腾讯iOS 导航SDK接入文档

导航版本号：5.2.1 支持iOS 9.0及以上版本

[腾讯iOS 导航SDK接入文档 0](#_Toc40433896)

[1.概述 1](#_Toc40433897)

[1.1路径规划 2](#_Toc40433898)

[1.2实时导航 3](#_Toc40433899)

[1.3模拟导航 5](#_Toc40433900)

[1.4 导航模式 5](#_Toc40433901)

[1.5自定义导航面板 8](#_Toc40433902)

[1.6添加控件 9](#_Toc40433903)

[2.工程配置 10](#_Toc40433904)

[2.1申请开发密钥 10](#_Toc40433905)

[2.2新建工程（已有可跳过） 12](#_Toc40433906)

[2.3设置腾讯地图、导航Key 13](#_Toc40433907)

[2.4 引入导航SDK、地图SDK 14](#_Toc40433908)

[2.5 权限设置 15](#_Toc40433909)

[3. 快速接入 16](#_Toc40433910)

[3.1导航初始化 16](#_Toc40433911)

[3.2 驾车路径规划 17](#_Toc40433912)

[3.3 开启、结束导航 18](#_Toc40433913)

[4. 功能概述 18](#_Toc40433914)

[4.1 设置设备号 18](#_Toc40433915)

[4.2 自定义UI 18](#_Toc40433916)

[4.2.1 路口放大图 18](#_Toc40433917)

[4.2.2 车道线 19](#_Toc40433918)

[4.2.3 顶部导航面板 19](#_Toc40433919)

[4.2.4 更换起点、终点、途径点的图标 19](#_Toc40433920)

[4.2.5 设置是否使用默认资源 19](#_Toc40433921)

[4.2.6 设置自车点位置 20](#_Toc40433922)

[4.2.7 设置导航日夜模式 20](#_Toc40433923)

[4.2.8 设置导航模式 20](#_Toc40433924)

[4.2.9 其余导航控件 21](#_Toc40433925)

[4.3 导航回调事件 23](#_Toc40433926)

[4.4 一键上报 29](#_Toc40433927)

[4.5 伴随路线 30](#_Toc40433928)

[5. 网约车场景 32](#_Toc40433929)

[5.1 快车业务 32](#_Toc40433930)

# 1.概述

导航SDK是一款针对在线导航的产品。该产品能够结合实时交通信息计算路径，并且提规划、实时导航、模拟导航、导航设置、自定义导航界面等功能，力求为供路径用户提供更加人性化的导航服务。

## 1.1路径规划

路径规划可根据起点、终点、途径点的经纬度以及驾车路线的参数配置，给用户提供出行路线策略。路径规划成功后得到一条或多条路线信息，根据路线信息可以在地图上进行位置标注、绘制路线等操作。

途经点最多可设置十个，驾车路线的参数配置包括是否走高速、是否躲避拥堵、是否避开收费道路、设置起点道路类型等。

起点道路类型包括在主路、在辅路、在桥上等，设置该字段可以提升路线规划起点位置的准确程度，默认为无详细起点路段类型。

路线信息中包含路线id、总距离、总预计时间等，详情可参照SDK中的“TNKCarRouteSearchTypes” 文件。

位置标注：可以在地图上用默认的大头针图标标注位置信息，也可以自定义图标。如Demo中的“起”、“终”、“经”图标，分别标注了路线中的起点、终点、途经点。

（地图上添加大头针和路线的详细方式可在地图SDK中查看。地图SDK链接<https://lbs.qq.com/ios_v1/index.html>）

例：设置起点为北京动物园，途经点为北京儿童医院，终点为北京西站，躲避拥堵和高速，起始点在主路时，路线规划效果如下图：（图中大头针颜色、路线颜色和宽度等可根据地图SDK的介绍进行更改）



## 1.2实时导航

实时导航是基于用户真实的定位信息来驱动的导航过程。路线规划完成后，就可以开始实时导航。导航状态下，页面展示导航面板、车道线、路口放大图、自车点罗盘等，开发者可设置其隐藏/显示。

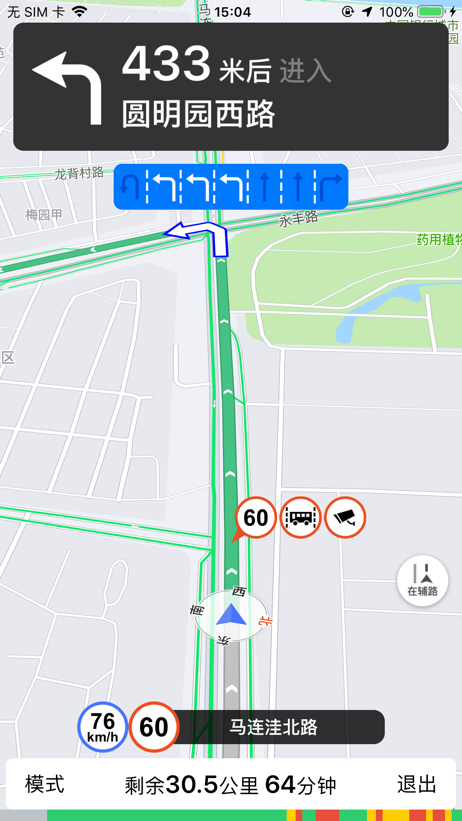
导航面板包含转弯图示、下一条道路名称和剩余距离等信息。

如下图所示，车道线通过白色高亮箭头指示用户可选择的车道类型，路口放大图能够显示道路走向、多岔路、道路具体形状、车道数量等，通过黄色高亮线段更清晰的标明路线，防止用户混淆。

自车点罗盘的自定义可参见本文档“快速入门指南”中的“自定义导航界面”。

导航时如果有可以切换的其他道路类型，可提示用户是否切换，道路类型有：在主路、在辅路、在对面等，可参照SDK中的TNKRoadType类。



## 1.3模拟导航

模拟导航中设置起点、终点和途经点的经纬度进行路径规划，路线规划成功后，根据得到的路线信息，模拟用户以76km/h的驾车速度进行导航。模拟导航时不获取用户的实时位置，开发者可以用于模拟导航测试。

## 1.4 导航模式

用户可以根据需求设置以下导航模式：

回弹模式：不需设置，导航状态中，用户使用手势操作地图后进入该模式，默认手势结束5秒后切换回之前的导航模式，直接设置为该模式不会被响应

2D模式：导航时地图始终朝北，车头方向按照实时传入的GPS点方向实时改变



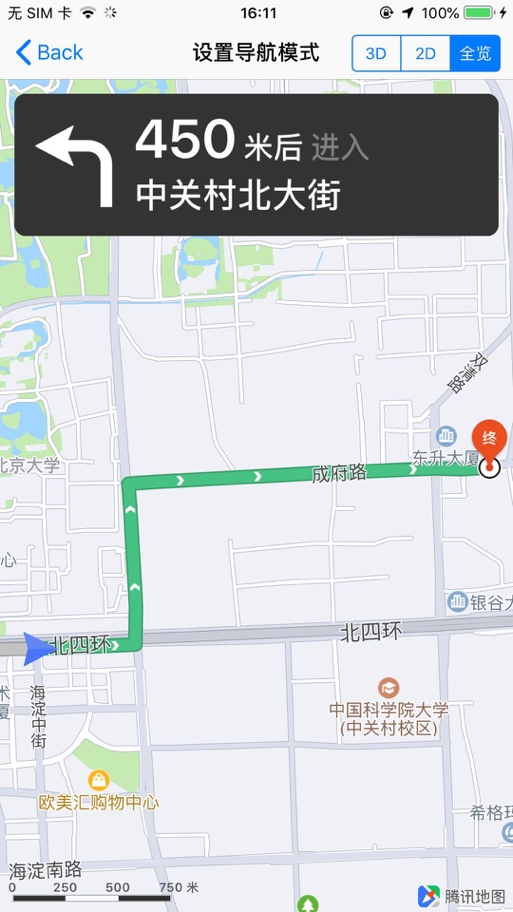
3D模式：导航时车头始终朝上，地图方向按照实时传入的GPS点旋转



全览模式：导航时地图显示导航的起点到终点的完整路线



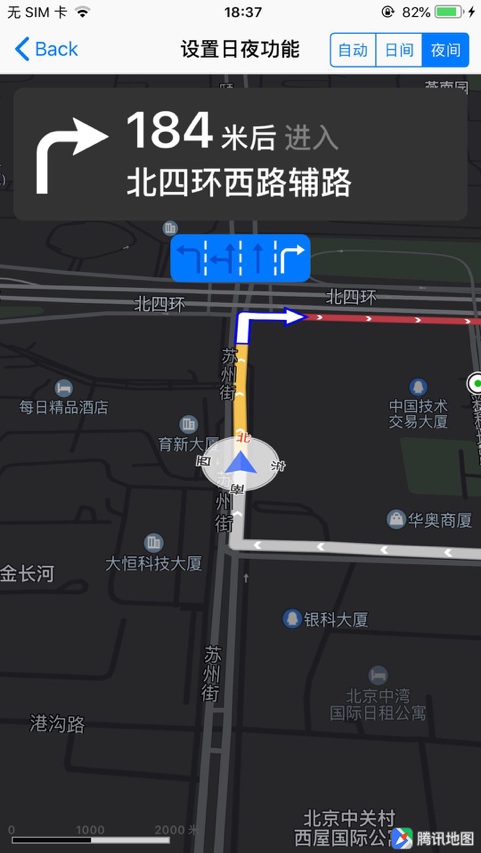
剩余全览模式：导航时地图只显示车辆的当前位置到终点的路线



日间模式：浅色系底图，设置该模式，则导航过程中始终为日间状态



夜间模式：深色系底图，设置该模式，则导航过程中始终为夜间状态



自动模式：设置该模式，导航SDK通过对当前系统时间和经纬度的判断自动切换到日间或夜间模式

例：2019年5月9日北京日夜间模式切换的时间为：

04：57切换为日间模式

19：25切换为夜间模式

## 1.5自定义导航面板

开发者可以根据需要自定义导航面板的位置、大小、内容等信息，也可设置其隐藏/显示。



## 1.6添加控件

开发者可对导航页面添加自己的控件，详见“快速入门指南”中的“添加控件”。

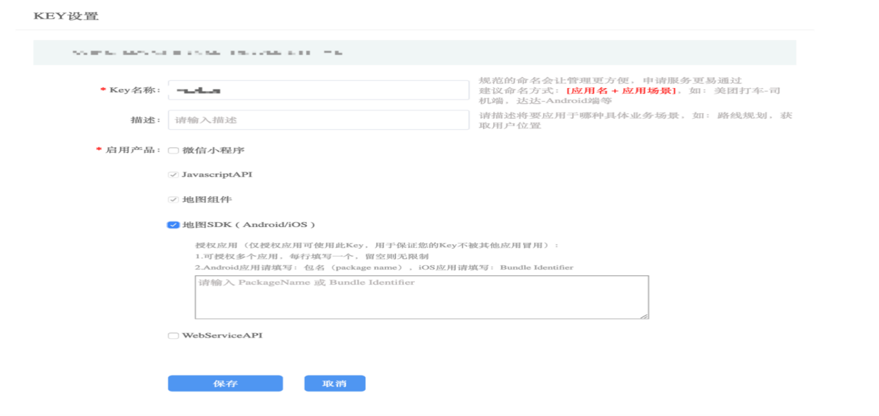


# 2.工程配置

## 2.1申请开发密钥

1) 创建开发者密钥：进入网站<https://lbs.qq.com/console/key.html>，创建开发者账号。如果之前已经创建过，可以直接登录。

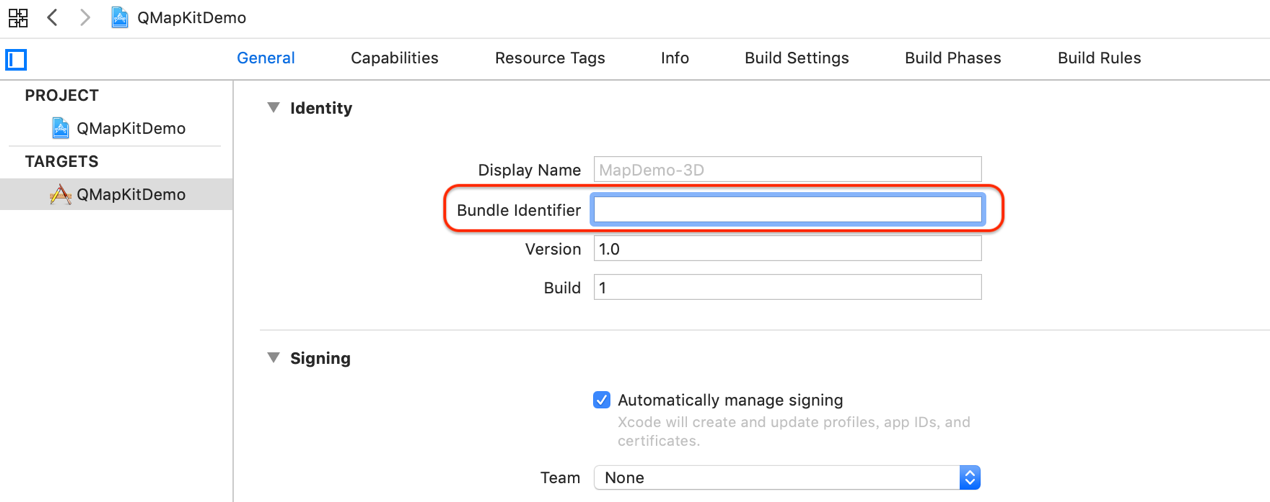


2) 进入控制台，在key管理中创建新密钥，输入新key名称和验证码，勾选“已阅读并同意以上条款”，会获得对应的key。

A screenshot of a social media post

Description automatically generated3) 进入key设置，启用产品中勾选“地图SDK”选项，在输入框中填写对应的.iOS应用请填写：Bundle Identifier点击保存，完成key的申请。此时用户即可用刚创建的key了。

获取Bundle Identifier 方法：打开Xcode，点击工程，如图



4）导航SDK的key设置完成后，需要联系您的导航SDK对接人员开通key的导航权限

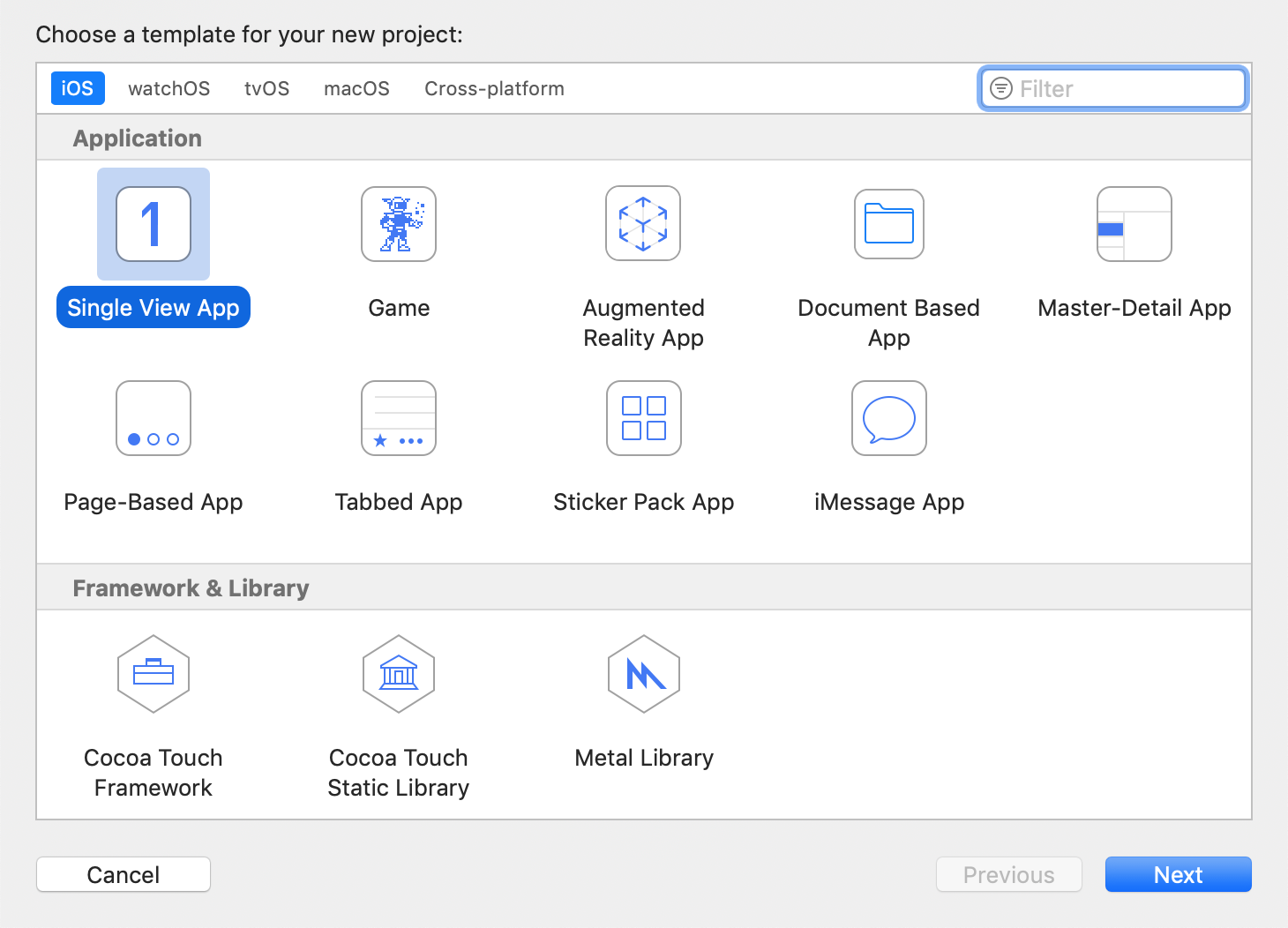
## 2.2新建工程（已有可跳过）

1. 如果您未下载Xcode开发工具，可以到App Store下载最新版本

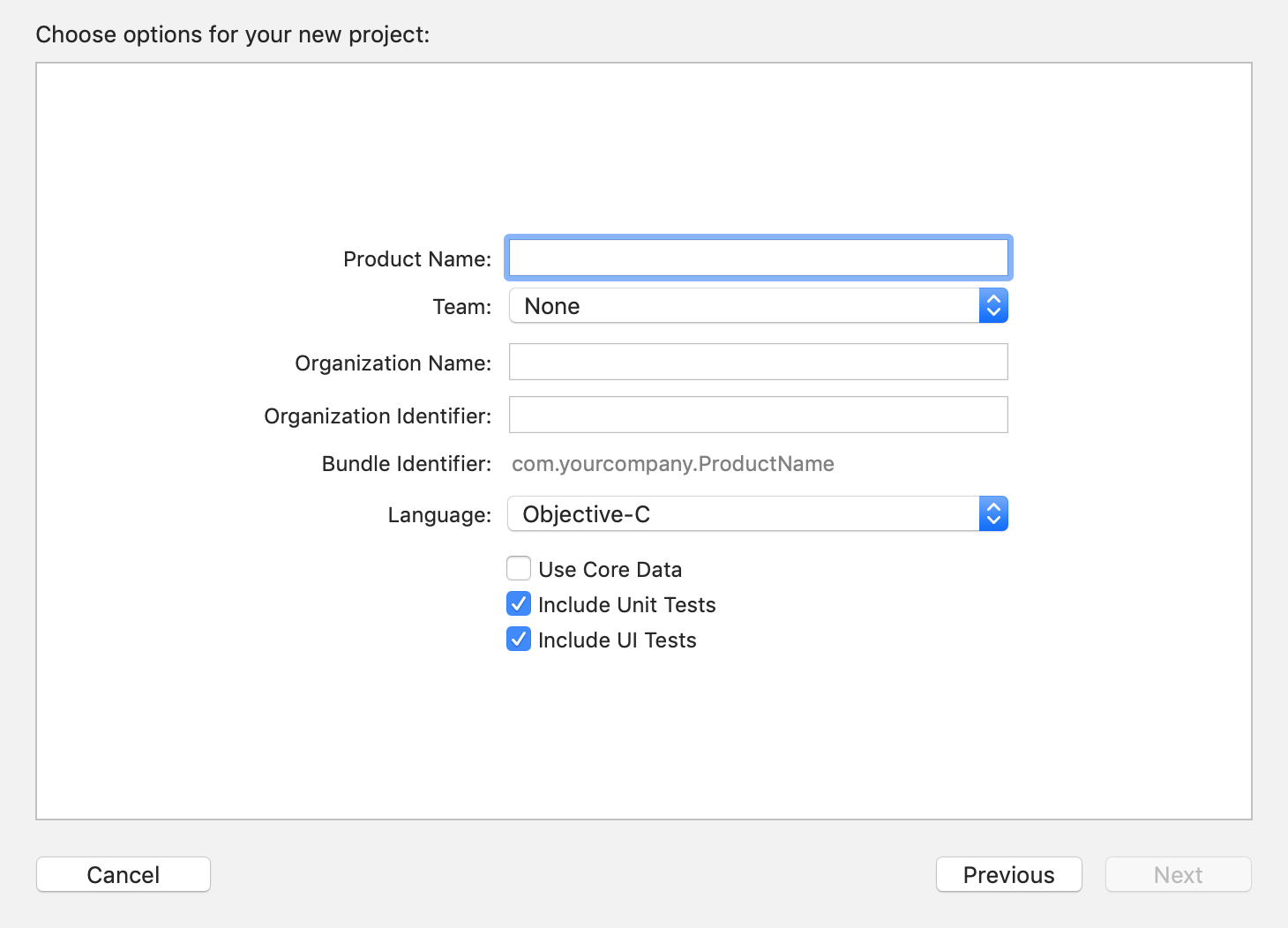


2、准备需要集成导航SDK的Xcode项目，已有可跳过

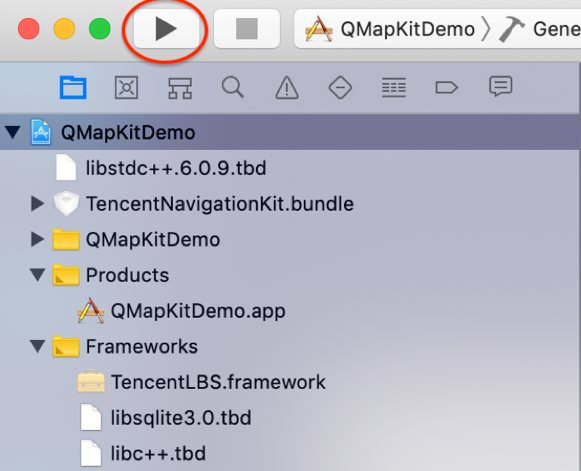
（1）打开Xcode，选择Single View App



填入自己工程的名称等信息



3、集成导航SDK前，可以点击Xcode左上角的运行按钮，提示“Success”则表示运行成功，方便集成导航SDK后的检测。



## 2.3设置腾讯地图、导航Key

在工程的“AppDelegate.m”中引入“#import<QMapKit/QMapKit.h>”和“#import <QMapKit/QMapKit.h>”

#import <QMapKit/QMapKit.h>”

#import<TNKNavigationKit/TNKNavigationKit.h>  
//配置地图和导航的key

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

{

//地图SDK的key配置

  [QMapServices sharedServices].APIKey = @"您的key";

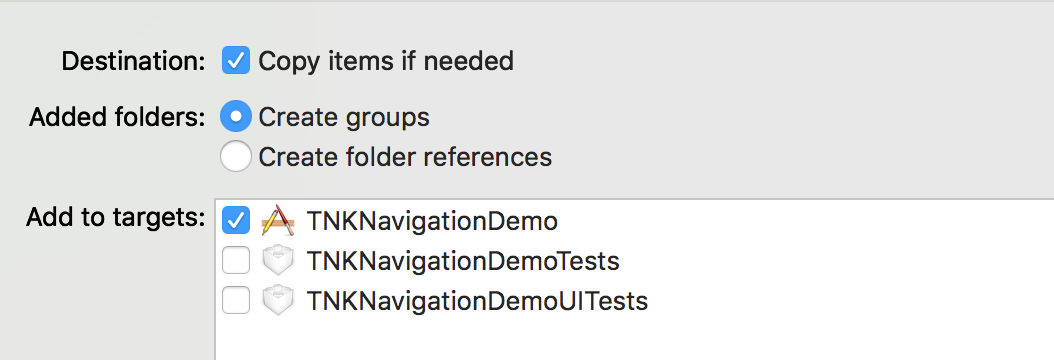
//导航SDK的key配置

[TNKNaviServices sharedServices].APIKey = @"您的key"

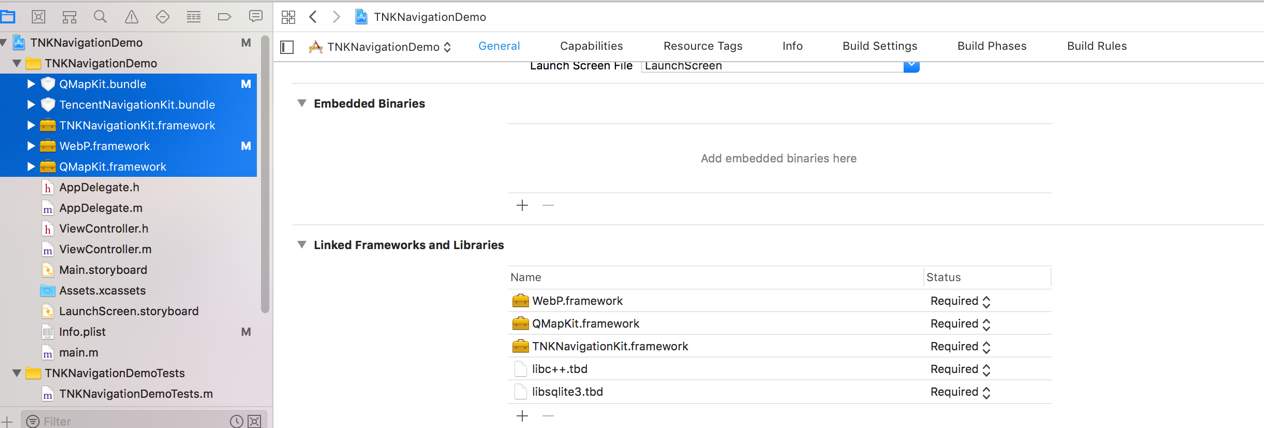
}

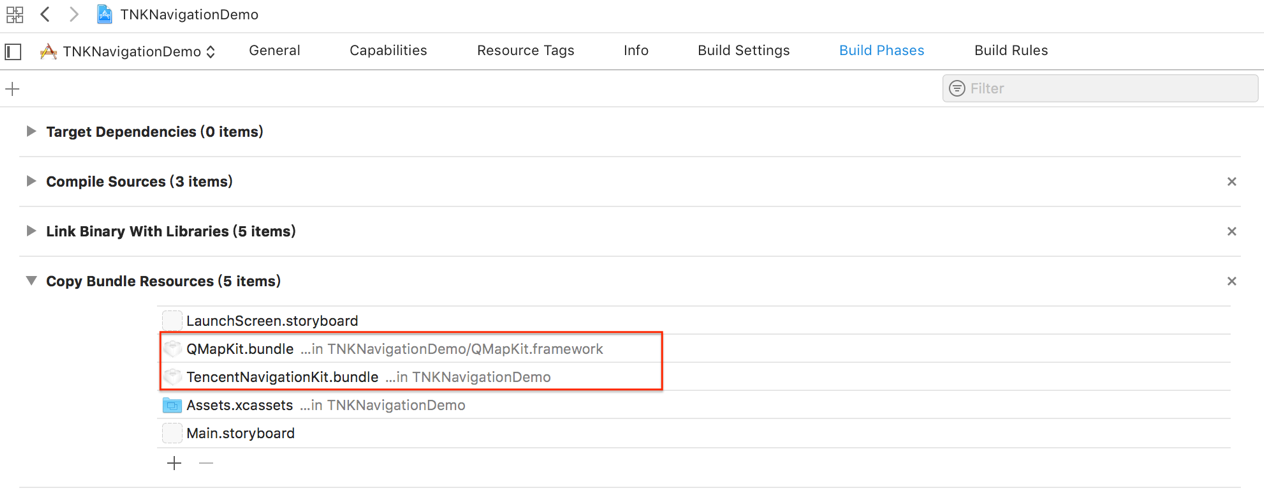
## 2.4 引入导航SDK、地图SDK

在Xcode工程中右键菜单中选择Add files to"..."，将WebP.framework、TNKNavigationKit.framework、QMapKit.framework以及TNKNavigationKit.framework中的TencentNavigationKit.bundle、QMapKit.framework中的QMapKit.bundle依次添加到工程中，注意勾选"Copy items if needed"



添加完成后检查项目中文件是否存在

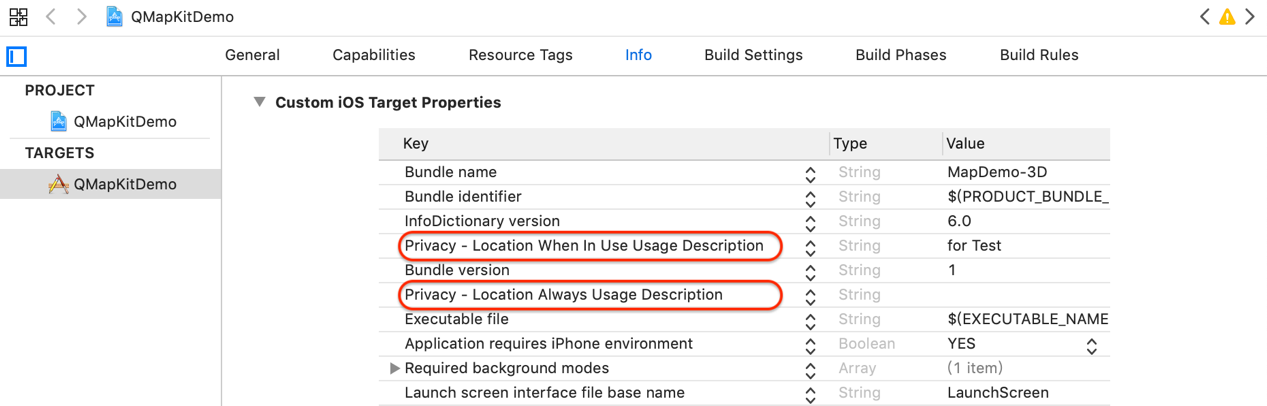


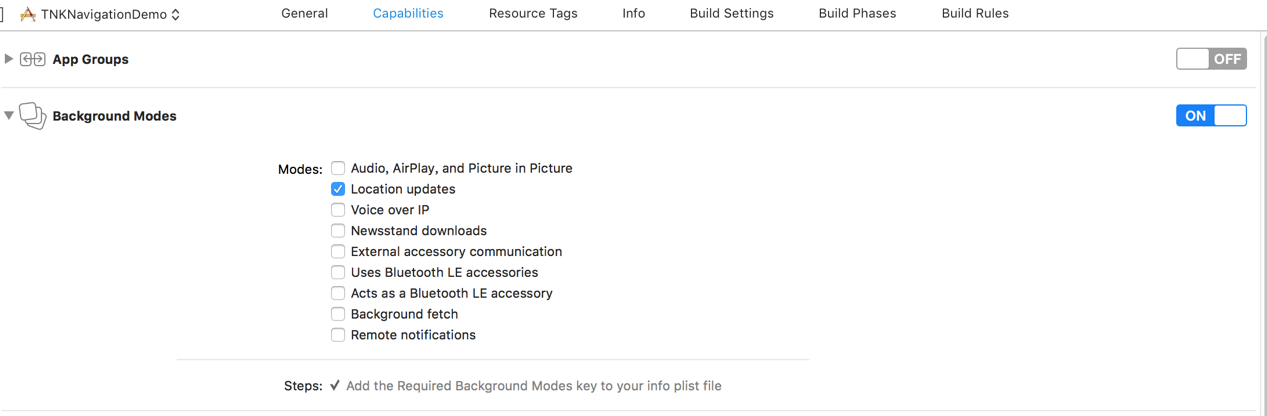


此时配置导航SDK部分已经完成，点击Xcode左上角的运行按钮，可看到“Success”的提示

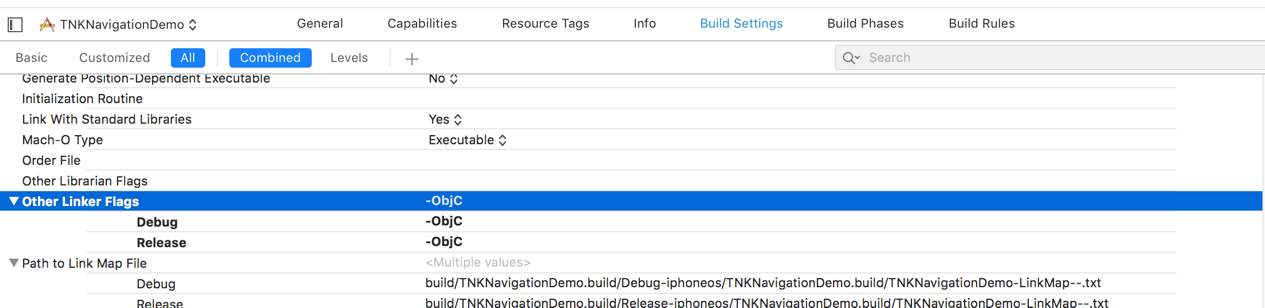
## 2.5 权限设置

(1) 若使用实时导航，需要在TARGETS-> Info中添加NSLocationWhenInUseUsageDescription和NSLocationAlwaysUsageDescription

 (2) 如果需要后台定位，需在IOS 9.0及以后设allowBackgroundLocationUpdates为YES，此时必须保证TARGETS->Capabilities->Background Modes->Location updates处于选中状态



(3)TARGETS->Building Settings -> Other Linker Flags中添加-ObjC



(4)如果需要一键上报功能的语音反馈功能，请查看[一键上报功能](#一键上报)

# 3. 快速接入

## 3.1导航初始化

需要初始化导航控制类和导航UI控制类

//初始化导航控制类

self.carManager = [[TNKCarNaviManager alloc] init];

// 注册代理，用以接收manager的回调

[self.carManager registerNaviDelegate:self];

//导航UI类初始化，用于显示默认导航UI,包括导航面板,路线,车标,起终点等

self.carNaviView = [[TNKCarNaviView alloc] initWithFrame:self.view.bounds];

// 展示导航默认UI元素

self.carNaviView.showUIElements = YES;

// TNKCarNaviView的回调,用于接收导航模式、日夜状态等变化

self.carNaviView.delegate = self;

// 调用该方法注册对象，用于接收导航UI数据更新

[self.carNaviManager registerUIDelegate:self.carNaviView];

## 3.2 驾车路径规划

发起路径规划代码如下

//发起驾车导航路线规划

[self.carManager searchNavigationRoutesWithRequest:request completion:^(TNKCarRouteSearchResult \*result, NSError \*error) {

// 可在该block中判断是否算路成功。若算路成功，可获取结果进行展示等操作

}];

其中参数说明如下表格所示，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 |  | 类型 | 必填 | 说明 |
| startPoint |  | TNKSearchNaviPoi | 是 | 起点，一般使用当前定位点赋值 |
|  | coordinate | CLLocationCoordinate2D | 是 | 地点的坐标 |
|  | uid | NSString | 否 | 该地点的唯一标识，即poi id |
| destinationPoint |  | TNKSearchNaviPoi | 是 | 目的地。 |
| wayPoints |  | NSArray<TNKSearchNaviPoi\*> \* | 否 | 途经点数组，最多设置10个。路线规划按照数组的顺序进行，不会按路径最短原则进行调整。 |
| searchOptions |  | TNKCarRouteSearchOption | 是 | 路线规划策略 |
|  | avoidToll | Bool | 否 | 是否避开收费站 |
|  | avoidHighway | Bool | 否 | 是否避开高速公路 |
|  | avoidTrafficJam | Bool | 否 | 是否避开拥堵 |
|  | navScene | int | 是 | 1:接驾；2:送驾 |
|  | preLocations | NSArray<CLLocation \*> \* | 否 | 前序点。最好传入路线规划之前的50个定位点，以提高起点的准确行。 |
|  | truckSearchOption | TNKTruckSearchOption | 否 | 货车算路属性。当有货车算路需求时使用，具体参考导航SDK接口文档 |

## 3.3 开启、结束导航

//选择index为0的路线开始导航

[self.carNaviManager startWithIndex:0];

// 结束导航

[self.carNaviManager stop];

模拟导航方法：

// 开启模拟导航，使用第1条路线进行导航

[self.carNaviManager startSimulateWithIndex:0 locationEntry:nil];

// 结束模拟导航

[self.carNaviManager stop];

# 4. 功能概述

## 4.1 设置设备号

用户可在APP启动时为导航设置设备唯一标识。出现问题时可向我们提供该设备唯一标识和发生问题的时间段，以便我们帮助您排查问题。设置代码如下：

[TNKNaviServices sharedServices].identifier = @"XXXXXX-XXXX-XXXX-XXXX-XXXXXX";

## 4.2 自定义UI

导航SDK提供了一些默认UI组件，开发者可以根据需要控制这些元素的现实和隐藏。

### 4.2.1 路口放大图

路口放大图默认开启，可使用如下方法控制：

// 设置不开启路口放大图功能

self.carManager.enableEnlargedIntersection = NO;

### 4.2.2 车道线

车道线默认开启，可使用如下方法控制：

// 设置不开启车道线功能

self.carManager.enableGuidedLane = NO;

### 4.2.3 顶部导航面板

顶部导航面板默认开启，可使用如下方法控制：

// 设置不隐藏默认导航面板

self.carNaviView.hideNavigationPanel = NO;

### 4.2.4 更换起点、终点、途径点的图标

// 更换起点图标

[self.carNaviView configStartPointMarkerPresentation:startPointMarker];

// 更换终点图标

[self.carNaviView configEndPointMarkerPresentation:endPointMarker];

// 更换途径点图标

[self.carNaviView configWayPointMarkerPresentation:wayPointMarkers];

### 4.2.5 设置是否使用默认资源

SDK中assets下的图片都可以自定义，如小车marker电子眼marker、车道线箭头图片等。使用时需要将同名文件放到assets目录下。

### 4.2.6 设置自车点位置

自车点即导航时车标在地图中的位置点，导航模式下车标位于地图宽高的比例，默认x为0.5，y为0.75，取值范围为[0.25, 0.75]，超过则取默认值

//3D导航时设置自车点位置

self.carNaviView.TNKCarNaviFixCarProportion2D = CGPointMake(0.5, 0.75);

//2D导航时设置自车点位置

self.carNaviView.TNKCarNaviFixCarProportion3D = CGPointMake(0.5, 0.75);

### 4.2.7 设置导航日夜模式

默认是根据太阳升起落下时间自动切换模式，也可以设置一直使用日间模式:

self.carNaviView.dayNightMode = TNKCarNaviDayNightModeAlwaysDay;

### 4.2.8 设置导航模式

默认为3D车头向上模式,导航模式包括:

* 3D最佳视野：TNKCarNaviUIMode3DCarTowardsUp，3D车头朝上模式.该模式下，车头始终保持指向屏幕上方，地图进行旋转并动态调整缩放级别.
* 2D最佳视野：TNKCarNaviUIMode2DMapTowardsNorth， 2D地图朝北模式.该模式下，车头进行旋转，地图保持上北下南并动态调整缩放级别.
* 2D全览模式：TNKCarNaviUIModeOverview，2D路线全览模式.该模式下,车头进行旋转，地图保持上北下南，同时会始终展示整条导航路线.
* 剩余全览模式：TNKCarNaviUIModeRemainingOverview,剩余路线全览模式。该模式下,车头进行旋转，地图保持上北下南，同时会始终展示整条导航路线的剩余部分。

设置方法如下，

//例：设置导航模式为3D车头朝上

self.carNaviView.mode = TNKCarNaviUIMode3DCarTowardsUp;

回弹模式：导航态中，用户使用手势操作地图后进入回弹模式，手势结束默认5秒后切换回之前的导航模式，可通过如下代码设置回弹时间:

//设置回弹时间为10秒

self.carNaviView.bounceTime = 10

### 4.2.9 其余导航控件

A screenshot of a cell phone screen with text

Description automatically generatedA close up of a map

Description automatically generated上图中红框部分是默认的UI组件，包括光柱图，地图缩放控件，当前车速，限速信息等，分别控制显隐，代码如下：

// 展示所有导航UI控件

self.carNaviView.showUIElements = YES;

// 隐藏底部工具栏

self.carNaviView.showBottomToolbar = NO;

// 隐藏主辅路切换按钮

self.carNaviView.showRoadTypeButton = NO;

// 隐藏当前车速UI

self.carNaviView.showCurrentSpeedView = NO;

// 隐藏限速、当前路名UI

self.carNaviView.showLimitSpeedAndCurrentRoadNameView = NO;

// 隐藏全览/非全览切换按钮

self.carNaviView.showOverviewButton = NO;

// 隐藏路况显隐按钮

self.carNaviView.showMapTrafficButton = NO;

// 隐藏路况光柱图按钮

self.carNaviView.showTrafficBar = NO;

接受导航退出按钮的点击事件：

- (void)carNaviViewCloseButtonClicked:(TNKCarNaviView \*)carNaviView;

## 4.3 导航回调事件

1）可实现TNKCarNaviUIDelegate协议从而获取导航展示信息的回调事件：

/\*\*

\* @brief 导航数据回调.

\* @param manager manager

\* @param data 导航数据

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager updateNavigationData:(TNKCarNavigationData \*)data;

/\*\*

\* @brief 显示路口放大图回调.

\* @param manager manager

\* @param image 路口放大图

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager showEnlargedIntersectionImage:(UIImage \*)image;

/\*\*

\* @brief 隐藏路口放大图回调.

\* @param manager manager

\*/

- (void)carNavigationManagerHideEnlargedIntersectionImage:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 显示车道线回调.

\* @param manager manager

\* @param image 车道线

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager showGuidedLaneImage:(UIImage \*)image;

/\*\*

\* @brief 隐藏车道线回调.

\* @param manager manager

\*/

- (void)carNavigationManagerHideGuidedLaneImage:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 更新路线路况数据回调. 其中包括路线总距离、路线剩余距离及路况情况.

\* @param manager manager

\* @param status 路线路况数据

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager updateRouteTrafficStatus:(TNKRouteTrafficStatus \*)status;

/\*\*

\* @brief 路线切换回调

\* @param manager manager

\* @param routePlan 路线数据

\* @param trafficStatus 路况数据

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager routeDidChange:(TNKCarRouteSearchRoutePlan \*)routePlan routeTrafficStatus:(TNKRouteTrafficStatus \*)trafficStatus;

其中NavigationData包含：当前道路路名、当前速度 、当前道路剩余距离、总剩余距离、总剩余时间、当前道路限速、下一道路路名、前方转向箭头图片、途经点信息等。

2）获得TNKNaviDelegate导航状态事件回调：

/\*\*

\* @brief 导航开始.

\* @param manager manager

\*/

- (void)carNavigationManagerDidStart:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 导航结束.

\* @param manager manager

\*/

- (void)carNavigationManagerDidStop:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 获取TTS回调.

\* @param manager manager

\* @param naviTTS tts信息

\*

\* @return 是否播报该文案.0未播报,1已播报.

\*/

- (int)carNavigationManager:(TNKCarNaviManager \*)manager

broadcastTTS:(TNKNaviTTS \*)naviTTS;

/\*\*

\* @brief 获取到达终点附近的回调. 尚未结束导航时, 这个回调可能执行多次.

\* @param manager manager

\*/

- (void)carNavigationManagerDidArriveDestination:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 获取偏航回调.

\* @param manager manager

\*/

- (void)carNavigationManagerDidOffRoute:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 获取GPS信号变弱的回调.

\* @param manager manager

\*/

- (void)carNavigationManagerDidGPSSignalBecomeWeak:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 获取GPS信号变强的回调.

\* @param manager manager

\*/

- (void)carNavigationManagerDidGPSSignalBecomeStrong:(TNKCarNaviManager \*)manager;

/\*\*

\* @brief 获取经过途经点回调.

\* @param manager manager

\* @param index 途经点索引

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didArriveWaypoint:(int)index;

/\*\*

\* @brief 获取道路类型更新回调.

\* @param manager manager

\* @param type 道路类型

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didUpdateRoadType:(TNKRoadType)type;

/\*\*

\* @brief 获取导航定位更新回调.

\* @param manager manager

\* @param location 定位信息

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didUpdateLocation:(TNKLocation \*)location;

/\*\*

\* @brief 获取定位失败的回调

\* @param manager manager

\* @param error 错误信息

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

updateLocationFailedWithError:(NSError \*)error;

/\*\*

\* @brief 获取重新算路请求开始回调.

\* @param manager manager

\* @param type 重新算路类型

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didStartRecaculateRoute:(TNKCarNaviManagerRecaculateType)type;

/\*\*

\* @brief 获取重新算路请求成功回调.

\* @param manager manager

\* @param type 重新算路类型

\* @param result 重新算路成功结果

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didSuccessRecaculateRoute:(TNKCarNaviManagerRecaculateType)type

result:(TNKCarRouteSearchResult \*)result;

/\*\*

\* @brief 获取重新算路请求失败回调.

\* @param manager manager

\* @param type 重新算路类型

\* @param error 重新算路失败原因

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didFailRecaculateRoute:(TNKCarNaviManagerRecaculateType)type

error:(NSError \*)error;

/\*\*

\* @brief 获取重新算路请求取消回调. 当偏航重新算路未完成但自车位置已驶回原路线时, 重新算路请求会取消并使用原路线继续导航.

\* @param manager manager

\* @param type 重新算路类型

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didCancelRecaculateRoute:(TNKCarNaviManagerRecaculateType)type;

/\*\*

\* @brief 获取路线切换回调. 当走向备选路线或者用户主动切换到备选路线上会触发此回调

\* @param manager manager

\* @param routePlan 路线信息

\* @param trafficStatus 路况信息

\*/

- (void)carNavigationManager:(TNKCarNaviManager \*)manager

didChangeRoute:(TNKCarRouteSearchRoutePlan \*)routePlan

routeTrafficStatus:(TNKRouteTrafficStatus \*)trafficStatus;

## 4.4 一键上报

A picture containing clock

Description automatically generatedA picture containing bus

Description automatically generated导航使用者遇到导航问题时可以进行问题反馈上报,包括：拥堵、施工、封路、事故、管制、语音反馈。

A picture containing clock

Description automatically generated

为了更好的定位问题，开发者可以将自己业务里的行政区域编码、用户id、订单id赋值给路线规划参数类TNKCarRouteSearchOption的adcode、userID、orderID。

// 设置城市编码：

carRouteSearchOptioin.adcode = @"";

//设置用户id

carRouteSearchOptioin.userID = @"";

// 设置订单id

carRouteSearchOptioin.orderID = @"";

// 可能造成语音冲突，可关闭语音反馈功能

self.carNaviManager.audioFeedbackEnabled = NO;

## 4.5 伴随路线

可开启导航过程中的伴随路线

A close up of a map

Description automatically generated

// 开启伴随路线

self.carNaviManager.multiRoutesEnabled = YES;

# 5. 网约车场景

## 5.1 快车业务

快车业务最简调用导航SDK、定位SDK、地图SDK、司乘同显SDK时序图如下所示，

路线规划策略类CarRouteSearchOptions在接送驾场景建议使用不同的策略。

1. 接驾路线规划建议策略:

navScene=1&avoidToll=true&avoidHighway=false&avoidTrafficJam=true

1. 送驾路线规划建议策略:

navScene=2&avoidToll=false&avoidHighway=false&avoidTrafficJam=true

路线规划最多返回3条路线，CarRouteSearchOptions还提供方法对3条路线重新排序:

// 0：默认； 1：距离优先； 2：价格优先

carRouteSearchOption.extraRankStrategy = 2;

// 设置单价：每公里2.6元

carRouteSearchOption.pricePerKilometer = 2.6;

// 设置单价：每分钟0.6元

carRouteSearchOption.pricePerMinuter = 0.6;