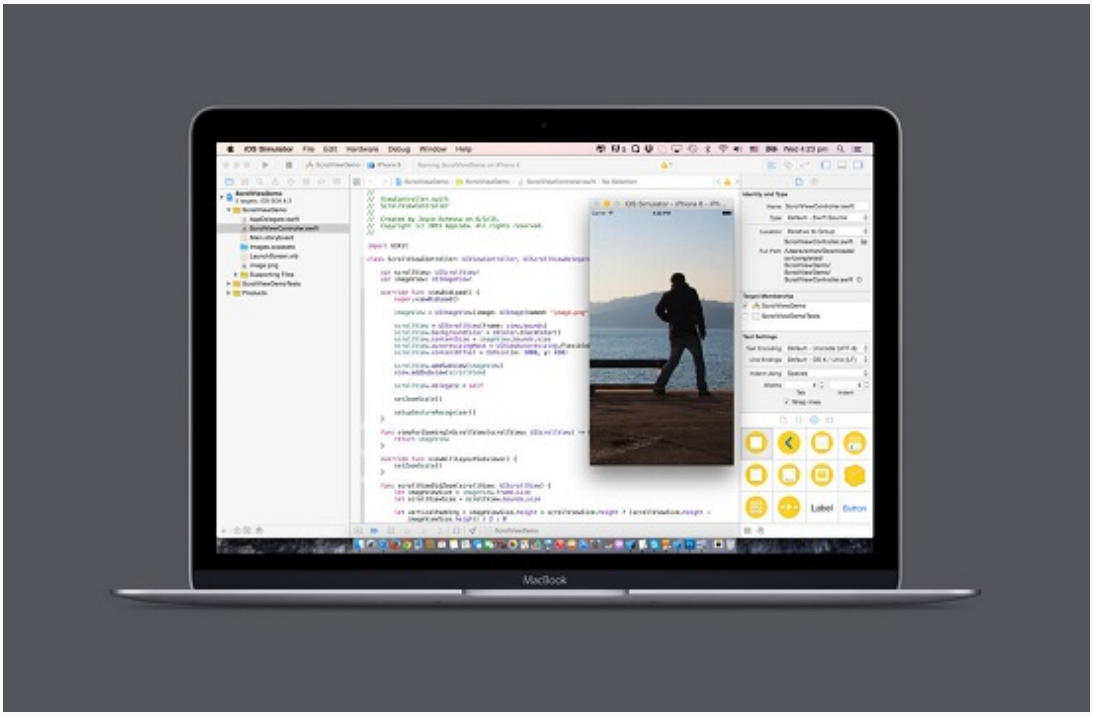
在iOS开发中，滚动视图（UIScrollView）通常用于显示内容尺寸大于屏幕尺寸的视图。滚动视图有以下两个主要作用：

- 让用户可以通过拖拽手势来观看想看到的内容
- 让用户可以通过捏合手势来放大或缩小观看的内容

在iOS应用中常见的表格视图（UITableView）就继承自滚动视图，并因此可以通过上下滚动来显示更多的内容。

在本篇教程中，我们将讨论滚动视图的诸多方面内容，主要包括：使用纯代码和可视化编程两种方式来创建一个滚动视图、实现滚动和缩放功能，以及如何嵌套使用滚动视图。



继续阅读之前，请先下载本文示例代码所需的资源文件，详见：资源文件地址。（译注：原文资源文件地址需要FQ访问，本人已转存到GitHub上，详见这里）

使用纯代码方式创建UIScrollView

UIScrollView同其他视图一样，可以通过纯代码和可视化编程两种方式来创建。在创建之后，只需要少量额外设置就可以让UIScrollView获得基本的滚动功能。

UIScrollView也和其他视图一样，应该被一个控制器管理或者添加到某个视图层级中。想要完成滚动功能还需要对UIScrollView进行以下两步设置：

- 必须设置UIScrollView的contentSize属性，它提供了UIScrollView的内容的大小，也就是可以滚动的区域的大小。
- 必须为UIScrollView添加一个或多个用于显示和滚动的子视图，这些视图提供了UIScrollView显示的内容。

你还可以根据应用的具体需求设置UIScrollView的一些显示效果，比如：是否显示水平和竖直方向的滚动条、滚动的

弹性效果、缩放的弹性效果，以及允许的滚动方向等。

接下来我们将在代码中创建一个UIScrollView。在下载的资源文件中打开ScrollViewDemo工程。它就是一个简单的Single View Application工程，只不过将storyboard中根控制器的类型绑定为自己新建的叫做ScrollViewController的控制器，还在项目中添加了一张我们要用到的图片，图片名称为image.png。

接下来打开ScrollViewController.swift文件，添加如下代码。

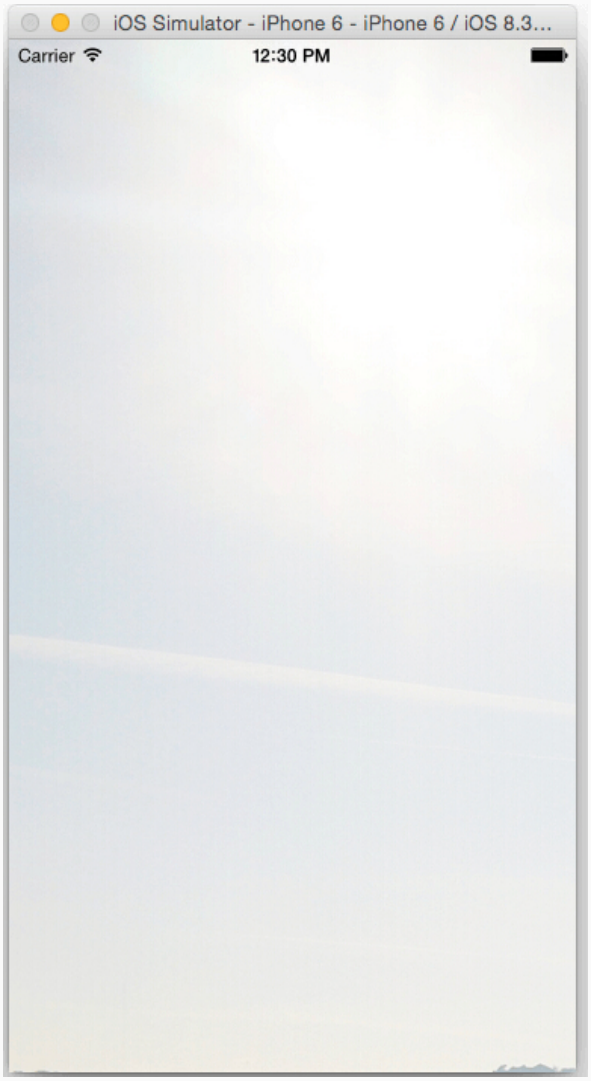
```
var scrollView: UIScrollView!
var imageView: UIImageView!
```

按如下代码所示修改viewDidLoad()方法。

```
override func viewDidLoad() {
    super.viewDidLoad()
    imageView = UIImageView(image: UIImage(named: "image.png"))
    scrollView = UIScrollView(frame: view.bounds)
    scrollView.backgroundColor = UIColor.blackColor()
    scrollView.contentSize = imageView.bounds.size
    scrollView.autoresizingMask = UIViewAutoresizing.FlexibleWidth | UIViewAutoresizing
    scrollView.addSubview(imageView)
    view.addSubview(scrollView)
}
```

上述代码创建了一个UIScrollView和UIImageView，UIImageView被设置为UIScrollView的子视图。contentSize属性控制滚动区域的大小，我们将它设置为跟图片的尺寸一样大（2000×1500）。我们将滚动视图的背景色设置为黑色，这样图片就像在一块黑色幕布上滚动一样。我们将滚动视图的autoresizingMask属性设置为.FlexibleWidth和.FlexibleHeight，使它能够在设备旋转之后自动适应新的宽度和高度。运行当前应用，你已经可以通过拖拽手势来滚动显示图片了。

当你启动应用后，你会发现图片初始显示区域是它左上角的部分。



这是因为滚动视图的bounds的起点默认为(0, 0)，代表了左上角。如果你想改变启动后显示的位置，你需要更改滚动视图的bounds的起点。因为这种需求经常被提起，所以UIScrollView专门提供了一个属性contentOffset用来实现这种需求。

在代码中添加如下语句，注意添加在设置autoresizingMask语句之后。

```
scrollView.contentOffset = CGPoint(x: 1000, y: 450)
```

重新运行应用，你会发现一开始就会显示图片的另一部分而不是左上角。你可以通过这种方式来决定程序启动后将要显示的内容。

缩放

我们已经添加了一个UIScrollView，并且能够让用户通过拖拽来观看尺寸大于屏幕尺寸的内容。相当棒，但如果视图能够缩放的话会带来更好的体验。

要支持缩放功能，你必须为UIScrollView设置一个代理，而且代理必须遵守UIScrollViewDelegate协议，代理还需要实现viewForZoomingInScrollView()方法，该方法返回想要被缩放的视图。

你还应该为缩放设置一个比例，可以通过UIScrollView的minimumZoomScale和maximumZoomScale这两个属性来实现，它们的默认值都是1.0。

按照如下代码更改ScrollViewController的定义：

```
class ScrollViewController: UIViewController, UIScrollViewDelegate {
```

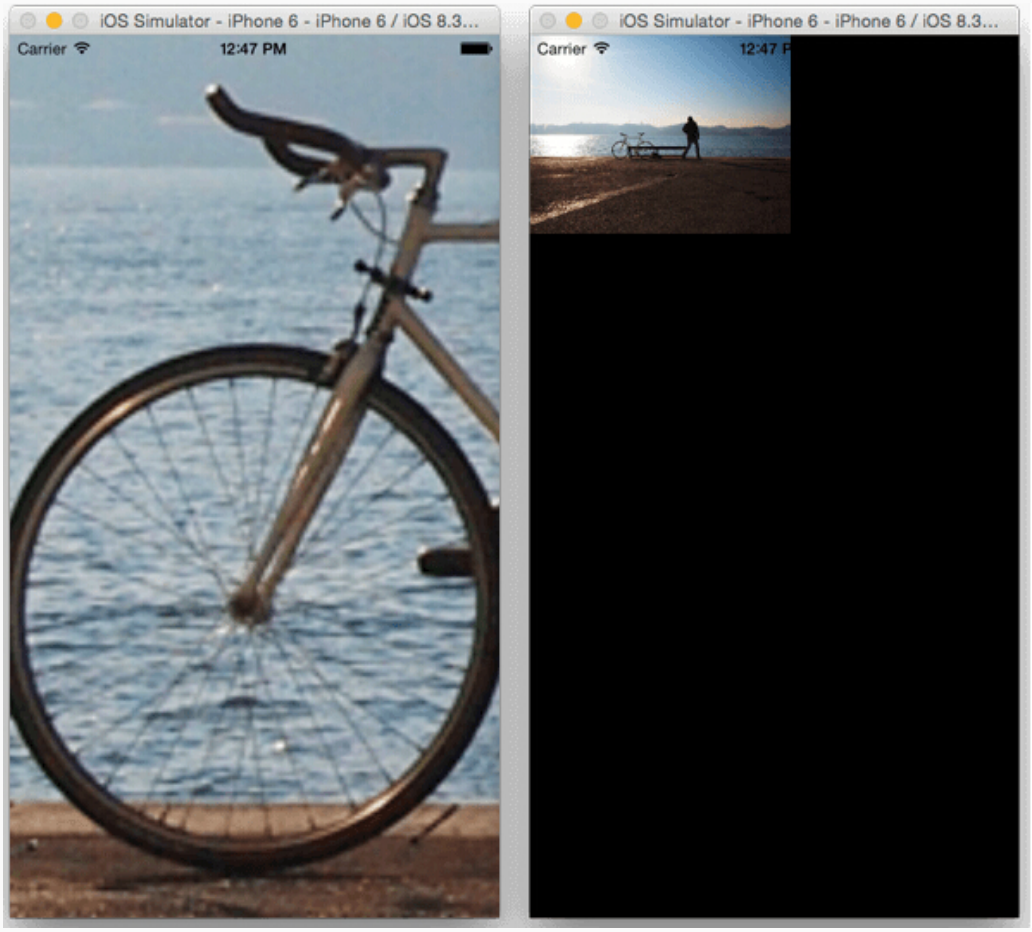
然后添加如下代码：

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```

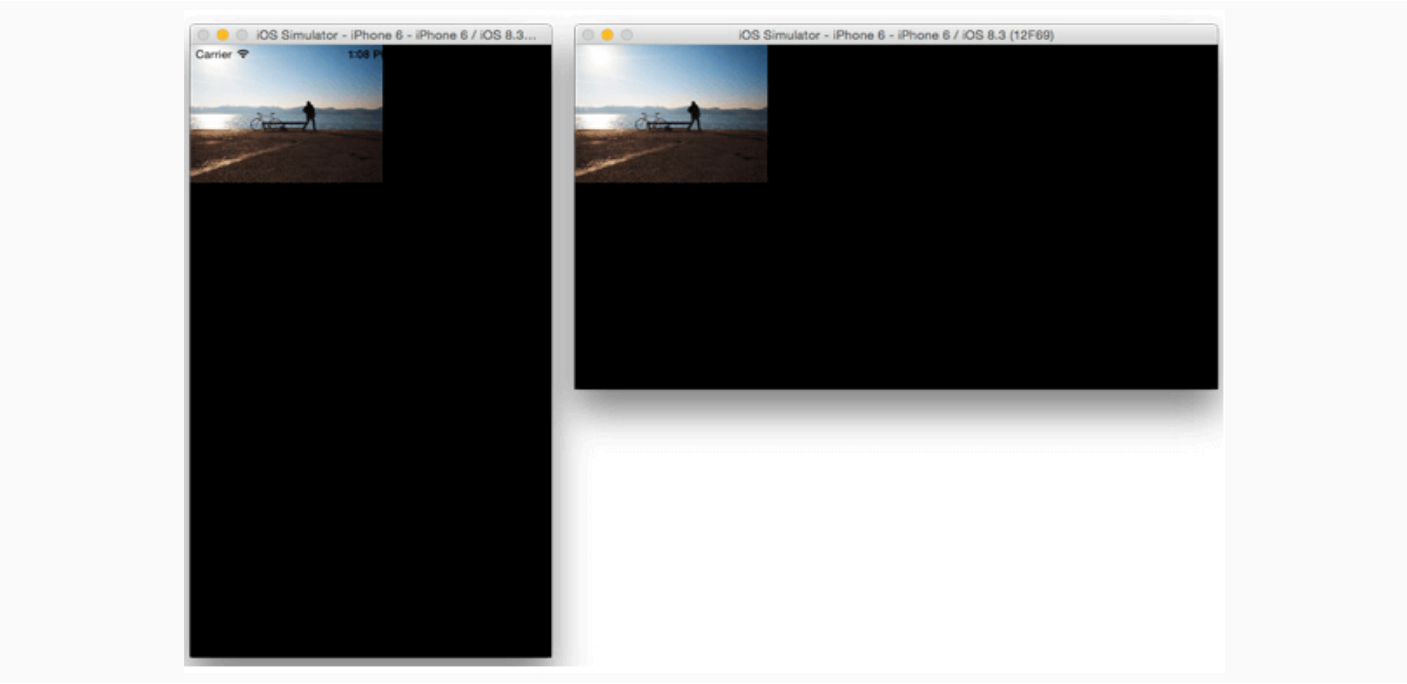
接下来在viewDidLoad()方法的最后添加如下代码：

```
scrollView.delegate = self
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
```

在上述代码中，我们设置了zoomScale为1.0，然后设置了缩放的最大和最小比例。在程序运行后，会按照图片的原始尺寸显示（因为zonmScale为1.0），当你使用捏合手势来操作图片时，你会发现图片可以被缩放了。我们设置了maximumZoomScale为4.0，所以图片最大只能放大到4倍。你也会发现，图片放大4倍后会变得很模糊，所以接下来我们会把它的缩放比例重新设置为1.0。



从上面的图片中我们可以发现，我们之前将minimumZoomScale设置为0.1实在是太小了，屏幕空出了很多空闲的地方。在横屏模式下，空闲的区域看上去更大。我们希望图片能在某一方向上能与屏幕相匹配，让图片既能完全显示，又能尽量减少屏幕的空闲空间。



要达到这样的效果，你必须通过图片尺寸和UIScrollView的尺寸来计算最小的缩放比例。

首先在viewDidLoad()方法中删除以下三行代码：

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
```

在控制器类中添加如下方法。在方法中，我们算出图片同UIScrollView的高度和宽度的比值，并将最小缩放比例设置为两者中更小的那个。注意，我们已经删除了maximumZoomScale的设置，所以它的默认值为1.0。

```
func setZoomScale() {
    let imageViewSize = imageView.bounds.size
    let scrollViewSize = scrollView.bounds.size
    let widthScale = scrollViewSize.width / imageViewSize.width
    let heightScale = scrollViewSize.height / imageViewSize.height
    scrollView.minimumZoomScale = min(widthScale, heightScale)
    scrollView.zoomScale = 1.0
}
```

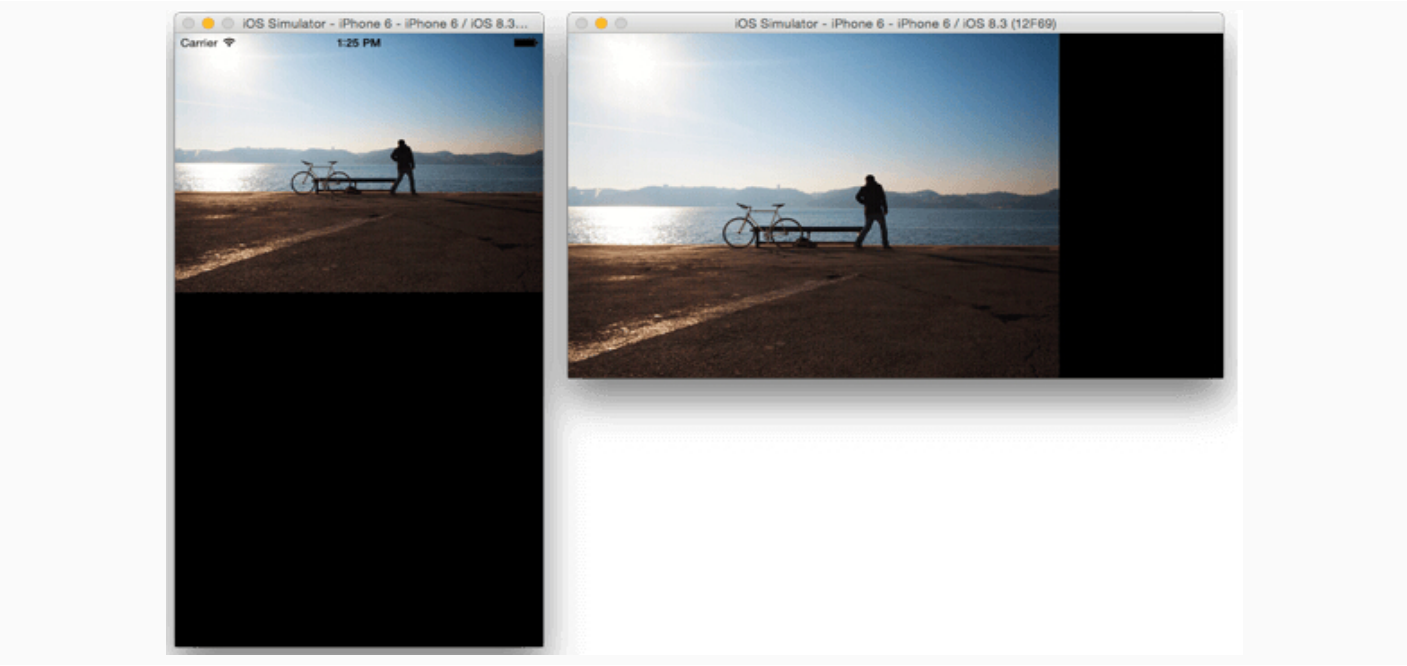
在viewDidLoad()方法最后调用这个方法：

```
setZoomScale()
```

在viewWillLayoutSubviews()方法中也需要调用该方法，这样当用户改变屏幕方向后，图片的尺寸仍然是正确的。

```
override func viewWillLayoutSubviews() {
    setZoomScale()
}
```

运行程序，现在你会发现无论你缩放到多小，图片都会完整显示并且尽量占满剩余的空间。



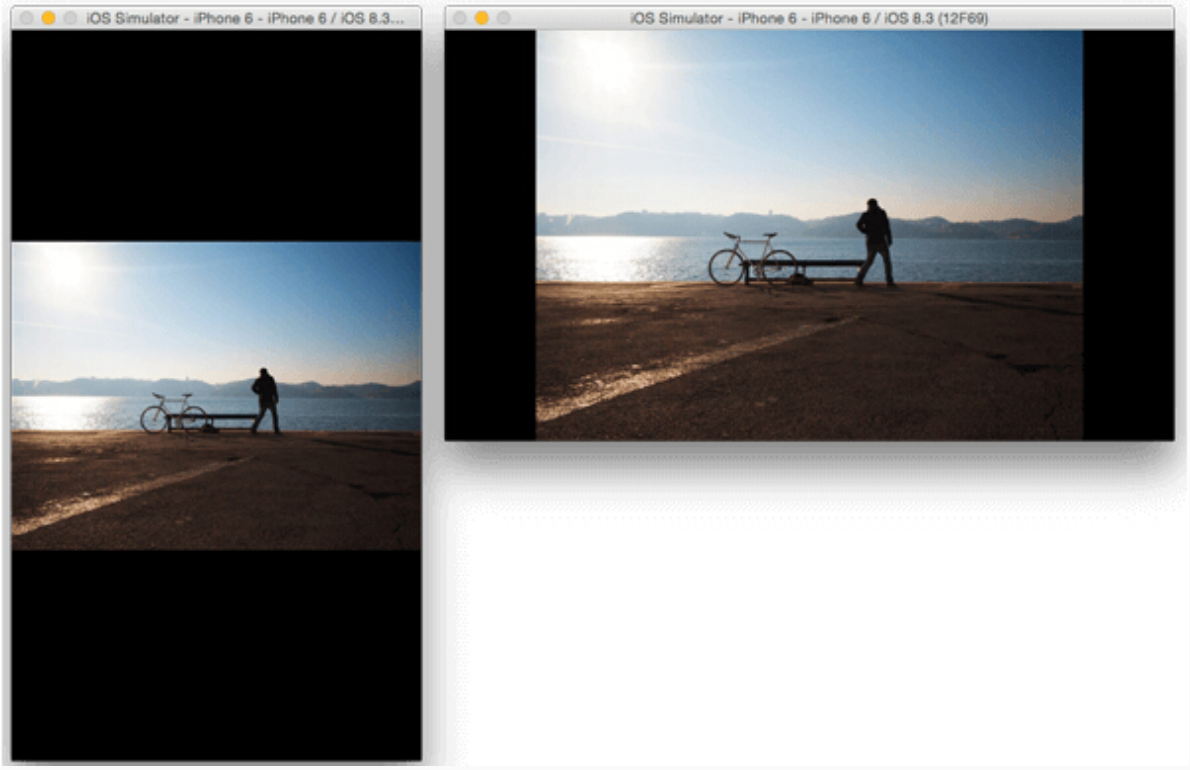
我们可以发现，图片是被定位在屏幕左上角的，我们希望将它放在屏幕中间。

在代码中添加如下方法。

```
func scrollViewDidZoom(scrollView: UIScrollView) {
        let imageViewSize = imageView.frame.size
        let scrollViewSize = scrollView.bounds.size
        let verticalPadding = imageViewSize.height < scrollViewSize.height ? (scrollView
        let horizontalPadding = imageViewSize.width < scrollViewSize.width ? (scrollView
        scrollView.contentInset = UIEdgeInsets(top: verticalPadding, left: horizontalPad
    }
```

这个方法在缩放的时候就会被调用，它会通知代理UIScrollView的缩放比例发生改变了。在上面的方法中，我们计算了图片在滚动视图中的内间距，从而使图片始终在屏幕的中间。对于上、下方向的内边距，我们首先判断图片视图的高度是否小于滚动视图的高度，如果是就将边距设为两者的差值的一半，否则设为0。水平间距我们采用同样的方式计算。然后通过contentInset属性设置所有方向的内边距，这个属性代表了UIScrollView的内容距离UIScrollView本身四周的距离。

运行程序，你会发现当你缩小图片时，图片始终保持在屏幕的中间。



通过双击来缩放

UIScrollView默认只支持通过捏合手势来实现缩放效果，如果想实现通过双击来缩放，则需要自己做些额外的设置。

iOS人机界面指南中介绍了可以通过双击手势来达到缩放的效果。使用双击手势进行缩放需要一定的前提：要缩放的视图只能在最大和最小比例两个固定值之间来回缩放，就像苹果官方的相册应用一样，当你双击图片时，图片放大至最大，当你再次双击时，图片缩小至最小，或者可以通过连续的双击使视图一点点达到最大，然后再次双击的时候，将视图恢复为全屏显示。但是大多数应用需要实现更灵活的双击缩放效果，例如地图应用，当你双击时会使其放大，继续双击会继续放大，想要缩小则可以使用双指捏合手势来实现。

要想在你的程序中实现双击缩放功能，你需要监听UIScrollView的手势并进行处理。在我们的程序中，我们将模仿苹果官方的相册应用的效果，当你双击时放大到最大值，再次双击时则缩小到最小值。

在代码中添加如下两个方法。

```
func setGestureRecognizer() {
    let doubleTap = UITapGestureRecognizer(target: self, action: "handleDoubleTap:")
    doubleTap.numberOfTapsRequired = 2
    scrollView.addGestureRecognizer(doubleTap)
}
func handleDoubleTap(recognizer: UITapGestureRecognizer) {
    if (scrollView.zoomScale > scrollView.minimumZoomScale) {
        scrollView.setZoomScale(scrollView.minimumZoomScale, animated: true)
    } else {
        scrollView.setZoomScale(scrollView.maximumZoomScale, animated: true)
    }
}
```

然后在viewDidLoad()方法最后调用上面的方法。

```
setGestureRecognizer()
```

在上面的代码中，我们为UIScrollView添加了一个双击手势的监听，然后根据图片当前的缩放比例，来判断是将图片放大或者缩小。

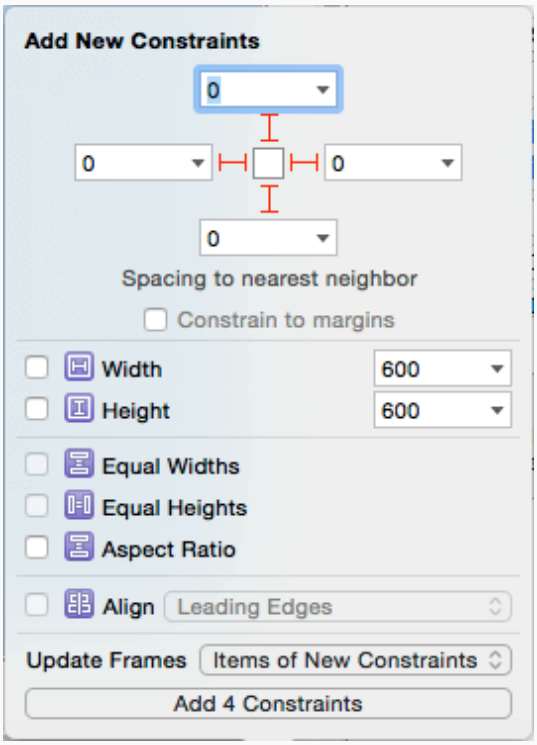
运行程序，你会发现已经能通过双击手势来缩放图片了。

用可视化编程方式创建UIScrollView

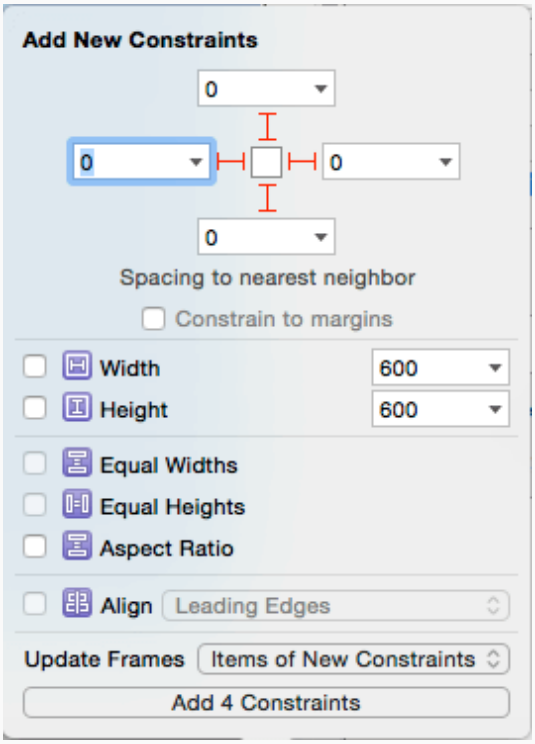
使用storyboard可以实现和我们上面使用代码方式实现的同样的功能，而且更为简单，代码量更少。

在Main.storyboard文件中，拖一个新的视图控制器，并将其设置为初始控制器（既可以将箭头拖到新控制器上，也可以在属性选项卡中选中Is Initial View Controller复选框）。

拖一个UIScrollView到新的控制器中，然后设置其边缘始终粘着屏幕。



然后拖一个UIImageView到刚才的UIScrollView中，将它的边缘设置为粘着UIScrollView。



要记住UIScrollView需要知道它的内容的大小，才可以实现滚动。当你为UIImageView设置图片时，UIScrollView的内容大小就会被自动设置为图片的大小。

在UIImageView的属性选项卡中，将Image属性设置为image.png，然后通过updating the frames解决自动布局的问题。运行程序，你会发现已经实现图片的滚动显示了，并且没有敲一行代码。你还可以在UIScrollView的属性选项卡中查看还有哪些属性可以设置，比如可以设置最大和最小的缩放比例。

如果想要实现缩放功能，你仍然需要通过代码，设置代理并实现viewForZoomingInScrollView()方法，同我们之前做过的一样，就不再重复一遍了。

UIScrollView的嵌套使用

可以在一个UIScrollView中嵌套另一个UIScrollView，两个UIScrollView既可以是相同方向滚动的，也可以是不同方向的。这部分内容的示例代码请使用NestedScrollViews项目。

相同方向的UIScrollView嵌套

相同方向的UIScrollView嵌套是指一个UIScrollView，它有另一个UIScrollView作为子控件，并且它们的滚动方向一致。你可以用相同方向的嵌套来实现这样的效果，比如在UIScrollView中添加多组要区分开的数据，你还可以通过它来实现两个UIScrollView同时滚动时的视差效果。在我们的示例中，我们将两个相同方向的UIScrollView设置不同的滚动速度，从而实现滚动时的视差效果。

打开NestedScrollViews项目中的storyboard文件，你将看到两个UIScrollView，分别叫做foreground和background。background里面添加了一个UIImageView，并将图片设置为image.png，foreground里面添加了一些标签和一个作为容器用的UIView，这些标签只是为了方便我们观看视图的滚动，容器视图我们将在下一节内容中才

用到。

我们的界面这样就算搭建完成了，现在运行程序的话，你会发现只有foreground视图在滚动，而background视图保持不动。接下来我们将要实现background的滚动，并且实现滚动的视差效果。

首先将foreground和background两个UIScrollView连线到控制器，之后代码会如下所示：

```
@IBOutlet weak var background: UIScrollView!
@IBOutlet weak var foreground: UIScrollView!
```

我们需要知道foreground视图滚动了多长的距离，用来计算background视图需要滚动多长的距离。所以我们需要为foreground视图设置一个代理，用来监听它的滚动。

```
class ViewController: UIViewController, UIScrollViewDelegate {
```

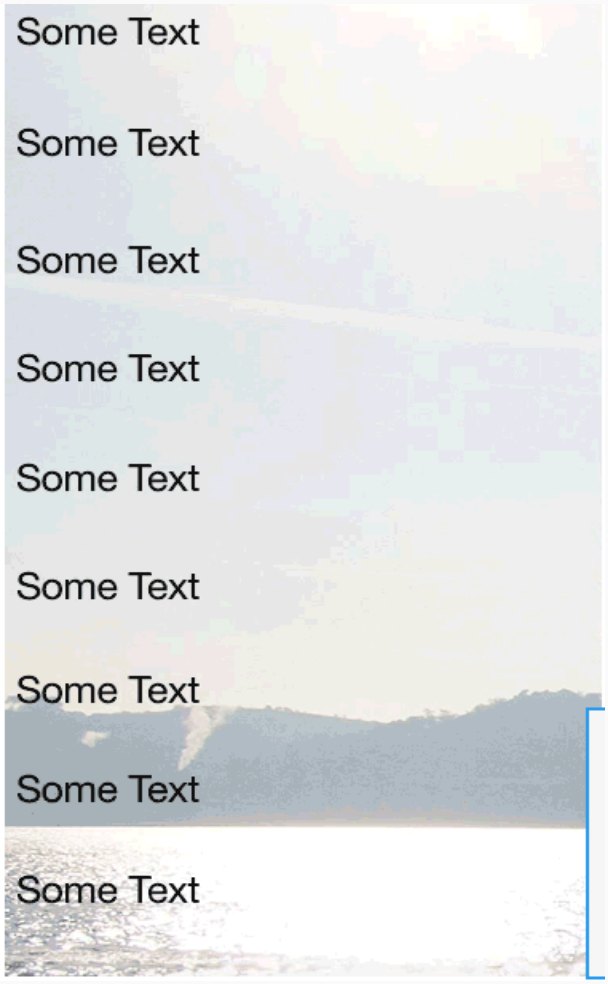
在viewDidLoad()方法中设置foreground视图的代理。

```
foreground.delegate = self
```

然后实现如下代理方法。

```
func scrollViewDidScroll(scrollView: UIScrollView) {
    let foregroundHeight = foreground.contentSize.height - CGRectGetHeight(foreground.bo
    let percentageScroll = foreground.contentOffset.y / foregroundHeight
    let backgroundHeight = background.contentSize.height - CGRectGetHeight(background.bo
    background.contentOffset = CGPoint(x: 0, y: backgroundHeight * percentageScroll)
}
```

在上面的代码中，我们获取了foreground视图可以滚动的最大高度，然后用当前滚动的距离除以它以获取滚动的比例，然后获取background视图可以滚动的最大高度，将其乘以滚动比例，就可以得到background应该滚动的距离。运行你的程序，在进行滚动时你会发现foreground和background两个视图都在滚动，并且background视图滚动的更快，从而有一种视差效果。



交叉方向的UIScrollView嵌套

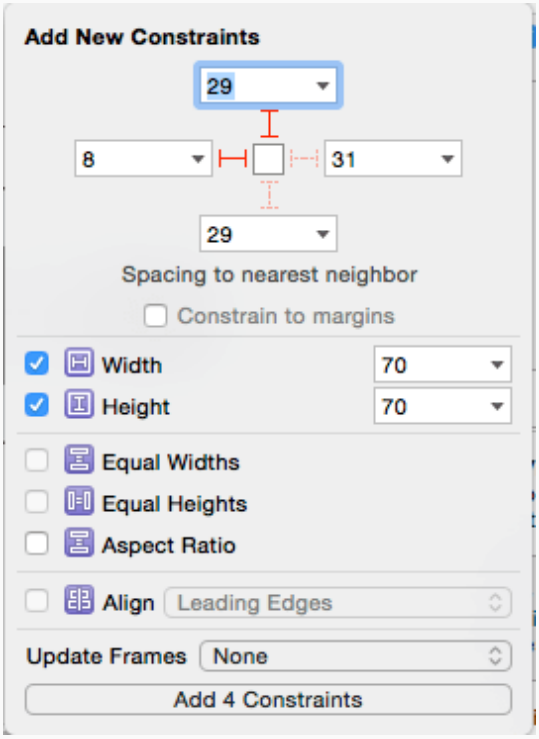
交叉方向的UIScrollView嵌套是指一个UIScrollView，它有另一个UIScrollView作为子控件，并且它们的滚动方向正好相差90°，接下来我们就演示一下这种情况。

在NestedScrollViews项目中，你会发现在foreground里面有一个Container View，我们将用它来设置我们水平滚动的UIScrollView。

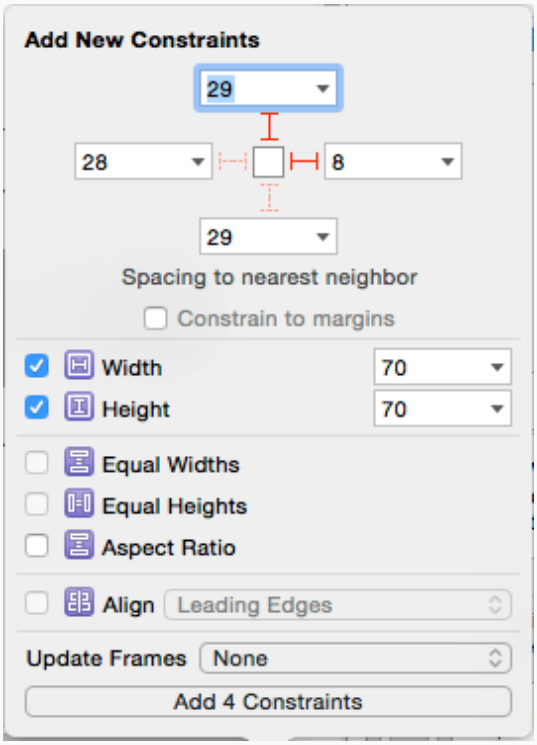
在storyboard中新拖入一个控制器，按住Control键从Container View拖到新的控制器，选择embed方式。然后选中这个控制器，将它的Size选项设为Freeform并将其高度设为128，因为Container View的高度就是128。

往新控制器中拖入一个UIScrollView，设置其边缘始终粘着父控件。然后在UIScrollView中拖入一个70×70的UIView，将其背景色设为灰色方便我们观看，然后复制多个，从左到右依次摆放在UIScrollView中。你不需要精确地去设置每一个UIView的位置，接下来我会教你们怎么去做。现在我们的控制器界面应该是这个样子。

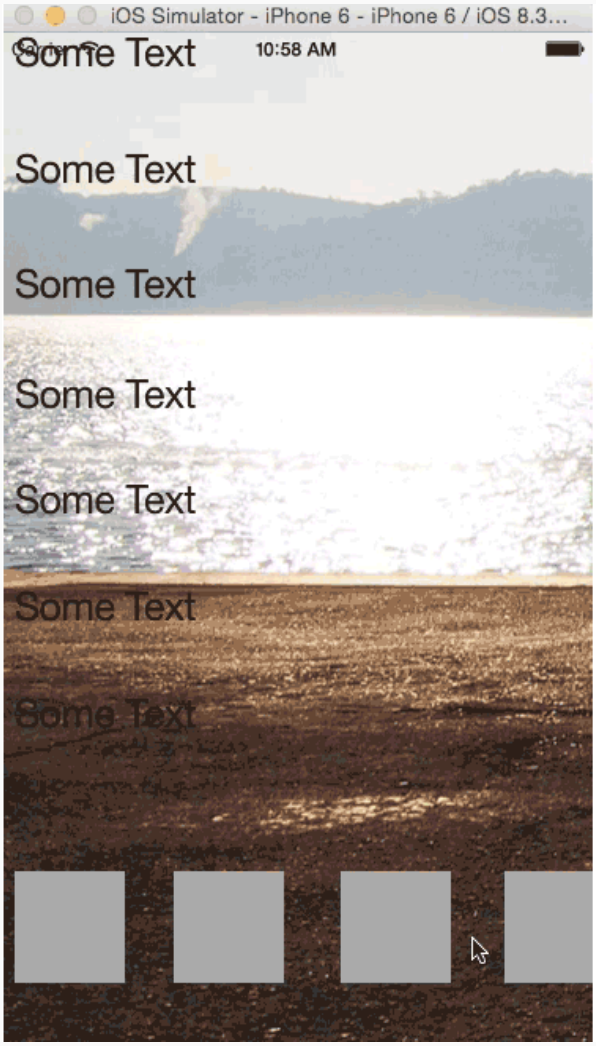
选择最左边的UIView，添加它上边和左边的约束，再添加宽度和高度约束。



再选择最右边的UIView，添加它的上边、右边、宽度和高度约束。



接下来，选中我们的UIScrollView，然后点击上面菜单栏中的Editor > Resolve Auto Layout Issues > All Views > Add Missing Constraints。这样我们所有的UIView就都添加好了约束。运行你的程序，竖直滚动到底部，你会看见我们的Container View，你可以水平滚动它里面的内容。下图中，我将控制器自身视图的背景色设置为透明，所以你看到的效果就是这样的。


iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.3...
Carrier
10:58 AM
Some Text
Some Text
Some Text
Some Text
Some Text
Some Text
Some Text
Some Text