高德地图 iOS SDK
开发指南

V2.4.1

高德软件有限公司
2014 年 12 月·北京

法律声明

高德地图

联系邮箱：api@autonavi.com

技术交流论坛：bbs.amap.com

官方微博：http://weibo.com/gaodedituapi

商务合作联系人：张先生　电话：010-84107170　电子邮箱： shiyue.zhang@autonavi.com

高德地图 API 欢迎用户的任何建议或意见。

目录

1 简介

1.1 什么是高德地图 iOS SDK？

高德地图 iOS SDK 是一套基于 iOS 5.1.1 及以上版本的地图应用程序开发接口。通过该接口，用户可使用高德地图数据和服务轻松构建功能丰富、交互性强的地图应用。

iOS SDK 提供的基础地图数据一共 17 个级别，包含建筑物、道路、医院、学校等信息。还支持使用 MATileOverlay 对基础地图数据附加额外的特性，制作自定义特色地图。

支持在地图上覆盖物来丰富地图显示，优化地图体验，覆盖物的种类丰富，包括：标注点、折线、多边形、圆、图片。从 2.3.0 开始，支持绘制带箭头、带纹理的矢量线，并且还能设置端点和连接点的类型，绘制效果更多元化。

支持双指缩放、双击方法、旋转等手势操作，并且有相应的接口开控制手势操作。

还提供了诸如 POI 搜索、路线规划、公交搜索以及坐标地址搜索等服务，用户可以根据自己的需要进行选择。

1.2 面向读者

此 SDK 是提供给具有一定 iOS 编程经验和了解面向对象概念的读者使用的。此外，读者还应该对地图产品有一定的了解。 用户在使用中遇到任何问题，都可以在 高德地图 API 开发者论坛 进行反馈。

1.3 兼容性

iOS SDK V2.3.0(含)之前版本适合 4.3 版本以上的 iOS 系统。

iOS SDK V2.4.0(含)之后版本适合 5.1.1 版本以上的 iOS 系统。

2 申请 Key

1. 访问 http://lbs.amap.com/console/key/。
2. 注册高德开发者账号，并认证成为开发者。（若已经是开发者，可直接到步骤 3）
3. 在“KEY 管理”页面点击上方的“获取 key”按钮，依次输入应用名，选择绑定的服务为“iOS 平台 SDK”，输入 Bundle Identifier（获取方法请参考：获取 Bundle Indentifier），如下图所示：

4. 点击下方的“获取 Key”按钮，在当前页面的“Key 列表”中可看到所申请的 Key，如下图所示：

3 地图显示

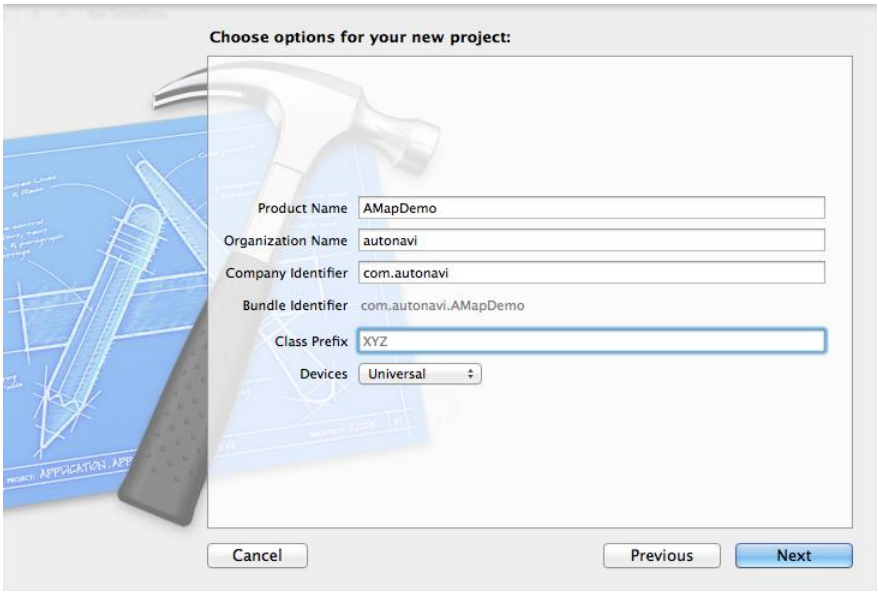
在“相关下载”页面中根据您的需求下载库文件并解压，包括：

- 3D 矢量地图库，点击下载。解压后得到 MAMapKit.framework 文件。
- 2D 栅格地图库，点击下载。解压后得到 MAMapKit.framework 文件。
- 搜索库，点击下载。解压后得到 AMapSearchKit.framework 文件。

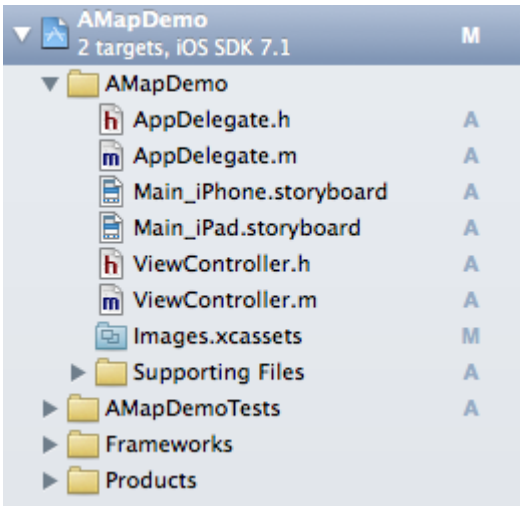
注意：3D 矢量地图和 2D 栅格地图只能选择一个使用，接口类似。

3.1 新建工程

新建一个 Single View Application 工程，如下图所示：



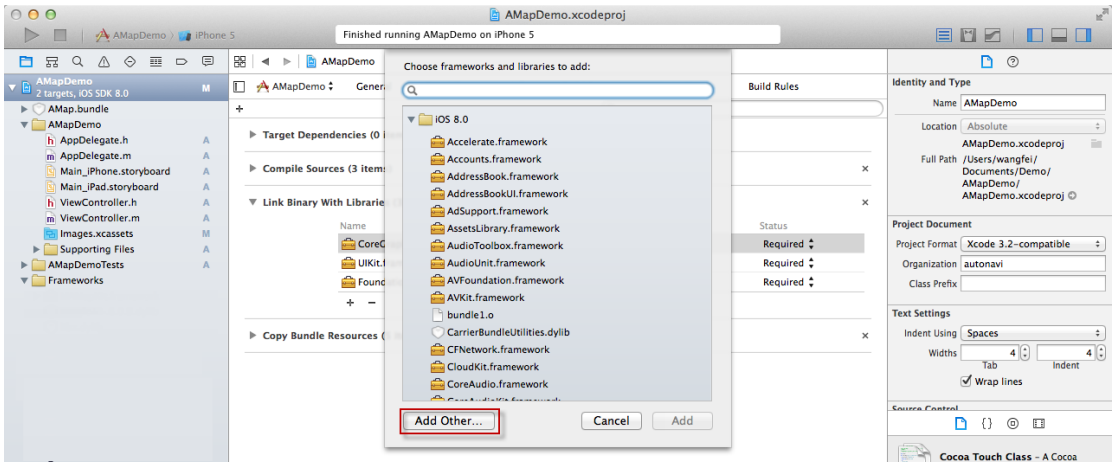
新建的工程如下图所示：



3.2 手动配置

3.2.1 引入地图库

左侧目录中选中工程名，在 TARGETS->Build Phases-> Link Binary With Libaries 中点击“+”按钮，在弹出的窗口中点击“Add Other”按钮，选择解压后的 MAMapKit.framework 文件添加到工程中。

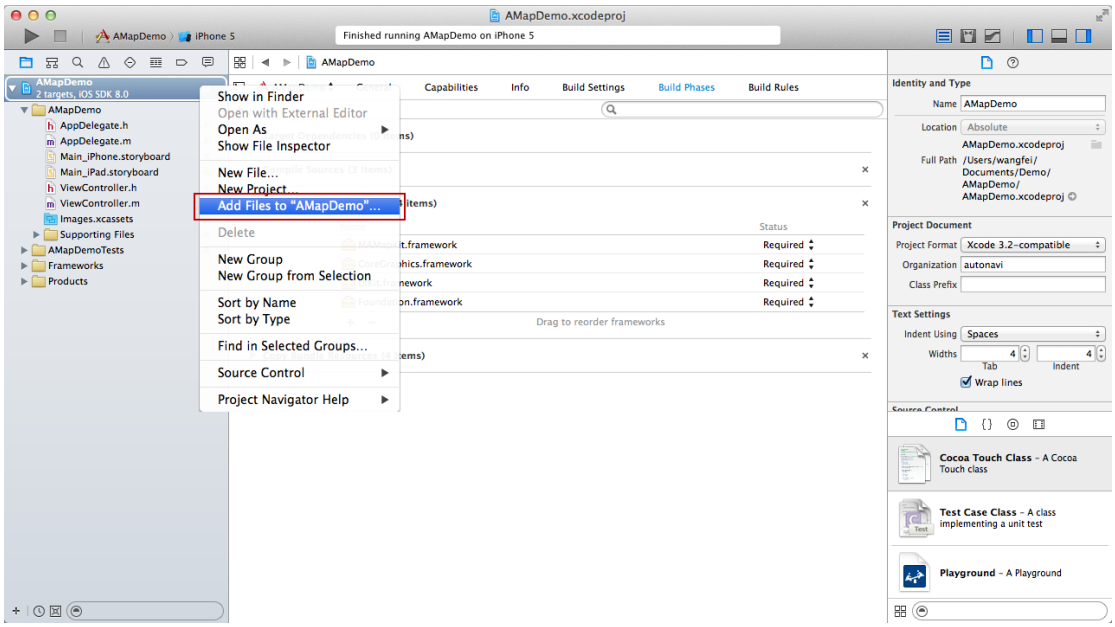


说明：搜索库 AMapSearchKit.framework 的引入方式同地图库。

3.2.2 引入 AMap.bundle 资源文件

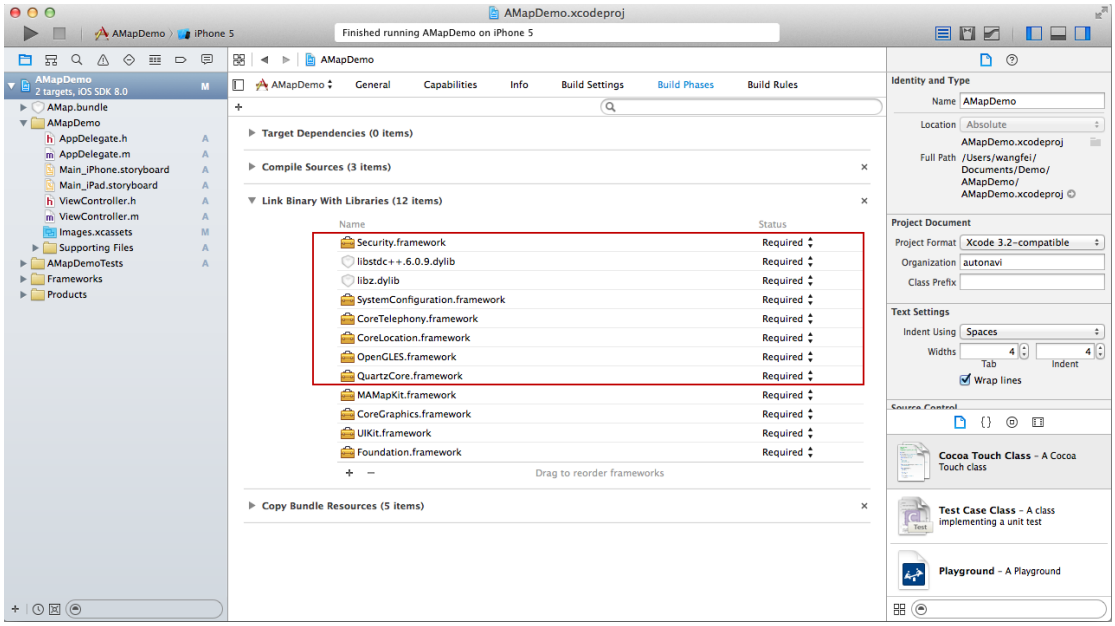
AMap.bundle 资源文件中存储了定位、默认大头针标注视图等图片，可利用这些资源图片进行开发。

左侧目录中选中工程名，在右键菜单中选择 Add Files to “工程名” ...，从 MAMapKit.framework->Resources 文件中选择 AMap.bundle 文件，并勾选“Copy items if needed”复选框，单击“Add”按钮，将资源文件添加到工程中。



3.2.3 引入系统库

左侧目录中选中工程名，在 TARGETS->Build Settings-> Link Binary With Libaries 中点击“+”按钮，在弹出的窗口中查找并选择所需的库（见下表），单击“Add”按钮，将库文件添加到工程中。

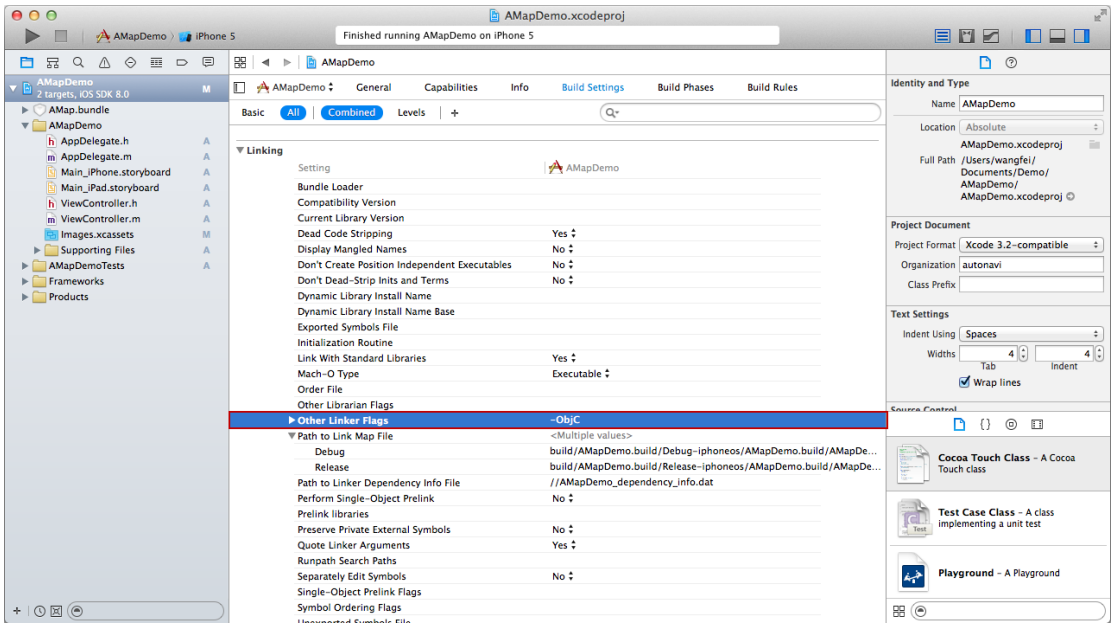


序号	库名称	备注
1	UIKit.framework	2D、3D、Search
2	Foundation.framework	2D、3D、Search
3	CoreGraphics.framework	2D、3D、Search
4	QuartzCore.framework	2D、3D
5	OpenGLES.framework	3D
6	CoreLocation.framework	2D、3D
7	CoreTelephony.framework	2D、3D、Search
8	SystemConfiguration.framework	2D、3D、Search
9	libz.dylib	2D、3D、Search
10	libstdc++6.09.dylib	2D、3D、Search
11	Security.framework	2D、3D
12	AdSupport.framework	3D

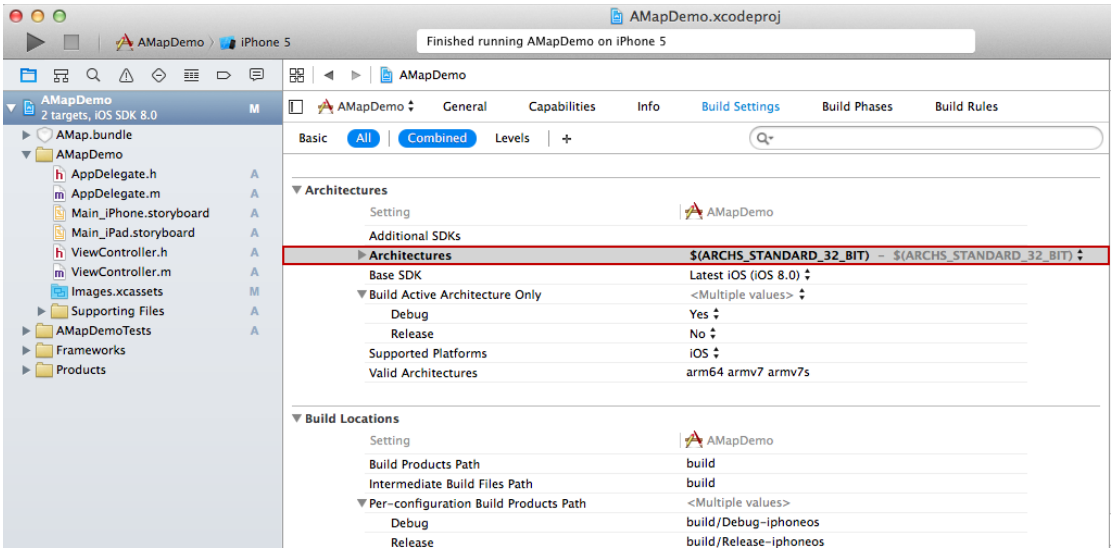
备注中：2D 表示使用 2D 栅格地图需要的系统文件，3D 表示使用 3D 矢量地图需要的系统文件、Search 表示使用搜索库需要的系统文件。

3.2.4 环境配置

在 TARGETS->Build Settings->Other Linker Flags 中添加-ObjC。



注意：V2.3.0（含）之前版本不支持 arm64，还需在 TARGETS->Build Settings->Architectures 点出选择框，选择 “Other”，将默认值修改为 $(ARCHS_STANDARD_32_BIT)。



3.3 自动配置

这里介绍一种自动部署 iOS SDK 的方法，省去你配置工程的时间，更高效的完成您的应用。

- 对于经熟悉 CocoaPods 的同学，使用如下命令：

```
pod 'AMap3DMap' #3D 地图 SDK
#pod 'AMap3DMap' #2D 地图 SDK(2D 地图和 3D 地图不能同时使用)
pod 'AMapSearch' #搜索服务 SDK
```

- 对初次使用 CocoaPods 的同学：

（1） 安装 CocoaPods（已安装请跳过）

在终端输入

```
sudo gem install cocoapods
```

如果安装成功，会有一个提示

```
Successfully installed cocoaPods
```

若很久没反应，则是因为安装被墙阻拦

1. 解决方法 1：打开 vpn 下载
2. 解决方法 2：请看详细指 http://code4app.com/article/cocoapods-install-usage

（2） 搜索高德 API 库

安装成功后，在终端输入以下命令：

3D 地图 SDK

```
pod search AMap3DMap
```

2D 地图 SDK

```
pod search AMap2DMap
```

搜索 SDK

```
pod search AMapSearch
```

若无返回结果，则先运行以下命令，再进行搜索

```
pod repo update
```

（3） 使用 cocoapod

在当前工程文件（.xcodeproj）所在文件夹下，打开 terminal

1. 创建 Podfile：

```
touch Podfile
```

2. 编辑 Podfile 内容，如下：

```
platform :ios, '7.0' #手机的系统
pod 'AMap3DMap'    #3D 地图 SDK
#pod 'AMap2DMap' #2D 地图 SDK (2D 和 3D 不能同时使用用)
pod 'AMapSearch' #搜索服务 SDK
```

3. 在 PodFile 所在的文件夹下输入命令：

```
pod install
```

若已经 install 过，使用以下命令更新版本。

```
pod update
```

成功以后，会出现如下记录：

```
localhost:yourWorkDir yourUserName$ pod install
```

```
Analyzing dependencies
Downloading dependencies
Installing AMap3DMap(2.4.0)
Installing AMapSearch (2.4.0)
Generating Pods project
Integrating client project !
[!] From now on use `yourProj.xcworkspace`.
```

打开 xcworkspace 文件，就可以开始开发您的应用了。

3.4 显示地图

3.4.1 配置用户 Key

在使用地图 SDK 时，需要对应用做 Key 机制验证，在地图初始化之前添加如下示例代码，配置之前在官网上申请的 Key：

```
[MAMapServices sharedServices].apiKey = @"用户 Key";
```

3.4.2 地图显示

（1） 修改 ViewController.m 文件，引入 MAMapKit.h 文件，继承 MAMapViewDelegate 协议，并定义 MAMapView 对象，示例代码如下所示：

```
#import <ViewController.m>
#import <MAMapKit/MAMapKit.h>

@interface ViewController ()<MAMapViewDelegate>
{
    MAMapView *_mapView;
}
@end
```

（2） 在 ViewController.m 文件相应的方法中进行地图初始化，初始化的步骤：

1. 构造 MAMapView 对象；
2. 设置代理；
3. 将 MAMapView 添加到 Subview 中。

- 对于 3D 矢量地图，在 viewDidLoad 方法中添加代码：

```
-(void) viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    //配置用户 Key
    [MAMapServices sharedServices].apiKey = @"用户 Key";

    _mapView = [[MAMapView alloc] initWithFrame:CGRectMake(0, 0,
CGRectGetWidth(self.view.bounds), CGRectGetHeight(self.view.bounds))];
    _mapView.delegate = self;
```

```
    [self.view addSubview:_mapView];
}
```

- 对于 2D 栅格地图，在 viewDidAppear 方法中添加代码：

```
-(void) viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    // Do any additional setup after loading the view, typically from a nib.
    //配置用户 Key
    [MAMapServices sharedServices].apiKey = @"用户 Key";

    _mapView = [[MAMapView alloc] initWithFrame:CGRectMake(0, 0,
CGRectGetWidth(self.view.bounds), CGRectGetHeight(self.view.bounds))];
    _mapView.delegate = self;

    [self.view addSubview:_mapView];
}
```

编译，运行工程，效果如下图所示：

4 地图图层

地图是由图层组成的，地图可以包括一个或多个图层，每个图层在每个级别都是由若干张图块组成。例如，您所看到的包括街道、兴趣点、学校、公园等内容的地图展现就是一个图层，另外实时交通的展现也是通过图层来实现的。

iOS SDK 支持多种地图图层的展现，包括矢量或栅格地图、卫星图层、实时交通图层等。

说明：以下示例代码都基于“显示地图”中初始化并添加到 Subview 中的_mapView 对象。

4.1 基本地图

高德地图 iOS SDK 为 3D 矢量地图 SDK 提供三种地图类型 MAMapTypeStandard、MAMapTypeSatellite 和 MAMapTypeStandardNight；为 2D 栅格地图 SDK 提供两种地图类型 MAMapTypeStandard 和 MAMapTypeSatellite。其中：MAMapTypeStandard 为标准地图，MAMapTypeSatellite 为卫星地图，MAMapTypeStandardNight 为夜景地图。

地图默认显示标准地图，从标准地图切换成卫星地图的方法如下：

```
//显示卫星地图
_mapView.mapType = MAMapTypeSatellite;
```

运行程序效果如下：

说明：地图类型切换只需将地图类型设置成相应的类型（MAMapTypeStandard/MAMapTypeSatellite/MAMapTypeStandardNight）。

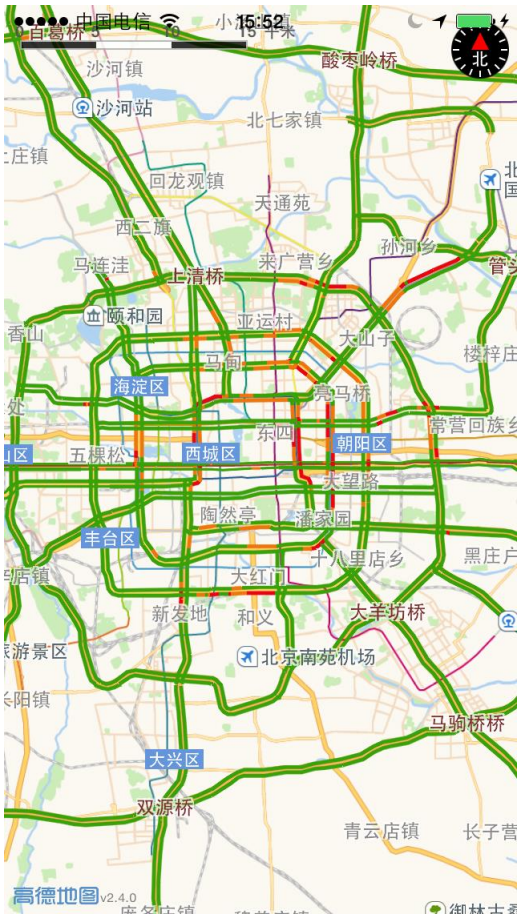
4.2 实时路况图

高德地图 iOS SDK 提供北京，上海，广州，深圳，武汉，沈阳，南京，宁波，重庆，杭州，青岛，成都，天津，大连，无锡，西安，石家庄，太原，常州，厦门，长春，福州，珠海，东莞，长沙，苏州，金华，佛山，济南，泉州，西宁，乌鲁木齐，嘉兴，香港，鄂尔多斯，南通，中山，惠州，镇江，郑州，合肥，昆明，德州，朝阳，抚顺，大同，荆州，温州，台州，绍兴，莆田，南平，漳州，宁德，三明，龙岩，烟台，阳江，江门，保定，临沂 61 个城市及城际间的实时交通路况。

显示实时交通路况的代码如下：

```
_mapView.showTraffic= YES;
```

运行后效果如下：



关闭实时路况的代码如下：

```
self.mapView.showTraffic= NO;
```

4.3 自定义图层

通过自定义图层可对基础底层地图添加额外的特性，如：某个商场的室内信息、某个景区的详情等等。自定义图层类是 MATileOverlay，它定义了能添加到基础底层地图的图片集合。

添加自定义图层的前提是使用球面墨卡托投影生成了相应的瓦片，并按照生成的格式部署在服务器上。在地图上显示自定义图层的步骤如下：

1. 根据 URL 模版（即指向相关图层图片的 URL）创建 MATileOverlay 对象。
2. 设置 MATileOverlay 的可见最大/最小 Zoom 值。
3. 设定 MATileOverlay 的可渲染区域。
4. 将 MATileOverlay 对象添加到 MAMapView 中。

我们提前准备了西单大悦城的 1-3 层楼的瓦片，并部署到了服务器上。在地图上显示朝阳大悦城 2 层瓦片的示例代码如下：

（1） 修改 ViewController.m 文件，在 viewDidLoad 方法中构造 2 层的对应的 MATileOverlay 对象，并添加到地图上。

```
#define TileOverlayViewControllerCoordinate CLLocationCoordinate2DMake(39.910695,
116.372830)

- (MATileOverlay *)constructTileOverlayWithFloor:(NSInteger)floor
{
    //
    /* 构建 tileOverlay 的 URL 模版. */
    NSString *URLTemplate = [NSString stringWithFormat:
@"http://sdkdemo.amap.com:8080/tileserver/Tile?x={x}&y={y}&z={z}&f=%ld", (long)floor];

    MATileOverlay *tileOverlay = [[MATileOverlay alloc] initWithURLTemplate:URLTemplate];

    tileOverlay.minimumZ = 18; //设置可见最小 Zoom 值
    tileOverlay.maximumZ = 20; //设置可见最大 Zoom 值

    tileOverlay.boundingMapRect =
MAMapRectForCoordinateRegion(MACoordinateRegionMakeWithDistance(TileOverlayViewC
ontrollerCoordinate, 200, 200)); //设置可渲染区域

    return tileOverlay;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.mapView.centerCoordinate    = TileOverlayViewControllerCoordinate;
    self.mapView.zoomLevel           = 19;

    [self.mapView addOverlay: [self constructTileOverlayWithFloor:2]];
}
```

（2） 实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，在瓦片显示在地图 View 上。示例代码如下：

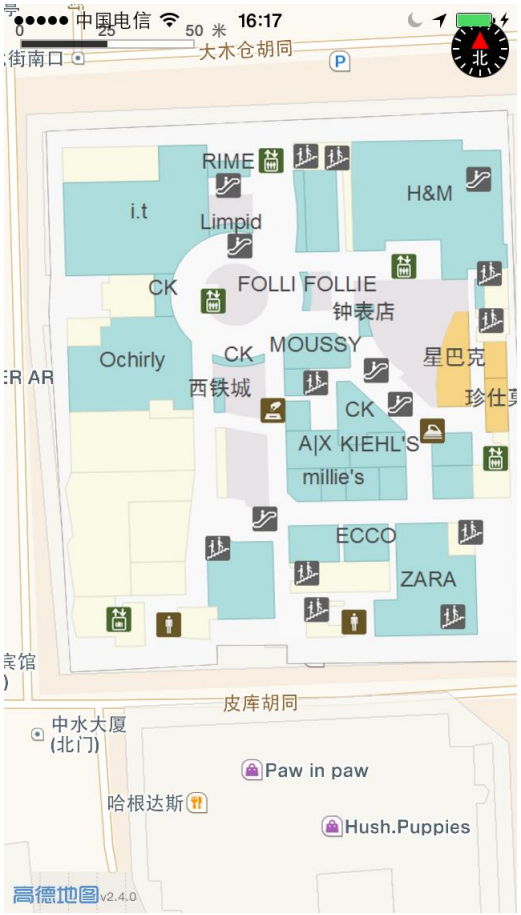
```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id
```

```
<MAOverlay>)overlay
{
    if ([overlay isKindOfClass:[MATileOverlay class]])
    {
        MATileOverlayView *tileOverlayView = [[MATileOverlayView alloc]
initWithTileOverlay:overlay];

        return tileOverlayView;
    }

    return nil;
}
```

运行效果如下图所示：



5 地图覆盖物

高德地图 iOS SDK 支持的地图覆盖物有：标注、折线、多边形、圆、大地曲线和图片覆盖物，您可以在地图上添加这些覆盖物来丰富地图显示，通过设置其属性实现各种功能，优化地图体验。

说明：以下示例代码都基于“显示地图”中初始化并添加到 Subview 中的_mapView 对象。

5.1 标注

iOS SDK 提供标注点的协议<MAAnnotation>,它包含一个标注的基本信息：标注 View 的中心点坐标、标题和副标题。同时，还封装了一个标注类 MAPointAnnotation，它定义了一个位于指定位置的标注数据对象。

MAAnnotationView 是标注对应的 View，它用于在地图显示标记。

5.1.1 大头针标注

iOS SDK 提供一个默认的大头针标注 View——MAPinAnnotationView，通过它可设置大头针标注是否可被拾起拖拽、是否以动画效果显示等等。

（1） 修改ViewController.m文件 ,在viewDidAppear方法中添加如下所示代码添加标注数据对象。

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    MAPointAnnotation *pointAnnotation = [[MAPointAnnotation alloc] init];
    pointAnnotation.coordinate = CLLocationCoordinate2DMake(39.989631, 116.481018);
    pointAnnotation.title = @"方恒国际";
    pointAnnotation.subtitle = @"阜通东大街 6 号";

    [_mapView addAnnotation:pointAnnotation];
}
```

（2） 实现 MAMapViewDelegate 的 mapView:viewForAnnotation:函数，设置标注样式。

```
- (MAAnnotationView *)mapView:(MAMapView *)mapView
viewForAnnotation:(id<MAAnnotation>)annotation

{
    if ([annotation isKindOfClass:[MAPointAnnotation class]])
    {
        static NSString *pointReuseIndetifier = @"pointReuseIndetifier";
        MAPinAnnotationView*annotationView = (MAPinAnnotationView*)[mapView
dequeueReusableAnnotationViewWithIdentifier:pointReuseIndetifier];
        if (annotationView == nil)
        {
            annotationView = [[MAPinAnnotationView alloc] initWithAnnotation:annotation
reuseIdentifier:pointReuseIndetifier];
        }
        annotationView.canShowCallout= YES;          //设置气泡可以弹出，默认为 NO
        annotationView.animatesDrop = YES;           //设置标注动画显示，默认为 NO
```

```
        annotationView.draggable = YES;          //设置标注可以拖动，默认为 NO
        annotationView.pinColor = MAPinAnnotationColorPurple;
        return annotationView;
    }
    return nil;
}
```

运行程序，在地图显示对应的标注点，点击标注弹出气泡，效果如图：

5.1.2 自定义标注

iOS SDK 可自定义标注的图标和弹出气泡的样式，均可通过 MAAnnotationView 来实现。

5.1.2.1 自定义标注图标

若大头针样式的标注不能满足您的需求，您可以自定义标注图标。

（1） 添加标注数据对象，可参考大头针标注的步骤（1）。

（2） 导入标记图片文件到工程中。这里我们导入一个名为 restauant.png 的图片文件 

（3） 在 MAMapViewDelegate 回调函数 mapView:viewForAnnotation:中修改 MAAnnotationView 对应的标注图片。示例代码如下：

```
- (MAAnnotationView *)mapView:(MAMapView *)mapView
```

```
viewForAnnotation:(id<MAAnnotation>)annotation
{
    if ([annotation isKindOfClass:[MAPointAnnotation class]])
    {
        static NSString *reuseIndetifier = @"annotationReuseIndetifier";
        MAAnnotationView *annotationView = (MAAnnotationView *)[mapView
dequeueReusableAnnotationViewWithIdentifier:reuseIndetifier];
        if (annotationView == nil)
        {
            annotationView = [[MAAnnotationView alloc] initWithAnnotation:annotation
reuseIdentifier:reuseIndetifier];
        }
        annotationView.image = [UIImage imageNamed:@"restaurant"];
        //设置中心心点偏移，使得标注底部中间点成为经纬度对应点
        annotationView.centerOffset = CGPointMake(0, -18);
        return annotationView;
    }
    return nil;
}
```

运行程序，标注点变成了餐馆图标，如下所示：

5.1.2.2 自定义气泡

气泡在 iOS 中又称为 callout，它由背景和气泡内容构成，如下图所示：

每个气泡显示的内容是根据您的需求定义的，这里我们按照如上图所示的气泡介绍实现一个自定义气泡的步骤。

（1） 新建自定义气泡类 CustomCalloutView，继承 UIView。

（2） 在 CustomCalloutView.h 中定义数据属性，包含：图片、商户名和商户地址。

```
@interface CustomCalloutView : UIView

@property (nonatomic, strong) UIImage *image; //商户图
@property (nonatomic, copy) NSString *title; //商户名
@property (nonatomic, copy) NSString *subtitle; //地址

@end
```

（3） 在 CustomCalloutView.m 中重写 UIView 的 drawRect 方法，绘制弹出气泡的背景。

```
#define kArrorHeight            10

- (void)drawRect:(CGRect)rect
{

    [self drawInContext:UIGraphicsGetCurrentContext()];

    self.layer.shadowColor = [[UIColor blackColor] CGColor];
    self.layer.shadowOpacity = 1.0;
    self.layer.shadowOffset = CGSizeMake(0.0f, 0.0f);

}

- (void)drawInContext:(CGContextRef)context
{

    CGContextSetLineWidth(context, 2.0);
    CGContextSetFillColorWithColor(context, [UIColor colorWithRed:0.3 green:0.3 blue:0.3
alpha:0.8].CGColor);

    [self getDrawPath:context];
    CGContextFillPath(context);

}

- (void)getDrawPath:(CGContextRef)context
{
    CGRect rrect = self.bounds;
```

```
    CGFloat radius = 6.0;
    CGFloat minx = CGRectGetMinX(rrect),
    midx = CGRectGetMidX(rrect),
    maxx = CGRectGetMaxX(rrect);
    CGFloat miny = CGRectGetMinY(rrect),
    maxy = CGRectGetMaxY(rrect)-kArrorHeight;

    CGContextMoveToPoint(context, midx+kArrorHeight, maxy);
    CGContextAddLineToPoint(context,midx, maxy+kArrorHeight);
    CGContextAddLineToPoint(context,midx-kArrorHeight, maxy);

    CGContextAddArcToPoint(context, minx, maxy, minx, miny, radius);
    CGContextAddArcToPoint(context, minx, minx, maxx, miny, radius);
    CGContextAddArcToPoint(context, maxx, miny, maxx, maxx, radius);
    CGContextAddArcToPoint(context, maxx, maxy, midx, maxy, radius);
    CGContextClosePath(context);
}
```

（4） 定义用于显示气泡内容的控件，并添加到 SubView 中。

如上图所示气泡，我们需要一个 UIImageView 和两个 UILabel，添加方法如下：

```
#define kPortraitMargin     5
#define kPortraitWidth      70
#define kPortraitHeight     50

#define kTitleWidth         120
#define kTitleHeight        20

@interface CustomCalloutView ()

@property (nonatomic, strong) UIImageView *portraitView;
@property (nonatomic, strong) UILabel *subtitleLabel;
@property (nonatomic, strong) UILabel *titleLabel;

@end

@implementation CustomCalloutView

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self)
    {
        self.backgroundColor = [UIColor clearColor];
        [self initSubViews];
    }
    return self;
}

- (void)initSubViews
{
    // 添加图片，即商户图
    self.portraitView = [[UIImageView alloc] initWithFrame:CGRectMake(kPortraitMargin,
kPortraitMargin, kPortraitWidth, kPortraitHeight)];

    self.portraitView.backgroundColor = [UIColor blackColor];
```

```
        [self addSubview:self.portraitView];

        // 添加标题，即商户名
        self.titleLabel = [[UILabel alloc] initWithFrame:CGRectMake(kPortraitMargin * 2 +
kPortraitWidth, kPortraitMargin, kTitleWidth, kTitleHeight)];
        self.titleLabel.font = [UIFont boldSystemFontOfSize:14];
        self.titleLabel.textColor = [UIColor whiteColor];
        self.titleLabel.text = @"titletitletitletitle";
        [self addSubview:self.titleLabel];

        // 添加副标题，即商户地址
        self.subtitleLabel = [[UILabel alloc] initWithFrame:CGRectMake(kPortraitMargin * 2 +
kPortraitWidth, kPortraitMargin * 2 + kTitleHeight, kTitleWidth, kTitleHeight)];
        self.subtitleLabel.font = [UIFont systemFontOfSize:12];
        self.subtitleLabel.textColor = [UIColor lightGrayColor];
        self.subtitleLabel.text = @"subtitleLabelsubtitleLabelsubtitleLabel";
        [self addSubview:self.subtitleLabel];
    }
```

（5） 在 CustomCalloutView.m 中给控件传入数据。

```
- (void)setTitle:(NSString *)title
{
    self.titleLabel.text = title;
}

- (void)setSubtitle:(NSString *)subtitle
{
    self.subtitleLabel.text = subtitle;
}

- (void)setImage:(UIImage *)image
{
    self.portraitView.image = image;
}
```

以上就是自定义气泡的全部过程，但是为了在点击标注时，弹出自定义的气泡，还需要自定义 AnnotationView。步骤如下：

（1） 新建类 CustomAnnotationView，继承 MAAnnotationView 或 MAPinAnnotationView。若继承 MAAnnotationView，则需要设置标注图标；若继承 MAPinAnnotationView，使用默认的大头针标注。

（2） 在 CustomAnnotationView.h 中定义自定义气泡属性，代码如下所示：

```
#import "CustomCalloutView.h"

@interface CustomAnnotationView : MAAnnotationView

@property (nonatomic, readonly) CustomCalloutView *calloutView;

@end
```

（3） 在 CustomAnnotationView.m 中修改 calloutView 属性，如下：

```
@interface CustomAnnotationView ()

@property (nonatomic, strong, readwrite) CustomCalloutView *calloutView;
```

```
@end
```

（4） 重写选中方法 setSelected。选中时新建并添加 calloutView ,传入数据 ;非选中时删除 calloutView。

```
#define kCalloutWidth       200.0
#define kCalloutHeight      70.0

- (void)setSelected:(BOOL)selected animated:(BOOL)animated
{
    if (self.selected == selected)
    {
        return;
    }

    if (selected)
    {
        if (self.calloutView == nil)
        {
            self.calloutView = [[CustomCalloutView alloc] initWithFrame:CGRectMake(0, 0,
kCalloutWidth, kCalloutHeight)];
            self.calloutView.center = CGPointMake(CGRectGetWidth(self.bounds) / 2.f +
self.calloutOffset.x,

-CGRectGetHeight(self.calloutView.bounds) / 2.f + self.calloutOffset.y);
        }

        self.calloutView.image = [UIImage imageNamed:@"building"];
        self.calloutView.title = self.annotation.title;
        self.calloutView.subtitle = self.annotation.subtitle;

        [self addSubview:self.calloutView];
    }
    else
    {
        [self.calloutView removeFromSuperview];
    }

    [super setSelected:selected animated:animated];
}
```

注意：提前导入 building.png 图片。

（5） 修改 ViewController.m，在 MAMapViewDelegate 的回调方法 mapView:viewForAnnotation 中的修改 annotationView 的类型，代码如下：

```
#import “CustomAnnotationView.h”

- (MAAnnotationView *)mapView:(MAMapView *)mapView
viewForAnnotation:(id<MAAnnotation>)annotation
{
    if ([annotation isKindOfClass:[MAPointAnnotation class]])
    {
        static NSString *reuseIndetifier = @"annotationReuseIndetifier";
        CustomAnnotationView *annotationView = (CustomAnnotationView *)[mapView
dequeueReusableAnnotationViewWithIdentifier:reuseIndetifier];
        if (annotationView == nil)
        {
            annotationView = [[CustomAnnotationView alloc] initWithAnnotation:annotation
```

```
reuseIdentifier:reuseIndetifier];
            }
            annotationView.image = [UIImage imageNamed:@"restaurant"];

            // 设置为 NO，用以调用自定义的 calloutView
            annotationView.canShowCallout = NO;

            // 设置中心点偏移，使得标注底部中间点成为经纬度对应点
            annotationView.centerOffset = CGPointMake(0, -18);
            return annotationView;
        }
        return nil;
}
```

运行程序，效果如下：

5.2 折线

折线类为 MAPolyline，由一组经纬度坐标组成，并以有序序列形式建立一系列的线段。iOS SDK 支持在 3D 矢量地图上绘制带箭头或有纹理等样式的折线。

在地图添加折线的步骤如下：

（1） 修改 ViewController.m 文件，在 viewDidLoad 方法中构造折线数据对象（一组经纬度坐标点）。

```
-(void) viewDidLoad
{
```

```
    //构造折线数据对象
    CLLocationCoordinate2D commonPolylineCoords[4];
    commonPolylineCoords[0].latitude = 39.832136;
    commonPolylineCoords[0].longitude = 116.34095;

    commonPolylineCoords[1].latitude = 39.832136;
    commonPolylineCoords[1].longitude = 116.42095;

    commonPolylineCoords[2].latitude = 39.902136;
    commonPolylineCoords[2].longitude = 116.42095;

    commonPolylineCoords[3].latitude = 39.902136;
    commonPolylineCoords[3].longitude = 116.44095;

    //构造折线对象
    MAPolyline *commonPolyline = [MAPolyline
polylineWithCoordinates:commonPolylineCoords count:4];

    //在地图上添加折线对象
    [_mapView addOverlay: commonPolyline];
}
```

（2） 继续在 ViewController.m 文件中，实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，设置折线的样式。示例代码如下：

```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id
<MAOverlay>)overlay
{
    if ([overlay isKindOfClass:[MAPolyline class]])
    {
        MAPolylineView *polylineView = [[MAPolylineView alloc] initWithPolyline:overlay];

        polylineView.lineWidth = 10.f;
        polylineView.strokeColor = [UIColor colorWithRed:0 green:0 blue:1 alpha:0.6];
        polylineView.lineJoinType = kMALineJoinRound;//连接类型
        polylineView.lineCapType = kMALineCapRound;//端点类型

        return polylineView;
    }
    return nil;
}
```

运行程序，效果如下所示：

在该回调函数中，调用 MAPolylineView 的 loadStrokeTextureImage 方法可以设置折线的纹理图片。

注意：若设置了纹理图片，设置线颜色、连接类型和端点类型将无效。

设置纹理图片的代码如下：

```
[polylineView loadStrokeTextureImage:[UIImage imageNamed:@"arrowTexture"]];
```

5.3 多边形

多边形类为 MAPolygon，与 MAPolyline 类似，包括有序序列的一系列坐标，但是多边形包含有内部区域。

在地图添加多边形的步骤如下：

（1） 修改 ViewController.m 文件，在 viewDidLoad 方法中构造多边形的数据对象（一组经纬度坐标点）。

```
-(void) viewDidLoad
{
    //构造多边形数据对象
    CLLocationCoordinate2D coordinates[4];
    coordinates[0].latitude = 39.810892;
    coordinates[0].longitude = 116.233413;

    coordinates[1].latitude = 39.816600;
    coordinates[1].longitude = 116.331842;
```

```
    coordinates[2].latitude = 39.762187;
    coordinates[2].longitude = 116.357932;

    coordinates[3].latitude = 39.733653;
    coordinates[3].longitude = 116.278255;

    MAPolygon *polygon = [MAPolygon polygonWithCoordinates:coordinates count:4];

    //在地图上添加折线对象
    [_mapView addOverlay: polygon];
}
```

（2） 继续在 ViewController.m 文件中，实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，设置多边形的样式。示例代码如下：

```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id
<MAOverlay>)overlay
{
    if ([overlay isKindOfClass:[MAPolygon class]])
    {
        MAPolygonView *polygonView = [[MAPolygonView alloc] initWithPolygon:overlay];

        polygonView.lineWidth = 5.f;
        polygonView.strokeColor = [UIColor colorWithRed:0.6 green:0.6 blue:0.6 alpha:0.8];
        polygonView.fillColor = [UIColor colorWithRed:0.77 green:0.88 blue:0.94 alpha:0.8];
        polygonView.lineJoinType = kMALineJoinMiter;//连接类型

        return polygonView;
    }
    return nil;
}
```

运行程序，效果如下所示：

5.4 圆

圆类为 MACircle，圆对象由中心点（经纬度）和半径（米）构成。

在地图绘制圆的步骤如下：

（1） 在 ViewController.m 的 viewDidLoad 方法中根据中心点和半径构造圆对象。

```
-(void) viewDidLoad
{
    //构造圆
    MACircle *circle = [MACircle
circleWithCenterCoordinate:CLLocationCoordinate2DMake(39.952136, 116.50095)
radius:5000];

    //在地图上添加圆
    [_mapView addOverlay: circle];
}
```

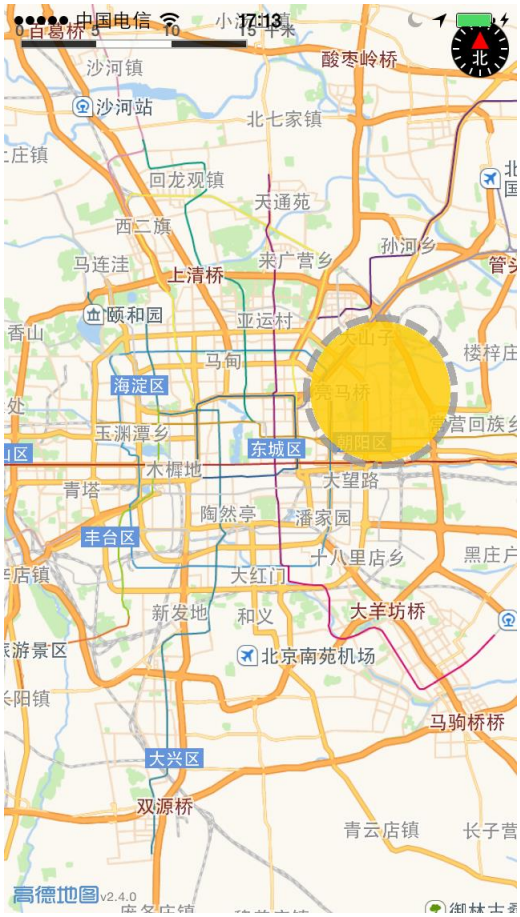
（2） 继续在 ViewController.m 文件中，实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，设置圆的样式。示例代码如下：

```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id
<MAOverlay>)overlay
{
    if ([overlay isKindOfClass:[MACircle class]])
    {
        MACircleView *circleView = [[MACircleView alloc] initWithCircle:overlay];
```

```
        circleView.lineWidth = 5.f;
        circleView.strokeColor = [UIColor colorWithRed:0.6 green:0.6 blue:0.6 alpha:0.8];
        circleView.fillColor = [UIColor colorWithRed:1.0 green:0.8 blue:0.0 alpha:0.8];
        circleView.lineDash = YES;

        return circleView;
    }
    return nil;
}
```

运行程序，效果如下所示：



5.5 大地曲线

大地曲线类为 MAGeodesicPolyline，继承自 MAPolyline。

在地图添加大地曲线的步骤如下：

（1） 修改 ViewController.m 文件，在 viewDidLoad 方法中构造折大地曲线数据对象（一组经纬度坐标点或地图投影点），下面以经纬度点为例：

```
- (void) viewDidLoad
{
    CLLocationCoordinate2D geodesicCoords[2];
    geodesicCoords[0].latitude = 39.905151;
    geodesicCoords[0].longitude = 116.401726;
```

```
    geodesicCoords[1].latitude = 38.905151;
    geodesicCoords[1].longitude = 70.401726;

    //构造大地曲线对象
    MAGeodesicPolyline *geodesicPolyline = [MAGeodesicPolyline
polylineWithCoordinates:geodesicCoords count:2];

    [_mapView addOverlay:geodesicPolyline];
}
```

（2） 在 ViewController.m 文件中，实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，设置曲线的样式。示例代码如下：

```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id
<MAOverlay>)overlay
{
    if ([overlay isKindOfClass:[MAPolyline class]])
    {
        MAPolylineView *polylineView = [[MAPolylineView alloc] initWithPolyline:overlay];

        polylineView.lineWidth = 8.f;
        polylineView.strokeColor = [UIColor colorWithRed:0 green:0 blue:1 alpha:0.8];

        return polylineView;
    }
    return nil;
}
```

运行程序，效果如下所示：

5.6 图片覆盖物

图片覆盖物类为 MAGroundOverlay ,可完成将一张图片以合适的大小贴在地图指定的位置上的功能。

绘制图片覆盖物的步骤如下：

（1） 在 ViewController.m 的 viewDidLoad 方法中根据范围和图片构造图片覆盖物。

```
- (void) viewDidLoad
{
    MACoordinateBounds coordinateBounds = MACoordinateBoundsMake(CLLocationCoordinate2DMake(39.939577, 116.388331),CLLocationCoordinate2DMake(39.935029, 116.384377));

    MAGroundOverlay *groundOverlay = [MAGroundOverlay groundOverlayWithBounds:coordinateBounds icon:[UIImage imageNamed:@"GWF"]];

    [_mapView addOverlay:groundOverlay];
    _mapView.visibleMapRect = groundOverlay.boundingMapRect;
}
```

（2） 实现 MAMapViewDelegate 的 mapView:viewForOverlay:函数，以在地图上显示图片覆盖物。

```
- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id <MAOverlay>)overlay{
    if ([overlay isKindOfClass:[MAGroundOverlay class]])
    {
        MAGroundOverlayView *groundOverlayView = [[MAGroundOverlayView alloc]
```

```
            initWithGroundOverlay:overlay];

            return groundOverlayView;
        }
        return nil;
}
```

运行效果如下所示：

6 地图控件

地图控件可帮您直观的了解当前的地图的状态，iOS SDK 提供“地图 Logo”、“指南针”和“比例尺”三种地图控件。

说明：以下示例代码都基于“显示地图”中初始化并添加到 Subview 中的_mapView 对象。

6.1 地图 Logo

iOS SDK 默认的 Logo 为“高德地图 v2.x.x”字样，显示在地图的左下方。地图 Logo 不能移除，但可通过 MAMapView.logoCenter 属性来调整 Logo 的显示位置。在 ViewController.m 的 viewDidLoad 方法添加如下如下：

```
_mapView.logoCenter = CGPointMake(CGRectGetWidth(self.view.bounds)-55, 450);
```

6.2 指南针

指南针默认是开启状态，显示在地图的右上角。

通过 MAMapView 的 showsCompass 属性用来控制指南针的可见性。compassOrigin 属性可改变指南针的显示位置。在 ViewController.m 的 viewDidLoad 方法添加如下如下：

```
_mapView.showsCompass= YES; // 设置成 NO 表示关闭指南针；YES 表示显示指南针

_mapView.compassOrigin= CGPointMake(_mapView.compassOrigin.x, 22); //设置指南针位置
```

6.3 比例尺

比例尺表示地图上两点间距离与实际与之对应的两点距离的比，在不同的缩放级别下，比例尺代表的长度也是不同的。

在 iOS SDK 中，比例尺默认显示在地图的左上角。MAMapView 的 showScale 属性用来控制比例尺的可见性，scaleOrigin 属性用来改变比例尺的显示位置。在 ViewController.m 的 viewDidLoad 方法添加如下代码：

```
_mapView.showScale= YES;  //设置成 NO 表示不显示比例尺；YES 表示显示比例尺

_mapView.scaleOrigin= CGPointMake(_mapView.scaleOrigin.x, 22);  //设置比例尺位置
```

运行代码，地图 Logo 显示到右下角，指南针和比例尺也不再被遮挡，如下图所示：


中国电信
16:45
0
5
10
15 千米
北
酸枣岭桥
沙河站
北七家镇
回龙观镇
天通苑
西二旗
马连洼
上清桥
来广营乡
孙河乡
颐和园
亚运村
大山子
马甸
海淀区
亮马桥
玉渊潭乡
东城区
朝阳区
木樨地
大望路
青塔
陶然亭
潘家园
丰台区
十八里店乡
大红门
新发地
和义
大羊坊桥
北京南苑机场
马驹桥桥
大兴区
双源桥
青云店镇
高德地图

7 地图操作

7.1 手势控制

iOS SDK 支持丰富的地图交互手势功能。手势功能默认都是开启的。

（1） 缩放手势

缩放手势可改变地图的缩放级别，地图响应的手势如下：

- 双击地图可以使缩放级别增加 1 (放大)
- 两个手指捏/拉伸

通过 MAMapView 的 scrollEnabled 属性可以禁用或启用缩放手势。这不会影响用户使用地图上的缩放控制按钮。禁用缩放手势的代码如下：

```
- (void)viewDidLoad {
{
    [super viewDidLoad];
    _mapView.zoomEnabled = NO;      //NO 表示禁用缩放手势，YES 表示开启
}
```

（2） 平移（滑动）手势

用户可以用手指拖动地图四处滚动（平移）或用手指滑动地图（动画效果）。通过 MAMapView 的 scrollEnabled 属性可以禁用或开启平移（滑动）手势。以下介绍展示如何禁用缩放手势，示例代码如下：

```
- (void)viewDidLoad {
{
    [super viewDidLoad];
    _mapView.scrollEnabled = NO;      //NO 表示禁用滑动手势，YES 表示开启
}
```

（3） 旋转手势(3D)

用户可以用两个手指在地图上转动，可以旋转 3D 矢量地图。通过调用类 MAMapView 的 rotateEnabled 属性禁用或开启旋转手势。

```
- (void)viewDidLoad {
{
    [super viewDidLoad];
    _mapView.rotateEnabled= NO;      //NO 表示禁用旋转手势，YES 表示开启
}
```

（4） 倾斜手势(3D)

用户可以在地图上放置两个手指，移动它们一起向下或向上去增加或减小倾斜角。通过 MAMapView 的 rotateCameraEnabled 属性禁用或启用倾斜手势。

```
- (void)viewDidLoad {
{
    [super viewDidLoad];
    _mapView.rotateEnabled= NO;      //NO 表示禁用倾斜手势，YES 表示开启
```

```
}
```

7.2 地图操作

除了通过手势操作地图，还可以通过 iOS SDK 中的方法来进行操作。

（1） 地图缩放

地图的缩放级别的范围是[3-19]，调用 MAMapView 的 setZoomLevel 方法设置地图的缩放级别，用来缩放地图。示例代码如下：

```
[_mapView setZoomLevel:17.5 animated:YES];
```

（2） 地图平移

地图平移时，缩放级别不变，可通过改变地图的中心点来移动地图，示例代码如下：

```
[_mapView setCenterCoordinate:center animated:YES];
```

（3） 地图旋转（3D）

旋转角度的范围是[0.f 360.f]，以逆时针为正向。调用 MAMapView 的 setRotationDegree 方法设置地图的旋转角度。示例代码如下：

```
[_mapView setRotationDegree:60.f animated:YES duration:0.5];
```

（4） 地图倾斜（3D）

倾斜角度范围为[0.f, 45.f]，调用 MAMapView 的 setCameraDegree 方法设置地图的倾斜角度。示例代码如下：

```
[_mapView setCameraDegree:30.f animated:YES duration:0.5];
```

7.3 地图截屏

IOS SDK 支持对选定的屏幕地图区域(CGRect)进行截屏，截取的内容包括：地图、覆盖物、弹出气泡。使用 MAMapView 中的 takeSnapshotInRect 方法进行截屏，该方法返回 UIImage 对象。示例代码如下：

```
CGRect inRect = CGRectMake(80,142,160,284);
UIImage *screenshotImage = [_mapView takeSnapshotInRect:inRect] ;
```

8 定位

iOS 系统不允许使用第三方定位，地图 SDK 的定位方法是对 iOS 系统定位的二次封装。通过封装。可将原始的定位点无偏差的显示在高德地图上。同时，可自定义定位图标和精度圈的样式。

注意：

iOS8 对定位功能进行了升级，为了您能正常使用定位功能，需要您参考如下方法进行处理：

V2.3.0(含)之前版本在 iOS8 中无法定位，请参考：
http://lbsbbs.amap.com/forum.php?mod=viewthread&tid=265&extra=page%3D1 ,自行适配定位。

自 V2.4.0 起 SDK 对定位功能进行了适配 ,在 info.plist 中追加 NSLocationWhenInUseUsageDescription 或 NSLocationAlwaysUsageDescription 字段，以申请相应的权限。

- NSLocationWhenInUseUsageDescription 表示应用在前台的时候可以搜到更新的位置信息。
- NSLocationAlwaysUsageDescription 表示应用在前台和后台(suspend 或 terminated)都可以获取到更新的位置数据。

8.1 开启定位

只要开启定位开关（MAMapView 的 showsUserLocation 属性）就可以开始定位。代码如下所示：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    _mapView.showsUserLocation = YES;    //YES 为打开定位，NO 为关闭定位
}
```

当位置更新时，会进定位回调，通过回调函数，能获取到定位点的经纬度坐标，示例代码如下：

```
-(void)mapView:(MAMapView *)mapView didUpdateUserLocation:(MAUserLocation
*)userLocation
updatingLocation:(BOOL)updatingLocation
{
    if(updateingLocation)
    {
        //取出当前位置的坐标

NSLog("latitude : %f,longitude: %f",userLocation.coordinate.latitude,userLocation.coordinate.l
ongitude);
    }
}
```

8.2 定位图层

将 MAMapView 添加到 Subview 中，开启定位后，会在地图上显示定位图层。

8.2.1 显示模式

定位图层有 3 种显示模式，分别为：

- MAUserTrackingModeNone：不跟随用户位置，仅在地图上显示。
- MAUserTrackingModeFollow：跟随用户位置移动，并将定位点设置成地图中心点。
- MAUserTrackingModeFollowWithHeading：跟随用户的位置和角度移动。

通过以下代码可改变定位图层的显示模式：

```
[_mapView setUserTrackingMode: MAUserTrackingModeFollow animated:YES]; //地图跟着位置移动
```

8.2.2 自定义定位图层

定位图层由定位点处的标注（MAUserLocation）和精度圈（MACircle）组成，高德地图 iOS SDK 可自定义定位图层的样式。

对于 3D 矢量地图

通过- (MAAnnotationView *)mapView:(MAMapView *)mapView viewForAnnotation:(id <MAAnnotation>)annotation 方法中自定义定位标注样式；通过- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id <MAOverlay>)overlay 方法自定义定位精度圈的样式。

注意：若想自定义定位经度圈样式，需先将 MAMapView 的 customizeUserLocationAccuracyCircleRepresentation 属性设置为 YES。

示例代码如下：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.

    _mapView.customizeUserLocationAccuracyCircleRepresentation = YES;

    _mapView.userTrackingMode = MAUserTrackingModeFollow;
}

- (MAOverlayView *)mapView:(MAMapView *)mapView viewForOverlay:(id <MAOverlay>)overlay
{
    /* 自定义定位精度对应的 MACircleView. */
    if (overlay == mapView.userLocationAccuracyCircle)
    {
        MACircleView *accuracyCircleView = [[MACircleView alloc] initWithCircle:overlay];

        accuracyCircleView.lineWidth    = 2.f;
        accuracyCircleView.strokeColor  = [UIColor lightGrayColor];
        accuracyCircleView.fillColor    = [UIColor colorWithRed:1 green:0 blue:0 alpha:.3];
```

```
            return accuracyCircleView;
        }
        return nil;
}

- (MAAnnotationView *)mapView:(MAMapView *)mapView
viewForAnnotation:(id<MAAnnotation>)annotation
{
        /* 自定义 userLocation 对应的 annotationView. */
        if ([annotation isKindOfClass:[MAUserLocation class]])
        {
            static NSString *userLocationStyleReuseIndetifier =
@"userLocationStyleReuseIndetifier";
            MAAnnotationView *annotationView = [mapView
dequeueReusableAnnotationViewWithIdentifier:userLocationStyleReuseIndetifier];
            if (annotationView == nil)
            {
                annotationView = [[MAAnnotationView alloc] initWithAnnotation:annotation

reuseIdentifier:userLocationStyleReuseIndetifier];
            }
            annotationView.image = [UIImage imageNamed:@"userPosition"];

            return annotationView;
        }
        return nil;
}
```

对于 2D 栅格地图

通过- (void)mapView:(MAMapView *)mapView didAddAnnotationViews:(NSArray *)views 方法自定义定位标注和精度圈的样式。示例代码如下：

```
- (void)viewDidAppear:(BOOL)animated
{
        [super viewDidAppear:animated];

        _mapView.showsUserLocation = YES;
        _mapView.userTrackingMode = MAUserTrackingModeFollowWithHeading;

        [_mapView setZoomLevel:16.1 animated:YES];
}

- (void)mapView:(MAMapView *)mapView didAddAnnotationViews:(NSArray *)views
{
        MAAnnotationView *view = views[0];

        // 放到该方法中用以保证 userlocation 的 annotationView 已经添加到地图上了。
        if ([view.annotation isKindOfClass:[MAUserLocation class]])
        {
            MAUserLocationRepresentation *pre = [[MAUserLocationRepresentation alloc] init];
            pre.fillColor = [UIColor colorWithRed:0.9 green:0.1 blue:0.1 alpha:0.3];
            pre.strokeColor = [UIColor colorWithRed:0.1 green:0.1 blue:0.9 alpha:1.0];
            pre.image = [UIImage imageNamed:@"location.png"];
            pre.lineWidth = 3;
            pre.lineDashPattern = @[@6, @3];
```

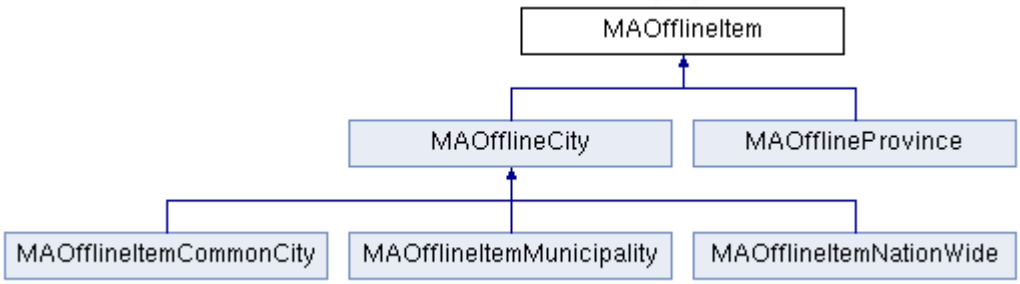
```
            [self.mapView updateUserLocationRepresentation:pre];

            view.calloutOffset = CGPointMake(0, 0);
        }
    }
```

9 离线地图(3D)

高德地图 IOS SDK 支持 3D 矢量地图数据的下载和更新。

离线地图数据以 MAOfflineItem 为单位进行下载，每个 MAOfflineItem 对象包含城市编码、城市名称、数据状态等基本信息，是离线数据省信息(MAOfflineProvince)和离线数据城市信息(MAOfflineCity) 的基类。离线数据城市信息（MAOfflineCity）又派生出三个子类，分别是全国概要图（MAOfflineItemNationWide）、直辖市（MAOfflineItemMunicipality）和普通城市（MAOfflineItemCommonCity）。各类的关系如下图所示：



按照下载项分项下载的离线数据效果如下图所示：

获取离线数据项的示例代码如下：

```
- (void)setupCities
{
    self.sectionTitles = @[@"全国", @"直辖市", @"省份"];
```

```
    self.cities = [MAOfflineMap sharedOfflineMap].cities;//普通城市和直辖市
    self.provinces = [MAOfflineMap sharedOfflineMap].provinces;//省
    self.municipalities = [MAOfflineMap sharedOfflineMap].municipalities;//直辖市

}

- (MAOfflineItem *)itemForIndexPath:(NSIndexPath *)indexPath
{
    MAOfflineItem *item = nil;
    switch (indexPath.section)
    {
        case 0:
        {
            item = [MAOfflineMap sharedOfflineMap].nationWide;//全国概要图
            break;
        }
        case 1:
        {
            item = self.municipalities[indexPath.row];//直辖市
            break;
        }
        case 2:
        {
            item = nil;
            break;
        }
        default:
        {
            MAOfflineProvince *pro = self.provinces[indexPath.section -
self.sectionTitles.count];
            if (indexPath.row == 0)
            {
                item = pro; //整个省
            }
            else
            {
                item = pro.cities[indexPath.row - 1]; //市
            }
            break;
        }
    }
    return item;
}
```

离线下载某个城市(参数 city)地图数据包示例代码如下：

```
- (void)download: (MAOfflineCity *) city
{
  [[MAOfflineMap sharedOfflineMap] downloadCity:city
downloadBlock:^(MAOfflineMapDownloadStatus downloadStatus, id info) {
  dispatch_async(dispatch_get_main_queue(), ^{
  if (downloadStatus == MAOfflineMapDownloadStatusWaiting)
  {
    NSLog(@"状态为: %@", @"等待下载");
  }
  else if(downloadStatus == MAOfflineMapDownloadStatusStart)
```

```
    {
        NSLog(@"状态为: %@", @"开始下载");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusProgress)
    {
        NSLog(@"状态为: %@", @"正在下载");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusCancelled) {
        NSLog(@"状态为: %@", @"取消下载");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusCompleted) {
        NSLog(@"状态为: %@", @"下载完成");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusUnzip) {
        NSLog(@"状态为: %@", @"下载完成，正在解压缩");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusError) {
        NSLog(@"状态为: %@", @"下载错误");
    }
    else if(downloadStatus == MAOfflineMapDownloadStatusFinished) {
        NSLog(@"状态为: %@", @"全部完成");
        [self.mapView reloadMap];                    //激活离线地图
    }
    }];
}
```

暂停某个城市(参数 city)离线地图下载的示例代码如下：

```
- (void)pause:(MAOfflineItem *)item
{
    NSLog(@"pause :%@", item.name);

    [[MAOfflineMap sharedOfflineMap] pauseItem:item];
}
```

暂停所有离线地图下载的函数为 ：[[MAOfflineMap sharedOfflineMap] cancelAll]。

10 搜索服务

高德地图 SDK 提供的搜索服务包括以下功能：POI 搜索、规划路径查询（驾车路线搜索、公交换成方案查询、步行路径检索）、地理编码、逆地理编码、输入提示语搜索。

搜索服务提供一个主搜索对象 AMapSearchAPI 和搜索协议<AMapSearchDelegate>，同时每个搜索功能都包含一个用于构造搜索参数的 Request 对象和一个用于接收搜索结果 Response 对象。使用搜索服务时，按照以下步骤进行：

（1） 初始化主搜索对象 AMapSearchAPI，并继承搜索协议<AMapSearchDelegate>。
（2） 构造 Request 对象，配置搜索参数。
（3） 通过主搜索对象以 Request 对象为参数，发起搜索。
（4） 实现搜索协议中搜索功能对应的回调函数，通过解析 Response 对象获取搜索结果。

10.1 POI 搜索

高德地图提供了千万级别的 POI（Point of Interesting，兴趣点）。在地图表达中，一个 POI 可代表一栋大厦、一家商铺、一处景点等等。

iOS SDK 包括 3 种类型的 POI 搜索：关键字搜索、周边搜索、指定区域搜索。不同类型的 POI 搜索，区别在于构造的搜索参数。其中：

- 关键字搜索：keywords 和 types 必设其一，searchType 为 AMapSearchType_PlaceKeyword。
- 周边搜索：keywords 和 types 必设其一，还必设 location（中心点坐标），searchType 为 AMapSearchType_PlaceAround。
- 指定区域搜索：keywords 和 types 必设其一，还必设 polygon（多边形），searchType 为 AMapSearchType_PlacePolygon。

下面以关键字搜索为例，介绍如何进行 POI 搜索。

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapPlaceSearchRequest 对象，配置关键字搜索参数
    AMapPlaceSearchRequest *poiRequest = [[AMapPlaceSearchRequest alloc] init];
    poiRequest.searchType = AMapSearchType_PlaceKeyword;
    poiRequest.keywords = @"俏江南";
    poiRequest.city = @[@"beijing"];
    poiRequest.requireExtension = YES;

    //发起 POI 搜索
    [_search AMapPlaceSearch: poiRequest];
```

```
}

//实现 POI 搜索对应的回调函数
- (void)onPlaceSearchDone:(AMapPlaceSearchRequest *)request
response:(AMapPlaceSearchResponse *)response
{
    if(response.pois.count == 0)
    {
        return;
    }

    //处理搜索结果
    NSString *strCount = [NSString stringWithFormat:@"count: %d",response.count];
    NSString *strSuggestion = [NSString stringWithFormat:@"Suggestion: %@",
response.suggestion];
    NSString *strPoi = @"";
    for (AMapPOI *p in response.pois) {
        strPoi = [NSString stringWithFormat:@"%@\nPOI: %@", strPoi, p.description];
    }
    NSString *result = [NSString stringWithFormat:@"%@ \n %@ \n %@", strCount,
strSuggestion, strPoi];
    NSLog(@"Place: %@", result);
}
```

10.2 路径规划查询

高德地图 iOS SDK 提供了驾车、步行和公交三种类型的路径规划方案。对于驾车路径搜索，可设置途经点、避让区域和避让道路，同时，还提供“时间最短”、“费用最少”、“距离最短”等多达 9 种的驾车策略，以完成不同业务需求的路径方案。对于公交换乘方案，提供“最经济”、“最少换乘”、“不乘地铁”等换乘策略，为您的出行提供多种选择。

无论哪种类型路径规划，origin（起点坐标）和 destination（终点坐标）为必设参数，其中：

- 驾车路径搜索，searchType 为 AMapSearchType_NaviDrive。
- 公交换乘查询，searchType 为 AMapSearchType_NaviBus。
- 步行路径搜索，searchType 为 AMapSearchType_NaviWalking。

下面以驾车路径规划为例，进行介绍。

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapNavigationSearchRequest 对象，配置查询参数
    AMapNavigationSearchRequest  *naviRequest=  [[AMapNavigationSearchRequest  alloc]
```

```
init];
    naviRequest.searchType = AMapSearchType_NaviDrive;
    naviRequest.requireExtension = YES;
    naviRequest.origin = [AMapGeoPoint locationWithLatitude:39.994949
longtitude:116.447265];
    naviRequest.destination = [AMapGeoPoint locationWithLatitude:39.990459
longtitude:116.481476];

    //发起路径搜索
    [_search AMapNavigationSearch: naviRequest];
}

//实现路径搜索的回调函数
- (void)onNavigationSearchDone:(AMapNavigationSearchRequest *)request
response:(AMapNavigationSearchResponse *)response
{
    if(response.route = nil)
    {
        return;
    }

    //处理搜索结果
    NSString *route = [NSString stringWithFormat:@"Navi: %@", response.route];
    NSLog(@"%@", route);
}
```

规划路径的结果构成如下图所示，可根据此结构图解析结果，准确展示线路。

10.3 地理编码

地理编码用于关联地址描述与空间坐标点，分为正向地理编码和逆地理编码。

10.3.1 正向地理编码

正向地理编码，又称为地址匹配，是从已知的地址描述到对应的经纬度坐标的转换过程。正向地理编码时 address（地址）为必设参数。

正向地理编码示例如下：

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapGeocodeSearchRequest 对象，address 为必选项，city 为可选项
    AMapGeocodeSearchRequest *geoRequest = [[AMapGeocodeSearchRequest alloc] init];
    geoRequest.searchType = AMapSearchType_Geocode;
    geoRequest.address = @"西单";
    geoRequest.city = @[@"beijing"];

    //发起正向地理编码
    [_search AMapGeocodeSearch: geoRequest];
}

//实现正向地理编码的回调函数
- (void)onGeocodeSearchDone:(AMapGeocodeSearchRequest *)request
response:(AMapGeocodeSearchResponse *)response
{
    if(response.geocodes.count == 0)
    {
        return;
    }

    //处理搜索结果
    NSString *strCount = [NSString stringWithFormat:@"count: %d", response.count];
NSString *strGeocodes = @"";
    for (AMapTip *p in response.geocodes) {
        strGeocodes = [NSString stringWithFormat:@"%@\ngeocode: %@", strGeocodes,
p.description];
    }
    NSString *result = [NSString stringWithFormat:@"%@ \n %@", strCount, strGeocodes];
    NSLog(@"Geocode: %@", result);
}
```

10.3.2 逆地理编码

逆地理编码，又称地址解析服务，是指从已知的经纬度坐标到对应的地址描述（如行政区划、街区、楼层、房间等）的转换。进行逆地编码时，location（坐标）为必设参数。

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
{
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapReGeocodeSearchRequest 对象，location 为必选项，radius 为可选项
    AMapReGeocodeSearchRequest *regeoRequest = [[AMapReGeocodeSearchRequest alloc]
init];
    regeoRequest.searchType = AMapSearchType_ReGeocode;
    regeoRequest.location = [AMapGeoPoint locationWithLatitude:39.990459
longtitude:116.481476];
    regeoRequest.radius = 10000;
    regeoRequest.requireExtension = YES;

    //发起逆地理编码
    [_search AMapReGoecodeSearch: regeoRequest];

}

//实现逆地理编码的回调函数
- (void)onReGeocodeSearchDone:(AMapReGeocodeSearchRequest *)request
response:(AMapReGeocodeSearchResponse *)response
{
    if(response.regeocode != nil)
    {
        //处理搜索结果
        NSString *result = [NSString stringWithFormat:@"ReGeocode: %@",
response.regeocode];
        NSLog(@"ReGeo: %@", result);
    }
}
```

10.4 公交查询

高德地图 iOS SDK 提供的公交查询包括：公交站查询、公交线路查询（根据线路关键字或线路 ID）。

10.4.1 公交站查询

根据输入的公交站名称的关键字，查询该公交站的位置信息以及途经该站的所有公交线路等。查询公交站时，keywords（关键字）为必设参数。示例代码如下：

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapBusStopSearchRequest 对象
    AMapBusStopSearchRequest *stopRequest = [[AMapBusStopSearchRequest alloc] init];
    stopRequest.keywords = @"望京西";
    stopRequest.city = @[@"beijing"];

    //发起公交站查询
    [_search AMapBusStopSearch: stopRequest];

}

//实现公交站查询的回调函数
-(void)onBusStopSearchDone:(AMapBusStopSearchRequest*)request
response:(AMapBusStopSearchResponse *)response
{
    if(response.busstops.count == 0)
    {
        return;
    }

    //处理查询结果
    NSString *strStop = @"";
    for (AMapBusStop *p in response.busstops) {
        strStop = [NSString stringWithFormat:@"%@\nStop: %@", strStop, p.description];
    }
    NSString *result = [NSString stringWithFormat:@"%@ \n %@ \n %@", strCount,
strSuggestion, strStop];
    NSLog(@"Stop: %@", result);
}
```

10.4.2 公交线路查询

iOS SDK 提供根据线路关键字和线路 ID 两种方式进行公交线路查询。其中：根据关键字查询时，keywords 为必设参数，searchType 为 AMapSearchType_LineKeyword；根据线路 ID 查询时，uid 为必设，searchType 为 AMapSearchType_LineID。

下面以线路关键字查询为例，介绍如何进行公交线路查询。示例代码如下：

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end
```

```
-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapBusLineSearchRequest 对象
    AMapBusLineSearchRequest *lineRequest = [[AMapBusLineSearchRequest alloc] init];
    ineRequest.keywords = @"445";
    lineRequest.requireExtension = YES;

    //发起公交线路查询
    [_search AMapBusLineSearch:lineRequest];
}

//实现公交线路查询的回调函数
-(void)onBusLineSearchDone:(AMapBusLineSearchRequest*)request
response:(AMapBusLineSearchResponse *)response
{
    if(response.buslines.count == 0)
    {
        return;
    }

    //处理查询结果
    NSString *strLine = @"";
    for (AMapBusLine *p in response.buslines) {
        strLine = [NSString stringWithFormat:@"%@\nLine: %@", strLine, p.description];
    }
    NSString *result = [NSString stringWithFormat:@"%@ \n %@ \n %@", strCount,
strSuggestion, strLine];
    NSLog(@"Line: %@", result);
}
```

10.5 输入提示搜索

输入提示是指根据用户输入的关键词，给出相应的提示信息，将最有可能的搜索词呈现给用户，以减少用户输入信息，提升用户体验。如：输入“方恒”，提示“方恒国际中心 A 座”，“方恒购物中心”等。

输入提示功能作为辅助功能，可配合地理编码、POI 关键字搜索等功能使用。

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapInputTipsSearchRequest 对象，keywords 为必选项，city 为可选项
    AMapInputTipsSearchRequest *tipsRequest= [[AMapInputTipsSearchRequest alloc] init];
    tipsRequest.searchType = AMapSearchType_InputTips;
    tipsRequest.keywords = @"望";
```

```
    tipsRequest.city = @[@"北京"];

    //发起输入提示搜索
    [_search AMapInputTipsSearch: tipsRequest];
}

//实现输入提示的回调函数
-(void)onInputTipsSearchDone:(AMapInputTipsSearchRequest*)request
response:(AMapInputTipsSearchResponse *)response
{
    if(response.tips.count == 0)
    {
        return;
    }

    //处理搜索结果
    NSString *strCount = [NSString stringWithFormat:@"count: %d", response.count];
    NSString *strtips = @"";
    for (AMapTip *p in response.tips) {
        strtips = [NSString stringWithFormat:@"%@\nTip: %@", strtips, p.description];
    }
    NSString *result = [NSString stringWithFormat:@"%@ \n %@", strCount, strtips];
    NSLog(@"InputTips: %@", result);
}
```

10.6 行政区划查询

根据县（区）级行政区划名称查询其下级区划的详细信息，如：中心点坐标、编码等等。

```
@interface ViewController ()<AMapSearchDelegate>
{
    AMapSearchAPI *_search;
}
@end

-(void)viewDidLoad
  {
    //初始化检索对象
    _search = [[AMapSearchAPI alloc] initWithSearchKey:@"您的 key" Delegate:self];

    //构造 AMapDistrictSearchRequest 对象，keywords 为必选项
    AMapDistrictSearchRequest *districtRequest = [[AMapDistrictSearchRequest alloc] init];
    districtRequest.keywords = name;
    districtRequest.requireExtension = YES;

    //发起行政区划查询
    [_search AMapDistrictSearch:districtRequest]; }

//实现行政区划查询的回调函数
- (void)onDistrictSearchDone:(AMapDistrictSearchRequest *)request
response:(AMapDistrictSearchResponse *)response
{
    NSLog(@"response: %@", response);
    [self handleDistrictResponse:response];
}
```

```
- (void)handleDistrictResponse:(AMapDistrictSearchResponse *)response
{
    if (response == nil)
    {
        return;
    }

    for (AMapDistrict *dist in response.districts)
    {
        MAPointAnnotation *poiAnnotation = [[MAPointAnnotation alloc] init];

        poiAnnotation.coordinate = CLLocationCoordinate2DMake(dist.center.latitude,
dist.center.longitude);
        poiAnnotation.title      = dist.name;
        poiAnnotation.subtitle   = dist.adcode;

        [self.mapView addAnnotation:poiAnnotation];

        if (dist.polylines.count > 0)
        {
            MAMapRect bounds = MAMapRectZero;

            for (NSString *polylineStr in dist.polylines)
            {
                MAPolyline *polyline = [CommonUtility
polylineForCoordinateString:polylineStr];
                [self.mapView addOverlay:polyline];

                bounds = MAMapRectUnion(bounds, polyline.boundingMapRect);
            }

            [self.mapView setVisibleMapRect:bounds animated:YES];
        }

        // 下级区划
        for (AMapDistrict *subdist in dist.districts)
        {
            MAPointAnnotation *subAnnotation = [[MAPointAnnotation alloc] init];

            subAnnotation.coordinate =
CLLocationCoordinate2DMake(subdist.center.latitude, subdist.center.longitude);
            subAnnotation.title      = subdist.name;
            subAnnotation.subtitle   = subdist.adcode;

            [self.mapView addAnnotation:subAnnotation];
        }

    }
}
```