2016

CustomContentAccessibility

中国信息无障碍产品联盟&信息无障碍研究会　译制

20160923

翻译声明

翻译机构：信息无障碍研究会(ARA) 中国信息无障碍产品联盟(CAPA)

译者：刘辉

审阅：刘彪、沈广荣、

本文档翻译自苹果 iOS Developer Library 的官方文档《CustomContentAccessibility》，如您对翻译文档内容有异议，请将原文文档做为主要参考，原文版权由 Apple Inc 持有并保留。

目录

关于 CustomContentAccessibility

最新版本：Version1.0，2013-12-18。

该样例展示了在一个 iOS App 中怎样支持无障碍和引导访问限制。

构建环境要求：Xcode5.0 及以后版本，iOS7 及以后版本。

运行环境要求：iOS7 及以后，只支持 iPad。

重要：该文档包含开发有关的 API 或技术的初级信息。该信息可能随时会改变，根据该文档实现的软件应该在终端操作系统软件中测试。

该样例，先前被称之为 WWDCMaps，展示了怎样在自绘 UIView 和 UIcontrol 上支持无障碍，演示了怎样为每个图形项目创建一个无障碍元素，并在容器视图上实现 UIAccessibilityContainer 协议与 iOS 无障碍系统进行交互。引导访问限制 API 在 iOS7 中被引入，为了在引导访问开启的时候限制功能，也在本样例中有展示。

点击样例，可下载该样例的样例代码。

1.简介

1.1 在 iOS 应用中支持无障碍

当一个 iOS 应用支持无障碍，存在 3 个基本场景：

1. 对于具有静态无障碍属性值的 UIView 对象，简单地在 Xcode Identity Inspector 的无障碍面板中设置即可。
2. 如果需要编程式改变无障碍属性的值，重写 UIAccessibilityElement 方法返回合适的值。
3. 对于自绘项目（不是 UIView 对象）为了支持无障碍，开发者必须为每一个项目创建一个无障碍元素（UIAccessibilityElement），并在容器视图中实现 UIAccessibility 协议。

该样例展示了怎样处理第二和第三种场景。本样例中的地图是使用代码绘制的，为了支持无障碍，该样例为每一个图形项目创建了一个无障碍元素（会议室、电梯，等等），并在平面视图类中实现 UIAccessibilityContainer，与 iOS 无障碍系统进行交互。基于不同用户交互模式，为 floor 和 coffee 切换控制，将 UIAccessibilityTraitAdjustable 和 UIAccessibilityTraitButton 应用到自定义 UIControl。

1.2 引导访问限制

引导访问是限制 iOS 只运行一个应用的功能，当禁用硬件按钮时。怎样在 iOS 上启用和配置引导访问的描述，参见 http://support.apple.com/kb/HT5509。

在引导访问模式下，UIGuideAccessRestrictionDelegate 协议允许 app 指定可以被用户禁用的额外特性。该样例实现了 UIGuideAccessRestrictionDelegate 协议来限制调整楼层和展示咖啡站的功能。

在引导模式下运行该样例，开发者可以三击 Home 键来展示引导访问选项。

开发者可以在引导访问选项条的右侧看到 CustomContentAccessibility 选项，并可以点击来禁用 floor 和 coffee 切换控制。

1.3 环境

构建环境要求：iOS SDK7.0

运行环境要求：iOS7.0 及以后的版本，只能运行在 iPad 设备中。

1.4 包清单

APLAppDelegate

应用门户，在引导访问模式下，实现 UIGuideAccessRestriction 协议来限制调整楼层和展示咖啡站。

APLViewController

主视图控制器，呈现地图和其他 UI 元素。

APLFloorPlanView

为绘制地图平面视图的自定义 UIView，为图形项目实现 UIAccessibilityContainer 来支持无障碍。

APLTitleView

绘制地图标题的自定义 UIView，重写 UIAccessibilityElement 方法为当前楼层提供 accessibilityValue。

APLElevatorControl

调整楼层的自定义 UIControl，实现 UIAccessibilityTraitAdjustable 特性。

APLCoffeeControl

展示/隐藏咖啡站的自定义 UIControl，实现 UIAccessibilityTraitButton 特性。

APLCommon

绘制地图的常见常量。

1.5 版本变更

1.0——第一个版本，是从 WWDC2013 的 WWDCMaps 样例转换来的。

2.免责声明

重要：这些苹果软件是由 Apple Inc.提供给使用者的，前提是使用者同意以下条款，且对于这些苹果软件的使用、安装、修改或再分配，仍需同意这些条款。如果使用者不同意以下条款，请不要使用、安装、修改或再分配这些苹果软件。

基于使用者同意遵守并遵循以下条款，苹果公司赋予开发者一个个人的、非排他的许可，在苹果的版权下，允许使用者使用、复制、修改和再分配这些苹果软件，无论有或没有修改，还是以原样式和/或二进制样式；如果使用者以整体地且不加修改地对该软件进行再分配，使用者必须在该苹果软件的所有这样的再分配版本中，保留该声明文本和免责条款。从这些苹果软件中抽取的，而没有得到苹果公司事先书面许可使用的名称、商标、服务标记或 Apple Inc.的标识，不能被用来宣传或推销产品。除本声明明确说明的例外，苹果公司没有授权其他权利或许可、明示或暗示，包含但不限于任何专利权利，使用者的衍生作品或其他整合这些苹果软件的作品，都可能会侵犯这些权利。

这些苹果软件是由苹果公司基于“AS IS”提供的。苹果公司未做任何保证、明示或暗示，包含但不限于未侵权的默认保证，适销性和针对特定目的的适用性，关于这些苹果软件或其单独地使用和操作或与使用者产品的整合。

在任何情况下，苹果公司不承担任何特殊的、间接的、偶然的或其他的损害赔偿（包含但不限于，替代品或服务的采购；使用、数据或利益的损失；或商业终端），这些损害是在这些软件的使用、复制、修改和/或再分配中所产生的，无论是否是基于合同、侵权（包含过失）、严格责任或其他引起的，尽管苹果公司已经提醒过这些损害的可能性。

Copyright (C) 2013 Apple Inc. All Rights Reserved。

3. APLAppDelegate.h

文件：APLAppDelegate.h

简介：应用的门户。

版本：1.0

```
#import <UIKit/UIKit.h>

@interface APLAppDelegate : UIResponder <UIApplicationDelegate,
UIGuidedAccessRestrictionDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

4. APLAppDelegate.m

文件：APLAppDelegate.h

简介：应用的门户。

版本：1.0

```
#import "APLAppDelegate.h"

#import "APLViewController.h"



static NSString *const kControlsRestrictionIdentifier =
@"com.apple.CustomContentAccessibility.ControlsRestrictionIdentifier";

@interface APLAppDelegate ()

@property (strong, nonatomic) APLViewController *viewController;

@end

@implementation APLAppDelegate

#pragma mark - UIApplicationDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions

{

    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen]
bounds]];

    self.viewController = [[APLViewController alloc]
initWithNibName:@"APLViewController" bundle:nil];

    self.window.rootViewController = self.viewController;

    [self.window makeKeyAndVisible];

    return YES;
```

```
}


#pragma mark - Implement UIGuidedAccessRestrictionDelegate
//返回一个本地化的、人类可读的字符串，为已提供的标识符提供额外的限制
//信息。
//
- (NSString *)detailTextForGuidedAccessRestrictionWithIdentifier:(NSString *)restrictionIdentifier
{
    if ( [restrictionIdentifier isEqualToString:kControlsRestrictionIdentifier] )
    {
        //该样例未被本地化，此处使用 NSLocalizedString 只用来告知返回字
        //符串将会被显示在引导访问 UI 中，且应该被本地化。
        return NSLocalizedString(@"detailTextForControlsRestriction", nil);
    }
    return nil;
}
//为应用中的每一个自定义访问限制返回一个具有标识符字符串的数组。
//每一个限制标识符必须是不同的字符串。例如：com.MyCompany.MyApp.SomeRestrictionIdentifier。
//
- (NSArray *)guidedAccessRestrictionIdentifiers
{
    return @[ kControlsRestrictionIdentifier ];
```

```
}


//每一次调用该限制，都会伴随标识符状态改变。

//此处，基于引导访问限制的状态，简单地启用或禁用这些控件。

//

- (void)guidedAccessRestrictionWithIdentifier:(NSString *)restrictionIdentifier


didChangeState:(UIGuidedAccessRestrictionState)newRestrictionState

{

        if ( [restrictionIdentifier isEqualToString:kControlsRestrictionIdentifier] )

        {

                [self.viewController    setControlsEnabled:(    newRestrictionState    !=
UIGuidedAccessRestrictionStateDeny )];

        }

}

//返回一个本地化字符串，描述与标识符相关联的限制。

//

-     (NSString     *)textForGuidedAccessRestrictionWithIdentifier:(NSString
*)restrictionIdentifier

{

        if ( [restrictionIdentifier isEqualToString:kControlsRestrictionIdentifier] )

        {

                //该样例未被本地化，此处使用 NSLocalizedString 只是用来告知返回
                //字符串将会被显示在引导访问 UI 中，且应该被本地化。
        return NSLocalizedString(@"textForControlsRestriction", nil);
    }
    return nil;
}


@end
```

5. APLCoffeeControl.h

文件：APLCoffeeControl.h

简介：在可见楼层平面，切换可见 coffee。

版本：1.0

```
#import <UIKit/UIKit.h>

@interface APLCoffeeControl : UIControl

@property (nonatomic, getter = isOn) BOOL on;

@end
```

6. APLCoffeeControl.m

文件：APLCoffeeControl.m

简介：在可见楼层平面，切换可见 coffee。

版本：1.0

```
#import "APLCoffeeControl.h"

#import "APLCommon.h"

@interface APLCoffeeControl ()

- (IBAction)handleSwipe:(UISwipeGestureRecognizer *)swipeGestureRecognizer;

@end

@implementation APLCoffeeControl

#pragma mark - UIControl overrides

//  当启用 coffer 控制的时候，使用自定义绘制方法来绘制 coffer 控件

//

- (void)drawRect:(CGRect)rect

{

    if ( self.enabled )

    {

        CGRect bounds = self.bounds;

        BOOL isOn = self.isOn;

        //  绘制标签

        NSAttributedString *labelAttributedString = [[NSAttributedString alloc] initWithString:@"Coffee" attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize * 2.0], NSForegroundColorAttributeName :
```

```
[UIColor blackColor] }];
        [labelAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(bounds) - labelAttributedString.size.width * 0.5, kPadding * 2.0)];
        //  绘制状态
        NSAttributedString *offStateAttributedString = [[NSAttributedString alloc] initWithString:@"OFF" attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize], NSForegroundColorAttributeName : ( !isOn ) ? [UIColor blackColor] : [UIColor lightGrayColor] }];
        NSAttributedString *onStateAttributedString = [[NSAttributedString alloc] initWithString:@"ON" attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize], NSForegroundColorAttributeName : ( isOn ) ? [UIColor blackColor] : [UIColor lightGrayColor] }];
        if ( isOn )
        {
            [offStateAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(bounds) + CGRectGetWidth(bounds) * 0.33 - offStateAttributedString.size.width * 0.5, CGRectGetMaxY(bounds) - kPadding * 2.0 - offStateAttributedString.size.height)];
            [onStateAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(bounds) - onStateAttributedString.size.width * 0.5, CGRectGetMaxY(bounds) - kPadding * 2.0 - onStateAttributedString.size.height)];
        }
        else
        {
            [offStateAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(bounds) -
offStateAttributedString.size.width * 0.5, CGRectGetMaxY(bounds) - kPadding * 2.0 - offStateAttributedString.size.height)];

            [onStateAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(bounds) - CGRectGetWidth(bounds) * 0.33 - onStateAttributedString.size.width * 0.5, CGRectGetMaxY(bounds) - kPadding * 2.0 - onStateAttributedString.size.height)];

        }

    }

}

//重写可用属性设置器来重新绘制控件

//

- (void)setEnabled:(BOOL)enabled

{

    [super setEnabled:enabled];

    [self setNeedsDisplay];

}

#pragma mark - UIAccessibilityElement (UIAccessibilityTraitButton)

- (BOOL)isAccessibilityElement

{

    return YES;

}

- (NSString *)accessibilityHint

{

    return @"Show or hide coffee locations";
```

```
}

- (NSString *)accessibilityLabel

{

        return @"Coffee";

}

- (NSString *)accessibilityValue

{

        return ( self.isOn ) ? @"On" : @"Off";

}

- (UIAccessibilityTraits)accessibilityTraits

{

        return UIAccessibilityTraitButton;

}

//  在一个元素上实现 accessibilityActivate ，来处理默认操作。

//例如，如果一个原生控件需要一个扫动手势，需要实现该方法，这样 VoiceOver 用户将会执行双击操作来激活该项目。

//如果该方法成功实现激活，返回 YES，否则，返回 NO。默认返回 NO。

//

- (BOOL)accessibilityActivate

{

        self.on = !self.isOn;

        return YES;

}

#pragma mark - Properties
```

```
- (void)setOn:(BOOL)on
{
    if ( on != _on )
    {
        _on = on;
        [self sendActionsForControlEvents:UIControlEventValueChanged];
        [self setNeedsDisplay];
    }
}
#pragma mark - Actions
- (IBAction)handleSwipe:(UISwipeGestureRecognizer *)swipeGestureRecognizer
{
    UISwipeGestureRecognizerDirection direction = swipeGestureRecognizer.direction;
    BOOL isOn = self.isOn;
    if ( (direction == UISwipeGestureRecognizerDirectionLeft && isOn) || (direction == UISwipeGestureRecognizerDirectionRight && !isOn) )
    {
        self.on = !isOn;
    }
}
@end
```

7. APLCommon.h

文件：APLCommon.h

简介：常用常量。

版本：1.0

```
extern const CGFloat       kFontSize;

extern const CGFloat       kLineWidth;

extern const NSInteger    kMaximumFloor;

extern const NSInteger    kMinimumFloor;

extern const CGFloat       kPadding;

extern const NSInteger    kFloorDefault;

extern const BOOL          kShowCoffeeDefault;
```

8. APLCommon.m

文件：APLCommon.m

简介：常用常量。

版本：1.0

```
#import "APLCommon.h"

const CGFloat      kFontSize = 16.0;

const CGFloat      kLineWidth = 2.0;

const NSInteger kMaximumFloor = 3;

const NSInteger kMinimumFloor = 1;

const CGFloat      kPadding = 4.0;


const NSInteger kFloorDefault = 2;

const BOOL            kShowCoffeeDefault = NO;
```

9. APLElevatorControl.h

文件：APLElevatorControl.h

简介：一个控制平面视图楼层的自定义 UIControl。

版本：1.0

```
#import <UIKit/UIKit.h>

@interface APLElevatorControl : UIControl

@property (nonatomic) NSInteger floor;

@end
```

10. APLElevatorControl.m

文件：APLElevatorControl.m

简介：一个控制平面视图楼层的自定义 UIControl。

版本：1.0

```
#import "APLElevatorControl.h"

#import "APLCommon.h"

typedef enum : NSUInteger {

        ControlDirectionDown = 1,

        ControlDirectionUp

} TControlDirection;

@interface APLElevatorControl ()

@property (nonatomic) CGPoint touchPoint;

@end

@implementation APLElevatorControl

#pragma mark - UIControl overrides

//当升降控件启用时，使用自定义绘制方法来绘制升降控件。

//

- (void)drawRect:(CGRect)rect

{

        if ( self.enabled )

        {

                BOOL downControlEnabled = [self controlEnabledForDirection:ControlDirectionDown];
```

```
    CGRect downControlRect = [self
controlRectForDirection:ControlDirectionDown];

    BOOL upControlEnabled = [self
controlEnabledForDirection:ControlDirectionUp];

    CGRect upControlRect = [self
controlRectForDirection:ControlDirectionUp];

    // 绘制向下的控件

    NSAttributedString *downControlAttributedString =
[[NSAttributedString alloc] initWithString:@"Down"
attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize],
NSForegroundColorAttributeName : ( !downControlEnabled ) ? [UIColor
lightGrayColor] : [UIColor blackColor] }];

    UIBezierPath *downControlBezierPath = [UIBezierPath
bezierPathWithOvalInRect:downControlRect];

    downControlBezierPath.lineWidth = kLineWidth;

    [( downControlEnabled && [self
controlTouchedForDirection:ControlDirectionDown] ) ? [UIColor darkGrayColor] :
[UIColor whiteColor] setFill];

    [downControlBezierPath fill];

    [( downControlEnabled ) ? [UIColor blackColor] : [UIColor
lightGrayColor] setStroke];

    [downControlBezierPath stroke];

    [downControlAttributedString
drawAtPoint:CGPointMake(CGRectGetMidX(downControlRect) -
downControlAttributedString.size.width * 0.5, CGRectGetMidY(downControlRect)
- downControlAttributedString.size.height * 0.5)];

    //绘制向上的控件

    NSAttributedString *upControlAttributedString = [[NSAttributedString
alloc] initWithString:@"Up" attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize], NSForegroundColorAttributeName : ( !upControlEnabled ) ? [UIColor lightGrayColor] : [UIColor blackColor] }];

            UIBezierPath *upControlBezierPath = [UIBezierPath bezierPathWithOvalInRect:upControlRect];

            upControlBezierPath.lineWidth = kLineWidth;

            [( upControlEnabled && [self controlTouchedForDirection:ControlDirectionUp] ) ? [UIColor darkGrayColor] : [UIColor whiteColor] setFill];

            [upControlBezierPath fill];

            [( upControlEnabled ) ? [UIColor blackColor] : [UIColor lightGrayColor] setStroke];

            [upControlBezierPath stroke];

            [upControlAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(upControlRect) - upControlAttributedString.size.width * 0.5, CGRectGetMidY(upControlRect) - upControlAttributedString.size.height * 0.5)];

      }

}

// 当升降控件被触摸时，返回 YES。

//

- (BOOL)beginTrackingWithTouch:(UITouch *)touch withEvent:(UIEvent *)event

{

      self.touchPoint = [touch locationInView:self];

      if ( [self controlTouchedForDirection:ControlDirectionDown] || [self controlTouchedForDirection:ControlDirectionUp] )
```

```
        {
                [self setNeedsDisplay];
                return YES;
        }
        return NO;
}
//  当触摸结束时，增加或降低楼层。
//
- (void)endTrackingWithTouch:(UITouch *)touch withEvent:(UIEvent *)event
{
        //  降低楼层
        if ( [self controlTouchedForDirection:ControlDirectionDown] )
        {
                [self decrementFloor];
        }
        //  增加楼层
        else if ( [self controlTouchedForDirection:ControlDirectionUp] )
        {
                [self incrementFloor];
        }
        self.touchPoint = CGPointZero;
}
//重写已启用的属性设置器，来重新绘制控件。
```

```
//
- (void)setEnabled:(BOOL)enabled
{
    [super setEnabled:enabled];
    [self setNeedsDisplay];
}
#pragma mark - UIAccessibilityElement (UIAccessibilityTraitAdjustable)
// 基础 UIAccessibilityElement 方法
//
- (BOOL)isAccessibilityElement
{
    return YES;
}
- (NSString *)accessibilityValue
{
    //返回一个格式化数字字符串
    NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
    [numberFormatter setNumberStyle:NSNumberFormatterDecimalStyle];
    return [numberFormatter stringFromNumber:[NSNumber numberWithInteger:self.floor]];
}
- (NSString *)accessibilityLabel
{
    return @"Floor";
}

- (UIAccessibilityTraits)accessibilityTraits

{

        return UIAccessibilityTraitAdjustable;

}

// 如果一个元素具有 UIAccessibilityTraitAdjustable 特性，该元素必须实现以下方法。增加（Incrementing）将会调整该元素使其内容值增加，减少（Decrementing）使其内容值减少。例如，accessibilityIncrement 将会增加一个 UISlider 的值，accessibilityDecrement 将会减少值。

//

- (void)accessibilityDecrement

{

        [self decrementFloor];

        // 增加楼层将会触发 UI 布局改变，所以会发送通知。

        UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification,
nil);

}

- (void)accessibilityIncrement

{

        [self incrementFloor];

        // 降低楼层将会触发 UI 布局改变，所以会发送通知

        UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification,
nil);

}

#pragma mark - Helpers
```

```
// 基于当前楼层，返回可用/禁用状态
//
- (BOOL)controlEnabledForDirection:(TControlDirection)controlDirection
{
    switch ( controlDirection )
    {
        case ControlDirectionDown:
            return self.floor > kMinimumFloor;
        case ControlDirectionUp:
            return self.floor < kMaximumFloor;
        default:
            //不支持的控制方向
            return NO;
    }
}
- (CGRect)controlRectForDirection:(TControlDirection)controlDirection
{
    CGRect bounds = self.bounds;
    CGFloat controlWidth = CGRectGetWidth(bounds) - kPadding * 2.0;
    switch ( controlDirection )
    {
        case ControlDirectionDown:
            return CGRectMake(kPadding, CGRectGetHeight(bounds) - 
kPadding - controlWidth, controlWidth, controlWidth);
```

```
        case ControlDirectionUp:
            return CGRectMake(kPadding, kPadding, controlWidth, controlWidth);
        default:
            //不支持的控制方向
            return CGRectZero;
    }
}
//如果该触摸点在控件矩形之内，返回 YES。
//
- (BOOL)controlTouchedForDirection:(TControlDirection)controlDirection
{
    return CGRectContainsPoint([self controlRectForDirection:controlDirection], self.touchPoint);
}
// 降低楼层并触发事件处理器
//
- (void)decrementFloor
{
    if ( [self controlEnabledForDirection:ControlDirectionDown] )
    {
        self.floor -= 1;
        [self sendActionsForControlEvents:UIControlEventValueChanged];
    }
```

```
}
//  增加楼层并触发事件处理器
//
- (void)incrementFloor
{
        if ( [self controlEnabledForDirection:ControlDirectionUp] )
        {
                self.floor += 1;
                [self sendActionsForControlEvents:UIControlEventValueChanged];
        }
}
#pragma mark - Properties
- (void)setFloor:(NSInteger)floor
{
        //  将楼层限制在[kMinimumFloor, kMaximumFloor]中
        _floor = MAX(MIN(floor, kMaximumFloor), kMinimumFloor);
        [self setNeedsDisplay];
}
@end
```

11. APLFloorPlanView.h

文件：APLFloorPlanView.h

简介：楼层平面视图，一个绘制楼层平面的自定义 UIView。

版本：1.0

```
#import <UIKit/UIKit.h>

@interface APLFloorPlanView : UIView

@property (nonatomic) NSInteger floor;

@property (nonatomic) BOOL showCoffee;

@end
```

12. APLFloorPlanView.m

文本：APLFloorPlanView.m

简介：楼层平面视图，一个绘制楼层平面的自定义 UIView。

版本：1.0

```
#import "APLFloorPlanView.h"

#import "APLCommon.h"

static NSString *const kFloorPlanBezierPath = @"FloorPlanBezierPath";

static NSString *const kFloorPlanString = @"FloorPlanString";

@interface APLFloorPlanView ()

@property (strong, nonatomic) NSMutableArray *accessibilityElements;

@end

@implementation APLFloorPlanView

#pragma mark - UIView overrides

// 自定义绘制方法，绘制楼层平面

//

- (void)drawRect:(CGRect)rect

{

    // 绘制楼层平面特征

    [[self floorPlanFeatures] enumerateObjectsUsingBlock:^(NSDictionary *floorPlanFeature, NSUInteger idx, BOOL *stop) {



        NSString *floorPlanFeatureString = floorPlanFeature[kFloorPlanString];

        BOOL isCoffee = [floorPlanFeatureString isEqualToString:@"Coffee"];
        BOOL isHall = [floorPlanFeatureString isEqualToString:@"Hall"];
        BOOL isLobby = [floorPlanFeatureString isEqualToString:@"Lobby"];
        NSAttributedString *floorPlanFeatureAttributedString = [[NSAttributedString alloc] initWithString:floorPlanFeatureString attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize], NSForegroundColorAttributeName : [UIColor blackColor] }];
        UIBezierPath *floorPlanFeatureBezierPath = floorPlanFeature[kFloorPlanBezierPath];
        floorPlanFeatureBezierPath.lineWidth = kLineWidth;
        [( isCoffee ) ? [UIColor colorWithRed:0.8 green:0.6 blue:0.4 alpha:1.0] : [UIColor whiteColor] setFill];
        [floorPlanFeatureBezierPath fill];
        [( isCoffee ) ? [UIColor colorWithRed:0.4 green:0.2 blue:0.0 alpha:1.0] : [UIColor lightGrayColor] setStroke];
        [floorPlanFeatureBezierPath stroke];
        if ( !isCoffee && !isHall && !isLobby )
        {
            [floorPlanFeatureAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(floorPlanFeatureBezierPath.bounds) - floorPlanFeatureAttributedString.size.width * 0.5, CGRectGetMidY(floorPlanFeatureBezierPath.bounds) - floorPlanFeatureAttributedString.size.height * 0.5)];
        }
    }];
}
#pragma mark - UIAccessibilityContainer
```

```
// 平面视图的内容是自定义绘制生成和呈现的，而不是使用 UIView。

// 对于像 room 和 coffee stop 的对象，为了支持无障碍，APLFloorPlanView 需要通过 UIAccessibilityContainer 协议与 iOS 无障碍系统进行交互。

// Helper 方法，如果对象不存在，在平面视图中为对象创建无障碍元素，并返回给调用者。一个无障碍元素的形态由 accessibilityPath 呈现。

//

- (NSArray *)accessibilityElements

{

    if ( _accessibilityElements == nil )

    {

        _accessibilityElements = [NSMutableArray array];



        // 创建无障碍元素

        [[self floorPlanFeatures] enumerateObjectsWithOptions:NSEnumerationReverse usingBlock:^(NSDictionary *floorPlanFeature, NSUInteger idx, BOOL *stop) {



            UIAccessibilityElement *accessibilityElement = [[UIAccessibilityElement alloc] initWithAccessibilityContainer:self];



            accessibilityElement.accessibilityLabel = floorPlanFeature[kFloorPlanString];



            accessibilityElement.accessibilityPath = UIAccessibilityConvertPathToScreenCoordinates(floorPlanFeature[kFloorPlanBezierPath], self);
```

```
                [_accessibilityElements addObject:accessibilityElement];
            }];
    }
    return _accessibilityElements;
}
//   无障碍容器必须将 NO 返回给-isAccessibilityElement。
//
- (BOOL)isAccessibilityElement
{
    return NO;
}
// 基于索引，按顺序返回无障碍元素。
// 默认返回 nil。
//
- (id)accessibilityElementAtIndex:(NSInteger)index
{
    return [self.accessibilityElements objectAtIndex:index];
}
// 返回容器内无障碍元素的个数。
//
- (NSInteger)accessibilityElementCount
{
    return self.accessibilityElements.count;
```

```
}
//  为一个无障碍元素返回有序索引值。
//  默认返回 NSNotFound。
//
- (NSInteger)indexOfAccessibilityElement:(id)element
{
        return [self.accessibilityElements indexOfObject:element];
}
#pragma mark - Helpers
//为平面视图设置对象路径。
//
- (NSArray *)floorPlanFeatures
{
        static UIBezierPath *exhibitHall1BezierPath = nil;
        static UIBezierPath *exhibitHall2BezierPath = nil;
        static UIBezierPath *lobby1BezierPath = nil;
        static UIBezierPath *lobby2BezierPath = nil;

        static UIBezierPath *elevatorsBezierPath = nil;
        static UIBezierPath *escalatorsBezierPath = nil;
        static UIBezierPath *restroomsBezierPath = nil;
        static UIBezierPath *stairsBezierPath = nil;
        static UIBezierPath *room2000BezierPath = nil;

        static UIBezierPath *room2001BezierPath = nil;

        static UIBezierPath *room2002BezierPath = nil;

        static UIBezierPath *room2003BezierPath = nil;

        static UIBezierPath *room2004BezierPath = nil;

        static UIBezierPath *room2005BezierPath = nil;

        static UIBezierPath *room2006BezierPath = nil;

        static UIBezierPath *room2007BezierPath = nil;

        static UIBezierPath *room2008BezierPath = nil;

        static UIBezierPath *room2009BezierPath = nil;

        static UIBezierPath *room2010BezierPath = nil;

        static UIBezierPath *room2011BezierPath = nil;

        static UIBezierPath *room2012BezierPath = nil;

        static UIBezierPath *room2014BezierPath = nil;

        static UIBezierPath *room2016BezierPath = nil;

        static UIBezierPath *room2018BezierPath = nil;

        static UIBezierPath *room2020BezierPath = nil;

        static UIBezierPath *room2022BezierPath = nil;

        static UIBezierPath *room2024BezierPath = nil;



        static UIBezierPath *coffee1001BezierPath = nil;

        static UIBezierPath *coffee1002BezierPath = nil;

        static UIBezierPath *coffee1003BezierPath = nil;
        static UIBezierPath *coffee1004BezierPath = nil;
        static UIBezierPath *coffee1005BezierPath = nil;
        static UIBezierPath *coffee2001BezierPath = nil;
        static UIBezierPath *coffee2002BezierPath = nil;
        static UIBezierPath *coffee2003BezierPath = nil;
        static UIBezierPath *coffee3001BezierPath = nil;
        static UIBezierPath *coffee3002BezierPath = nil;
        static UIBezierPath *coffee3003BezierPath = nil;

        static dispatch_once_t floorPlanBezierPathsOnceToken;
        dispatch_once(&floorPlanBezierPathsOnceToken, ^{
            exhibitHall1BezierPath = [UIBezierPath bezierPath];
            [exhibitHall1BezierPath moveToPoint:CGPointMake(115.0, 70.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(480.0, 70.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(480.0, 130.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(535.0, 130.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(535.0, 300.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(480.0, 300.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(480.0, 375.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(518.0, 375.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(518.0, 405.0)];
            [exhibitHall1BezierPath addLineToPoint:CGPointMake(115.0, 405.0)];
            [exhibitHall1BezierPath closePath];
```

```
exhibitHall2BezierPath = [UIBezierPath bezierPath];
[exhibitHall2BezierPath moveToPoint:CGPointMake(115.0, 70.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 70.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 164.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(535.0, 164.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(535.0, 300.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 300.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 375.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(518.0, 375.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(518.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(480.0, 444.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(428.0, 462.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(428.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(376.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(376.0, 444.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(324.0, 462.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(324.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(272.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(272.0, 444.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(220.0, 462.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(220.0, 405.0)];
```

```
[exhibitHall2BezierPath addLineToPoint:CGPointMake(168.0, 405.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(168.0, 444.0)];
[exhibitHall2BezierPath addLineToPoint:CGPointMake(115.0, 462.0)];
[exhibitHall2BezierPath closePath];

lobby1BezierPath = [UIBezierPath bezierPath];
[lobby1BezierPath moveToPoint:CGPointMake(535.0, 130.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(538.0, 130.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(538.0, 50.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(610.0, 50.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(616.0, 86.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(644.0, 86.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(644.0, 410.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(536.0, 410.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(536.0, 405.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(518.0, 405.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(518.0, 375.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(538.0, 375.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(538.0, 230.0)];
[lobby1BezierPath addLineToPoint:CGPointMake(535.0, 230.0)];
[lobby1BezierPath closePath];

lobby2BezierPath = [UIBezierPath bezierPath];
```

```
[lobby2BezierPath moveToPoint:CGPointMake(535.0, 164.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(538.0, 164.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(538.0, 50.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(610.0, 50.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(670.0, 410.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(536.0, 458.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(536.0, 405.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(518.0, 405.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(518.0, 375.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(538.0, 375.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(538.0, 230.0)];
[lobby2BezierPath addLineToPoint:CGPointMake(535.0, 230.0)];
[lobby2BezierPath closePath];


elevatorsBezierPath = [UIBezierPath bezierPath];
[elevatorsBezierPath moveToPoint:CGPointMake(538.0, 98.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(535.0, 98.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(535.0, 95.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(518.0, 95.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(518.0, 125.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(535.0, 125.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(535.0, 122.0)];
[elevatorsBezierPath addLineToPoint:CGPointMake(538.0, 122.0)];
```

```
[elevatorsBezierPath closePath];


escalatorsBezierPath = [UIBezierPath bezierPath];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 232.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 232.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 340.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(602.0, 340.0)];
[escalatorsBezierPath closePath];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 238.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 238.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 244.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 244.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 250.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 250.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 256.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 256.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 262.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 262.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 268.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 268.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 274.0)];
[escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 274.0)];
[escalatorsBezierPath moveToPoint:CGPointMake(602.0, 280.0)];
```

```
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 280.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 286.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 286.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 292.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 292.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 298.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 298.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 304.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 304.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 310.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 310.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 316.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 316.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 322.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 322.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 328.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 328.0)];
        [escalatorsBezierPath moveToPoint:CGPointMake(602.0, 334.0)];
        [escalatorsBezierPath addLineToPoint:CGPointMake(630.0, 334.0)];

        restroomsBezierPath = [UIBezierPath bezierPath];
        [restroomsBezierPath moveToPoint:CGPointMake(115.0, 395.0)];
        [restroomsBezierPath addLineToPoint:CGPointMake(80.0, 395.0)];
```

```
[restroomsBezierPath addLineToPoint:CGPointMake(80.0, 370.0)];
[restroomsBezierPath addLineToPoint:CGPointMake(72.0, 370.0)];
[restroomsBezierPath addLineToPoint:CGPointMake(72.0, 318.0)];
[restroomsBezierPath addLineToPoint:CGPointMake(112.0, 318.0)];
[restroomsBezierPath addLineToPoint:CGPointMake(112.0, 370.0)];
[restroomsBezierPath addLineToPoint:CGPointMake(115.0, 370.0)];
[restroomsBezierPath closePath];

stairsBezierPath = [UIBezierPath bezierPath];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 54.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 54.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 90.0)];
[stairsBezierPath addLineToPoint:CGPointMake(578.0, 90.0)];
[stairsBezierPath closePath];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 60.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 60.0)];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 66.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 66.0)];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 72.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 72.0)];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 78.0)];
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 78.0)];
[stairsBezierPath moveToPoint:CGPointMake(578.0, 84.0)];
```

```
[stairsBezierPath addLineToPoint:CGPointMake(598.0, 84.0)];


room2000BezierPath = [UIBezierPath bezierPath];
[room2000BezierPath moveToPoint:CGPointMake(465.0, 85.0)];
[room2000BezierPath addLineToPoint:CGPointMake(465.0, 158.0)];
[room2000BezierPath addLineToPoint:CGPointMake(425.0, 158.0)];
[room2000BezierPath addLineToPoint:CGPointMake(425.0, 85.0)];
[room2000BezierPath closePath];


room2001BezierPath = [UIBezierPath bezierPath];
[room2001BezierPath moveToPoint:CGPointMake(425.0, 202.0)];
[room2001BezierPath addLineToPoint:CGPointMake(490.0, 202.0)];
[room2001BezierPath addLineToPoint:CGPointMake(490.0, 284.0)];
[room2001BezierPath addLineToPoint:CGPointMake(425.0, 284.0)];
[room2001BezierPath closePath];


room2002BezierPath = [UIBezierPath bezierPath];
[room2002BezierPath moveToPoint:CGPointMake(425.0, 70.0)];
[room2002BezierPath addLineToPoint:CGPointMake(425.0, 158.0)];
[room2002BezierPath addLineToPoint:CGPointMake(372.0, 158.0)];
[room2002BezierPath addLineToPoint:CGPointMake(372.0, 70.0)];
[room2002BezierPath closePath];
```

```
room2003BezierPath = [UIBezierPath bezierPath];
[room2003BezierPath moveToPoint:CGPointMake(372.0, 202.0)];
[room2003BezierPath addLineToPoint:CGPointMake(425.0, 202.0)];
[room2003BezierPath addLineToPoint:CGPointMake(425.0, 284.0)];
[room2003BezierPath addLineToPoint:CGPointMake(372.0, 284.0)];
[room2003BezierPath closePath];

room2004BezierPath = [UIBezierPath bezierPath];
[room2004BezierPath moveToPoint:CGPointMake(372.0, 70.0)];
[room2004BezierPath addLineToPoint:CGPointMake(372.0, 158.0)];
[room2004BezierPath addLineToPoint:CGPointMake(320.0, 158.0)];
[room2004BezierPath addLineToPoint:CGPointMake(320.0, 70.0)];
[room2004BezierPath closePath];

room2005BezierPath = [UIBezierPath bezierPath];
[room2005BezierPath moveToPoint:CGPointMake(320.0, 202.0)];
[room2005BezierPath addLineToPoint:CGPointMake(372.0, 202.0)];
[room2005BezierPath addLineToPoint:CGPointMake(372.0, 284.0)];
[room2005BezierPath addLineToPoint:CGPointMake(320.0, 284.0)];
[room2005BezierPath closePath];

room2006BezierPath = [UIBezierPath bezierPath];
[room2006BezierPath moveToPoint:CGPointMake(320.0, 70.0)];
```

```
            [room2006BezierPath addLineToPoint:CGPointMake(320.0, 158.0)];
            [room2006BezierPath addLineToPoint:CGPointMake(268.0, 158.0)];
            [room2006BezierPath addLineToPoint:CGPointMake(268.0, 70.0)];
            [room2006BezierPath closePath];

            room2007BezierPath = [UIBezierPath bezierPath];
            [room2007BezierPath moveToPoint:CGPointMake(268.0, 202.0)];
            [room2007BezierPath addLineToPoint:CGPointMake(320.0, 202.0)];
            [room2007BezierPath addLineToPoint:CGPointMake(320.0, 284.0)];
            [room2007BezierPath addLineToPoint:CGPointMake(268.0, 284.0)];
            [room2007BezierPath closePath];

            room2008BezierPath = [UIBezierPath bezierPath];
            [room2008BezierPath moveToPoint:CGPointMake(268.0, 70.0)];
            [room2008BezierPath addLineToPoint:CGPointMake(268.0, 158.0)];
            [room2008BezierPath addLineToPoint:CGPointMake(216.0, 158.0)];
            [room2008BezierPath addLineToPoint:CGPointMake(216.0, 70.0)];
            [room2008BezierPath closePath];

            room2009BezierPath = [UIBezierPath bezierPath];
            [room2009BezierPath moveToPoint:CGPointMake(216.0, 202.0)];
            [room2009BezierPath addLineToPoint:CGPointMake(268.0, 202.0)];
            [room2009BezierPath addLineToPoint:CGPointMake(268.0, 284.0)];
```

```
[room2009BezierPath addLineToPoint:CGPointMake(216.0, 284.0)];
[room2009BezierPath closePath];

room2010BezierPath = [UIBezierPath bezierPath];
[room2010BezierPath moveToPoint:CGPointMake(216.0, 85.0)];
[room2010BezierPath addLineToPoint:CGPointMake(216.0, 158.0)];
[room2010BezierPath addLineToPoint:CGPointMake(164.0, 158.0)];
[room2010BezierPath addLineToPoint:CGPointMake(164.0, 85.0)];
[room2010BezierPath closePath];

room2011BezierPath = [UIBezierPath bezierPath];
[room2011BezierPath moveToPoint:CGPointMake(164.0, 202.0)];
[room2011BezierPath addLineToPoint:CGPointMake(216.0, 202.0)];
[room2011BezierPath addLineToPoint:CGPointMake(216.0, 284.0)];
[room2011BezierPath addLineToPoint:CGPointMake(164.0, 284.0)];
[room2011BezierPath closePath];

room2012BezierPath = [UIBezierPath bezierPath];
[room2012BezierPath moveToPoint:CGPointMake(164.0, 85.0)];
[room2012BezierPath addLineToPoint:CGPointMake(164.0, 158.0)];
[room2012BezierPath addLineToPoint:CGPointMake(115.0, 158.0)];
[room2012BezierPath addLineToPoint:CGPointMake(115.0, 85.0)];
[room2012BezierPath closePath];
```

```
room2014BezierPath = [UIBezierPath bezierPath];
[room2014BezierPath moveToPoint:CGPointMake(425.0, 284.0)];
[room2014BezierPath addLineToPoint:CGPointMake(490.0, 284.0)];
[room2014BezierPath addLineToPoint:CGPointMake(490.0, 300.0)];
[room2014BezierPath addLineToPoint:CGPointMake(480.0, 300.0)];
[room2014BezierPath addLineToPoint:CGPointMake(480.0, 370.0)];
[room2014BezierPath addLineToPoint:CGPointMake(425.0, 370.0)];
[room2014BezierPath closePath];

room2016BezierPath = [UIBezierPath bezierPath];
[room2016BezierPath moveToPoint:CGPointMake(372.0, 284.0)];
[room2016BezierPath addLineToPoint:CGPointMake(425.0, 284.0)];
[room2016BezierPath addLineToPoint:CGPointMake(425.0, 370.0)];
[room2016BezierPath addLineToPoint:CGPointMake(372.0, 370.0)];
[room2016BezierPath closePath];

room2018BezierPath = [UIBezierPath bezierPath];
[room2018BezierPath moveToPoint:CGPointMake(320.0, 284.0)];
[room2018BezierPath addLineToPoint:CGPointMake(372.0, 284.0)];
[room2018BezierPath addLineToPoint:CGPointMake(372.0, 370.0)];
[room2018BezierPath addLineToPoint:CGPointMake(320.0, 370.0)];
[room2018BezierPath closePath];
```

```
room2020BezierPath = [UIBezierPath bezierPath];
[room2020BezierPath moveToPoint:CGPointMake(268.0, 284.0)];
[room2020BezierPath addLineToPoint:CGPointMake(320.0, 284.0)];
[room2020BezierPath addLineToPoint:CGPointMake(320.0, 370.0)];
[room2020BezierPath addLineToPoint:CGPointMake(268.0, 370.0)];
[room2020BezierPath closePath];

room2022BezierPath = [UIBezierPath bezierPath];
[room2022BezierPath moveToPoint:CGPointMake(216.0, 284.0)];
[room2022BezierPath addLineToPoint:CGPointMake(268.0, 284.0)];
[room2022BezierPath addLineToPoint:CGPointMake(268.0, 370.0)];
[room2022BezierPath addLineToPoint:CGPointMake(216.0, 370.0)];
[room2022BezierPath closePath];

room2024BezierPath = [UIBezierPath bezierPath];
[room2024BezierPath moveToPoint:CGPointMake(164.0, 284.0)];
[room2024BezierPath addLineToPoint:CGPointMake(216.0, 284.0)];
[room2024BezierPath addLineToPoint:CGPointMake(216.0, 370.0)];
[room2024BezierPath addLineToPoint:CGPointMake(164.0, 370.0)];
[room2024BezierPath closePath];

coffee1001BezierPath = [UIBezierPath bezierPath];
```

```
[coffee1001BezierPath moveToPoint:CGPointMake(348.0, 202.0)];
[coffee1001BezierPath addLineToPoint:CGPointMake(352.0, 224.0)];
[coffee1001BezierPath addLineToPoint:CGPointMake(364.0, 224.0)];
[coffee1001BezierPath addLineToPoint:CGPointMake(368.0, 202.0)];
[coffee1001BezierPath closePath];

coffee1002BezierPath = [UIBezierPath bezierPath];
[coffee1002BezierPath moveToPoint:CGPointMake(130.0, 182.0)];
[coffee1002BezierPath addLineToPoint:CGPointMake(134.0, 204.0)];
[coffee1002BezierPath addLineToPoint:CGPointMake(146.0, 204.0)];
[coffee1002BezierPath addLineToPoint:CGPointMake(150.0, 182.0)];
[coffee1002BezierPath closePath];

coffee1003BezierPath = [UIBezierPath bezierPath];
[coffee1003BezierPath moveToPoint:CGPointMake(130.0, 262.0)];
[coffee1003BezierPath addLineToPoint:CGPointMake(134.0, 284.0)];
[coffee1003BezierPath addLineToPoint:CGPointMake(146.0, 284.0)];
[coffee1003BezierPath addLineToPoint:CGPointMake(150.0, 262.0)];
[coffee1003BezierPath closePath];

coffee1004BezierPath = [UIBezierPath bezierPath];
[coffee1004BezierPath moveToPoint:CGPointMake(368.0, 282.0)];
[coffee1004BezierPath addLineToPoint:CGPointMake(372.0, 304.0)];
```

```
[coffee1004BezierPath addLineToPoint:CGPointMake(384.0, 304.0)];
[coffee1004BezierPath addLineToPoint:CGPointMake(388.0, 282.0)];
[coffee1004BezierPath closePath];

coffee1005BezierPath = [UIBezierPath bezierPath];
[coffee1005BezierPath moveToPoint:CGPointMake(268.0, 332.0)];
[coffee1005BezierPath addLineToPoint:CGPointMake(272.0, 354.0)];
[coffee1005BezierPath addLineToPoint:CGPointMake(284.0, 354.0)];
[coffee1005BezierPath addLineToPoint:CGPointMake(288.0, 332.0)];
[coffee1005BezierPath closePath];

coffee2001BezierPath = [UIBezierPath bezierPath];
[coffee2001BezierPath moveToPoint:CGPointMake(558.0, 222.0)];
[coffee2001BezierPath addLineToPoint:CGPointMake(562.0, 244.0)];
[coffee2001BezierPath addLineToPoint:CGPointMake(574.0, 244.0)];
[coffee2001BezierPath addLineToPoint:CGPointMake(578.0, 222.0)];
[coffee2001BezierPath closePath];

coffee2002BezierPath = [UIBezierPath bezierPath];
[coffee2002BezierPath moveToPoint:CGPointMake(598.0, 382.0)];
[coffee2002BezierPath addLineToPoint:CGPointMake(602.0, 404.0)];
[coffee2002BezierPath addLineToPoint:CGPointMake(614.0, 404.0)];
[coffee2002BezierPath addLineToPoint:CGPointMake(618.0, 382.0)];
```

```
[coffee2002BezierPath closePath];


coffee2003BezierPath = [UIBezierPath bezierPath];

[coffee2003BezierPath moveToPoint:CGPointMake(128.0, 412.0)];

[coffee2003BezierPath addLineToPoint:CGPointMake(132.0, 434.0)];

[coffee2003BezierPath addLineToPoint:CGPointMake(144.0, 434.0)];

[coffee2003BezierPath addLineToPoint:CGPointMake(148.0, 412.0)];

[coffee2003BezierPath closePath];


coffee3001BezierPath = [UIBezierPath bezierPath];

[coffee3001BezierPath moveToPoint:CGPointMake(558.0, 202.0)];

[coffee3001BezierPath addLineToPoint:CGPointMake(562.0, 224.0)];

[coffee3001BezierPath addLineToPoint:CGPointMake(574.0, 224.0)];

[coffee3001BezierPath addLineToPoint:CGPointMake(578.0, 202.0)];

[coffee3001BezierPath closePath];


coffee3002BezierPath = [UIBezierPath bezierPath];

[coffee3002BezierPath moveToPoint:CGPointMake(568.0, 362.0)];

[coffee3002BezierPath addLineToPoint:CGPointMake(572.0, 384.0)];

[coffee3002BezierPath addLineToPoint:CGPointMake(584.0, 384.0)];

[coffee3002BezierPath addLineToPoint:CGPointMake(588.0, 362.0)];

[coffee3002BezierPath closePath];
```

```
            coffee3003BezierPath = [UIBezierPath bezierPath];
            [coffee3003BezierPath moveToPoint:CGPointMake(438.0, 418.0)];
            [coffee3003BezierPath addLineToPoint:CGPointMake(442.0, 440.0)];
            [coffee3003BezierPath addLineToPoint:CGPointMake(454.0, 440.0)];
            [coffee3003BezierPath addLineToPoint:CGPointMake(458.0, 418.0)];
            [coffee3003BezierPath closePath];
    });
    NSMutableArray *floorPlanFeatures = [NSMutableArray array];
    switch ( self.floor )
    {
        case 1:
        {
            [floorPlanFeatures
addObjectsFromArray:@[ @{ kFloorPlanBezierPath : exhibitHall1BezierPath,
kFloorPlanString : @"Hall" },
                            @{ kFloorPlanBezierPath : lobby1BezierPath,
kFloorPlanString : @"Lobby" },
                            @{ kFloorPlanBezierPath : elevatorsBezierPath,
kFloorPlanString : @"Elevators" },
                            @{ kFloorPlanBezierPath : escalatorsBezierPath,
kFloorPlanString : @"Escalators" },
                            @{ kFloorPlanBezierPath : restroomsBezierPath,
kFloorPlanString : @"Restrooms" },
                            @{ kFloorPlanBezierPath : stairsBezierPath,
kFloorPlanString : @"Stairs" } ]];
```

```
                if ( self.showCoffee )
                {
                    [floorPlanFeatures
addObjectsFromArray:@[ @{ kFloorPlanBezierPath : coffee1001BezierPath,
kFloorPlanString : @"Coffee" },
                            @{ kFloorPlanBezierPath : coffee1002BezierPath,
kFloorPlanString : @"Coffee" },
                            @{ kFloorPlanBezierPath : coffee1003BezierPath,
kFloorPlanString : @"Coffee" },
                            @{ kFloorPlanBezierPath : coffee1004BezierPath,
kFloorPlanString : @"Coffee" },
                            @{ kFloorPlanBezierPath : coffee1005BezierPath,
kFloorPlanString : @"Coffee" } ]];
                }
                break;
            }
            case 2:
            {
                [floorPlanFeatures
addObjectsFromArray:@[ @{ kFloorPlanBezierPath : exhibitHall2BezierPath,
kFloorPlanString : @"Hall" },
                            @{ kFloorPlanBezierPath : lobby2BezierPath,
kFloorPlanString : @"Lobby" },
                            @{ kFloorPlanBezierPath : elevatorsBezierPath,
kFloorPlanString : @"Elevators" },
                            @{ kFloorPlanBezierPath : escalatorsBezierPath,
kFloorPlanString : @"Escalators" },
                              @{ kFloorPlanBezierPath : restroomsBezierPath,
kFloorPlanString : @"Restrooms" },
                              @{ kFloorPlanBezierPath : stairsBezierPath,
kFloorPlanString : @"Stairs" },
                              @{ kFloorPlanBezierPath : room2000BezierPath,
kFloorPlanString : @"2000" },
                              @{ kFloorPlanBezierPath : room2001BezierPath,
kFloorPlanString : @"2001" },
                              @{ kFloorPlanBezierPath : room2002BezierPath,
kFloorPlanString : @"2002" },
                              @{ kFloorPlanBezierPath : room2003BezierPath,
kFloorPlanString : @"2003" },
                              @{ kFloorPlanBezierPath : room2004BezierPath,
kFloorPlanString : @"2004" },
                              @{ kFloorPlanBezierPath : room2005BezierPath,
kFloorPlanString : @"2005" },
                              @{ kFloorPlanBezierPath : room2006BezierPath,
kFloorPlanString : @"2006" },
                              @{ kFloorPlanBezierPath : room2007BezierPath,
kFloorPlanString : @"2007" },
                              @{ kFloorPlanBezierPath : room2008BezierPath,
kFloorPlanString : @"2008" },
                              @{ kFloorPlanBezierPath : room2009BezierPath,
kFloorPlanString : @"2009" },
                              @{ kFloorPlanBezierPath : room2010BezierPath,
kFloorPlanString : @"2010" },
                              @{ kFloorPlanBezierPath : room2011BezierPath,
kFloorPlanString : @"2011" },

                              @{ kFloorPlanBezierPath : room2012BezierPath,
kFloorPlanString : @"2012" },

                              @{ kFloorPlanBezierPath : room2014BezierPath,
kFloorPlanString : @"2014" },

                              @{ kFloorPlanBezierPath : room2016BezierPath,
kFloorPlanString : @"2016" },

                              @{ kFloorPlanBezierPath : room2018BezierPath,
kFloorPlanString : @"2018" },

                              @{ kFloorPlanBezierPath : room2020BezierPath,
kFloorPlanString : @"2020" },

                              @{ kFloorPlanBezierPath : room2022BezierPath,
kFloorPlanString : @"2022" },

                              @{ kFloorPlanBezierPath : room2024BezierPath,
kFloorPlanString : @"2024" } ]];

                if ( self.showCoffee )

                {

                     [floorPlanFeatures
addObjectsFromArray:@[ @{ kFloorPlanBezierPath : coffee2001BezierPath,
kFloorPlanString : @"Coffee" },

                              @{ kFloorPlanBezierPath : coffee2002BezierPath,
kFloorPlanString : @"Coffee" },

                              @{ kFloorPlanBezierPath : coffee2003BezierPath,
kFloorPlanString : @"Coffee" } ]];

                }

                break;
```

```
        }

        case 3:

        {

            [floorPlanFeatures
addObjectsFromArray:@[ @{ kFloorPlanBezierPath : exhibitHall2BezierPath,
kFloorPlanString : @"Hall" },

                         @{ kFloorPlanBezierPath : lobby2BezierPath,
kFloorPlanString : @"Lobby" },

                         @{ kFloorPlanBezierPath : elevatorsBezierPath,
kFloorPlanString : @"Elevators" },

                         @{ kFloorPlanBezierPath : escalatorsBezierPath,
kFloorPlanString : @"Escalators" },

                         @{ kFloorPlanBezierPath : restroomsBezierPath,
kFloorPlanString : @"Restrooms" },

                         @{ kFloorPlanBezierPath : stairsBezierPath,
kFloorPlanString : @"Stairs" },

                         @{ kFloorPlanBezierPath : room2000BezierPath,
kFloorPlanString : @"3000" },

                         @{ kFloorPlanBezierPath : room2001BezierPath,
kFloorPlanString : @"3001" },

                         @{ kFloorPlanBezierPath : room2002BezierPath,
kFloorPlanString : @"3002" },

                         @{ kFloorPlanBezierPath : room2003BezierPath,
kFloorPlanString : @"3003" },

                         @{ kFloorPlanBezierPath : room2004BezierPath,
kFloorPlanString : @"3004" },

                         @{ kFloorPlanBezierPath : room2005BezierPath,
kFloorPlanString : @"3005" },
                                    @{   kFloorPlanBezierPath   :   room2006BezierPath,
kFloorPlanString : @"3006" },
                                    @{   kFloorPlanBezierPath   :   room2007BezierPath,
kFloorPlanString : @"3007" },
                                    @{   kFloorPlanBezierPath   :   room2008BezierPath,
kFloorPlanString : @"3008" },
                                    @{   kFloorPlanBezierPath   :   room2009BezierPath,
kFloorPlanString : @"3009" },
                                    @{   kFloorPlanBezierPath   :   room2010BezierPath,
kFloorPlanString : @"3010" },
                                    @{   kFloorPlanBezierPath   :   room2011BezierPath,
kFloorPlanString : @"3011" },
                                    @{   kFloorPlanBezierPath   :   room2012BezierPath,
kFloorPlanString : @"3012" },
                                    @{   kFloorPlanBezierPath   :   room2014BezierPath,
kFloorPlanString : @"3014" },
                                    @{   kFloorPlanBezierPath   :   room2016BezierPath,
kFloorPlanString : @"3016" },
                                    @{   kFloorPlanBezierPath   :   room2018BezierPath,
kFloorPlanString : @"3018" },
                                    @{   kFloorPlanBezierPath   :   room2020BezierPath,
kFloorPlanString : @"3020" },
                                    @{   kFloorPlanBezierPath   :   room2022BezierPath,
kFloorPlanString : @"3022" },
                                    @{   kFloorPlanBezierPath   :   room2024BezierPath,
kFloorPlanString : @"3024" } ]];
```

```
            if ( self.showCoffee )

            {

                [floorPlanFeatures
addObjectsFromArray:@[  @{  kFloorPlanBezierPath  :  coffee3001BezierPath,
kFloorPlanString : @"Coffee" },

                            @{  kFloorPlanBezierPath  :  coffee3002BezierPath,
kFloorPlanString : @"Coffee" },

                            @{  kFloorPlanBezierPath  :  coffee3003BezierPath,
kFloorPlanString : @"Coffee" } ]];

            }

            break;

        }

        default:

            //  不支持的楼层

            break;

    }

    return floorPlanFeatures;

}

#pragma mark - Properties

// floop 属性设置器，保证楼层值在 [kMinimumFloor, kMaximumFloor]之间，并
为当前楼层重新设置无障碍元素。

- (void)setFloor:(NSInteger)floor

{

    //  将楼层限制在[kMinimumFloor, kMaximumFloor]中。

    _floor = MAX(MIN(floor, kMaximumFloor), kMinimumFloor);
```

```
        self.accessibilityElements = nil;

        [self setNeedsDisplay];

}

// showCoffee 设置器，如果状态未改变，重新设置无障碍元素。

//

- (void)setShowCoffee:(BOOL)showCoffee

{

        _showCoffee = showCoffee;

        self.accessibilityElements = nil;

        [self setNeedsDisplay];

}

@end
```

13. APLTitleView.h

文件：APLTitleView.h

简介：样例地图的标题视图。

版本：1.0

```
#import <UIKit/UIKit.h>

@interface APLTitleView : UIView

@property (nonatomic) NSInteger floor;

@end
```

14. APLTitleView.m

文件：APLTitleView.m

简介：样例地图的标题视图。

版本：1.0

```
#import "APLTitleView.h"

#import "APLCommon.h"

@interface APLTitleView ()

@property (weak, nonatomic) IBOutlet UILabel *titleLabel;

@end

@implementation APLTitleView

#pragma mark - UIView overrides

// 自定义绘制方法，绘制标识符。

//

- (void)drawRect:(CGRect)rect

{

    for ( NSInteger i = kMinimumFloor; i <= kMaximumFloor; ++i )

    {

        BOOL indicatorHighlighted = [self indicatorHighlightedForFloor:i];

        CGRect indicatorRect = [self indicatorRectForFloor:i];

        NSString *indicatorString = [NSString stringWithFormat:@"%ld",
(long)i];

        // 绘制标识符

        NSAttributedString *indicatorAttributedString = [[NSAttributedString
alloc] initWithString:indicatorString attributes:@{ NSFontAttributeName : [UIFont boldSystemFontOfSize:kFontSize], NSForegroundColorAttributeName : ( indicatorHighlighted ) ? [UIColor blackColor] : [UIColor lightGrayColor] }];

            UIBezierPath *indicatorBezierPath = [UIBezierPath bezierPathWithOvalInRect:indicatorRect];

            indicatorBezierPath.lineWidth = kLineWidth;

            [[UIColor whiteColor] setFill];

            [indicatorBezierPath fill];

            [( indicatorHighlighted ) ? [UIColor blackColor] : [UIColor lightGrayColor] setStroke];

            [indicatorBezierPath stroke];

            [indicatorAttributedString drawAtPoint:CGPointMake(CGRectGetMidX(indicatorRect) - indicatorAttributedString.size.width * 0.5, CGRectGetMidY(indicatorRect) - indicatorAttributedString.size.height * 0.5)];

      }

}

#pragma mark - UIAccessibilityElement

// 重写来编程式提供无障碍属性值。

// 这是提供动态无障碍属性的方法。

// 对于静态属性值，我们可以在中 Xcode (Interface Builder)设置。

- (BOOL)isAccessibilityElement

{

      return YES;

}
```

```
- (NSString *)accessibilityLabel

{

    return self.titleLabel.text;

}

//基于楼层提供无障碍值。

//

- (NSString *)accessibilityValue

{

    // 返回一个格式化数字字符串。

    NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];

    [numberFormatter setNumberStyle:NSNumberFormatterDecimalStyle];

    return [numberFormatter stringFromNumber:[NSNumber
numberWithInteger:self.floor]];

}

#pragma mark - Helpers

//为特定楼层返回标识符的突出状态。

//

- (BOOL)indicatorHighlightedForFloor:(NSInteger)floor

{

    return floor == self.floor;

}

//为特定楼层返回标识符矩形框。

//

- (CGRect)indicatorRectForFloor:(NSInteger)floor
```

```
{
    CGRect bounds = self.bounds;
    CGFloat indicatorWidth = CGRectGetHeight(bounds) - kPadding * 2.0;
    switch ( floor )
    {
        case 1:
            return CGRectMake(CGRectGetMidX(bounds) + CGRectGetWidth(bounds) * 0.25 - indicatorWidth * 1.5 - kPadding * 2.0, kPadding, indicatorWidth, indicatorWidth);
        case 2:
            return CGRectMake(CGRectGetMidX(bounds) + CGRectGetWidth(bounds) * 0.25 - indicatorWidth * 0.5, kPadding, indicatorWidth, indicatorWidth);
        case 3:
            return CGRectMake(CGRectGetMidX(bounds) + CGRectGetWidth(bounds) * 0.25 + indicatorWidth * 0.5 + kPadding * 2.0, kPadding, indicatorWidth, indicatorWidth);
        default:
            //不支持的楼层。
            return CGRectZero;
    }
}
#pragma mark - Properties
// floor 属性设置器，保证楼层值在[kMinimumFloor, kMaximumFloor]之间。
//
```

```
- (void)setFloor:(NSInteger)floor
{
        //  将 floor 限制在[kMinimumFloor, kMaximumFloor]中。
        _floor = MAX(MIN(floor, kMaximumFloor), kMinimumFloor);
        [self setNeedsDisplay];
}
@end
```

15. APLViewController.h

文件：APLViewController.h

简介：app 的主视图控制器。

版本：1.0

```
#import <UIKit/UIKit.h>

#import "APLCoffeeControl.h"

#import "APLElevatorControl.h"

@interface APLViewController : UIViewController

//  操作

- (IBAction)coffeeChanged:(APLCoffeeControl *)coffeeControl;

- (IBAction)floorChanged:(APLElevatorControl *)elevatorControl;

//  限制

- (void)setControlsEnabled:(BOOL)enabled;

@end
```

16. APLViewController.m

文件：APLViewController.m

摘要：app 的主视图控制器。

版本：1.0

```
#import "APLViewController.h"

#import "APLCommon.h"

#import "APLFloorPlanView.h"

#import "APLTitleView.h"

@interface APLViewController ()

@property (weak, nonatomic) IBOutlet APLCoffeeControl *coffeeControl;

@property (weak, nonatomic) IBOutlet APLElevatorControl *elevatorControl;

@property (weak, nonatomic) IBOutlet APLFloorPlanView *floorPlanView;

@property (weak, nonatomic) IBOutlet APLTitleView *titleView;

@end

@implementation APLViewController

#pragma mark - UIViewController overrides

- (void)viewDidLoad

{

    [super viewDidLoad];

    //  为 UI 元素设置初始状态

    self.coffeeControl.on = kShowCoffeeDefault;

    self.elevatorControl.floor = kFloorDefault;

    self.floorPlanView.floor = kFloorDefault;
    self.floorPlanView.showCoffee = kShowCoffeeDefault;
    self.titleView.floor = kFloorDefault;
}
#pragma mark - Restrictions
// 为引导访问设置限制。当引导访问限制状态改变(-(void)guidedAccessRestrictionWithIdentifier:didChangeState:)时，实现该限制。
//
- (void)setControlsEnabled:(BOOL)enabled
{
    self.coffeeControl.enabled = enabled;
    self.elevatorControl.enabled = enabled;
}
#pragma mark - Actions
- (IBAction)coffeeChanged:(APLCoffeeControl *)coffeeControl
{
    self.floorPlanView.showCoffee = coffeeControl.isOn;
}
- (IBAction)floorChanged:(APLElevatorControl *)elevatorControl
{
    self.floorPlanView.floor = elevatorControl.floor;
    self.titleView.floor = elevatorControl.floor;
}
@end
```

17. main.m

版本：main.m

摘要：标准主函数。

版本：1.0

```
#import <UIKit/UIKit.h>

#import "APLAppDelegate.h"

int main(int argc, char * argv[])

{

    @autoreleasepool {

        return UIApplicationMain(argc, argv, nil, NSStringFromClass([APLAppDelegate class]));

    }

}
```

18. 文档修订历史

日期	事项
2013-12-18	该样例展示了在一个 iOS 应用中怎样支持无障碍和引导访问限制。