腾讯Android 导航SDK接入文档

导航版本号5.2.1.1支持 Android 4.0 及以上系统

[Android 导航SDK接入文档 0](#_Toc40258436)

[1.概述 2](#_Toc40258437)

[1.1路径规划 2](#_Toc40258438)

[1.2实时导航 3](#_Toc40258439)

[1.3模拟导航 4](#_Toc40258440)

[1.4导航设置 4](#_Toc40258441)

[1.5自定义导航面板 8](#_Toc40258442)

[1.6添加控件 9](#_Toc40258443)

[2.工程配置 10](#_Toc40258444)

[2.1申请开发者秘钥 10](#_Toc40258445)

[2.2新建工程(已有可跳过) 11](#_Toc40258446)

[2.3设置腾讯地图、导航Key 11](#_Toc40258447)

[2.4 引入导航SDK、地图SDK、定位SDK 12](#_Toc40258448)

[2.5权限设置 12](#_Toc40258449)

[2.6混淆配置 13](#_Toc40258450)

[3.快速接入 14](#_Toc40258451)

[3.1 实例化导航地图 14](#_Toc40258452)

[3.2 实例化导航管理类 15](#_Toc40258453)

[3.3驾车路径规划 16](#_Toc40258454)

[3.4开启、结束导航 17](#_Toc40258455)

[4.功能概述 18](#_Toc40258456)

[4.1 设置设备号 18](#_Toc40258457)

[4.2 自定义UI 19](#_Toc40258458)

[4.2.1 路口放大图 19](#_Toc40258459)

[4.2.2 车道线 19](#_Toc40258460)

[4.2.3顶部导航面板 19](#_Toc40258461)

[4.2.4更换起点、终点、途经点的图标 19](#_Toc40258462)

[4.2.5设置是否使用默认资源 20](#_Toc40258463)

[4.2.6设置自车点位置 20](#_Toc40258464)

[4.2.7设置导航日夜模式 20](#_Toc40258465)

[4.2.8设置导航模式 21](#_Toc40258466)

[4.2.9其余导航控件 22](#_Toc40258467)

[4.3导航回调事件 23](#_Toc40258468)

[4.4.一键上报 25](#_Toc40258469)

[4.5 伴随路线 26](#_Toc40258470)

[4.6 自动灌点 27](#_Toc40258471)

[5.网约车场景 28](#_Toc40258472)

[5.1 快车业务 28](#_Toc40258473)

# 1.概述

腾讯导航 SDK 是一款针对在线导航的产品。该产品的路径计算与实时交通信息相结合，提供路径规划、实时导航、模拟导航、导航设置、自定义导航界面等功能，力求为用户提供更加人性化的导航服务。依赖库包括：腾讯地图SDK和腾讯定位SDK。

## 1.1路径规划

路径规划可根据起点、终点、途径点的经纬度和PoiId（PoiId可根据场景需求决定是否使用）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。

Marker(是用来表示一个点位置的可见元素)，可以在地图上用默认marker标注位置信息，也可以自定义图标。如Demo中的“起”、“终”、“经”分别标注了路线中的起点、终点、途经点。

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”。

## 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，进行模拟导航。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导航设置

用户可以根据需求来设置相应的导航模式、日夜间模式、主辅路切换功能，

具体效果如下：

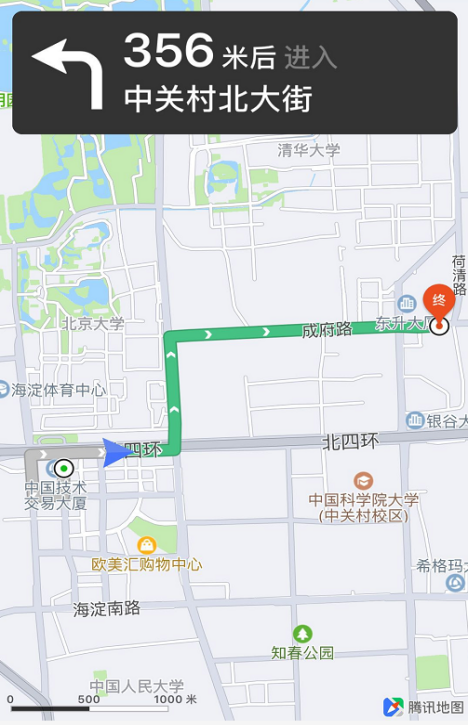
2D模式：导航时地图始终朝北，车头实时改变方向



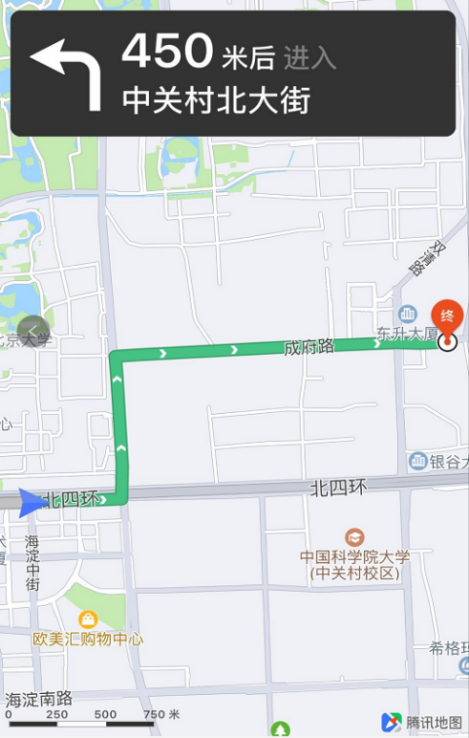
3D模式：导航时车头始终朝上，地图实时旋转



全览模式：导航时地图显示导航的起点到终点的完整路线



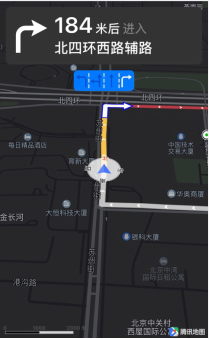
剩余全览模式：导航时地图显示车辆的当前位置点到终点的路线



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

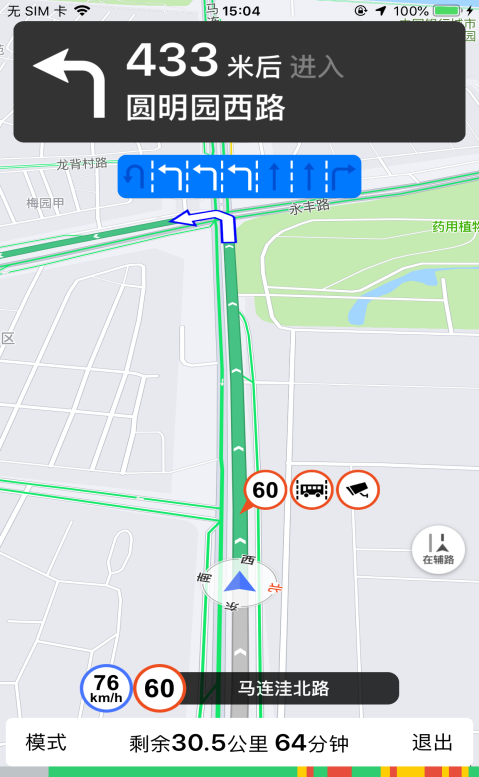
例：2019年5月9日北京日夜间模式切换的时间为：

04：57切换为日间模式

19：25切换为夜间模式

回弹模式：回弹模式：默认值，导航态中，用户使用手势操作地图后进入该模式，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。

主辅路切换：在用户进行路线规划时，通过配置起点路段类型属性能极大地提高路线规划的准确性。其中路段类型包括：在桥上、在桥下、在主路、在辅路、在对面、在桥下主路、在桥下辅路。当用户处于导航过程之中时，用户也能够通过回调方法拿到当前行驶路段可切换的道路类型信息，具体方法方法见下文。



## 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，隐藏，等信息



## 1.6添加控件

用户可对导航页面添加自己的控件



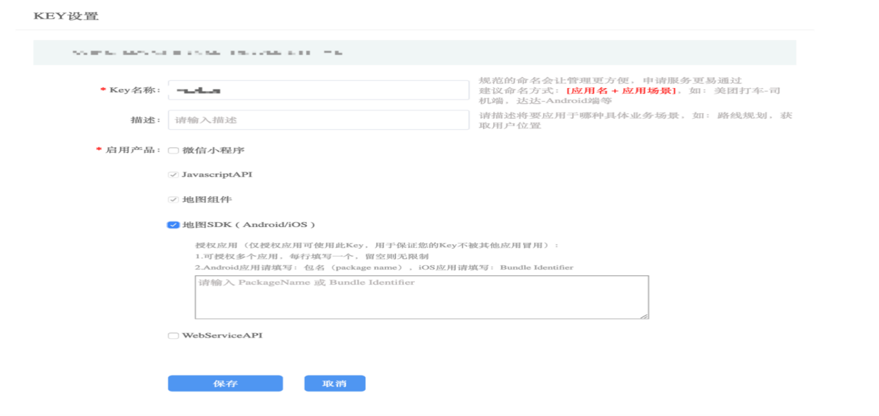
# 2.工程配置

## 2.1申请开发者秘钥

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。



3) 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.Android应用请填写：包名（package name），点击保存，完成key的申请。此时用户即可用刚创建的key了。



4) 导航SDK的权限需联系小助手开通权限。

## 2.2新建工程(已有可跳过)

**第 1 步，下载并安装 Android Studio。**按照指南[下载](https://developer.android.com/sdk/index.html)并[安装](http://developer.android.com/sdk/installing/index.html?pkg=studio) Android Studio。（注：下载地址为Google官方网站）<https://developer.android.com/studio?pkg=studio>

**第 2 步，创建项目。**按以下步骤新建一个 Empty Activity 的应用项目。

1、启动 Android Studio。如果您看到 Welcome to Android Studio 对话框，请选择 Start a new Android Studio project，否则，请点击 Android Studio 菜单栏中的 File，然后点击 New->New Project，按提示输入您的应用名称、公司域和项目位置。 然后点击 Next。

2、选择您的应用所需的机型。 如果您不能确定自己的需要，只需选择 Phone and Tablet。然后点击 Next。

3、在“Add an activity to Mobile”对话框中选择 Empty Activity。 然后点击 Next。

4、按提示输入 Activity 名称、布局名称和标题。 使用默认值即可。 然后点击 Finish。

## 2.3设置腾讯地图、导航Key

在Application标签中的配置如下所示：

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<meta-data

android:name="TencentMapSDK"

android:value="申请的Key"/>

</application>

## 2.4 引入导航SDK、地图SDK、定位SDK

1) Maven引入导航SDK和地图SDK:

repositories {

maven {

url "https://oss.sonatype.org/content/groups/public"

}

}

dependencies {

// 地图SDK

implementation 'com.tencent.map:tencent-map-vector-sdk:4.3.3.5'

// 导航SDK

implementation 'com.tencent.map:tencent-map-nav-sdk:5.2.1.1'

// 导航支持库

implementation 'com.tencent.map:tencent-map-nav-surport:1.0.2.4'

}

2) 引用定位SDK v8.5.5。使用导航SDK推荐使用定制版定位SDK，可联系小助手获得。

## 2.5权限设置

SDK需要使用网络，访问硬件存储等系统权限，在AndroidManifest.xml文件里，添加如下权限：

<!-- 通过GPS得到精确位置 -->

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<!-- 通过网络得到粗略位置 -->

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<!-- 支持A-GPS辅助定位 -->

<uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />

<!-- 访问WiFi状态 -->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<!-- 修改WiFi状态，发起WiFi扫描 -->

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />

<!-- 访问网络状态, 检测网络的可用性，需要网络运营商相关信息用于网络定位 -->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<!-- 访问网络的变化 -->

<uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE" />

<!-- 读写手机SD卡权限 -->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<!-- 访问网络 -->

<uses-permission android:name="android.permission.INTERNET" />

## 2.6混淆配置

在module里找到proguard-rules.pro文件，添加：

-dontwarn com.qq.taf.\*\*

-keep class com.qq.taf.\*\* { \*; }

-keep public class com.tencent.map.ama.navigation.data.NavigationJNI {\*;}

-keep public class com.google.webp.libwebpJNI {\*;}

-dontwarn sun.misc.Unsafe

-keep, includedescriptorclasses public class com.tencent.map.lib.gl.JNI { \*;}

-keep, includedescriptorclasses public class com.tencent.map.lib.gl.\* { \*;}

-keep, includedescriptorclasses public class com.tencent.tencentmap.mapsdk.maps.a.\* { \*;}

# 地图SDK

-keep class com.tencent.tencentmap.\*\*{\*;}

-keep class com.tencent.map.\*\*{\*;}

-keep class com.tencent.beacontmap.\*\*{\*;}

-keep class navsns.\*\*{\*;}

-dontwarn com.qq.\*\*

-dontwarn com.tencent.\*\*

# 定位SDK

-keepclassmembers class \*\* {

public void on\*Event(...);

}

-keep class c.t.\*\*{\*;}

-keep class com.tencent.map.geolocation.\*\*{\*;}

-dontwarn org.eclipse.jdt.annotation.\*\*

-dontwarn c.t.\*\*

# 导航SDK

-dontwarn sun.misc.Unsafe

-keep, includedescriptorclasses public class com.tencent.map.lib.gl.JNI { \*;}

-keep, includedescriptorclasses public class com.tencent.map.lib.gl.\* { \*;}

-keep, includedescriptorclasses public class com.tencent.tencentmap.mapsdk.maps.a.\* { \*;}

-keep class com.iflytek.tts.TtsService.\*\* { \*; }

-keep class com.tencent.map.navi.surport.\*\* { \*; }

-keep class com.tencent.map.screen.\*\* { \*; }

-keep class com.tencent.beacon.\*\* { \*; }

然后在module的build.gradle文件中引用该混淆文件：

buildTypes {

release {

minifyEnabled true

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

# 3.快速接入

## 3.1 实例化导航地图

在layout文件中定义CarNaviView

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<com.tencent.map.navi.car.CarNaviView

android:id="@+id/car\_navi\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

</RelativeLayout>

地图生命周期控制。生命周期管理方法，用户需要在对应生命周期回调方法中，主动调用。

@Override

protected void onStart() {

mCarNaviView.onStart();

super.onStart();

}

@Override

protected void onRestart() {

mCarNaviView.onRestart();

super.onRestart();

}

@Override

protected void onResume() {

mCarNaviView.onResume();

super.onResume();

}

@Override

protected void onPause() {

mCarNaviView.onPause();

super.onPause();

}

@Override

protected void onStop() {

mCarNaviView.onStop();

super.onStop();

}

@Override

protected void onDestroy() {

mCarNaviView.onDestroy();

super.onDestroy();

}

## 3.2 实例化导航管理类

初始化TencentCarNaviManager实例，推荐使用单例模式:

// 实例化导航管理器

mTencentCarNaviManager = new TencentCarNaviManager(this);

// 设置回调监听

Private TencentRouteSearchCallback mTencentSearchCallback = new TencentRouteSearchCallback() {

@Override

public void onRouteSearchFailure(int errorCode, String errorMessage) {

// 算路失败回调

}

@Override

public void onRouteSearchSuccess(ArrayList<RouteData> routes) {

//最多三条路线数据:路线距离，路线点串，预估时间，路况数据等

}

};

// 注册导航地图，接受导航事件

mTencentCarNaviManager.addNaviView(mCarNaviView);

// 开启内置的语音播报模块

mTencentCarNaviManager.setInternalTtsEnabled(true);

## 3.3驾车路径规划

发起路线规划代码如下

mTencentCarNaviManager.searchRoute(from,to, wayPoints, carSearchOptions, mTencentSearchCallback);

其中参数说明如下表格所示，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 |  | 类型 | 必填 | 说明 |
| from |  | NaviPoi | 是 | 起点，一般使用当前定位点赋值 |
|  | latitude | double | 是 | 纬度 |
|  | longitude | double | 是 | 经度 |
|  | poiId | String | 否 | 兴趣点的id |
| to |  | NaviPoi | 是 | 目的地。 |
| wayPoints |  | List<NaviPoi> | 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。 |
| carSearchOptions |  | CarRouteSearchOptions | 是 | 路线规划策略 |
|  | avoidToll | Boolean | 否 | 是否避开收费站 |
|  | avoidHighway | Boolean | 否 | 是否避开高速公路 |
|  | avoidCongestion | Boolean | 否 | 是否避开拥堵 |
|  | navScene | int | 是 | 1:接驾；2:送驾 |
|  | preLocations | List<GpsLocation> | 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。 |
|  | truckRouteSearchParams | TruckRouteSearchParams | 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 |

## 3.4开启、结束导航

// 使用第routeIndex条路开启导航

mTencentCarNaviManager.startNavi(routeIndex);

// 导航过程中将定位SDK回调获得到的定位点传给导航SDK，设置回调频率为每秒1次

private TencentLocationListener mTencentLocationListener = new TencentLocationListener() {

@Override

public void onLocationChanged(TencentLocation tencentLocation, int error, String reason) {

if (error != TencentLocation.ERROR\_OK || tencentLocation == null) {

// 定位失败

return;

}

mTencentCarNaviManager.updateLocation(LocationUtils.convertToGpsLocation(tencentLocation), error, reason);

}

@Override

public void onStatusUpdate(String name, int status, String desc) {

}

};

// 结束导航

mTencentCarNaviManager.stopNavi();

模拟导航方法：

//开启模拟导航，使用第1条路线进行导航

mTencentCarNaviManager.startSimulateNavi(0);

//关闭模拟导航

mTencentCarNaviManager.stopSimulateNavi() ;

# 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：

TencentNavi.Config config = new TencentNavi.Config(); config.setDeviceId(“XXXXXX-XXXX-XXXX-XXXX-XXXXXX”);

TencentNavi.init(this, config);

定位SDK设置代码如下:

TencentLocationRequest request = TencentLocationRequest.create();

Request.setUniqueId(“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

mTencentCarNaviManager.setEnlargedIntersectionEnabled(false)

### 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

mTencentCarNaviManager.setGuidedLaneEnabled(false)

### 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：

// 设置不展示顶部导航面板

mCarNaviView.setNaviPanelEnabled(false);

### 4.2.4更换起点、终点、途经点的图标

// 更换起点图标

mCarNaviView.configStartPointMarkerpresentation(fromMarkerBitmap, realFromMarkerBitmap);

// 更换终点图标

mCarNaviView.configEndPointMarkerpresentation(toMarkerBitmap,realToMarkerBitmap);

// 更换途经点图标

mCarNaviView.configWayPointMarkerpresentation(wayMarkerBitmaps);

### 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module下的assets目录下。 然后设置可替换默认资源：

mTencentCarNaviManager.setIsDefaultRes(false);



### 4.2.6设置自车点位置

// 设置导航过程中3d模式下，车标位于地图宽高的比例，默认x坐标为0.5 ，y坐标为0.75

mCarNaviView.setNaviFixingProportion3D(0.5,0.5);

// 设置导航过程中2d模式下，车标位于地图宽高的比例，默认x坐标为0.5 ，y坐标为0.75

mCarNaviView.setNaviFixingProportion2D(0.5,0.5);

### 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:

//设置导航一直为日间模式

mCarNaviView.setDayNightMode(DayNightMode.DAY\_MODE);

### 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：MODE\_3DCAR\_TOWARDS\_UP ，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：MODE\_2DMAP\_TOWARDS\_NORTH ， 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：MODE\_OVERVIEW ，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：MODE\_REMAINING\_OVERVIEW,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，

// 设置3D车头朝上

mCarNaviView.setNaviMode(NaviMode.MODE\_3DCAR\_TOWARDS\_UP);

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:

// 回弹时间改为10s

mCarNaviView.setBounceTime(10);

### 4.2.9其余导航控件

地图的截图

描述已自动生成 地图的截图

描述已自动生成

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，可以分别控制显隐，代码如下：

// 展示所有导航UI控件

CarNaviInfoPanel carNaviInfoPanel = mCarNaviView.showNaviInfoPanel();

CarNaviInfoPanel.NaviInfoPanelConfig naviInfoPanelConfig = new CarNaviInfoPanel.NaviInfoPanleConfig();

// 隐藏底部退出、设置控件 naviInfoPanelConfig.setButtomPanelEnable(false);

// 隐藏主辅路切换按钮

naviInfoPanelConfig.setChangeRoadEnable(false);

// 隐藏当前车速UI naviInfoPanelConfig.setCurrentSpeedEnable(false);

// 隐藏限速、当前路名UI naviInfoPanelConfig.setLimitAndRoadEnable(false);

// 隐藏全览/非全览切换按钮 naviInfoPanelConfig.setShowFullViewEnable(false);

// 隐藏路况显隐按钮

naviInfoPanelConfig.setTrafficBarEnable(false);

// 更新以上设置 carNaviInfoPanel.setNaviInfoPanelConfig(naviInfoPanelConfig);

接收导航退出按钮的点击事件:

carNaviInfoPanel.setOnNaviInfoListener(new carNaviInfoPanel.OnNaviInfoListener() {

@Override

public void onBackClick() {

}

}

);

## 4.3导航回调事件

1)可实现INaviView协议从而获取导航展示信息的回调事件:

private INaviView mINaviView = new INaviView() {

@Override

public void onUpdateNavigationData(NavigationData navigationData) {

// 获取导航过程相关数据

}

@Override

public void onShowEnlargedIntersection(Bitmap bitmap) {

//获取路口放大图资源

}

@Override

public void onHideEnlargedIntersection() {

// 隐藏路口放大图

}

@Override

public void onShowGuidedLane(Bitmap bitmap) {

// 获取车道线资源

}

@Override

public void onHideGuidedLane() {

// 隐藏车道线

}

@Override

public void onUpdateTraffic(String routId, int totalDistance, int leftDistance, ArrayList<LatLng> points, ArrayList<TrafficItems> trafficItems, Boolean isCurrentRoute) {

// 获取更新导航线路路况的信息。

}

@Override

public void onGpsRssiChanged(int rssi) {

// GPS信号强度变化. 0:无信号，1：信号弱，2：信号中，3：信号强

}

}

// 添加事件监听

mTencentCarNaviManager.addNaviView(mINaviView);

// 适时移除事件监听

mTencentCarNaviManager.removeNaviView(mINaviView);

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)获得导航状态事件回调：

private TencentNaviCallback mTencentNaviCallback = new TencentNaviCallback() {

@Override

public int onVoiceBroadcast(NaviTts tts) {

// 语音播报文案

return 1;

}

@Override

public void onUpdateAttachedLocation(AttachedLocation location) {

// 导航过程中定位点的信息回调，其中包含原始定位和吸附定位

}

@Override

public void onArrivedDestination() {

// 即将到达目的地

}

@Override  
 public void onStartNavi() {

// 开启导航

}

@Override

public void onStopNavi() {

// 结束导航

}

@Override

public void onOffRoute() {

// 发生偏航

}

@Override

public void onRecalculateRouteStarted(int type) {

// 偏航发起重新算路，开发者无需处理此信息

}

@Override

public void onRecalculateRouteCanceled() {

// 偏航重新算路请求取消，可能是由于误偏航之后马上纠正，从而取消请求

}

@Override

public void onRecalculateRouteSuccess(int type, ArrayList<RouteData> routeDataList) {  
 // 偏航重新路线规划成功，导航默认选择了第一条路进行导航  
 }

@Override

public void onRecalculateRouteFailure(int type, int errorCode, String errorMessage) {

// 偏航重新路线规划失败，开发者无需处理此信息  
 }

@Override

public void onPassedWayPoint(int passPointIndex) {

// 达到第passPointIndex个途经点回调

}

@Override

public void onUpdateRoadType(int roadType) {

// 主辅路切换提示: 0为无提示，1为在桥上，2为在桥下，3为在主路，4为在辅路，5为在对面，6为桥下主路，7为桥下辅路

}

## 4.4.一键上报

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。

。手机屏幕截图

描述已自动生成 一些文字和图片的手机截图

描述已自动生成

// 设置城市编码：

UploadPercentor.*setAdCode*(**"城市编码"**);  
// 设置apikey

UploadPercentor.*setApiKey*(**"xxxxxx-xxxx-xxxx-xxxxxx"**);  
// 设置订单id

UploadPercentor.*setOrderId*(**"xxx"**);  
// 设置用户id

UploadPercentor.*setUserId*(**"xxx"**);

// 可能造成语音冲突，可关闭语音反馈功能

UploadPercentor.setHideVioce();

以上配置必须设置，否则会导致上传失败。开发者需要自己创建上报按钮，点击上报按钮调用如下方法：

OneKeyReportManager.*getInstance*().showOneKeyReportDialog(mContext);

## 4.5 伴随路线

可开启导航过程中的伴随路线

地图的截图

描述已自动生成

// 开启伴随路线

mTencentCarNaviManager.setMulteRoutes(true);

// 接受相关事件回调

mCarNaviView.setNaviMapActionCallback(mTencentCarNaviManager);

## 4.6 自动灌点

以前导航开启之后需要每秒调用如下方法给导航SDK设置定位点，以驱动导航工作：

mTencentCarNaviManager.updateLocation(location, error, reason);

自动灌点的功能是导航SDK耦合腾讯定位SDK，然后导航SDK自己去自动获得定位点，开发者不需要调用上述方法给导航SDK设置定位点了。

开启自动灌点功能代码如下，

// 开启自动灌点开关

mTencentCarNaviManager.setUseExtraLocationData (false);

// 定位管理类关联上下文

TNKLocationManager.getInstance().setContext(getApplicationContext());

开发者需要使用导航SDK的定位管理类来获取定位点信息，无需直接调用定位SDK的TencentLocationManager：

// 添加定位数据回调，如果此时未开启定位SDK，会自动开启定位SDK

TNKLocationManager.getInstance().addLocationListener(new ITNKLocationCallBack() {

@Override

public void requestLocationUpdatesResult(int errorCode) {

}

@Override

public void onLocationChanged(TencentLocation tencentLocation, int errorcode, String reason) {

}

});

// 移除定位数据回调监听

TNKLocationManager.getInstance().removeLicationListener(...);

# 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&avoidCongestion=true

1. 送驾路线规划建议策略:

navScene=2&avoidToll=false&avoidHighway=false&avoidCongestion=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

mSearchOptions.setRouteTraticeType(2);

// 设置单价：每公里2.6元、每分钟0.6元

mSearchOptions.setPrice(2.5,0.6);