百度地图 v3.1.0

sdk 介绍与集成

1. 简述：

什么是百度地图iOS SDK?

百度地图iOS SDK是一套基于armv7、armv7s、arm64（自v2.5.0版本）处理器设备的应用程序接口，不仅提供构建地图的基本接口，还提供POI搜索、地理编码、路线规划、定位、本地覆盖物绘制、周边雷达等服务，自v2.0.0开始为矢量渲染的3D地图，并新增了矢量离线地图下载功能接口。

自v2.7.0版本开始，iOS 地图SDK向广大开发者提供了 .framework形式的开发包，此种形式配置简单、使用方便，请广大开发者使用。自2.9.0版本起，iOS 地图SDK不再提供 .a形式的开发包!

您可以使用百度地图iOS SDK开发适用于移动设备的地图应用，通过接口，您可以轻松访问百度服务和数据，构建功能丰富、交互性强的地图应用程序。百度地图iOS SDK提供的功能如下：

- **地图**：提供地图展示和地图操作功能；
- **室内图**：提供展示公众建筑物室内地图的展示功能；
- **POI检索**：支持周边检索、区域检索和城市内兴趣点检索；
- **室内POI检索**：支持设置城市和当前建筑物的室内POI检索；
- **地理编码**：提供经纬度和地址信息相互转化的功能接口；
- **线路规划**：支持公交、驾车、步行、骑行，四种方式的线路规划；
- **覆盖物图层**：支持在地图上添加覆盖物（标注、几何图形、热力图、地形图图层等），展示更丰富的LBS信息；
- **定位**：获取当前位置信息，并在地图上展示（支持普通、跟随、罗盘三种模式）；
- **离线地图**：使用离线地图可节省用户流量，提供更好的地图展示效果；
- **调启百度地图**：利用SDK接口，直接在本地打开百度地图客户端或WebApp，实现地图功能；
- **周边雷达**：利用周边雷达功能，开发者可在App内低成本、快速实现查找周边使用相同App的用户位置的功能；
- **LBS云检索**：支持查询存储在LBS云内的自有数据；
- **瓦片图层**： 支持在地图上添加自有瓦片数据。
- **特色功能**：提供短串分享、Place详情检索、热力图等特色功能，帮助开发者搭建功能更加强大的应用；

2. 申请密钥

简介：用户在使用 SDK 前需获取百度地图移动开发者密钥(key)，此 key 与引用 sdk 的程序包名有关，地图初始化时候需要用到该 key。

第一步：打开百度地图 API 控制台

第二步：点击创建应用

第三步：填写应用名称，应用类型选 IOS SDK，安全吗为 bundleId，提交

第四步：控制台列表中“访问应用(ak)“即为开发者密钥

3. 注意事项

(1) 静态库中采用 ObjectC++实现，因此需要您保证您工程中至少有一个.mm 后缀的源文件(您可以将任意一个.m 后缀的文件改名为.mm)，或者在工程属性中指定编

译方式，即在 Xcode 的 Project -> Edit Active Target -> Build Setting 中找到 Compile Sources As，并将其设置为"Objective-C++"

(2) 如果您只在 Xib 文件中使用了 BMKMapView，没有在代码中使用 BMKMapView，编译器在链接时不会链接对应符号，需要在工程属性中显式设定：在 Xcode 的 Project -> Edit Active Target -> Build Setting -> Other Linker Flags 中添加-ObjC

(3) 在使用 Xcode6 进行 SDK 开发过程中，需要在 info.plist 中添加：Bundle display name ，且其值不能为空（Xcode6 新建的项目没有此配置，若没有会造成 manager start failed）

(4) 由于iOS9改用更安全的https，为了能够在iOS9中正常使用地图SDK，请在"Info.plist"中进行如下配置，否则影响SDK的使用。

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

(5) 如果在iOS9中使用了调起百度地图客户端功能，必须在"Info.plist"中进行如下配置，否则不能调起百度地图客户端。

```
<key>LSApplicationQueriesSchemes</key>
<array>
    <string>baidumap</string>
</array>
```

(6) 管理地图的生命周期：自2.0.0起，BMKMapView新增viewWillAppear、viewWillDisappear方法来控制BMKMapView的生命周期，并且在一个时刻只能有一个BMKMapView接受回调消息，因此在使用BMKMapView的viewController中需要在viewWillAppear、viewWillDisappear方法中调用BMKMapView的对应的方法，并处理delegate，代码如下：

```
(void)viewWillAppear:(BOOL)animated
{
    [_mapView viewWillAppear];
    _mapView.delegate = self; // 此处记得不用的时候需要置nil，否则影响内存的释放
}
```

```
-(void)viewWillDisappear:(BOOL)animated
{
    [_mapView viewWillDisappear];
    _mapView.delegate = nil; // 不用时，置nil
}
```

(7) 自iOS SDK v2.5.0起，为了对iOS8的定位能力做兼容，做了相应的修改，开发者在使用过程中注意事项如下： 需要在info.plist里添加（以下二选一，两个都添加默认使用NSLocationWhenInUseUsageDescription）
NSLocationWhenInUseUsageDescription ，允许在前台使用时获取GPS的描述
NSLocationAlwaysUsageDescription ，允许永久使用GPS的描述

(8) 确认项目中添加mapapi.bundle文件以及添加方法正确，不能删除或随意更改其中files文件夹下的内容：
注：mapapi.bundle中存储了定位、默认大头针标注View及路线关键点的资源图片，还存储了矢量地图绘制必需的资源文件。如果您不需要使用内置的图片显示功能，则可以删除bundle文件中的image文件夹。您也可以根据具体需求任意替换或删除该bundle中image文件夹的图片文件。添加方式：将mapapi.bundle拷贝到您的工程目录，直接将该bundle文件托拽至Xcode工程左侧的Groups&Files中即可。若您需要替换定位、指南针的图标，请保留原文件名称，否则不显示替换的新图片，默认大头针标注与路线关键点的新图片名称可自定义名称。

(9) 注意BMKManager对象的生命周期管理，在使用地图SDK期间不能释放该对象，尤其在arc情况下注意避免提前被自动释放，否则，该对象一旦被释放，网络模块将不可用，地图无法加载，检索失败。

4. 配置开发环境

(1) 使用CocoaPods

注：此种方式只支持导入全量包的SDK，包含百度地图iOS SDK所有功能

一、前提：安装CocoaPods

在终端输入

```
sudo gem install cocoapods
```

如果安装成功，会有一个提示

```
Successfully installed cocoaPods
```

二、使用CocoaPods导入地图SDK

在当前工程文件（.xcodeproj）所在文件夹下，打开terminal

1.创建Podfile：

```
touch Podfile
```

编辑文件，内容如下：

target ‘工程名’do

```
pod 'BaiduMapKit' #百度地图SDK
```

，

end

3.在Podfile所在的文件夹下输入命令：

```
pod install （这个可能比较慢，请耐心等待……）
```

(2) 手动配置.framework形式开发包

1) 根据需要导入.framework包，其中base为必须包

2) 引入所需系统库：

CoreLocation.framework和QuartzCore.framework、OpenGLES.framework、SystemConfiguration.framework、CoreGraphics.framework、Security.framework、libsqlite3.0.tbd、CoreTelephony.framework 、libstdc++.6.0.9.tbd

在项目根目录添加Frameworks文件夹，系统库添加时会自动添加至该目录

(3) 环境配置：在TARGETS->Build Settings->Other Linker Flags 中添加-ObjC。

(4) 引入mapapi.bundle资源文件，方法：选中工程名，在右键菜单中选择Add Files to “工程名”…，从BaiduMapAPI_Map.framework||Resources文件中选择mapapi.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

(5) 引入头文件

(6) 初始化sdk

在您的AppDelegate.h文件中添加BMKMapManager的定义

```
@interface BaiduMapApiDemoAppDelegate : NSObject <UIApplicationDelegate> {
        UIWindow *window;
        UINavigationController *navigationController;
        BMKMapManager* _mapManager;
}
```

在您的AppDelegate.m文件中添加对BMKMapManager的初始化，并填入您申请的授权Key，示例如下

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {        // 要使用百度地
请先启动BaiduMapManager
    _mapManager = [[BMKMapManager alloc]init];
// 如果要关注网络及授权验证事件，请设定     generalDelegate参数
    BOOL ret = [_mapManager start:@"在此处输入您的授权Key"  generalDelegate:nil];
    if (!ret) {
        NSLog(@"manager start failed!");
    }
// Add the navigation controller's view to the window and display.
    [self.window addSubview:navigationController.view];
    [self.window makeKeyAndVisible];
   return YES;
}
```

sdk 具体接口调用

1. 定位功能

(1) 简介：

由于系统原因，iOS不允许使用第三方定位，因此地图SDK中的定位方法，本质上是对原生定位的二次封装。通过封装，开发者可更便捷的使用。此外，地图SDK中还提供了相应的定位图层（支持定位三态效果），帮助开发者显示当前位置信息。

注：自iOS8起，系统定位功能进行了升级，SDK为了实现最新的适配，自v2.5.0起也做了相应的修改，开发者在使用定位功能之前，需要在info.plist里添加（以下二选一，两个都添加默认使用NSLocationWhenInUseUsageDescription）：

NSLocationWhenInUseUsageDescription ，允许在前台使用时获取GPS的描述

NSLocationAlwaysUsageDescription ，允许永久使用GPS的描述

(2) 定位显示类型

4.1起，新增一种定位方法BMKUserTrackingModeHeading，普通定位+定位罗盘模式。

目前为止，BMKMapView的定位模式(userTrackingMode)有4种分别是：

1) BMKUserTrackingModeNone :普通定位模式，显示我的位置，我的位置图标和地图都不会旋转

2）BMKUserTrackingModeFollow ：定位跟随模式，我的位置始终在地图中心，我的位置图标会旋转，地图不会旋转

3）BMKUserTrackingModeFollowWithHeading ：定位罗盘模式，我的位置始终在地图中心，我的位置图标和地图都会跟着旋转

4）BMKUserTrackingModeHeading：普通定位+定位罗盘模式，显示我的位置，我的位置始终在地图中心，我的位置图标会旋转，地图不会旋转。即在普通定位模式的基础上显示方向。

(3) 获取位置信息

定位功能可以和地图功能分离使用，单独的定位功能使用方式如下：

```
-(void)viewDidLoad
{
    //初始化BMKLocationService
    _locService = [[BMKLocationService alloc]init];
    _locService.delegate = self;
    //启动LocationService
    [_locService startUserLocationService];
}
//实现相关delegate 处理位置信息更新
//处理方向变更信息
- (void)didUpdateUserHeading:(BMKUserLocation *)userLocation
{
    //NSLog(@"heading is %@",userLocation.heading);
}
//处理位置坐标更新
- (void)didUpdateBMKUserLocation:(BMKUserLocation *)userLocation
{
    //NSLog(@"didUpdateUserLocation lat %f,long %f",userLocation.location.coordinate.
latitude,userLocation.location.coordinate.longitude);
}
```

(4) 展示位置信息

展示定位信息的功能位于 “基础地图（Map）”这个功能模块，开发者使用时请注意选择

```
//普通态
//以下_mapView为BMKMapView对象
_mapView.showsUserLocation = YES;//显示定位图层
[_mapView updateLocationData:userLocation];
```

2. 地理编码

(1) 简介

地理编码指的是将地址信息建立空间坐标关系的过程。又可分为正向地理编码和反向地理编码。

1)正向地理编码指的是由地址信息转换为坐标点的过程。

2)反向地理编码服务实现了将地球表面的地址坐标转换为标准地址的过程反向地理编码提供了坐标定位引擎，帮助用户通过地面某个地物的坐标值来反向查询得到该地物所在的行政区划、所处街道、以及最匹配的标准地

址信息。通过丰富的标准地址库中的数据，可帮助用户在进行移动端查询、商业分析、规划分析等领域创造无限价值

(2) 发起正向/逆向地理编码请求

```
-(void)viewDidLoad
{
    //初始化检索对象
    _searcher =[[BMKGeoCodeSearch alloc]init];
    _searcher.delegate = self;
    BMKGeoCodeSearchOption *geoCodeSearchOption = [[BMKGeoCodeSearchOption alloc]init
];
    geoCodeSearchOption.city= @"北京市";
    geocodeSearchOption.address = @"海淀区上地10街10号";
    BOOL flag = [_searcher geoCode:geoCodeSearchOption];
    [geoCodeSearchOption release];
    if(flag)
    {
        NSLog(@"geo检索发送成功");
    }
    else
    {
        NSLog(@"geo检索发送失败");
    }

    //发起反向地理编码检索
    //CLLocationCoordinate2D pt = (CLLocationCoordinate2D){39.915, 116.404};
    //BMKReverseGeoCodeOption *reverseGeoCodeSearchOption = [[
    //BMKReverseGeoCodeOption alloc]init];
    //reverseGeoCodeSearchOption.reverseGeoPoint = pt;
    //BOOL flag = [_searcher reverseGeoCode:reverseGeoCodeSearchOption];
    //[reverseGeoCodeSearchOption release];
    //if(flag)
    //{
    //  NSLog(@"反geo检索发送成功");
    //}
    //else
    //{
    //  NSLog(@"反geo检索发送失败");
    //}
 }
```

(3) 实现代理处理回调结果

```
//实现Deleage处理回调结果
//接收正向编码结果

- (void)onGetGeoCodeResult:(BMKGeoCodeSearch *)searcher result:(BMKGeoCodeResult *)r
esult errorCode:(BMKSearchErrorCode)error{
    if (error == BMK_SEARCH_NO_ERROR) {
        //在此处理正常结果
    }
    else {
        NSLog(@"抱歉，未找到结果");
    }
}

//接收反向地理编码结果
//-(void) onGetReverseGeoCodeResult:(BMKGeoCodeSearch *)searcher result:
//(BMKReverseGeoCodeResult *)result
//errorCode:(BMKSearchErrorCode)error{
//  if (error == BMK_SEARCH_NO_ERROR) {
//      在此处理正常结果
//  }
//  else {
//      NSLog(@"抱歉，未找到结果");
//  }
//}

//不使用时将delegate设置为 nil
-(void)viewWillDisappear:(BOOL)animated
{
    _searcher.delegate = nil;
}
```

3. 路径规划

(1) 简介

1) 百度地图iOS SDK为开发者提供了公交换乘、驾车、骑行和步行 4种类型的线路规划方案，同时根据不同的方案还可以选择“时间最短”、“距离最短”等策略来完成最终的线路规划。开发者可根据自己实际的业务需求来自由使用。

2) 自v2.7.0版本起，我们针对驾车线路规划，增加了返回多条线路结果的能力，具体使用方法，可参考我们的官网Demo。

3) 自v2.10.0起，新增骑行线路规划功能，具体使用方法请参看demo。

4) 自v3.0起,驾车路线规划结果新增3个属性：打车费用信息、拥堵米数、红绿灯个数。

5) 自v3.1.0起，新增跨城综合公交线路规划MassTransit，支持起点、终点坐标在不同城市的公交线路规划，可检索火车、飞机、公交、大巴等公共交通线路，同时可以根据不同的方案选择多种策略来完成跨城公共交通线路规划。此跨城公交线路规划包含旧公交线路规划transit（支持城市内公交规划）的全部功能，旧公交线路规划将废弃，建议使用新版跨城公交线路规划。

(2) 公交换乘的线路规划

1) 初始化并发起检索

```
    //初始化检索对象
    _routeSearch = [[BMKRouteSearch alloc] init];
    //设置delegate，用于接收检索结果
    _routeSearch.delegate = self;
    //构造公共交通路线规划检索信息类
    BMKPlanNode* start = [[BMKPlanNode alloc]init];
    start.name = @"北京";
    start.cityName = @"天安门";
    BMKPlanNode* end = [[BMKPlanNode alloc]init];
    end.name = @"天津";
    end.cityName = @"天津站";
    BMKMassTransitRoutePlanOption *option = [[BMKMassTransitRoutePlanOption alloc]ini
t];
    option.from = start;
    option.to = end;

    //发起检索
    BOOL flag = [_routesearch massTransitSearch:option];

    if(flag) {
        NSLog(@"公交交通检索（支持垮城）发送成功");
    } else {
        NSLog(@"公交交通检索（支持垮城）发送失败");
    }
```

2) 实现delegate，获取结果

```
/**
 *返回公共交通路线检索结果（new）
 *@param searcher 搜索对象
 *@param result 搜索结果，类型为BMKMassTransitRouteResult
 *@param error 错误号，@see BMKSearchErrorCode
 */
- (void)onGetMassTransitRouteResult:(BMKRouteSearch*)searcher result:(BMKMassTransit
RouteResult*)result errorCode:(BMKSearchErrorCode)error
{
    NSLog(@"onGetMassTransitRouteResult error:%d", (int)error);
    if (errorCode == BMK_SEARCH_NO_ERROR) {
        //成功获取结果
    } else {
        //检索失败
    }
}
```

4. 打开百度地图进行路径搜索

```
BMKPlanNode* from= [[BMKPlanNode alloc]init];
BMKPlanNode* to = [[BMKPlanNode alloc]init];
from.name = fromName;
from.pt = fromLocation;
to.name = toName;
to.pt = toLocation;

switch (searchType) {
  case BMKRouteSearchTypeTransiting:{
    BMKOpenTransitRouteOption *option = [[BMKOpenTransitRouteOption alloc] init];
    option.appScheme = @"baidumapsdk://mapsdk.baidu.com";//用于调起成功后，返回原应用
    option.startPoint = from;
    option.endPoint = to;
    return  [BMKOpenRoute openBaiduMapTransitRoute:option];
  }
    break;
  case BMKRouteSearchTypeDriving:{
    BMKOpenDrivingRouteOption *option = [[BMKOpenDrivingRouteOption alloc] init];
    option.appScheme = @"baidumapsdk://mapsdk.baidu.com";//用于调起成功后，返回原应用
    option.startPoint = from;
    option.endPoint = to;
    return  [BMKOpenRoute openBaiduMapDrivingRoute:option];
  }
    break;
  case BMKRouteSearchTypeWalking:{
    BMKOpenWalkingRouteOption *option = [[BMKOpenWalkingRouteOption alloc] init];
    option.appScheme = @"baidumapsdk://mapsdk.baidu.com";//用于调起成功后，返回原应用
    option.startPoint = from;
    option.endPoint = to;
    return  [BMKOpenRoute openBaiduMapWalkingRoute:option];
  }
```

1）百度地图SDK提供简单的接口调用，通过调起百度地图客户端，实现复杂的业务逻辑。

2）目前所支持的调启类型有：POI周边检索、POI详情检索、POI全景、步行线路规划、驾车线路规划、公交线路规划、驾车导航、步行导航、骑行导航（调启步行导航、骑行导航，需百度地图App 8.8以上版本才可以支持）。。

5. 周边雷达功能

(1) 简介

周边雷达功能，是面向移动端开发者的一套SDK功能接口。它的本质是一个连接百度LBS开放平台前端SDK产品和后端LBS云的中间服务。

开发者利用周边雷达功能，可以便捷的在自己的应用内，帮助用户实现查找周边跟“我”使同样一款App的人、查看周边用户在听什么歌、看什么文章、有什么新动态、查看周边有什么最新发生的新闻、资讯

(2) 使用周边雷达功能的流程

3) 注册周边雷达

注册周边雷达是使用其相应功能的基础前提。通过注册可实现一个或多个应用之间的关系绑定，实现相互之间的位置信息查看。

点击如下按钮，进入我的周边雷达管理及新建页面。

4) 初始化周边雷达功能

在使用位置信息上传和检索周边位置信息之前，需要对周边雷达功能模块进行初始化操作。初始化的核心代码如下：

1、周边雷达管理类使用了单例模式，并且通过引用计数的方式管理这个实例。可以使用使用下边的方法获取实例（引用计数加1）：

```
BMKRadarManager *_radarManager = [BMKRadarManagergetRadarManagerInstance];
在不需要时，通过下边的方法使引用计数减1
[BMKRadarManagerreleaseRadarManagerInstance];
```

2、在上传和拉取位置信息前，需要设置userid，否则会自动生成.设置userid的代码如下：

```
[_radarManager.userId = @"baidu_mapsdk_radarid"];
```

3、通过添加radar delegate获取自动上传时的位置信息，以及获得雷达操作结果，代码如下：

```
[_radarManageraddRadarManagerDelegate:self];//添加radar delegate
```

在不需要时，需要移除radar delegate，否则会影响内存的释放。代码如下：

```
[_radarManagerremoveRadarManagerDelegate:self];//不用需移除，否则影响内存释放
```

5) 位置信息上传

周边雷达功能模块，支持将用户的位置等信息上传到百度LBS云服务，从而实现应用内部及应用之间的位置信息查看。

目前支持单次位置信息上传和位置信息连续自动上传两种模式。

单次位置信息上传的核心代码如下：

```
//构造我的位置信息
BMKRadarUploadInfo *myinfo = [[BMKRadarUploadInfoalloc] init];
myinfo.extInfo = @"hello,world";//扩展信息
myinfo.pt = CLLocationCoordinate2DMake(39.916, 116.404);//我的地理坐标
//上传我的位置信息
BOOL res = [_radarManageruploadInfoRequest:myinfo];
if (res) {
NSLog(@"upload 成功");
    } else {
NSLog(@"upload 失败");
}
```

位置信息连续自动上传的核心代码如下：

```
//启动自动上传用户位置信息,需要实现getRadarAutoUploadInfo获取我的位置信息
[_radarManagerstartAutoUpload:5];
```

6）周边位置检索

利用周边雷达功能，可实现周边（处于同一个周边雷达关系内）用户位置信息检索的能力。 检索过程支持距离、时间等约束条件；返回结果支持按照距离、时间远近的排序。

第一步，发起检索请求，核心代码如下：

```
BMKRadarNearbySearchOption *option = [[BMKRadarNearbySearchOptionalloc] init]
    ;
option.radius = 8000;//检索半径
option.sortType = BMK_RADAR_SORT_TYPE_DISTANCE_FROM_NEAR_TO_FAR;//排序方式
option.centerPt = _CLLocationCoordinate2DMake(39.916, 116.404);//检索中心点
//发起检索
BOOL res = [_radarManagergetRadarNearbySearchRequest:option];
if (res) {
NSLog(@"get 成功");
    } else {
NSLog(@"get 失败");
}
```

第二步，实现BMKRadarManagerDelegate回调方法获取结果，核心代码如下：

```
- (void)onGetRadarNearbySearchResult:(BMKRadarNearbyResult *)result error:(BMKRadarE
rrorCode)error {
NSLog(@"onGetRadarNearbySearchResult  %d", error);
if (error == BMK_RADAR_NO_ERROR) {
    }
}
```

6. 关键词检索(在线建议查询)

（1） 简介：

在线建议查询是指根据关键词查询在线建议词。为了帮助开发者实现检索出来的关键词快速定位到地图上，SDK自3.5.0版本起，开放了检索结果的经纬度信息及对应POI点的UID信息。

注意：

1）. 在线建议检索的本质是根据部分关键是检索出来可能的完整关键词名称，如果需要这些关键词对应的POI的具体信息，请使用POI检索来完成；

2). 在线检索结果的第一条可能存在没有经纬度信息的情况，该条结果为文字联想出来的关键词结果，并不对应任何确切POI点。例如输入“肯”，第一条结果为“肯德基”，这条结果是一个泛指的名称，不会带有经纬度等信息。

(2) 初始化检索功能

```
_searcher =[[BMKSuggestionSearch alloc]init];
_searcher.delegate = self;
BMKSuggestionSearchOption* option = [[BMKSuggestionSearchOption alloc] init];
option.cityname = @"北京";
option.keyword  = @"中关村";
BOOL flag = [_searchersuggestionSearch:option];
    [option release];
    if(flag)
    {
        NSLog(@"建议检索发送成功");
    }
    else
    {
        NSLog(@"建议检索发送失败");
    }
}
```

(3) 实现delegate处理回调

```
- (void)onGetSuggestionResult:(BMKSuggestionSearch*)searcher result:(BMKSuggestionResult*)result errorCode:(BMKSearchErrorCode)error{
        if (error == BMK_SEARCH_NO_ERROR) {
                //在此处理正常结果
        }
        else {
                NSLog(@"抱歉，未找到结果");
        }
}

//不使用时将delegate设置为 nil
-(void)viewWillDisappear:(BOOL)animated
{
    _searcher.delegate = nil;
}
```

7. poi检索

(1) 简介：

POI（Point of Interest），中文可以翻译为“兴趣点”。在地理信息系统中，一个POI可以是一栋房子、一个商铺、一个邮筒、一个公交站等。百度地图SDK提供三种类型的POI检索：周边检索、区域检索和城市内检索。下面将以周边检索为例，向大家介绍如何使用检索服务。

(2) 初始化检索功能

```
-(void)viewDidLoad
 {
   //初始化检索对象
   _searcher =[[BMKPoiSearch alloc]init];
   _searcher.delegate = self;
   //发起检索
   BMKNearbySearchOption *option = [[BMKNearbySearchOption alloc]init];
   option.pageIndex = curPage;
   option.pageCapacity = 10;
   option.location = CLLocationCoordinate2D{39.915, 116.404};
   option.keyword = @"小吃";
   BOOL flag = [_searcher poiSearchNearBy:option];
   [option release];
   if(flag)
   {
       NSLog(@"周边检索发送成功");
   }
   else
   {
       NSLog(@"周边检索发送失败");
   }

 }
```

(3) 实现代理处理回调

```
 - (void)onGetPoiResult:(BMKPoiSearch*)searcher result:(BMKPoiResult*)poiResultList
errorCode:(BMKSearchErrorCode)error
 {
    if (error == BMK_SEARCH_NO_ERROR) {
        //在此处理正常结果
    }
    else if (error == BMK_SEARCH_AMBIGUOUS_KEYWORD){
         //当在设置城市未找到结果，但在其他城市找到结果时，回调建议检索城市列表
         // result.cityList;
         NSLog(@"起始点有歧义");
    } else {
         NSLog(@"抱歉，未找到结果");
    }
}
//不使用时将delegate设置为 nil
-(void)viewWillDisappear:(BOOL)animated
{
     _searcher.delegate = nil;
}
```

8. poi详情检索

(1) 发起检索

```
//初始化检索服务
_poisearch = [[BMKPoiSearch alloc] init];
_poisearch.delegate = self;
//POI详情检索
BMKPoiDetailSearchOption* option = [[BMKPoiDetailSearchOption alloc] init];
option.poiUid = @"此处为POI的uid";//POI搜索结果中获取的uid
BOOL flag = [_poisearch poiDetailSearch:option];
[option release];
if(flag)
{
    //详情检索发起成功
}
else
{
    //详情检索发送失败
}
```

(2) 设置结果监听

```
-(void)onGetPoiDetailResult:(BMKPoiSearch *)searcher result:(BMKPoiDetailResult *)po
iDetailResult errorCode:(BMKSearchErrorCode)errorCode
{
if(errorCode == BMK_SEARCH_NO_ERROR){
//在此处理正常结果
}
}
```

9. 公交详情信息检索

(1) 简介：

POI检索返回的POI结果中，epoitype字段表示POI类型，epoitype字段值为2标示公交路线，4表示地铁路线，把这两种类型的POI的uid传给公交信息检索接口，可以得到该POI所代表的路线的详细信息（如：该公交线有多少个站点，每个站点的名称，位置、参考票价和上下线行信息）。

(2) 发起检索

```
    _searcher =[[BMKBusLineSearch alloc]init]
    _searcher.delegate = self;
    //发起检索
    BMKBusLineSearchOption *buslineSearchOption = [[BMKBusLineSearchOption alloc]init
];
    buslineSearchOption.city= @"北京";
    buslineSearchOption.busLineUid= @"your bus line UID";
    BOOL flag = [_searcher busLineSearch:buslineSearchOption];
    [buslineSearchOption release];
    if(flag)
    {
         NSLog(@"busline检索发送成功");
    }
    else
    {
        NSLog(@"busline检索");
    }
```

(3) 实现PoiSearchDelegate

```
    - (void)onGetBusDetailResult:(BMKBusLineSearch*)searcher result:(BMKBusLineResul
t*)busLineResult errorCode:(BMKSearchErrorCode)error
    {
        if (error == BMK_SEARCH_NO_ERROR) {
            //在此处理正常结果
        }
        else {
            NSLog(@"抱歉，未找到结果");
        }
    }
    //不使用时将delegate设置为 nil
    -(void)viewWillDisappear:(BOOL)animated
    {
        _searcher.delegate = nil;
    }
```

10. 行政区边界数据检索

(1) 发起检索

```
//初始化检索对象
_districtSearch = [[BMKDistrictSearch alloc] init];
//设置delegate，用于接收检索结果
_districtSearch.delegate = self;
//构造行政区域检索信息类
BMKDistrictSearchOption *option = [[BMKDistrictSearchOption alloc] init];
option.city = @"北京";
option.district = @"海淀";
//发起检索
BOOL flag = [_districtSearch districtSearch:option];
if (flag) {
    NSLog(@"district检索发送成功");
} else {
    NSLog(@"district检索发送失败");
}
```

(2) 实现delegate，获取结果

```
/**
 *返回行政区域搜索结果
 *@param searcher 搜索对象
 *@param result 搜索结BMKDistrictSearch果
 *@param error 错误号, @see BMKSearchErrorCode
 */
- (void)onGetDistrictResult:(BMKDistrictSearch *)searcher result:(BMKDistrictResult
*)result errorCode:(BMKSearchErrorCode)error {
    NSLog(@"onGetDistrictResult error: %d", error);
if (error == BMK_SEARCH_NO_ERROR) {
//code
    }
}
```

(3) 可以进行边界多边形绘制

11. 短串分享

(1) 简介：短串分享是指，用户搜索查询后得到的每一个地理位置结果将会对应一条短串（短链接），用户可以通过短信、邮件或第三方分享组件（如微博、微信等）把短串分享给其他用户从而实现地理位置信息的分享。当其他用户收到分享的短串后，点击短串即可打开手机上的百度地图客户端或者手机浏览器进行查看。

(2) 例如，用户搜索“百度大厦”后通过短信使用短串分享功能把该地点分享给好友，好友点击短信中的短串“http://j.map.baidu.com/BkmBk” 后可以调起百度地图客户端或者手机浏览器查看“百度大厦”的地理位置信息。

(3) 目前短串分享功能暂时开放了“POI详情分享”、“驾车/公交/骑行/步行路线规划分享”和“位置信息分享”，日后会开放更多的功能，欢迎广大开发者使用短串分享功能。

(4) 发起检索

```
    //初始化检索对象
    _searcher =[[BMKShareURLSearch alloc]init];
    _searcher.delegate = self;
    //发起短串搜索获取poi分享url
    BMKPoiDetailShareURLOption *detailShareUrlSearchOption = [[BMKPoiDetailShareURLOp
tion alloc]init];
    //从poi检索得到的poi的uid
    detailShareUrlSearchOption.uid==@"your poi uid";
    BOOL flag = [_searcher requestPoiDetailShareURL:detailShareUrlSearchOption];
    [detailShareUrlSearchOption release];
    if(flag)
    {
        NSLog(@"详情url检索发送成功");
    }
    else
    {
        NSLog(@"详情url检索发送失败");
    }
    //发起位置信息分享URL检索
    //    BMKLocationShareURLOption *option = [[BMKLocationShareURLOption alloc]init];
    //    option.snippet = @"上地10街10号";
    //    option.name = @"百度大厦";
    //    option.location = CLLocationCoordinate2D{40.055,116.037};
    //    BOOL flag = [_searcher requestLocationShareURL:option];
    //    [option release];
    //    if(flag)
    //    {
    //         NSLog(@"位置信息分享URL检索发送成功");
    //    }
    //    else
    //    {
    //        NSLog(@"位置信息分享URL发送失败");
    //     }

    }
```

```
    //初始化检索服务
    _shareurlsearch = [[BMKShareURLSearch alloc]init];
    //设置delegate
    _shareurlsearch.delegate = self;

    //构建路线规划短串分享检索信息类
    BMKRoutePlanShareURLOption *option = [[BMKRoutePlanShareURLOption alloc] init];

    //起点
    BMKPlanNode *fromNode = [[BMKPlanNode alloc] init];
    fromNode.name = @"百度大厦";
    fromNode.cityID = 131;
    option.from = fromNode;

    //终点
    BMKPlanNode *toNode = [[BMKPlanNode alloc] init];
    toNode.name = @"天安门";
    toNode.cityID = 131;
    option.to = toNode;
```

```
//    option.routePlanType = BMK_ROUTE_PLAN_SHARE_URL_TYPE_DRIVE;//驾车
//    option.routePlanType = BMK_ROUTE_PLAN_SHARE_URL_TYPE_WALK;//步行
//    option.routePlanType = BMK_ROUTE_PLAN_SHARE_URL_TYPE_RIDE;//骑行
    option.routePlanType = BMK_ROUTE_PLAN_SHARE_URL_TYPE_TRANSIT;//公交
    option.cityID = 131;//当进行公交路线规划短串分享且起终点通过关键字指定时，必须指定
    option.routeIndex = 0;//公交路线规划短串分享时使用，分享的是第几条线路

    //发起检索
    BOOL flag = [_shareurlsearch requestRoutePlanShareURL:option];
    if (flag) {
        NSLog(@"routePlanShortUrlShare检索发送成功");
    } else {
        NSLog(@"routePlanShortUrlShare检索发送失败");
    }
```

(5) 实现代理方法，获取结果、

```
    //实现Delegate处理回调结果
    //处理Poi详情分享URL结果
    - (void)onGetPoiDetailShareUrlResult:(BMKShareUrlSearch *)searcher result:(BMKSh
areURLResult *)result errorCode:(BMKSearchErrorCode)error{
        if (error == BMK_SEARCH_NO_ERROR) {
            //在此处理正常结果
        }
       else {
            NSLog(@"抱歉，未找到结果");
        }
    }

    //处理位置信息你分享了URL结果
    //- (void)onGetLocationShareUrlResult:(BMKShareUrlSearch *)searcher result:(BMKSh
areURLResult *)result errorCode:(BMKSearchErrorCode)error{
    //    if (error == BMK_SEARCH_NO_ERROR) {
    //        在此处理正常结果
    //    }
    //    else {
    //        NSLog(@"抱歉，未找到结果");
    //    }
    //}


    //不使用时将delegate设置为 nil
    -(void)viewWillDisappear:(BOOL)animated
    {
        _searcher.delegate = nil;
    }
```

```
/**
 *返回路线规划分享url
 *@param searcher 搜索对象
 *@param result 返回结果
 *@param error 错误号，@see BMKSearchErrorCode
 */
- (void)onGetRoutePlanShareURLResult:(BMKShareURLSearch *)searcher result:(BMKShareU
RLResult *)result errorCode:(BMKSearchErrorCode)error {
    NSLog(@"onGetRoutePlanShareURLResult error:%d", (int)error);
    if (error == BMK_SEARCH_NO_ERROR) {
        //share shore url is result.url
        //share code
    }
}
```

12. 空间计算

根据用户指定的两个坐标点，计算这两个点的实际地理距离。核心代码如下:

```
BMKMapPoint point1 = BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.915,116.4
04));
BMKMapPoint point2 = BMKMapPointForCoordinate(CLLocationCoordinate2DMake(38.915,115.4
04));
CLLocationDistance distance = BMKMetersBetweenMapPoints(point1,point2);
```

13. 坐标转换

百度地图SDK采用的是百度自有的地理坐标系（bdll09），因此开发者在做位置标注的时候，需要将其他类型的坐标转换为百度坐标。相应的接口和转换方式如下:

```
CLLocationCoordinate2D coor = CLLocationCoordinate2DMake(39.90868, 116.3956);//原始坐
标
//转换 google地图、soso地图、aliyun地图、mapabc地图和amap地图所用坐标至百度坐标
NSDictionary* testdic = BMKConvertBaiduCoorFrom(coor,BMK_COORDTYPE_COMMON);
//转换GPS坐标至百度坐标(加密后的坐标)
testdic = BMKConvertBaiduCoorFrom(coor,BMK_COORDTYPE_GPS);
NSLog(@"x=%@,y=%@",[testdic objectForKey:@"x"],[testdic objectForKey:@"y"]);
//解密加密后的坐标字典
CLLocationCoordinate2D baiduCoor = BMKCoorDictionaryDecode(testdic);//转换后的百度坐标
```

14. 空间关系判断

提供相应的接口能力，判断点与圆或多边形的位置关系。

判断点与圆位置关系的示例代码如下：

```
BOOL ptInCircle = BMKCircleContainsCoordinate(CLLocationCoordinate2DMake(39.918,116.4
08), CLLocationCoordinate2DMake(39.915,116.404), 1000);
```

除以上位置关系判断方法外，SDK还提供获取折线上与折线外指定位置最近点的方法。核心代码如下：

```
BMKMapPoint *polylinePoints = new BMKMapPoint[4];
polylinePoints[0]= BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.915,116.404
));
polylinePoints[1]= BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.915,116.454
));;
polylinePoints[2]= BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.975,116.524
));;
polylinePoints[3]= BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.855,116.554
));
BMKMapPoint point = BMKMapPointForCoordinate(CLLocationCoordinate2DMake(39.815,116.50
4));
BMKMapPoint nearestPoint = BMKGetNearestMapPointFromPolyline(point, polylinePoints, 4
);
```

15. 收藏夹功能

iOS地图SDK自v2.8.0版本起，向开发者开放了本地收藏夹功能，帮助开发者更好的收藏、管理本地空间点信息数据。

具体使用方法如下：

1、 初始化收藏夹管理类：

```
BMKFavPoiManager *_favManager = [[BMKFavPoiManageralloc] init];//初始化收藏夹管理类
```

2、 添加一个收藏点，核心代码如下：

```
//构造收藏点信息
BMKFavPoiInfo *poiInfo = [[BMKFavPoiInfoalloc] init];
    poiInfo.pt = CLLocationCoordinate2DMake(39.908, 116.204);//收藏点坐标
poiInfo.poiName = @"收藏点名称";//收藏点名称
//添加收藏点(收藏点功后会得到favId)
NSInteger res = [_favManageraddFavPoi:poiInfo];
```

3、 获取收藏点，核心代码如下：

```
//获取所有收藏点
NSArray *allFavPois = [_favManagergetAllFavPois];
//获取某个收藏点(收藏点成功后会得到favId)
BMKFavPoiInfo *favPoi = [_favManagergetFavPoi:favId];
```

4、 删除收藏的点，核心代码如下：

```
//删除所有收藏点
BOOL res = [_favManagerclearAllFavPois];
//删除某个收藏点(收藏点成功后会得到favId)
BOOL res = [_favManagerdeleteFavPoi:favId];
```

16. 基础地图

(1) 简介

1) 开发者可利用SDK提供的接口，使用百度为您提供的基础地图数据。目前百度地图SDK所提供的地图等级为3-21级，所包含的信息有建筑物、道路、河流、学校、公园等内容。

2) 百度地图支持多点触摸、双击放大、多点单击缩小、旋转等手势操作，此外自2.2.0版本起，支持相应的控制接口来开启/关闭这些手势操作；此外，在该版本地图对象实现了多实例特性，即开发者可以在一个页面中建立多个地图对象，并且针对这些对象分别操作且不会产生相互干扰。具体使用方法请参考MultiMapViewDemo的介绍。

3) 地图上自定义的标注点和覆盖物我们统称为地图覆盖物。您可以通过定制BMKAnnotation和BMKOverlay来添加对应的标注点和覆盖物。地图覆盖物的设计遵循数据与View分离的原则，BMKAnnotation和BMKOverlay系列的类主要用来存放覆盖物相关的数据，BMKAnnotaionView和BMKOverlayView系列类为覆盖物对应的View。

4) SDK支持画点、折线、圆、多边形（包括凹凸两种）、图片图层和自定义覆盖物。从2.0.0开始矢量地图采用OpenGL绘制，新增支持OpenGL绘制的基本线绘制、面绘制接口。详见AnnotationDemo，SDK内置的BMKPolylineOverlay、BMKPolygonOverlay，BMKCircleOverlay均采用OpenGL绘制。

(2) 地图类型

百度地图SDK为您提供了3种类型的地图资源（普通矢量地图、卫星图和空白地图），开发者可以利用BMKMapView中的mapType属性来设置地图类型。空白地图，基础地图瓦片将不会被渲染。在地图类型中设置为BMKMapTypeNone，将不会使用流量下载基础地图瓦片图层。使用场景：与瓦片图层一起使用，节省流量，提升自定义瓦片图下载速度。

设置空白地图的方法如下：

```
_mapView.mapType = BMKMapTypeNone;//设置地图为空白类型
```

开启卫星图的方法如下：

```
//切换为卫星图
[_mapView setMapType:BMKMapTypeSatellite];
```

由卫星图切换为普通矢量图的核心代码如下：

```
//切换为普通地图
[_mapView setMapType:BMKMapTypeStandard];
```

(3) 实时交通图

利用地图SDK所提供的接口，开发者可显示当前地图城市内的实时路况信息此外，自2.0.0版本起，SDK还支持城际（城市之间，如高速）路况。//打开实时路况图层

```
[_mapView setTrafficEnabled:YES];
```

关闭实时路况的核心代码如下：

```
//关闭实时路况图层
[_mapView setTrafficEnabled:NO];
```

(4) 地图控制和手势

地图Logo

默认在左下角显示，不可以移除。 通过logoPosition属性，使用枚举类型控制显示的位置，共支持6个显示位置(左下，中下，右下，左上，中上，右上)。

地图Logo不允许遮挡，可通过mapPadding属性可以设置地图边界区域，来避免UI遮挡。

指南针

指南针默认为开启状态，可以关闭显示。

比例尺

比例尺默认为开启状态，可以关闭显示。同时支持设置MaxZoomLevel和minZoomLevel。

地图平移

控制是否启用或禁用平移的功能，默认开启。如果启用，则用户可以平移地图

地图缩放

控制是否启用或禁用缩放手势，默认开启。如果启用，用户可以双指点击或缩放地图视图。

地图俯视（3D）

控制是否启用或禁用俯视（3D）功能，默认开启。如果启用，则用户可使用双指 向下或向上滑动到俯视图。

地图旋转

控制是否启用或禁用地图旋转功能，默认开启。如果启用，则用户可使用双指 旋转来旋转地图。

3D-Touch手势

自2.10.0起，支持3D Touch回调，可控制是否开启或关闭回调3D-Touch手势，默认为关闭。

禁止所有手势

控制是否一并禁止所有手势，默认关闭。如果启用，所有手势都将被禁用。

(5) 地图标注

BMKAnnotation为标注对应的protocal，您可以自定义标注类实现该protocal。百度地图SDK也预置了基本的标注点（BMKPointAnnotation）和一个大头针标注View（BMKPinAnnotationView），您可以直接使用来显示标注

第一步，修改您的ViewController.h文件，添加以下代码，使您的ViewController实现BMKMapViewDelegate协议：

```
@interface AnnotationDemoViewController : UIViewController <bmkmapviewdelegate> {
    IBOutlet BMKMapView* _mapView;
}
@end
```

第二步，修改您的ViewController.m文件，实现BMKMapViewDelegate的_mapView:viewForAnnotation:函数，并在viewDidAppear添加标注数据对象，核心代码如下：

```
- (void) viewDidAppear:(BOOL)animated {
        // 添加一个PointAnnotation
        BMKPointAnnotation* annotation = [[BMKPointAnnotation alloc]init];
        CLLocationCoordinate2D coor;
        coor.latitude = 39.915;
        coor.longitude = 116.404;
        annotation.coordinate = coor;
        annotation.title = @"这里是北京";
        [_mapView addAnnotation:annotation];
    }
// Override
- (BMKAnnotationView *)mapView:(BMKMapView *)mapView viewForAnnotation:(id <BMKAnnota
tion>)annotation
    {
        if ([annotation isKindOfClass:[BMKPointAnnotation class]]) {
            BMKPinAnnotationView *newAnnotationView = [[BMKPinAnnotationView alloc]
initWithAnnotation:annotation reuseIdentifier:@"myAnnotation"];
            newAnnotationView.pinColor = BMKPinAnnotationColorPurple;
            newAnnotationView.animatesDrop = YES;// 设置该标注点动画显示
            return newAnnotationView;
        }
        return nil;
    }
```

通过以上几步简单的操作，您就可以实现在地图上添加标注了，相应的删除标注方法如下：

```
if (annotation != nil) {
    [_mapView removeAnnotation:annotation];
}
```

(6) 点聚合

1、声明点聚合管理类为全局变量，并初始化，核心代码如下

```
BMKClusterManager *_clusterManager;//点聚合管理类

//初始化点聚合管理类

_clusterManager = [[BMKClusterManager alloc] init];
```

2、向点聚合管理类中添加点，核心代码如下

```
CLLocationCoordinate2D coor = CLLocationCoordinate2DMake(39.915, 116.404);
    //向点聚合管理类中添加标注
    for (NSInteger i = 0; i < 20; i++) {
        double lat =  (arc4random() % 100) * 0.001f;
        double lon =  (arc4random() % 100) * 0.001f;
        BMKClusterItem *clusterItem = [[BMKClusterItem alloc] init];
        clusterItem.coor = CLLocationCoordinate2DMake(coor.latitude + lat, coor.longi
tude + lon);
        [_clusterManager addClusterItem:clusterItem];
    }
```

3.获取聚合后的点，并添加到地图中，核心代码如下

```
///获取聚合后的标注
    NSArray *array = [_clusterManager getClusters:_clusterZoom];
    NSMutableArray *clusters = [NSMutableArray array];
    for (BMKCluster *item in array) {
        ClusterAnnotation *annotation = [[ClusterAnnotation alloc] init];
        annotation.coordinate = item.coordinate;
        annotation.size = item.size;
        annotation.title = [NSString stringWithFormat:@"我是%ld个", item.size];
        [clusters addObject:annotation];
    }
    [_mapView removeAnnotations:_mapView.annotations];
    [_mapView addAnnotations:clusters];
```

具体源码请在ClusterDemo中查看。

(7) 底图标注

自v2.9.0版本起，SDK给BMKMapView提供了控制底图标注的showMapPoi方法默认显示底图标注。利用此方法可得到仅显示道路信息的地图。

```
//设置隐藏地图标注

[_mapView setShowMapPoi:NO];
```

(8) 几何图形绘制

1) 折线

第一步，修改您的ViewController.h文件，添加以下代码，使您的ViewController实现BMKMapViewDelegate协议：

```
@interface OverlayDemoViewController : UIViewController <BMKMapViewDelegate>{
    IBOutlet BMKMapView* _mapView;
}
@end
```

第二步，修改您的ViewController.m文件，实现BMKMapViewDelegate的mapView:viewForOverlay:函数，并在viewDidLoad添加折线数据对象：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // 添加折线覆盖物
    CLLocationCoordinate2D coors[2] = {0};
    coors[0].latitude = 39.315;
    coors[0].longitude = 116.304;
    coors[1].latitude = 39.515;
    coors[1].longitude = 116.504;
    BMKPolyline* polyline = [BMKPolyline polylineWithCoordinates:coors count:2];
    [_mapView addOverlay:polyline];
}
// Override
- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id <BMKOverlay>)ove
rlay{
   if ([overlay isKindOfClass:[BMKPolyline class]]){
        BMKPolylineView* polylineView = [[[BMKPolylineView alloc] initWithOverlay:ove
rlay] autorelease];
        polylineView.strokeColor = [[UIColor purpleColor] colorWithAlphaComponent:1];
        polylineView.lineWidth = 5.0;

        return polylineView;
   }
   return nil;
}
```

2) 多纹理折线

```
//构建顶点数组
CLLocationCoordinate2Dcoords[5] = {0};
coords[0].latitude = 39.965;
coords[0].longitude = 116.404;
coords[1].latitude = 39.925;
coords[1].longitude = 116.454;
coords[2].latitude = 39.955;
coords[2].longitude = 116.494;
coords[3].latitude = 39.905;
coords[3].longitude = 116.654;
coords[4].latitude = 39.965;
coords[4].longitude = 116.704;
//构建分段纹理索引数组
NSArray *textureIndex = [NSArrayarrayWithObjects:
                                [NSNumbernumberWithInt:0],
                                [NSNumbernumberWithInt:1],
                                [NSNumbernumberWithInt:2],
                                [NSNumbernumberWithInt:1], nil];

//构建BMKPolyline,使用分段纹理
BMKPolyline* polyLine = [BMKPolylinepolylineWithCoordinates:coordscount:5textureIndex
:textureIndex];
//添加分段纹理绘制折线覆盖物
    [_mapViewaddOverlay:polyLine];
```

第二步，实现BMKMapViewDelegate的mapView:viewForOverlay:回调，核心代码如下：

```
- (BMKOverlayView*)mapView:(BMKMapView *)map viewForOverlay:(id<BMKOverlay>)overlay
{
if ([overlay isKindOfClass:[BMKPolylineclass]]) {
BMKPolylineView* polylineView = [[BMKPolylineViewalloc] initWithOverlay:overlay];
polylineView.lineWidth = 5;
polylineView.isFocus = YES;// 是否分段纹理绘制（突出显示），默认YES
//加载分段纹理图片，必须否则不能进行分段纹理绘制
        [polylineView loadStrokeTextureImages:
         [NSArrayarrayWithObjects:[UIImageimageNamed:@"road_blue_arrow.png"],
          [UIImageimageNamed:@"road_green_arrow.png"],
          [UIImageimageNamed:@"road_red_arrow.png"],nil]];
returnpolylineView;
    }
returnnil;
}
```

3) 多颜色折线

第一步，同2)的第一步

第二步，实现BMKMapViewDelegate回调，核心代码如下：

```
//根据overlay生成对应的View
- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id <BMKOverlay>)ove
rlay
{
    if ([overlay isKindOfClass:[BMKPolyline class]]) {
        BMKPolylineView* polylineView = [[BMKPolylineView alloc] initWithOverlay:over
lay];
        polylineView.lineWidth = 5;
        /// 使用分段颜色绘制时，必须设置（内容必须为UIColor）
        polylineView.colors = [NSArray arrayWithObjects:[UIColor greenColor], [UIColo
r redColor], [UIColor yellowColor], nil];
        return polylineView;
    }

    return nil;
}
```

4) 弧线

百度地图iOS SDK自v2.1.1本起，新增了绘制弧线的方法。用户可以根据三个有序点唯一确定一条弧线，满足您的业务需求。

```
    - (void)viewDidLoad {
        [super viewDidLoad];
    //添加弧线覆盖物
    //传入的坐标顺序为起点、途经点、终点
    CLLocationCoordinate2D coords[3] = {0};
    coords[0].latitude = 39.9374;
    coords[0].longitude = 116.350;
    coords[1].latitude = 39.9170;
    coords[1].longitude = 116.360;
    coords[2].latitude = 39.9479;
    coords[2].longitude = 116.373;

    BMKArcline *arcline = [BMKArcline arclineWithCoordinates:coords];
       [_mapView addOverlay:arcline];

    }
    //根据overlay生成对应的View
    - (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id<bmkoverlay>)
overlay
    {
    if ([overlay isKindOfClass:[BMKArcline class]])
        {
            BMKArclineView* arclineView = [[[BMKArclineView alloc] initWithOverlay:ov
erlay] autorelease];
            arclineView.strokeColor = [[UIColor blueColor] colorWithAlphaComponent:0.
5];
            arclineView.lineWidth = 5.0;

            return arclineView;
            }
        return nil;
    }
```

5）多边形

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // 添加多边形覆盖物
    CLLocationCoordinate2D coords[3] = {0};
    coords[0].latitude = 39;
    coords[0].longitude = 116;
    coords[1].latitude = 38;
    coords[1].longitude = 115;
    coords[2].latitude = 38;
    coords[2].longitude = 117;
    BMKPolygon* polygon = [BMKPolygon polygonWithCoordinates:coords count:3];

    [_mapView addOverlay:polygon];
}
// Override
- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id <BMKOverlay>)ove
rlay{
    if ([overlay isKindOfClass:[BMKPolygon class]]){
        BMKPolygonView* polygonView = [[[BMKPolygonView alloc] initWithOverlay:overla
y] autorelease];
        polygonView.strokeColor = [[UIColor purpleColor] colorWithAlphaComponent:1];
        polygonView.fillColor = [[UIColor cyanColor] colorWithAlphaComponent:0.2];
        polygonView.lineWidth = 5.0;

        return polygonView;
    }
    return nil;
}
```

6）圆

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // 添加圆形覆盖物
    CLLocationCoordinate2D coor;
    coor.latitude = 39.915;
    coor.longitude = 116.404;
    BMKCircle* circle = [BMKCircle circleWithCenterCoordinate:coor radius:5000];

    [_mapView addOverlay:circle];
}
```

```
// Override
- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id <BMKOverlay>)ove
rlay{
    if ([overlay isKindOfClass:[BMKCircle class]]){
        BMKCircleView* circleView = [[[BMKCircleView alloc] initWithOverlay:overlay]
autorelease];
        circleView.fillColor = [[UIColor cyanColor] colorWithAlphaComponent:0.5];
        circleView.strokeColor = [[UIColor blueColor] colorWithAlphaComponent:0.5];
        circleView.lineWidth = 10.0;

        return circleView;
    }
    return nil;
}
```

7) 地形图图层

自V2.1.0开始，新增图片图层，开发者可在地图的指定位置上添加图片。该图片可随地图的平移、缩放、旋转等操作做相应的变换。图片图层是一种特殊的Overlay， 它位于底图和底图标注层之间（即图片图层不会遮挡地图标注信息）， 此外，图片图层的添加顺序不会影响其他图层（例如：POI搜索图层、我的位置图层等）的叠加关系。

图片图层对象初始化的方法有两种：(1)根据指定经纬度坐标生成 (2)根据指定区域生成。

第一步，修改您的ViewController.h文件，添加以下代码，使您的ViewController实现BMKMapViewDelegate协议：

```
@interface AnnotationDemoViewController : UIViewController<BMKMapViewDelegate>{
        IBOutlet BMKMapView* _mapView;
   }
end
```

第二步，修改您的ViewController.m文件，在viewDidLoad添加图片图层对象：

```
- (void)viewDidLoad {
        [super viewDidLoad];
        //添加图片图层覆盖物(第一种:根据指定经纬度坐标生成)
        CLLocationCoordinate2D coors;
        coors.latitude = 39.800;
        coors.longitude = 116.404;
        BMKGroundOverlay* ground = [BMKGroundOverlay groundOverlayWithPosition:coors
        zoomLevel:11 anchor:CGPointMake(0.0f,0.0f)
        icon:[UIImage imageWithName:@"test.png"]];
        [_mapView addOverlay:ground];

        //添加图片图层覆盖物(第二种:根据指定区域生成)
        CLLocationCoordinate2Dcoords[2] = {0};
        coords[0].latitude = 39.815;
        coords[0].longitude = 116.404;
        coords[1].latitude = 39.915;
        coords[1].longitude = 116.504;
        BMKCoordinateBounds bound;
        bound.southWest = coords[0];
        bound.northEast = coords[1];
        BMKGroundOverlay* ground2 = [BMKGroundOverlay groundOverlayWithBounds: bound
        icon:[UIImage imageWithName:@"test.png"]];
        [_mapView addOverlay:ground2];

}
```

第三步，修改您的ViewController.m文件，实现BMKMapViewDelegate的_mapView:viewForOverlay:函数：

```
- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id<BMKOverlay>)over
lay{
        if ([overlay isKindOfClass:[BMKGroundOverlay class]]){
            BMKGroundOverlayView* groundView = [[[BMKGroundOverlayView alloc]
initWithOverlay:overlay] autorelease];
            return groundView;
        }
        return nil;
}
```

(9) 热力图功能

热力图是用不同颜色的区块叠加在地图上描述人群分布、密度和变化趋势的一个产品，百度地图SDK将绘制热力图的能力为广大开发者开放，帮助开发者利用自有数据，构建属于自己的热力图，提供丰富的展示效果。

注意：此处的“热力图功能”不同于“百度城市热力图”。百度城市热力图通过简单的接口调用，开发者可展示百度数据的热力图层。而此处的热力图功能，需要开发者传入自己的位置数据，然后SDK会根据热力图绘制规则为开发者做本地的热力图渲染绘制

(10) 自定义覆盖物

从2.0.0开始，地图渲染采用OpenGL方式实现，因此覆盖物基类BMKOverlayView新增glRender接口，以及绘制基本线renderLinesWithPoints、面renderRegionWithPoints的接口来实现对覆盖物的OpenGL渲染。绘制自定义overlay时，继承BMKOverlayView的子类需实现glRender接口，在glRender中通过调用renderLinesWithPoints、renderRegionWithPoints来组合自己想要实现的图形。

(11) OpenGL绘制功能

自v2.6.0起，iOS地图SDK为广大开发者开放了OpenGL绘制功能，开发者可利用OpenGL的绘制来实现更多复杂的覆盖物绘制

(12) 瓦片图层

iOS地图SDK自v2.9.0起，新增瓦片图层（tileOverlay），该图层支持开发者添加自有瓦片数据，包括本地加载和在线下载两种方式。

该图层可随地图的平移、缩放、旋转等操作做相应的变换，它仅位于底图之上（即瓦片图层将会遮挡底图，不遮挡其他图层），瓦片图层的添加顺序不会影响其他图层（例如：POI搜索图层、我的位置图层等）的叠加关系，适用于开发者拥有某一区域的地图，并希望使用此区域地图覆盖相应位置的百度地图

(13) 设置地图区域边界

v2.10.0起,新增了mapPadding方法，支持设置地图区域边界，在被定义边距范围内，对显示和操作地图，做了如下两方面的定义。

(1) 百度logo、比例尺、指南针等，可被控制在自定义的地图区域边界内。

(2) 自适应MapStatus中心点坐标，由原屏幕中心点调整至设置的区域中心点。

当设计的UI与地图部分重叠时，可以设置地图的操作和显示范围,以防止UI遮挡和重叠

给地图设置边界，设置方法如下：

```
// paddingLeft、 paddingTop、 paddingRight、 paddingBottom
// 表示距离屏幕左、上、右、下边距离，单位为屏幕坐标下的像素密度

_mapView.mapPadding = UIEdgeInsetsMake(0, 0, 28, 0);
```

(14) 设置地图显示范围

v2.10.0起新增了设置地图显示范围，手机屏幕仅显示设定的地图范围，当前不支持旋转地图的情况，请与"禁用旋转手势"配合使用。

使用场景：针对需要展示部分固定范围的地图，如希望设置仅显示北京市区地图，可使用此功能。

使用方法如下:

```
CLLocationCoordinate2D center = CLLocationCoordinate2DMake(39.924257, 116.403263);
BMKCoordinateSpan span = BMKCoordinateSpanMake(0.038325, 0.028045);
_mapView.limitMapRegion = BMKCoordinateRegionMake(center, span);////限制地图显示范围
_mapView.rotateEnabled = NO;//禁用旋转手势
```

17. 室内地图
    (1) 室内图
    (2) 室内poi
    (3) 室内路线规划
18. 个性化地图

自v2.10.0起，支持使用个性化地图模板，改变底图颜色和样式。

使用个性化模板，实现地图元素的颜色设置，地图元素包含大地、水系、草地、高速、普通道路、铁路、地铁，poi等，以及poi和道路的文字颜色设置。通过可见属性，控制显示地图元素。

V2.10.2提供3个标准模板样式文件和一个空白样式文件 （点击下载） 供大家使用，后续还会增加模板。下个版本，我们将进一步优化此功能，让广大开发者可以通过可视化编辑器方式编辑样式模板。

个性化地图功能位于“基础地图（Map）”这个功能模块，开发者使用时请注意选择。

使用个性化地图样式文件的方法如下：

1、 将所需模板(如：custom_config)放入工程目录

2、 设定地图样式文件的路径，方法如下：

```
注：必须在BMKMapView对象初始化之前设置

    //个性化地图模板文件路径

    NSString* path = [[NSBundle mainBundle] pathForResource:@"custom_config" ofType:@
""];

//设置个性化地图样式

[BMKMapView customMapStyle:path];
```

自v3.0起，支持个性化地图和普通地图切换。设置个性化地图后，个性化地图默认为开启状态。

关闭个性化地图方法如下：

```
 [BMKMapView enableCustomMapStyle:NO];//关闭个性化地图
```

通过此功能，可实现App的夜间和普通地图切换的需求。

19. LBS.云检索
20. 离线地图