转的方向和旋转的弧度。当用户的手势开始和结束的时候分别会用到如下的两个属性.UIGestureRecognizerStateBegan,UIGestureRecognizerStateEnded.

一些旋转的视图元素其实是从 UIView 这个类衍生出来的，你可以利用 CGAffineTransformMakeRotation 这个方法来进行一个旋转手势的事件传递.

代码如下:

```
#import <UIKit/UIKit.h>
@interface Detecting_Rotation_GesturesViewController : UIViewController
@property (nonatomic, strong)
    UIRotationGestureRecognizer *rotationGestureRecognizer;
@property (nonatomic, strong)
    UILabel *helloWorldLabel;
@property (nonatomic, unsafe_unretained)
    CGFloat rotationAngleInRadians;
@end
```

首先让我们来看一看这段代码中的几个属性，以及解释一下他们的作用

helloWorldLabel

这是我们为这个视图创建的一个标签，我们将添加一些代码以便我们能够在这个标签上使用手势来模拟旋转的一个效果。

rotationGestureRecognizer
这个不用多说，是我们的旋转手势的监听捕获实例对象。

rotationAngleInRadians
这个对象就是我们在旋转的时候需要为我们的标签对象设置的一个位置对象信息，当我们每一次旋转的时候我们都会把一个新的位置值保存在这个对象里面，达到一个旋转的效果。

其实你也不用考虑太多，虽然标签的位置并不重要，我们还是让标签在他的周围进行一个旋转的，无论我们的视图和标签在什么地方，唯一值得我们注意的就是，在不同的设备，由于屏幕尺寸（ipad，iphone）不一样，所以我们都是需要来自动进行计算，计算下一次旋转的值达到旋转的效果。

通过使用标签的 center 这个属性，然后设置一个中心位置值给我们的窗体界面，我们将会把我们的标签居中显示，然后我们在旋转这个标签的时候也是依据这个中心位置进行一个旋转。下面一起来看看相关的一些代码。

```
#import "Detecting_Rotation_GesturesViewController.h"
@implementation Detecting_Rotation_GesturesViewController
@synthesize rotationGestureRecognizer;
@synthesize helloWorldLabel;
@synthesize rotationAngleInRadians;
...
```

下面是需要修改的.m 文件

```
- (void) handleRotations:(UIRotationGestureRecognizer *)paramSender{

    if (self.helloWorldLabel == nil){
        return;
    }

    /* Take the previous rotation and add the current rotation to it */
    self.helloWorldLabel.transform = CGAffineTransformMakeRotation(self.rotationAngleInRadians +
                                                                   paramSender.rotation);

    /* At the end of the rotation, keep the angle for later use */
    if (paramSender.state == UIGestureRecognizerStateEnded){
        self.rotationAngleInRadians += paramSender.rotation;
    }

}
```

这个旋转手势识别器将会给我们传递一组旋转的角度，旋转手势识别器一直还在进行一个监听的动作，因此他会一边监听我们手势旋转的角度，一边把这些捕获到的角度传递给我们，然后我们可以利用这些角度信息，进行一些位置的计算，然后调整下一个显示的位置。

注意，如果你使用的是模拟器，并不是真机，你仍然可以来测试这个旋转的手势，你需要

使用 Option 这个键，然后你会看到在模拟器中间有两个圆环，你可以通过调整圆环的位置，来进行手势的旋转，在旋转的时候，你还必须使用到 Alt 键和 Shift 键，这样你才能够正常的旋转。

在添加完如上代码之后，我们可以根据旋转的手势角度给我们的这个标签设置一个旋转的角度，但是你可能会遇到如下问题，当一个旋转的动作结束了，另外的一个旋转动作又开始了，第二个旋转的手势角度将会把我们第一次的旋转手势角度给替换掉，因此在第一个旋转动作没有发生的时候，第二个旋转的角度就把第一个旋转的角度覆盖掉。所以我们需要无论这个旋转的手势动作什么时候结束，我们都必须要让当前的这次旋转动作进行下去，因此我们就需要添加一个旋转角度的队列，让我们的标签按照这个队列中的旋转角度依次的进行旋转运动。

我们在前文中提到一个 CGAffineTransformMakeRotation 的方法，这个方法可用来创建依序变化的队列。在 iOS SDK 中所有以"CG"开都的都需要引入,CoreGraphice 框架包，当我们把这个包添加到工程之后，然后在运行我们的项目。看看最新的效果。

7.3. 捕获类似拖拽的手势

7.3.1. 问题

你需要在你的应用中添加利用手指来达到图层拖拉的应用。

7.3.2. 方案

利用 UIPanGestureRecognizer 这个手势识别器，

请参考如下代码。

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

    self.view.backgroundColor = [UIColor whiteColor];

    /* Let's first create a label */
    CGRect labelFrame = CGRectMake(0.0f,       /* X */
                                   0.0f,       /* Y */
                                   150.0f,    /* Width */
                                   100.0f); /* Height */

    self.helloWorldLabel = [[UILabel alloc] initWithFrame:labelFrame];
    self.helloWorldLabel.text = @"Hello World";
    self.helloWorldLabel.backgroundColor = [UIColor blackColor];
    self.helloWorldLabel.textColor = [UIColor whiteColor];
    self.helloWorldLabel.textAlignment = UITextAlignmentCenter;

    /* Make sure to enable user interaction; otherwise, tap events
      won't be caught on this label */
    self.helloWorldLabel.userInteractionEnabled = YES;

    /* And now make sure this label gets displayed on our view */
    [self.view addSubview:self.helloWorldLabel];

    /* Create the Pan Gesture Recognizer */
    self.panGestureRecognizer = [[UIPanGestureRecognizer alloc]
                                 initWithTarget:self
```

```
                                    action:@selector(handlePanGestures:)];

    /* At least and at most we need only one finger to activate
      the pan gesture recognizer */
    self.panGestureRecognizer.minimumNumberOfTouches = 1;
    self.panGestureRecognizer.maximumNumberOfTouches = 1;

    /* Add it to our view */
    [self.helloWorldLabel addGestureRecognizer:self.panGestureRecognizer];

}
- (void) viewDidUnload{
    [super viewDidUnload];
    self.panGestureRecognizer = nil;
    self.helloWorldLabel = nil;
}
```

7.3.3. 讨论

UIPanGestureRecognizer 这个手势识别器类，就像他的名字一样，可以捕获监测拖动的手势。

拖动的手势一般都是用手指来操作产生的，一般当拖动手势产生的时候会顺序的经过如下几个阶段。

1. UIGestureRecognizerStateBegan
2. UIGestureRecognizerStateChanged
3. UIGestureRecognizerStateEnded

我们一般会实现手势识别时的处理方法，如下的代码，将会实现利用手指来移动屏幕中间的一个标签。

```
- (void) handlePanGestures:(UIPanGestureRecognizer*)paramSender{

    if (paramSender.state != UIGestureRecognizerStateEnded &&
          paramSender.state != UIGestureRecognizerStateFailed){
        CGPoint location = [paramSender locationInView:paramSender.view.superview];
        paramSender.view.center = location;
    }

}
```

为了能达到移动我们屏幕中的标签，我们需要获取手指在屏幕中的坐标，并不是标签的坐标。因此，我们需要调用 locationInView 这个方法，并把我们的标签作为参数传递进去。

通过使用 locationInView 这个方法，来获取到我们手势的坐标，为了能够捕获到多个手指的位置，我们就需要使用 locationOfTouch:inView: 这个方法。然后通过使用 UIPanGestureRecognizer 这个对象的 minimumNumberOfTouches 和 maximumNumberOfTouches 参数。你就可以在同一个时间来捕获多个手指的拖拽的动作了。

DevDiv翻译

在我们的实例中，为了能够更好的演示，我们就只添加了一个手指的拖拽动作监听。

当手势产生的时候，并且处于 UIGestureRecognizerStateEnded 这个状态的时候，反馈出来的坐标 X 和 Y 肯能并不是数字类型的，可能是 NAN 空。因此在这个状态的时候，我们一边建议不要采用这个状态反馈的坐标值。

7.4. 监听并捕获到长久按住屏幕的手势动作

7.4.1. 问题

你想监听并捕获到用户用手指长久按住屏幕的某一个地方的手势事件。

7.4.2. 方案

创建一个 UILongPressGestureRecognizer 的实例对象，并把这个对象绑定在你的视图控制器里面。代码如下

```
#import <UIKit/UIKit.h>
@interface Detecting_Long_Press_GesturesViewController : UIViewController
@property (nonatomic, strong)
   UILongPressGestureRecognizer *longPressGestureRecognizer;
@property (nonatomic, strong) UIButton *dummyButton;
@end
```

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

   self.view.backgroundColor = [UIColor whiteColor];

   self.dummyButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
   self.dummyButton.frame = CGRectMake(0.0f,
                                       0.0f,
                                       72.0f,
                                       37.0f);
   self.dummyButton.center = self.view.center;
   [self.view addSubview:self.dummyButton];

   /* First create the gesture recognizer */
   self.longPressGestureRecognizer =
   [[UILongPressGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(handleLongPressGestures:)];
       /* The number of fingers that must be present on the screen */
   self.longPressGestureRecognizer.numberOfTouchesRequired = 2;

   /* Maximum 100 pixels of movement allowed before the gesture
     is recognized */
   self.longPressGestureRecognizer.allowableMovement = 100.0f;
```

```
    /* The user must press 2 fingers (numberOfTouchesRequired) for
      at least 1 second for the gesture to be recognized */
    self.longPressGestureRecognizer.minimumPressDuration = 1.0;

    /* Add this gesture recognizer to our view */
    [self.view addGestureRecognizer:self.longPressGestureRecognizer];

}
- (void) viewDidUnload{
    [super viewDidUnload];
    self.longPressGestureRecognizer = nil;
    self.dummyButton = nil;
}
```

7.4.3. 讨论

iOS SDK 提供了一个类来监听长久连贯的轻轻点击的手势事件，这个类就是 UILongTapGestureRecognizer，当用户用手指一下连一下的点击视图的时候就会触发这个事件。因此，你就能够捕获到用户点击的次数，等相关的数据.

下面我们将会看一下这个类中的几个属性。

numberOfTapsRequired

这个属性就保存了用户点击的次数，当这个手势动作触发之前，并没有一个手势是在屏幕是上的，当第一次把手指点击在屏幕上，然后拿开，这个动作可以称为一次点击时间。

numberOfTouchesRequired

这个属性保存了有多少个手指点击了屏幕，因此你要确保你每次的点击手指数目是一样的，默认值是为 0.

minimumPressDuration

这个参数表示，两次点击之间间隔的时间长度。

在我们的实例中，我们是按照如下方式来设置的。

- numberOfTapsRequired: default (we are not changing this value)
- numberOfTouchesRequired: 2
- allowableMovement: 100
- minimumPressDuration: 1

当这些参数的值都设置好了之后，我们就需要按照这个设置来进行一些手势的点击动作，我们要间隔一秒钟点一次，连续点击 100 次。

当我们按照这个约束条件完成了动作之后，我们的 handleLongPressGestures 方法将会被调用。这个时间我们就能够在这个方法里面添加一些处理动作，代码如下。

```
- (void) handleLongPressGestures:(UILongPressGestureRecognizer *)paramSender{

    /* Here we want to find the mid point of the two fingers
      that caused the long press gesture to be recognized. We configured
      this number using the numberOfTouchesRequired property of the
      UILongPressGestureRecognizer that we instantiated in the
      viewDidLoad instance method of this View Controller. If we
      find that another long press gesture recognizer is using this
      method as its target, we will ignore it */

    if ([paramSender isEqual:self.longPressGestureRecognizer]){

        if (paramSender.numberOfTouchesRequired == 2){
```

DevDiv翻译

```
        CGPoint touchPoint1 =
        [paramSender locationOfTouch:0
                                inView:paramSender.view];

        CGPoint touchPoint2 =
        [paramSender locationOfTouch:1
                                inView:paramSender.view];

        CGFloat midPointX = (touchPoint1.x + touchPoint2.x) / 2.0f;
        CGFloat midPointY = (touchPoint1.y + touchPoint2.y) / 2.0f;

        CGPoint midPoint = CGPointMake(midPointX, midPointY);

        self.dummyButton.center = midPoint;

    } else {
      /* This is a long press gesture recognizer with more
        or less than 2 fingers */

    }
  }

}
```

一个比较典型的用手指来长时间的点击屏幕就是地图的这个应用，当你看不同地方的时候，通过你的手指按住屏幕，然后不要松手，长久按下去，那么不过一会就会有一个锚点标记在地图上。

7.5. 监听捕获轻击的手势动作

7.5.1. 问题

你想监听捕获用户点击了屏幕的事件。

7.5.2. 方案

创建一个 UITapGestureRecognizer 类的实例对象，然后把他绑定到我们的视图上面，代码如下。

```
#import <UIKit/UIKit.h>
@interface Detecting_Tap_GesturesViewController : UIViewController
@property (nonatomic, strong)
   UITapGestureRecognizer *tapGestureRecognizer;
@end
```

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

   self.view.backgroundColor = [UIColor whiteColor];

   /* Create the Tap Gesture Recognizer */
```

```
    self.tapGestureRecognizer = [[UITapGestureRecognizer alloc]
                                 initWithTarget:self
                                 action:@selector(handleTaps:)];

    /* The number of fingers that must be on the screen */
    self.tapGestureRecognizer.numberOfTouchesRequired = 2;

    /* The total number of taps to be performed before the
      gesture is recognized */
    self.tapGestureRecognizer.numberOfTapsRequired = 3;

    /* Add this gesture recognizer to our view */
    [self.view addGestureRecognizer:self.tapGestureRecognizer];

}
- (void) viewDidUnload{
    [super viewDidUnload];
    self.tapGestureRecognizer = nil;
}
```

7.5.3. 讨论

点击的手势是最常见的一个手势动作，当用户用手指接触屏幕，到离开屏幕，一个点击动作就算完成了。

UITapGestureRecognizer 类的 locationInView 这个方法一般用来获取点击的位置，如果是多个点击的手势动作，调用这个方法将会得到一组数据的点的位置，在我们上面的实例中，我们给 numberOfTouches 这个属性值设置为 2，通过设置这个值，就表示，我们在点击的时候是需要两个手指同时来完成动作的。同样 numberOfTapsRequired 这个值设置为 3，就表示，我们需要连续的点击三次。我们提供了一个 handleTaps 方法，来处理手势发生之后需要的处理动作，代码如下

```
- (void) handleTaps:(UITapGestureRecognizer*)paramSender{

    NSUInteger touchCounter = 0;
    for (touchCounter = 0;
         touchCounter < paramSender.numberOfTouchesRequired;
         touchCounter++){
        CGPoint touchPoint =
        [paramSender locationOfTouch:touchCounter
                              inView:paramSender.view];
        NSLog(@"Touch #%lu: %@",
              (unsigned long)touchCounter+1,
              NSStringFromCGPoint(touchPoint));
    }

}
```

在这段代码中，我们通过两个手指来点击屏幕的手势动作，获取每一次点击的位置，因此当我们运行这段代码，并且手势动作完成之后，我们会得到我们点击的屏幕位置。

```
Touch #1: {107, 186}
Touch #2: {213, 254}
```

如果你是在使用模拟器测试的，那么你可以通过按住 Option 键来模拟出两个手指出来，然后通过你的鼠标来进行一个点击的动作。

上端代码中有一个 NSStringFromCGPoint 的方法，这个方法其实就是把一个物理的位置信息转化成一个 NSString 类型的数据，这样我们就可以直接打印出来以便我们的查看~!

7.6. 放大和缩小的手势监听和处理

7.6.1. 问题

你想捕获并处理类似放大和缩小的手势动作。

7.6.2. 方案

创建一个 UIPinchGestureRecognizer 类的实例话对象，然后把这个对象添加到你的视图中去，代码如下

```
#import <UIKit/UIKit.h>
@interface Detecting_Pinch_GesturesViewController : UIViewController
@property (nonatomic, strong)
   UIPinchGestureRecognizer *pinchGestureRecognizer;
@property (nonatomic, strong) UILabel *myBlackLabel;
@property (nonatomic, unsafe_unretained) CGFloat currentScale;
@end
```

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

   self.view.backgroundColor = [UIColor whiteColor];

   CGRect labelRect = CGRectMake(0.0f,         /* X */
                                 0.0f,         /* Y */
                                 200.0f,       /* Width */
                                 200.0f);      /* Height */

   self.myBlackLabel = [[UILabel alloc] initWithFrame:labelRect];
   self.myBlackLabel.center = self.view.center;
   self.myBlackLabel.backgroundColor = [UIColor blackColor];

   /* Without this line, our pinch gesture recognizer
     will not work */
   self.myBlackLabel.userInteractionEnabled = YES;
   [self.view addSubview:self.myBlackLabel];

   /* Create the Pinch Gesture Recognizer */
   self.pinchGestureRecognizer =    [[UIPinchGestureRecognizer alloc]
                                     initWithTarget:self
                                     action:@selector(handlePinches:)];
      /* Add this gesture recognizer to our view */
   [self.myBlackLabel
```

```
    addGestureRecognizer:self.pinchGestureRecognizer];

}
- (void) viewDidUnload{
   [super viewDidUnload];
   self.myBlackLabel = nil;
   self.pinchGestureRecognizer = nil;
}
```

7.6.3. 讨论

通过聚拉的手势动作可以很轻松的来改变视图元素的一个比例，例如，我们在通过 Safari 浏览器打开网页的时候，有时间为了能够更清晰的看到其中的一些文字，那么我们就需要放大我们的屏幕了，有时候我们又需要缩放页面。

手势的动作状态有如下三种，一般是按照顺序来进行转换的。

1. UIGestureRecognizerStateBegan
2. UIGestureRecognizerStateChanged
3. UIGestureRecognizerStateEnded

一旦聚拉的手势动作产生了之后，我们就需要在捕获的事件中进行一个页面调整。其中有两个比较重要的变量 scale 和 velocity，前者是一个比例范围，后者是一个变化速率的，也就是说每次变化的一个像素点。

一般设置代码如下

```
- (void) handlePinches:(UIPinchGestureRecognizer*)paramSender{

   if (paramSender.state == UIGestureRecognizerStateEnded){
      self.currentScale = paramSender.scale;
   } else if (paramSender.state == UIGestureRecognizerStateBegan &&
                      self.currentScale != 0.0f){
      paramSender.scale = self.currentScale;
   }

   if (paramSender.scale != NAN &&
         paramSender.scale != 0.0){
      paramSender.view.transform =
      CGAffineTransformMakeScale(paramSender.scale,
                                          paramSender.scale);
   }

}
```

由于 scale 这个属性的值是每次都在变的，所以我们需要用另外一个变量来保存当前的一个 scale 的值，这个变量叫做 currentScale,,这样我们就行进行一个缩小，变大的视图效果了

点击这里访问：DevDiv.com 移动开发论坛