
百度地图移动版 API for ios 开发指南

2012.3.29

Copyright @ Baidu.com

目 录

1 简介.....	3
1.1 什么是百度地图 API.....	3
1.2 获取 API Key	3
1.3 兼容性.....	3
2 在您的程序中显示地图	3
2.1 引入百度 MapApi 的头文件	3
2.2 引入静态库文件	4
2.3 引入 CoreLocation.framework 和 QuartzCore.framework	4
2.4 引入 mapapi.bundle 资源文件.....	5
2.5 初始化 BMKMapManager.....	5
2.6 创建 BMKMapView	6
3 注意事项.....	6
4 卫星图图层.....	7
5 实时路况图层.....	7
6 地图覆盖物.....	8
5.1 添加标注.....	8
5.2 删除标注	10
5.3 添加折线.....	11
5.4 添加多边形	12
5.5 添加圆.....	13
5.6 删除 Overlay	14
7 服务类.....	15
6.1 POI 检索	16
6.2 公交方案检索	18
6.3 驾车路线检索	19
6.4 步行路线检索	21
6.5 地理编码.....	22
6.6 反地理编码.....	23
8 定位.....	24
9 离线地图.....	24

1 简介

1.1 什么是百度地图 API

百度地图移动版 API (IOS) 是一套基于 iOS3.0 及以上设备的应用程序接口, 通过该接口, 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。百度地图移动版 API 不仅包含构建地图的基本接口, 还提供了诸如地图定位、本地搜索、路线规划等数据服务, 您可以根据自己的需要进行选择, 目前支持 iPhone3.0 以上的版本, 对 iPad 暂不支持。

面向的读者

API 是提供给那些具有一定 iOS 编程经验和了解面向对象概念的读者使用。此外, 读者还应该对地图产品有一定的了解。您在使用中遇到任何问题, 都可以通过 [API 贴吧](#)或[交流群](#)反馈给我们。

1.2 获取 API Key

用户在使用 API 之前需要获取百度地图移动版 API Key, APIKey 可跨平台使用, 如果您已经有 Android 平台的授权 Key, 可直接在 iOS 平台使用。该 Key 与您的百度账户相关联, 您必须先有百度帐户, 才能获得 API KEY。并且, 该 KEY 与您引用 API 的程序名称有关, 具体流程请参照[获取密钥](#)。

1.3 兼容性

支持 iOS3.0 及以上系统, 百度地图 API 接口与 iOS 内置的 MapKit 包兼容, 开发者只需很小的改动即可完成从 MapKit 到百度地图 API 的迁移。并且迁移到百度地图 API 之后很多 MapKit 中只有 iOS4.0 以上版本才能使用的特性接口也可以正常使用了。

2 在您的程序中显示地图

完整的 Demo 例程可参考[相关下载](#)。

2.1 引入百度 MapApi 的头文件

首先将百度 MapAPI 提供的头文件和静态库(.a)文件拷贝到您的工程目录下, 在 XCode 中添加新的文件 Group, 引入百度 MapAPI 提供的头文件(请使用 xcode 4.X 以上平台)。

在您需要使用百度 MapAPI 的文件中添加以下代码

```
#import "BMapKit.h"
```

2.2 引入静态库文件

百度 MapAPI 提供了模拟器和真机两中环境所使用的静态库文件，分别存放在 `libs/Release-iphonesimulator` 和 `libs/Release-iphoneos` 文件夹下。有两种方式可以引入静态库文件：

第一种方式：直接将对应平台的.a 文件拖拽至 XCode 工程左侧的 Groups&Files 中，缺点是每次在真机和模拟器编译时都需要重新添加.a 文件；

第二种方式：使用 `lipo` 命令将设备和模拟器的.a 合并成一个通用的.a 文件，将合并后的通用.a 文件拖拽至工程中即可，具体命令如下：

```
lipo -create Release-iphoneos/libbaidumapapi.a Release-iphonesimulator/libbaidumapapi.a -  
output libbaidumapapi.a
```

第三种方式：

1.将 API 的 libs 文件夹拷贝到您的 Application 工程跟目录下

2.在 XCode 的 Project -> Edit Active Target -> Build -> Linking -> Other Linker Flags 中添加 `-lbaidumapapi`

3.设置静态库的链接路径，在 XCode 的 Project -> Edit Active Target -> Build -> Search Path -> Library Search Paths 中添加您的静态库目录，比如

`"$(SRCROOT)/../libs/Release$(EFFECTIVE_PLATFORM_NAME)"`，`$(SRCROOT)`宏代表您的工程文件目录，`$(EFFECTIVE_PLATFORM_NAME)`宏代表当前配置是 OS 还是 simulator

注：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，即将 XCode 的 Project -> Edit Active Target -> Build -> GCC4.2 - Language -> Compile Sources As 设置为 **"Objective-C++"**

2.3 引入 CoreLocation.framework 和 QuartzCore.framework

百度 MapAPI 中提供了定位功能和动画效果，因此您需要在您的 XCode 工程中引入 `CoreLocation.framework` 和 `QuartzCore.framework`。添加方式：右键点击 Xcode 工程左侧的 Frameworks 文件夹，add->Existing Frameworks,在弹出窗口中选中这两个 framework，点击 add 即可。

2.4 引入 mapapi.bundle 资源文件

该步骤为可选，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。如果您不需要使用内置的图片显示功能，则可以不添加此 bundle 文件。您也可以根据具体需求任意替换或删除该 bundle 中的图片文件。

添加方式：将 mapapi.bundle 拷贝到您的工程目录，直接将该 bundle 文件拖拽至 XCode 工程左侧的 Groups&Files 中即可。

2.5 初始化 BMKMapManager

在您的 AppDelegate.h 文件中添加 BMKMapManager 的定义

```
1. @interface BaiduMapApiDemoAppDelegate : NSObject
2. <UIApplicationDelegate> {
3.     UIWindow *window;
4.     UINavigationController *navigationController;
5.     BMKMapManager* _mapManager;
6. }
```

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

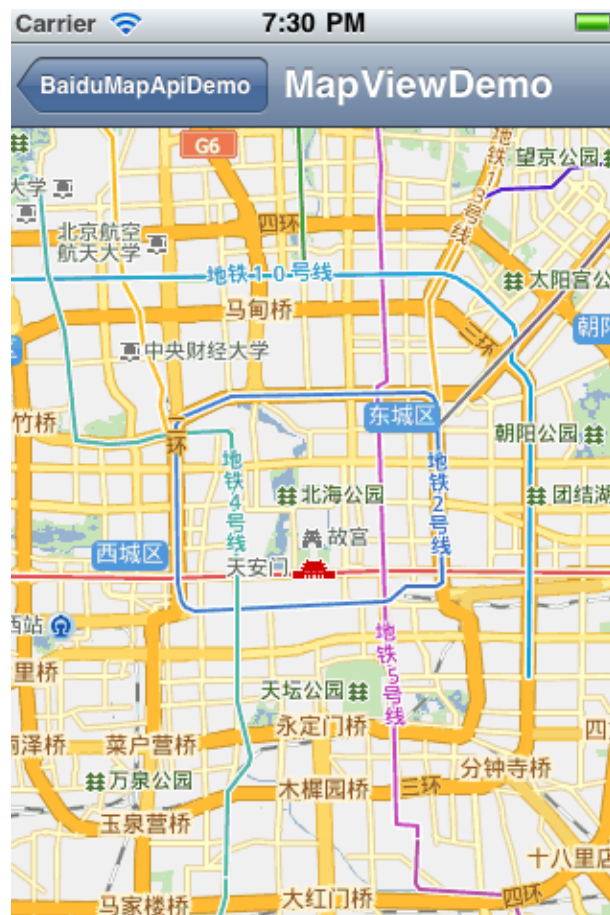
```
1. - (BOOL)application:(UIApplication *)application
2. didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
3.     // 要使用百度地图，请先启动 BaiduMapManager
4.     _mapManager = [[BMKMapManager alloc] init];
5.     // 如果要关注网络及授权验证事件，请设定 generalDelegate 参数
6.     BOOL ret = [_mapManager start:@"在此处输入您的授权 Key"
7.         generalDelegate:nil];
8.     if (!ret) {
9.         NSLog(@"manager start failed!");
10.    }
11.    // Add the navigation controller's view to the window
12.    // and display.
13.    [self.window addSubview:navigationController.view];
14.    [self.window makeKeyAndVisible];
15.    return YES;
16. }
```

2.6 创建 BMKMapView

在您的 ViewController.m 文件中添加 BMKMapView 的创建代码，示例如下

```
1. - (void) viewDidLoad {  
2.     [super viewDidLoad];  
3.     BMKMapView* mapView = [[BMKMapView  
        alloc] initWithFrame: CGRectMake(0, 0, 320, 480)];  
4.     self.view = mapView;  
5. }
```

编译，运行，效果如下图所示：



默认地图已经可以支持多点触摸，双击放大，多点单击缩小等操作，并都附带动画效果。

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，即将 XCode 的

Project -> Edit Active Target -> Build -> GCC4.2 - Language -> Compile Sources As 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合，需要在工程属性中显式设定：在 XCode 的 Project -> Edit Active Target -> Build -> Linking -> Other Linker Flags 中添加-all_load

3.授权 Key 的申请：授权 Key 可跨平台使用，如果您已经申请过 Android 的 key，可直接在 iOS 中使用；如果还没有授权 Key，请到 <http://dev.baidu.com/wiki/static/imap/key/> 页面申请

4 卫星图图层

```
[mapView setMapType:BMKMapTypeSatellite];
```

运行后效果如下：



5 实时路况图层

目前支持以下 11 个城市的实时路况信息：北京，上海，广州，深圳，南京，南昌，成都，重庆，武汉，大连，常州。在地图中通过以下代码设置显示实时路况图层：

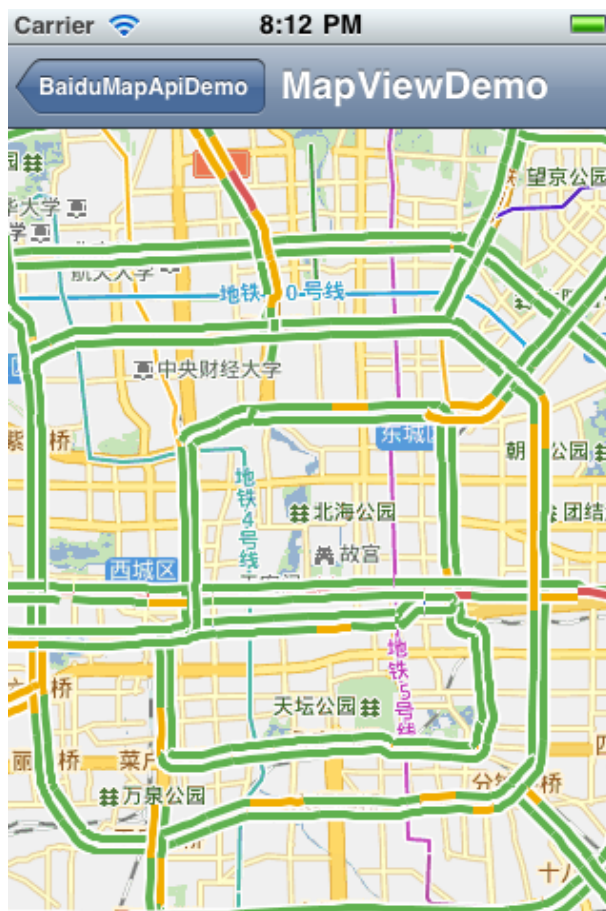
打开实时路况图层：

```
[mapView setMapType:BMKMapTypeTrafficOn];
```

关闭实时路况图层:

```
[mapView setMapType:BMKMapTypeTrafficOff];
```

运行后效果如下:



6 地图覆盖物

覆盖物概述

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，**BMKAnnotationView** 和 **BMKOverlayView** 系列类为覆盖物对应的 **View**。

5.1 添加标注

BMKAnnotation 为标注对应的 **protocol**，您可以自定义标注类实现该 **protocol**。百度地图 API 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。示例如下：

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

```
1. #import <UIKit/UIKit.h>
2. #import "BMapKit.h"
3. @interface AnnotationDemoViewController : UIViewController
    <BMKMapViewDelegate>{
4.     IBOutlet BMKMapView* mapView;
5. }

6. @end
```

修改您的 **ViewController.m** 文件，实现 **BMKMapViewDelegate** 的 **mapView:viewForAnnotation:** 函数，并在 **viewDidLoad** 添加标注数据对象

```
1. // Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
2. - (void) viewDidLoad {
3.     [super viewDidLoad];
4.     // 设置 mapView 的 Delegate
5.     mapView.delegate = self;
6.     // 添加一个 PointAnnotation
7.     BMKPointAnnotation* annotation = [[BMKPointAnnotation alloc] init];
8.     CLLocationCoordinate2D coor;
9.     coor.latitude = 39.915;
10.     coor.longitude = 116.404;
11.     annotation.coordinate = coor;
12.     annotation.title = @"这里是北京";
13.     [mapView addAnnotation:annotation];
14. }
15. // Override
16. - (BMKAnnotationView *) mapView: (BMKMapView *) mapView viewForAnnotation: (id <BMKAnnotation>) annotation
17. {
```

```

18.         if ([annotation isKindOfClass:[BMKPointAnnotation class]])
19.         {
20.             BMKPinAnnotationView *newAnnotationView = [[BMKPinA
nnotationView alloc] initWithAnnotation:annotation reuseIdentifier:
@"myAnnotation"];
21.             newAnnotationView.pinColor = BMKPinAnnotationColorP
urple;
22.             newAnnotationView.animatesDrop = YES; // 设置该标注点动
画显示
23.             return newAnnotationView;
24.         }
25.         return nil;
26.     }

```

运行后，会在地图显示对应的标注点，点击会弹出气泡，效果如图：



5.2 删除标注

通过 `removeAnnotation:` 函数实现对已添加标注的删除功能，示例如下：

```
1.         if (annotation != nil) {
2.             [mapView removeAnnotation:annotation];
3.         }
```

5.3 添加折线

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

```
1. #import <UIKit/UIKit.h>
2. #import "BMapKit.h"

3. @interface OverlayDemoViewController : UIViewController <BMKMapViewDe
    delegate>{
4.     IBOutlet BMKMapView* mapView;
5. }
6. @end
```

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：

```
1. - (void)viewDidLoad {
2.     [super viewDidLoad];
3.     // 设置 delegate
4.     mapView.delegate = self;
5.     // 添加折线覆盖物
6.     CLLocationCoordinate2D coors[2] = {0};
7.     coors[0].latitude = 39.315;
8.     coors[0].longitude = 116.304;
9.     coors[1].latitude = 39.515;
10.    coors[1].longitude = 116.504;
11.    BMKPolyline* polyline = [BMKPolyline polylineWithCoordinate
        s:coors count:2];
12.    [mapView addOverlay:polyline];
```

```

13. }
14. // Override
15. - (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id
    <BMKOverlay>)overlay{
16.     if ([overlay isKindOfClass:[BMKPolyline class]]){
17.         BMKPolylineView* polylineView = [[[BMKPolylineView a
            lloc] initWithOverlay:overlay] autorelease];
18.         polylineView.strokeColor = [[UIColor purpleColo
            r] colorWithAlphaComponent:1];
19.         polylineView.lineWidth = 5.0;
20.         return polylineView;
21.     }
22.     return nil;
23. }

```

运行后，效果如图：



5.4 添加多边形

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

修改您的 ViewController.m 文件，实现 BMKMapViewDelegate 的 mapView:viewForOverlay:函数，并在 viewDidLoad 添加多边形数据对象：

```

1. - (void)viewDidLoad {
2.     [super viewDidLoad];
3.     // 设置 delegate
4.     mapView.delegate = self;
5.     // 添加多边形覆盖物
6.     CLLocationCoordinate2D coords[3] = {0};

```

```

7. coords[0].latitude = 39;
8. coords[0].longitude = 116;
9. coords[1].latitude = 38;
10.     coords[1].longitude = 115;
11.     coords[2].latitude = 38;
12.     coords[2].longitude = 117;
13.     BMKPolygon* polygon = [BMKPolygon polygonWithCoordinates:coords
    count:3];
14.     [mapView addOverlay:polygon];
15. }
16. // Override
17. - (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:
    (id <BMKOverlay>)overlay{
18.     if ([overlay isKindOfClass:[BMKPolygon class]]){
19.         BMKPolygonView* polygonView = [[[BMKPolygonView alloc] initWithOverlay:overlay] autorelease];
20.         polygonView.strokeColor = [[UIColor purpleColor] colorWithAlphaComponent:1];
21.         polygonView.fillColor = [[UIColor cyanColor] colorWithAlphaComponent:0.2];
22.         polygonView.lineWidth = 5.0;
23.         return polygonView;
24.     }
25.     return nil;
26. }

```

运行后，效果如图：



5.5 添加圆

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

修改您的 `ViewController.m` 文件，实现 `BMKMapViewDelegate` 的 `mapView:viewForOverlay:` 函数，并在 `viewDidLoad` 添加园数据对象：

```

1. - (void)viewDidLoad {

```

```

2. [super viewDidLoad]
3. // 设置 delegate
4. mapView.delegate = self;
5. // 添加圆形覆盖物
6. CLLocationCoordinate2D coor;
7. coor.latitude = 39.915;
8. coor.longitude = 116.404;
9. BMKCircle* circle = [BMKCircle circleWithCenterCoordinate:coor radius:5000];
10. [mapView addOverlay:circle];
11.}
12.// Override
13.- (BMKOverlayView *)mapView:(BMKMapView *)mapView viewForOverlay:(id <BMKOverlay>)overlay{
14.    if ([overlay isKindOfClass:[BMKCircle class]]){
15.        BMKCircleView* circleView = [[BMKCircleView alloc] initWithOverlay:overlay autorelease];
16.        circleView.fillColor = [[UIColor cyanColor] colorWithAlphaComponent:0.5];
17.        circleView.strokeColor = [[UIColor blueColor] colorWithAlphaComponent:0.5];
18.        circleView.lineWidth = 10.0;
19.        return circleView;
20.    }
21.    return nil;
22.}

```

运行后，效果如图：



5.6 删除 Overlay

通过 `removeOverlay` 函数实现对已添加标注的删除功能，示例如下：

```
1.         if (overlay != nil) {
2.             [mapView removeOverlay:overlay];
3.         }
```

7 服务类

百度地图 API 提供的搜索服务包括：POI 检索，多关键字检索，公交方案检索，驾车路线检索，步行路线检索，地理编码，反地理编码。

所有检索请求接口均为异步接口，您必须实现 `BMKSearchDelegate` 协议，在检索到结果后，API 会回调 `BMKSearchDelegate` 对应的接口，通知调用者检索结果数据。

`BMKSearchDelegate` 对应的接口如下：

```
1. /**
2.  * 返回 POI 搜索结果
3.  * @param poiResultList 搜索结果列表，成员类型为 BMKPoiResult
4.  * @param type 返回结果类型：
5.  *             BMKTypePoiList, BMKTypeAreaPoiList, BMKAreaMultiPoiList
6.  * @param error 错误号，@see BMKErrorCode
7.  */
8. - (void) onGetPoiResult: (NSArray*) poiResultList searchType: (int) type
9.   errorCode: (int) error {
10. }
11. /**
12.  * 返回公交搜索结果
13.  * @param result 搜索结果
14.  * @param error 错误号，@see BMKErrorCode
15.  */
16. - (void) onGetTransitRouteResult: (BMKPlanResult*) result
17.   errorCode: (int) error {
18. }
19. /**
20.  * 返回驾乘搜索结果
21.  * @param result 搜索结果
22.  * @param error 错误号，@see BMKErrorCode
23.  */
24. - (void) onGetDrivingRouteResult: (BMKPlanResult*) result
25.   errorCode: (int) error {
26. }
```

```

24. /**
25.  *返回步行搜索结果
26.  *@param result 搜索结果
27.  *@param error 错误号, @see BMKErrorCode
28.  */
29. - (void) onGetWalkingRouteResult: (BMKPlanResult*) result
    errorCode: (int) error {
30. }
31. /**
32.  *返回地址信息搜索结果
33.  *@param result 搜索结果
34.  *@param error 错误号, @see BMKErrorCode
35.  */
36. - (void) onGetAddrResult: (BMKAddrInfo*) result errorCode: (int) error {
37. }

```

6.1 POI 检索

百度地图 API 提供以下几类 POI 检索类型：城市内检索，周边检索，范围检索，多关键字检索。

此处以城市内检索为例说明：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```

1. @interface PoiSearchDemoViewController : UIViewController<BMKMapView
    Delegate, BMKSearchDelegate> {
2.     IBOutlet BMKMapView* _mapView;
3.     BMKSearch* _search;
4. }

```

在 ViewController.m 的 viewDidLoad 中创建 BMKSearch 对象，设置对应的 delegate，并实现 BMKSearchDelegate 协议中获取 POI 结果的方法，代码如下：

```

1. - (void) viewDidLoad {
2.     [super viewDidLoad];
3.     _mapView.delegate = self;
4.     _search = [[BMKSearch alloc] init];
5.     _search.delegate = self;

```



```
6. //发起 POI 检索
7. [_search poiSearchInCity:@"北京" withKey:@"西单" pageIndex:0];
8. }
9. - (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
   e errorCode:(int)error
10. {
11.     if (error == BMKErrorOk) {
12.         BMKPoiResult* result = [poiResultList objectAtIndex:0];
13.         for (int i = 0; i < result.poiInfoList.count; i++)
14.         {
15.             BMKPoiInfo* poi = [result.poiInfoList objectAtIndex:i];
16.             BMKPointAnnotation* item = [[BMKPointAnnotation alloc] init];
17.             item.coordinate = poi.pt;
18.             item.title = poi.name;
19.             [_mapView addAnnotation:item];
20.             [item release];
21.         }
22. }
```

运行效果如图:


```

11.         [_search transitSearch:@"北京" startNode:start endNode:end];
12.         [start release];
13.         [end release];
14.     }
15. - (void)onGetTransitRouteResult:(BMKPlanResult*)result errorCode:(int)error
16. {
17.     // 在此处添加您对公交方案结果的处理
18. }

```

将公交方案对应的路线和关键点绘制在地图上，效果如下图：



示例代码请参考[相关下载](#) demo 工程中的 RouteSearchDemoViewController.mm 文件

6.3 驾车路线检索

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码参考 POI 检索中的示例代码。

在 `ViewController.m` 中创建 `BMKSearch` 对象，设置对应的 `delegate`，并实现 `BMKSearchDelegate` 协议中获取驾车路线结果的方法，代码如下：

```
1. - (void)viewDidLoad {
2.     [super viewDidLoad];
3.     _mapView.delegate = self;
4.     _search = [[BMKSearch alloc] init];
5.     _search.delegate = self;
6.     //发起公交检索
7.     BMKPlanNode* start = [[BMKPlanNode alloc] init];
8.     start.name = @"天安门";
9.     BMKPlanNode* end = [[BMKPlanNode alloc] init];
10.    end.name = @"百度大厦";
11.    [_search drivingSearch:@"北京" startNode:start endCity:@"北京" endNode:end];
12.    [start release];
13.    [end release];
14. }
15. - (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:
    (int)error
16. {
17.     // 在此处添加您对驾车方案结果的处理
18. }
```

将驾车方案对应的路线和关键点绘制在地图上，效果如下图：



示例代码请参考[相关下载](#) demo 工程中的 RouteSearchDemoViewController.mm 文件

6.4 步行路线检索

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码参考 POI 检索中的示例代码。

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，并实现 BMKSearchDelegate 协议中获取步行路线结果的方法，代码如下：

```
1. - (void)viewDidLoad {
2. [super viewDidLoad];
3. _mapView.delegate = self;
4. _search = [[BMKSearch alloc] init];
5. _search.delegate = self;
6. //发起步行检索
7. BMKPlanNode* start = [[BMKPlanNode alloc] init];
8. start.name = @"天安门";
9. BMKPlanNode* end = [[BMKPlanNode alloc] init];
```

```

10.         end.name = @"百度大厦";
11.         [_search walkingSearch:@"北京" startNode:start endCity:@"北京" endNode:end];
12.         [start release];
13.         [end release];
14.     }
15. - (void)onGetWalkingRouteResult:(BMKPlanResult*)result errorCode:(int)error
16. {
17.     // 在此处添加您对步行方案结果的处理
18. }

```

将步行方案对应的路线和关键点绘制在地图上，效果如下图：



示例代码请参考[相关下载](#) demo 工程中的 RouteSearchDemoViewController.mm 文件

6.5 地理编码

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码参考 [POI 检索](#) 中的示例代码。

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，并实现 BMKSearchDelegate 协议中获取地理编码结果的方法，代码如下：

```
1. - (void) viewDidLoad {
2.     [super viewDidLoad];
3.     _mapView.delegate = self;
4.     _search = [[BMKSearch alloc] init];
5.     _search.delegate = self;
6.     //发起地理编码
7.     [_search geocode:@"东长安街 33 号" withCity:@"北京"];
8. }
9. - (void) onGetAddrResult:(BMKAddrInfo*) result errorCode:(int) error
10. {
11.     // 在此处添加您对地理编码结果的处理
12. }
```

完整的示例代码请参考[相关下载](#) demo 工程中的 GeocodeDemoViewController.mm 文件

6.6 反地理编码

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码参考 [POI 检索](#) 中的示例代码。

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，并实现 BMKSearchDelegate 协议中获取反地理编码结果的方法，代码如下：

```
1. - (void) viewDidLoad {
2.     [super viewDidLoad];
3.     _mapView.delegate = self;
4.     _search = [[BMKSearch alloc] init];
5.     _search.delegate = self;
6.     //发起反地理编码
7.     CLLocationCoordinate2D pt = (CLLocationCoordinate2D){39.915101, 116.
8.         403981};
9.     [_search reverseGeocode:pt];
10. }
11. - (void) onGetAddrResult:(BMKAddrInfo*) result errorCode:(int) error
12. {
13.     // 在此处添加您对反地理编码结果的处理
14. }
```

```
13. }
```

完整的示例代码请参考[相关下载](#) demo 工程中的 GeocodeDemoViewController.mm 文件

8 定位

您可以通过以下代码来开启定位功能：

```
1. [mapView setShowUserLocation:YES];
```

定位成功后，可以通过 `mapView.userLocation` 来获取位置数据。

完整的示例代码请参考[相关下载](#) demo 工程中的 LocationDemoViewController.mm 文件

9 离线地图

SDK v1.1 以后支持离线地图导入，从官网下载对应的离线包，通过 `itunes` 导入对应程序的共享目录，对于越狱的手机可以通过 `91 助手` 拷到对应程序目录下的 `document` 目录：

```
1. _offlineMap = [[BMKOfflineMap alloc] init];  
2. _offlineMap.delegate = self;
```

完整的示例代码请参考[相关下载](#) demo 工程中的 OfflineDemoViewController.m 文件