第 3 章　开　关　类

对于 iPhone 或者 iPad 而言，用户除了单击操作外，还有拖动和单击并存的操作。对于这种操作，常用于开关控件中。本章将主要讲解开关控件的有关实例。

实例 17　自定义开关的外观

【实例描述】

本实例实现的功能是显示一个外观发生改变的开关。用户可以通过拖动或者单击的方式实现开关状态的切换，并显示不同效果。其中，使用标签显示开关当前状态信息。运行效果如图 3.1 所示。

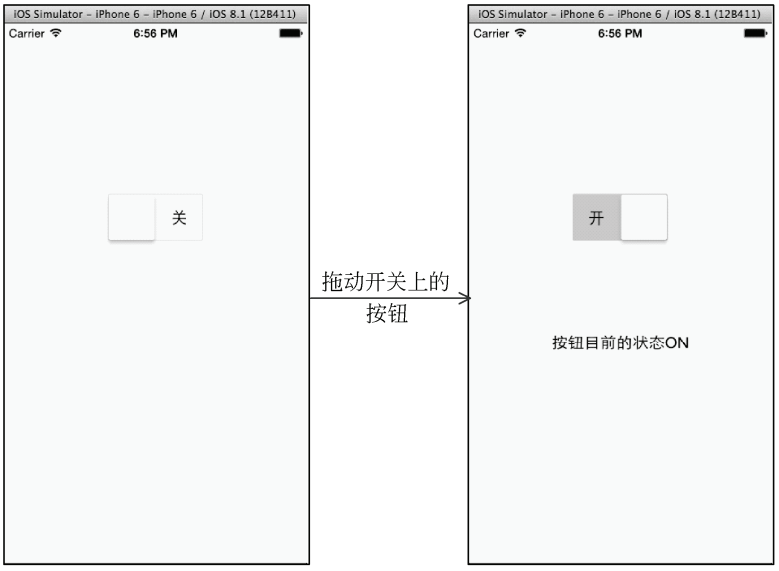

图 3.1　实例 17 运行效果

【实现过程】

（1）创建一个项目，命名为“自定义开关控制器的外观”。

（2）创建一个基于 UIControl 类的 Switch 类。

（3）打开 Switch 文件，编写代码。实现对象、实例变量、属性以及方法的声明。程序代码如下：

```
#import <UIKit/UIKit.h>
@interface Switch : UIControl{
```

```
    //对象
    UILabel *label1;
    UILabel *label2;
    UIView *background;
    UIView *knob;
    //实例变量
    double startTime;
    BOOL isAnimating;
}
//属性
@property (nonatomic, assign) BOOL on;
@property (nonatomic, strong) UIColor *inactiveColor;
@property (nonatomic, strong) UIColor *activeColor;
……
@property (nonatomic,strong)  UILabel *label1;
@property (nonatomic,strong)  UILabel *label2;
//方法
- (void)setOn:(BOOL)on animated:(BOOL)animated;
- (BOOL)isOn;
- (void)showOn:(BOOL)animated;
- (void)showOff:(BOOL)animated;
- (void)setup;                                    //创建开关
@end
```

（4）打开 Switch.m 文件，编写代码。使用的方法如表 3-1 所示。

表 3-1　Switch.m文件中方法总结

方　　法	功　　能
init	初始化
initWithCoder:	初始化实例化对象
initWithFrame:	自定义开关的初始化
setup	创建开关
beginTrackingWithTouch: withEvent:	开始触摸
continueTrackingWithTouch: withEvent:	继续触摸
endTrackingWithTouch: withEvent:	结束触摸
cancelTrackingWithEvent:	取消触摸
setInactiveColor:	设置不活动的颜色
setOnColor:	设置开关打开的颜色
setBorderColor:	设置边框的颜色
setKnobColor:	设置开关上按钮的颜色
setShadowColor:	设置阴影的颜色
setOn:	设置开关是开还是关
setOn:animated:	设置开关的状态，并设置过渡动画
isOn	获取开关是否为开或关的状态
showOn:	显示开关的开位置，并带有动画
showOff:	显示开关的关闭位置，并带有动画

在本文件中代码分为创建开关控件和实现开关控件中的开、关功能两个部分。这里需要讲解几个重要的方法（其他方法请读者参考源代码）。其中创建开关控件由 init、initWithCoder:、initWithFrame:、setup、setInactiveColor:、setOnColor:、setBorderColor:、setKnobColor:、setShadowColor:、setOn:、setOn:animated:、showOn:和 showOff:方法共同

实现。其中 initWithFrame:方法对自定义开关进行初始化。程序代码如下：

```
- (id)initWithFrame:(CGRect)frame
{
    CGRect initialFrame;
    if (CGRectIsEmpty(frame)) {
        initialFrame = CGRectMake(0, 0, 50, 30);            //设置框架
    }
    else {
        initialFrame = frame;                               //设置框架
    }
    self = [super initWithFrame:initialFrame];
    if (self) {
        [self setup];
    }
    return self;
}
```

setup 方法实现自定义开关控件的创建。程序代码如下：

```
- (void)setup {
    self.on = NO;
    self.inactiveColor = [UIColor clearColor];
    self.activeColor = [UIColor colorWithRed:0.89f green:0.89f blue:0.89f
    alpha:1.00f];
    self.onColor = [UIColor colorWithRed:1.0f green:0.85f blue:0.39f alpha:
    1.00f];
    self.borderColor = [UIColor colorWithRed:0.89f green:0.89f blue:0.91f
    alpha:1.00f];                                           //设置边框颜色
    self.knobColor = [UIColor whiteColor];
    self.shadowColor = [UIColor grayColor];                 //设置阴影的颜色
    //背景
    background = [[UIView alloc] initWithFrame:CGRectMake(0, 0, self.frame.
    size.width, self.frame.size.height)];
    background.backgroundColor = [UIColor clearColor];      //设置背景颜色
    background.layer.cornerRadius = self.frame.size.height * 0.5;
    background.layer.borderColor = self.borderColor.CGColor;//设置边框颜色
    background.layer.borderWidth = 1.0;                     //设置边框宽度
    background.userInteractionEnabled = NO;
    [self addSubview:background];
    //标签
    label1 = [[UILabel alloc] initWithFrame:CGRectMake(0, 0, self.frame.
    size.width -
self.frame.size.height, self.frame.size.height)];
    label1.alpha = 1;                                       //设置透明度
    label1.textAlignment =NSTextAlignmentCenter;
    [self addSubview:label1];
    label2=  [[UILabel  alloc]  initWithFrame:CGRectMake(self.frame.size.
height,  0,  self.frame.size.width  -  self.frame.size.height,  self.frame.
size.height)];
    label2.alpha = 1.0;
    label2.textAlignment =NSTextAlignmentCenter;            //设置对齐方式
    [self addSubview:label2];
    //开关上的按钮
    knob = [[UIView alloc] initWithFrame:CGRectMake(1, 1, self.frame.
    size.height - 2,
self.frame.size.height - 2)];
    knob.backgroundColor = self.knobColor;                  //设置背景颜色
    knob.layer.cornerRadius = (self.frame.size.height * 0.5) - 1;
```

```
    knob.layer.shadowColor = self.shadowColor.CGColor; //设置阴影的颜色
    knob.layer.shadowRadius = 2.0;
    knob.layer.shadowOpacity = 0.5;
    knob.layer.shadowOffset = CGSizeMake(0, 3);         //设置阴影的偏移量
    knob.layer.masksToBounds = NO;
    knob.userInteractionEnabled = NO;
    [self addSubview:knob];                             //添加视图对象
    background.layer.cornerRadius = 2;
    knob.layer.cornerRadius = 2;
    isAnimating = NO;
}
```

setOn:animated:方法对开关的动画进行设置。程序代码如下：

```
- (void)setOn:(BOOL)isOn animated:(BOOL)animated {
    on = isOn;
    //判断开关是否打开
    if (isOn) {
        [self showOn:animated];
    }
    else {
        [self showOff:animated];
    }
}
```

showOn:方法对开关的打开位置进行显示，并带有动画。程序代码如下：

```
- (void)showOn:(BOOL)animated {
    CGFloat normalKnobWidth = self.bounds.size.height - 2;
    CGFloat activeKnobWidth = normalKnobWidth + 5;
    //判断是否需要动画
    if (animated) {
        isAnimating = YES;                                  //有动画效果
        //动画实现
        [UIView animateWithDuration:0.3 delay:0.0 options:UIViewAnimation
OptionCurveEaseOut|UIViewAnimationOptionBeginFromCurrentState animations:^{
            if (self.tracking)
                knob.frame = CGRectMake(self.bounds.size.width - (activeKnob
Width + 1), knob.frame.origin.y, activeKnobWidth, knob.frame.size.height);
                                                            //设置框架
            else
                knob.frame = CGRectMake(self.bounds.size.width - (normalKnob
Width + 1), knob.frame.origin.y, normalKnobWidth, knob.frame.size.height);
            background.backgroundColor = self.onColor;      //设置背景颜色
            background.layer.borderColor = self.onColor.CGColor;
        } completion:^(BOOL finished) {
            isAnimating = NO;                               //没有动画效果
        }];
    }
    else {
        if (self.tracking)
            knob.frame = CGRectMake(self.bounds.size.width - (activeKnob
Width + 1), knob.frame.origin.y, activeKnobWidth, knob.frame.size.height);
                                                            //设置框架
        else
            knob.frame = CGRectMake(self.bounds.size.width - (normalKnob
Width + 1), knob.frame.origin.y, normalKnobWidth, knob.frame.size.height);
        background.backgroundColor = self.onColor;          //设置背景颜色
        background.layer.borderColor = self.onColor.CGColor;
```

```
    }
}
```

showOff:方法对开关的关闭位置进行显示，并带有动画。程序代码如下：

```
- (void)showOff:(BOOL)animated {
    CGFloat normalKnobWidth = self.bounds.size.height - 2;
    CGFloat activeKnobWidth = normalKnobWidth + 5;
    //判断是否需要动画
    if (animated) {
        isAnimating = YES;
        //动画的实现
        [UIView animateWithDuration:0.3 delay:0.0 options:
UIViewAnimationOptionCurveEaseOut|UIViewAnimationOptionBeginFromCurrent
State animations:^{
            if (self.tracking) {
                knob.frame = CGRectMake(1, knob.frame.origin.y, activeKnobWidth,
knob.frame.size.height);                                    //设置框架
                background.backgroundColor = self.activeColor;//设置背景颜色
            }
            else {
                knob.frame = CGRectMake(1, knob.frame.origin.y, normalKnobWidth,
knob.frame.size.height);
                background.backgroundColor = self.inactiveColor;
            }
            background.layer.borderColor = self.borderColor.CGColor;
                                                                //设置边框颜色
            } completion:^(BOOL finished) {
            isAnimating = NO;
        }];
    }
    else {
        if (self.tracking) {
            knob.frame = CGRectMake(1, knob.frame.origin.y, activeKnobWidth,
knob.frame.size.height);
            background.backgroundColor = self.activeColor;  //设置背景颜色
        }
        else {
            knob.frame = CGRectMake(1, knob.frame.origin.y, normalKnobWidth,
knob.frame.size.height);                                    //设置框架
            background.backgroundColor = self.inactiveColor;
        }
        background.layer.borderColor = self.borderColor.CGColor; //设置边框颜色
    }
}
```

这时一个开关控件就创建好了，但是此时的开关控件是无法实现开和关功能的，需要使用 beginTrackingWithTouch:withEvent:、continueTrackingWithTouch:withEvent:、endTrackingWithTouch:withEvent:和 cancelTrackingWithEvent:方法实现。其中，beginTrackingWithTouch:withEvent:方法实现触摸开始的功能。程序代码如下：

```
- (BOOL)beginTrackingWithTouch:(UITouch *)touch withEvent:(UIEvent *)event
{
    [super beginTrackingWithTouch:touch withEvent:event];
    startTime = [[NSDate date] timeIntervalSince1970];      //实例化日期对象
    CGFloat activeKnobWidth = self.bounds.size.height - 2 + 5;
    isAnimating = YES;                                      //具有动画效果
    //动画实现
```

```
    [UIView  animateWithDuration:0.3  delay:0.0  options:UIViewAnimation
OptionCurveEaseOut|UIViewAnimationOptionBeginFromCurrentState animations:^{
        if (self.on) {
            knob.frame = CGRectMake(self.bounds.size.width - (activeKnobWidth +
1), knob.frame.origin.y, activeKnobWidth, knob.frame.size.height);
                                                                    //设置框架
            background.backgroundColor = self.onColor;
        }
        else {
            knob.frame = CGRectMake(knob.frame.origin.x, knob.frame.origin.
y, activeKnobWidth, knob.frame.size.height);                        //设置框架
            background.backgroundColor = self.activeColor;
        }
    } completion:^(BOOL finished) {
        isAnimating = NO;
    }];
    return YES;
}
```

ontinueTrackingWithTouch:withEvent 方法实现手指在屏幕上移动的动作。程序代码如下：

```
- (BOOL)continueTrackingWithTouch:(UITouch *)touch withEvent:(UIEvent *)event {
    [super continueTrackingWithTouch:touch withEvent:event];
    //获取触摸的位置
    CGPoint lastPoint = [touch locationInView:self];
    if (lastPoint.x > self.bounds.size.width * 0.5)
        [self showOn:YES];
    else
        [self showOff:YES];
    return YES;
}
```

endTrackingWithTouch:withEvent:方法实现触摸结束。程序代码如下：

```
- (void)endTrackingWithTouch:(UITouch *)touch withEvent:(UIEvent *)event {
    [super endTrackingWithTouch:touch withEvent:event];
    double endTime = [[NSDate date] timeIntervalSince1970];
    double difference = endTime - startTime;
    BOOL previousValue = self.on;
    //判断用户是否轻拍开关或者保持轻拍很久
    if (difference <= 0.2) {
        CGFloat normalKnobWidth = self.bounds.size.height - 2;
        knob.frame = CGRectMake(knob.frame.origin.x, knob.frame.origin.y,
normalKnobWidth, knob.frame.size.height);
        [self setOn:!self.on animated:YES];
    }
    else {
        CGPoint lastPoint = [touch locationInView:self];    //获取触摸位置
        if (lastPoint.x > self.bounds.size.width * 0.5)
            [self setOn:YES animated:YES];
        else
            [self setOn:NO animated:YES];
    }
    if (previousValue != self.on)
        [self sendActionsForControlEvents:UIControlEventValueChanged];
}
```

cancelTrackingWithEvent:方法，实现取消触摸的功能。程序代码如下：

```
- (void)cancelTrackingWithEvent:(UIEvent *)event {
    [super cancelTrackingWithEvent:event];
    //判断开关是否打开
    if (self.on)
        [self showOn:YES];
    else
        [self showOff:YES];
}
```

（5）打开 ViewController.h 文件，编写代码。实现头文件、插座变量以及对象的声明。程序代码如下：

```
#import <UIKit/UIKit.h>
#import "Switch.h"                                          //头文件
@interface ViewController : UIViewController{
    IBOutlet UILabel *lab;                                  //插座变量
    Switch *mySwitch;                                       //对象
}
@end
```

（6）打开 Main.storyboard 文件。在 View Controller 视图控制器的设计界面中放入一个标签控件。双击，将此标签的文本内容删除，将插座变量 lab 与此标签控件进行关联。

（7）打开 ViewController.m 文件，编写代码。实现创建自定义选择器对象并放入到当前的视图中，以及实现显示开关的状态功能。程序代码如下：

```
- (void)viewDidLoad
{
    mySwitch = [[Switch alloc] initWithFrame:CGRectMake(0, 0, 100, 50)];
    mySwitch.center = CGPointMake(self.view.bounds.size.width * 0.5,
    self.view.bounds.size.height * 0.5 - 80);               //设置中心点
    [mySwitch addTarget:self action:@selector(switchChanged:) forControlEvents:
UIControlEventValueChanged];                                //添加动作
    [self.view addSubview:mySwitch];
    mySwitch.label1.text=@"开";                             //设置文本内容
    mySwitch.label2.text=@"关";
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}
- (void)switchChanged:(Switch *)sender {
    NSString *a=[NSString stringWithFormat:@"按钮目前的状态%@",sender.on ?
    @"ON" : @"OFF"];
    lab.text=a;                                             //设置文本内容
}
```

【代码解析】

由于本实例中的代码和方法非常多，为了方便读者的阅读，笔者绘制了一些执行流程图，如图 3.2 和图 3.3 所示。其中，单击运行按钮后一直到在模拟器上显示自定义开关，使用了 viewDidLoad、initWithFrame:、setup、setOn:、setOn:animated:、showOff:、setInactiveColor:、setOnColor:、setBorderColor:、setKnobColor:和 setShadowColor:方法共同实现。以下将实现，在界面上显示一个状态为关的按钮。它们的执行流程如图 3.2 所示。

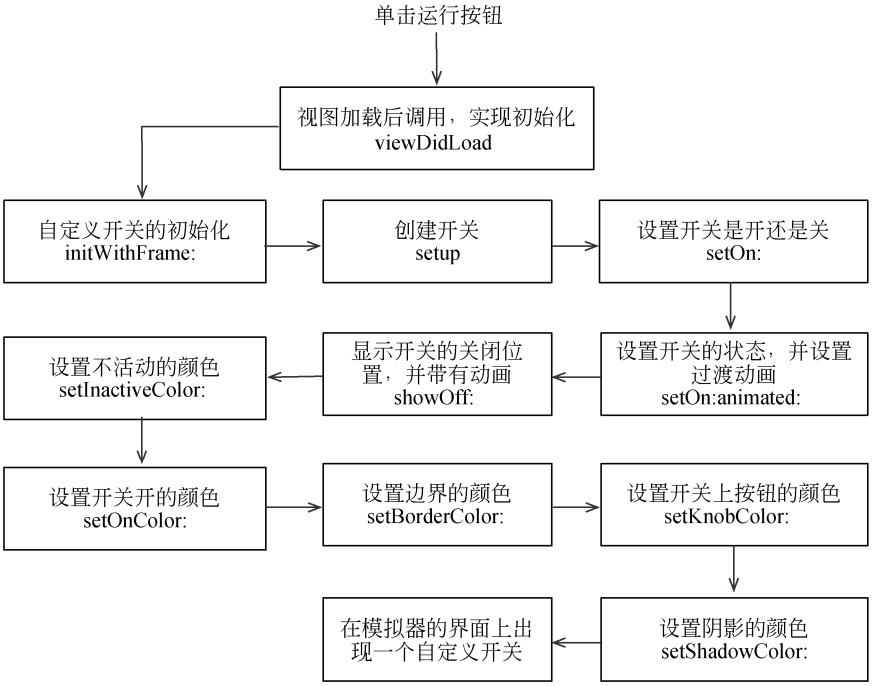

图 3.2　实例 17 程序执行流程 1

如果想要实现轻拍或者拖动开关上的按钮，并将当前的开关状态显示在界面上，需要使用 beginTrackingWithTouch:withEvent:、continueTrackingWithTouch:withEvent:、endTrackingWithTouch:withEvent:、showOn:、showOff:、setOn:animated:和 switchChanged:方法共同实现。以下将实现，通过拖动将按钮的当前状态关改变为开。它们的执行流程如图 3.3 所示。

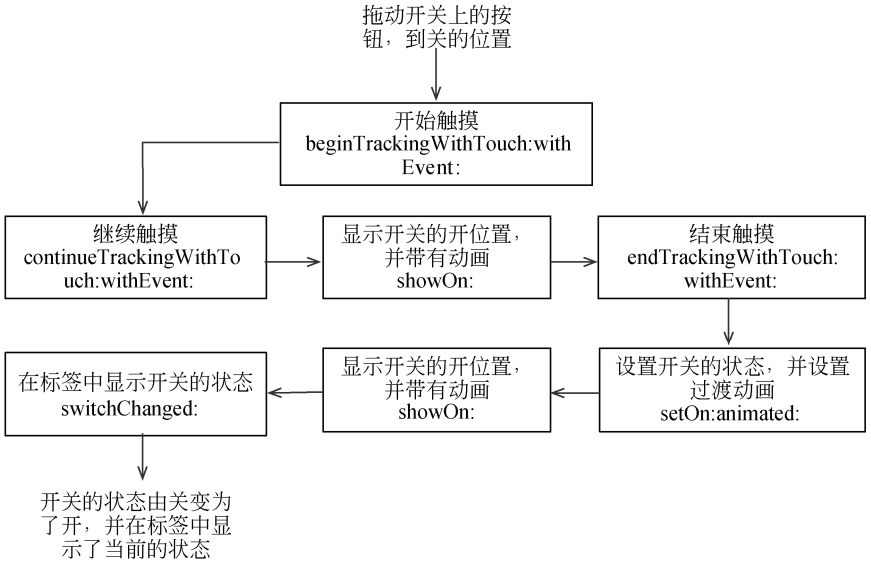

图 3.3　实例 17 程序执行流程 2

实例 18　实现滑块窗口滑动切换的效果

【实例描述】

本实例的功能是使用自定义的开关实现一个通过滑块实现切换效果的功能。运行效果如图 3.4 所示。

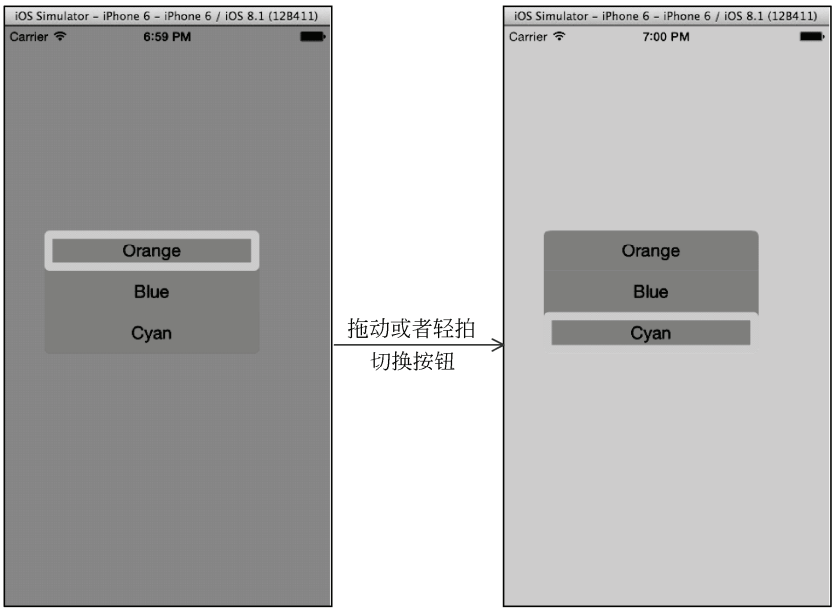

图 3.4　实例 18 运行效果

【实现过程】

在用户界面，首先显示一个自定义的开关，当用户滑动自定义开关上的滑块到指定的某一项时，实现背景颜色的切换。具体的实现步骤如下。

（1）创建一个项目，命名为“实现 UISwitch 窗口滑动切换效果”。

（2）创建一个基于 UIView 类的 Switch 类。

（3）打开 Switch.h 文件，编写代码。实现协议、属性和方法的声明。程序代码如下：

```
#import <UIKit/UIKit.h>
@class Switch;
//协议
@protocol SwitchDelegate <NSObject>
-(void)slideView:(Switch *)slideswitch switchChangedAtIndex:(NSUInteger)index;
@end
@interface Switch : UIView
//属性
@property(nonatomic,retain)UILabel *labelOne,*labelTwo,*labelThree;
@property(nonatomic,retain)UIButton *toggleButton;
@property(nonatomic,assign)id<SwitchDelegate> delegate;
//方法
- (void)setFrameVertical:(CGRect)frame withCornerRadius:(CGFloat)cornerRadius;
```

```
- (void)setText:(NSString *)text forTextIndex:(NSInteger *)number;
                                                          //设置字体
- (void)wasDraggedVertical:(UIButton *)button withEvent:(UIEvent *)event;
- (void)finishedDraggingVertical:(UIButton *)button withEvent:(UIEvent *)event;
- (void)setFrameBackgroundColor:(UIColor *)color;
- (void)setSwitchFrameColor:(UIColor *)color;
- (void)setTextColor:(UIColor *)color;                    //设置文本颜色
-(void)setSwitchBorderWidth:(CGFloat)width;
@end
```

（4）打开 Switch.m 文件，编写代码。使用的方法如表 3-2 所示。

表 3-2　Switch.m文件中方法总结

方　　法	功　　能
initWithFrame:	开关的初始化
hitTest:withEvent:	实现触碰测试
setFrameVertical: withCornerRadius:	设置垂直的框架
setSwitchBorderWidth:	设置框架的边框宽度
setFrameBackgroundColor:	设置框架的背景颜色
setSwitchFrameColor:	设置开关框架的颜色
didTapLabelOneWithGesture:	实现第一个标签的轻拍
didTapLabelTwoWithGesture:	实现第二个标签的轻拍
didTapLabelThreeWithGesture:	实现第三个标签的轻拍
finishedDraggingVertical:	按钮完成拖曳
wasDraggedVertical: withEvent:	按钮的拖曳
setText: forTextIndex:	设置文本

在本文件中代码分为了对开关的设置以及实现滑动切换的功能两个部分。这里需要讲解几个重要的方法（其他方法请读者参考源代码）。其中创建开关控件由 initWithFrame:、setFrameVertical: withCornerRadius:、setSwitchBorderWidth:、setFrameBackgroundColor:、setSwitchFrameColor: 和 setText: forTextIndex: 方法共同实现。其中 setFrameVertical:withCornerRadius: 方法，实现使用圆角半径对框架进行初始化。在此方法中代码分为三个部分：创建标签、创建手势识别器并添加到标签中、创建切换按钮。其中创建标签的程序代码如下：

```
    long double height;
    long double f=(long double)frame.size.height;
    height=f/3;
    //创建 3 个标签
    labelOne = [[UILabel alloc] initWithFrame:CGRectMake(frame.origin.x,
frame.origin.y, frame.size.width, height)];
    labelOne.textAlignment=NSTextAlignmentCenter;          //标签的对齐方式
    [labelOne.layer setBorderColor:[[UIColor darkGrayColor] CGColor]];
                                                           //设置边框颜色
    CAShapeLayer *maskLayerLeft = [CAShapeLayer layer];
    UIBezierPath  *maskPathLeft=[UIBezierPath  bezierPathWithRoundedRect:
labelOne.bounds byRoundingCorners:(UIRectCornerTopLeft | UIRectCornerTopRight)
cornerRadii:CGSizeMake(cornerRadius,cornerRadius)];
    maskLayerLeft.path = maskPathLeft.CGPath;              //设置路径
    labelOne.layer.mask = maskLayerLeft;
    [self addSubview:labelOne];                            //添加视图对象
```

```
    labelTwo = [[UILabel alloc] initWithFrame:CGRectMake(frame.origin.x,
frame.origin.y+height, frame.size.width, height)];
    labelTwo.textAlignment=NSTextAlignmentCenter;          //对齐方式
    [self addSubview:labelTwo];
    labelThree = [[UILabel alloc] initWithFrame:CGRectMake(labelTwo.frame.
origin.x, labelTwo.frame.origin.y+height, frame.size.width, height)];
    labelThree.textAlignment=NSTextAlignmentCenter;
    [labelThree.layer setBorderColor:[[UIColor darkGrayColor] CGColor]];
                                                           //设置边框的颜色
    CAShapeLayer *maskLayerRight = [CAShapeLayer layer];
    UIBezierPath *maskPathRight=[UIBezierPath bezierPathWithRoundedRect:
labelThree.bounds byRoundingCorners:(UIRectCornerBottomRight | UIRectCorner
BottomLeft) cornerRadii:CGSizeMake(cornerRadius,cornerRadius)];
    maskLayerRight.path = maskPathRight.CGPath;            //设置路径
    labelThree.layer.mask = maskLayerRight;
    [self addSubview:labelThree];                          //添加视图对象
    labelOne.userInteractionEnabled=YES;
    labelTwo.userInteractionEnabled=YES;
    labelThree.userInteractionEnabled=YES;
```

创建手势识别器，并将创建的手势识别器添加到创建的标签中。程序代码如下：

```
    UITapGestureRecognizer *tapGestureLabelone =[[UITapGestureRecognizer
alloc] initWithTarget:
    self action:@selector(didTapLabelOneWithGesture:)]; //创建轻拍手势识别器
    [labelOne addGestureRecognizer:tapGestureLabelone];
                                    //将轻拍手势识别器添加到 labelOne 标签中
    UITapGestureRecognizer *tapGestureLabelTwo =[[UITapGestureRecognizer
alloc] initWithTarget:self action:@selector(didTapLabelTwoWithGesture:)];
    [labelTwo addGestureRecognizer:tapGestureLabelTwo]; //添加手势识别器对象
    UITapGestureRecognizer *tapGestureLabelThree =[[UITapGestureRecognizer
alloc] initWithTarget:self action:@selector(didTapLabelThreeWithGesture:)];
    [labelThree addGestureRecognizer:tapGestureLabelThree];
```

创建切换按钮的程序代码如下：

```
    toggleButton = [UIButton buttonWithType:UIButtonTypeCustom];
    [toggleButton setTitle:@"" forState:UIControlStateNormal]; //设置标题
     [toggleButton addTarget:self action:@selector(wasDraggedVertical:withEvent:)
          forControlEvents:UIControlEventTouchDragInside];     //添加动作
    [toggleButton addTarget:self action:@selector(finishedDraggingVertical:withEvent:)
          forControlEvents:UIControlEventTouchUpInside];       //添加动作
     toggleButton.frame = CGRectMake(frame.origin.x, frame.origin.y, frame.
     size.width, height);                                     //设置框架
    toggleButton.backgroundColor=[UIColor colorWithRed:0.0 green:0.0 blue:0.0
    alpha:0.0];                                               //设置背景颜色
    [toggleButton.layer setBorderWidth:4.0];
    [toggleButton.layer setBorderColor:[[UIColor lightGrayColor] CGColor]];
    toggleButton.layer.cornerRadius=cornerRadius;
    [self addSubview:toggleButton];                           //添加视图对象
```

setText:forTextIndex:方法，实现对文本的设置。程序代码如下：

```
- (void)setText:(NSString *)text forTextIndex:(NSInteger *)number
{
    int labelnumber=(int)number;
    //判断 labelnumber 是否为 1
    if(labelnumber==1)
    {
```

```
        labelOne.text=text;
        [self.delegate slideView:self switchChangedAtIndex:0];
    }
    //判断 labelnumber 是否为 2
    if(labelnumber==2)
    {
        labelTwo.text=text;
    }
    //判断 labelnumber 是否为 3
    if(labelnumber==3)
    {
        labelThree.text=text;
    }
}
```

要想实现滑动切换效果，需要使用 hitTest:withEvent:、didTapLabelOneWithGesture:、didTapLabelTwoWithGesture:、didTapLabelThreeWithGesture:和 finishedDraggingVertical:方法共同实现。其中，hitTest:withEvent:方法实现触碰测试，它可以找到触碰的第一响应者。程序代码如下：

```
-(UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event
{
    NSEnumerator *reverseE = [self.subviews reverseObjectEnumerator];
    UIView *iSubView;
    while ((iSubView = [reverseE nextObject])) {
        UIView *viewWasHit = [iSubView hitTest:[self convertPoint:point
toView:iSubView] withEvent:event];                          //实例化对象
        if(viewWasHit) {
            return viewWasHit;                                  //返回
        }
    }
    return [super hitTest:point withEvent:event];
}
```

didTapLabelOneWithGesture:方法，对 LabelOne 的标签实现轻拍。程序代码如下：

```
-(void)didTapLabelOneWithGesture:(UITapGestureRecognizer *)tapGesture {
    [UIView animateWithDuration:0.6
                    animations:^{
                        [toggleButton setFrame:labelOne.frame];  //设置框架
                    }
                    completion:^(BOOL finished) {

                    }
     ];
    [self.delegate slideView:self switchChangedAtIndex:0];
}
```

finishedDraggingVertical:方法实现按钮完成拖动，并在所有选项和切换按钮之间找出最短距离的选项，将按钮以动画的形式移动到此选项上。程序代码如下：

```
- (void)finishedDraggingVertical:(UIButton *)button withEvent:(UIEvent *)event
{
    float diffone,difftwo,diffthree;
    //获取切换按钮和选项之间的距离
    diffone=fabsf(button.frame.origin.y-labelOne.frame.origin.y);
    difftwo=fabsf(labelTwo.frame.origin.y-button.frame.origin.y);
    diffthree=fabsf(labelThree.frame.origin.y-button.frame.origin.y);
```

```
    //判断距离，并移动切换按钮到距离最近的选项上
    if (diffone==difftwo) {
        [UIView animateWithDuration:0.6
                        animations:^{
                            [button setFrame:labelOne.frame];     //设置框架
                        }
                        completion:^(BOOL finished) {
                        }
         ];
    }
    if (difftwo==diffthree) {
        [UIView animateWithDuration:0.6
                        animations:^{
                            [button setFrame:labelTwo.frame];     //设置框架
                        }
                        completion:^(BOOL finished) {
                        }
         ];
    }
    if(diffone<difftwo && diffone<diffthree)
    {
        [UIView animateWithDuration:0.6
                        animations:^{
                            [button setFrame:labelOne.frame];     //设置框架
                        }
                        completion:^(BOOL finished) {
                        }
         ];
    }
    if(difftwo<diffone && difftwo<diffthree)
    {
        [UIView animateWithDuration:0.6
                        animations:^{
                            [button setFrame:labelTwo.frame];     //设置框架
                        }
                        completion:^(BOOL finished) {
                        }
         ];
    }
    if(diffthree<diffone && diffthree<difftwo)
    {
        [UIView animateWithDuration:0.6
                        animations:^{
                            [button setFrame:labelThree.frame];   //设置框架
                        }
                        completion:^(BOOL finished) {
                        }
         ];
    }
}
```

wasDraggedVertical:方法，实现对按钮的拖动。程序代码如下：

```
- (void)wasDraggedVertical:(UIButton *)button withEvent:(UIEvent *)event
{
    UITouch *touch = [[event touchesForView:button] anyObject];//实例化对象
    CGPoint previousLocation = [touch previousLocationInView:button];
                                                          //获取前一次触摸的位置
    CGPoint location = [touch locationInView:button]; //获取触摸的位置
    CGFloat delta_y = location.y - previousLocation.y;
```

```
    //设置中心位置
    button.center = CGPointMake(button.center.x ,
                                button.center.y +delta_y);
    //判断按钮的 y 值是否大于标签对象 labelThree 的 y 值
    if (button.frame.origin.y>labelThree.frame.origin.y) {
        [button setFrame:labelThree.frame];
    }
    //判断按钮的 y 值是否大于标签对象 labelOne 的 y 值
    if (button.frame.origin.y<labelOne.frame.origin.y) {
        [button setFrame:labelOne.frame];
    }
    //判断按钮的 y 值是否等于标签对象 labelOne 的 y 值
    if(button.frame.origin.y==labelOne.frame.origin.y)
    {
        [self.delegate slideView:self switchChangedAtIndex:0];
    }
    else
        //判断按钮的 y 值是否等于标签对象 labelTwo 的 y 值
        if (button.frame.origin.y==labelTwo.frame.origin.y)
        {
            [self.delegate slideView:self switchChangedAtIndex:1];
        }
        else
            //判断按钮的 y 值是否等于标签对象 labelThree 的 y 值
            if (button.frame.origin.y==labelThree.frame.origin.y)
            {
                [self.delegate slideView:self switchChangedAtIndex:2];
            }
}
```

（5）打开 ViewController.h 文件，编写代码。实现头文件和属性等的声明。程序代码如下：

```
#import <UIKit/UIKit.h>
#import "Switch.h"
@interface ViewController : UIViewController<SwitchDelegate>
@property(nonatomic,retain) Switch *switchV;
@end
```

（6）打开 ViewController.m 文件，实现创建 Switch 类的对象并放入到当前的视图中，并实现当前视图的颜色的切换。程序代码如下：

```
- (void)viewDidLoad
{
    switchV=[[Switch alloc]init];
    switchV.delegate=self;
    [switchV setFrameVertical:(CGRectMake(40, 200, 210, 120)) withCorner
    Radius:5.0];
    [switchV setFrameBackgroundColor:[UIColor grayColor]]; //设置背景
    [switchV setSwitchFrameColor:[UIColor lightTextColor]];
    [switchV setSwitchBorderWidth:8.0];                    //设置边框的宽度
    //设置文本内容
    [switchV setText:@"Orange" forTextIndex:1];
    [switchV setText:@"Blue" forTextIndex:2];
    [switchV setText:@"Cyan" forTextIndex:3];
    [self.view addSubview:switchV];
    [super viewDidLoad];
     //Do any additional setup after loading the view, typically from a nib.
}
```

```
//实现切换
-(void)slideView:(Switch *)slideswitch switchChangedAtIndex:(NSUInteger)index
{
    //判断 index 是否等于 0
    if (index==0) {
        self.view.backgroundColor=[UIColor orangeColor];    //设置背景颜色
    }
    else
    //判断 index 是否等于 1
        if (index==1) {
            self.view.backgroundColor=[UIColor blueColor];
        }
        else
            //判断 index 是否等于 2
            if (index==2) {
                self.view.backgroundColor=[UIColor cyanColor];
            }
}
```

【代码解析】

本实例关键功能是颜色的相应切换。下面将详细讲解这个知识点。

当按钮到达不同的选项，就会出现不同的颜色切换。它的实现需要使用按钮的 y 值和每一个标签的 y 值进行判断，从而调用索引不同的 slideView: switchChangedAtIndex:方法。程序代码如下：

```
    if(button.frame.origin.y==labelOne.frame.origin.y)
    {
        [self.delegate slideView:self switchChangedAtIndex:0];
    }
    else
        if (button.frame.origin.y==labelTwo.frame.origin.y)
        {
            [self.delegate slideView:self switchChangedAtIndex:1];
        }
        else
            if (button.frame.origin.y==labelThree.frame.origin.y)
            {
                [self.delegate slideView:self switchChangedAtIndex:2];
            }
```