腾讯 Android 导航 SDK 接入文档

导航版本号 5.2.2 支持 Android 4.0 及以上系统

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。

Demo 地址: https://github.com/TencentLBS/TencentNaviDemo_Android

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

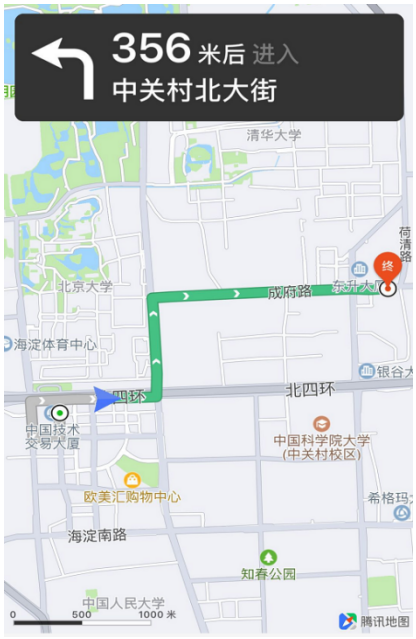
用户可以根据需求来设置相应的导航模式、日夜间模式、主辅路切换功能，

具体效果如下：

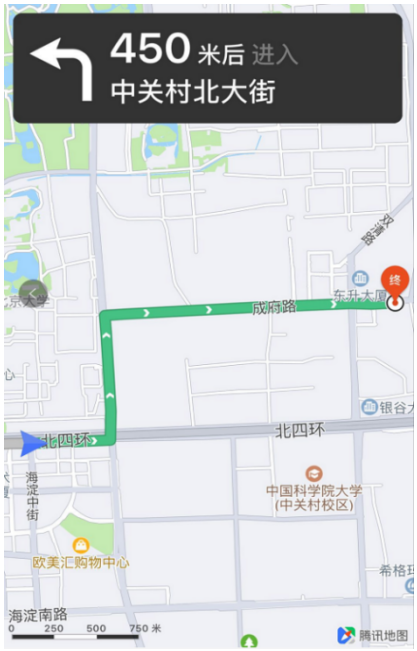
2D 模式：导航时地图始终朝北，车头实时改变方向

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

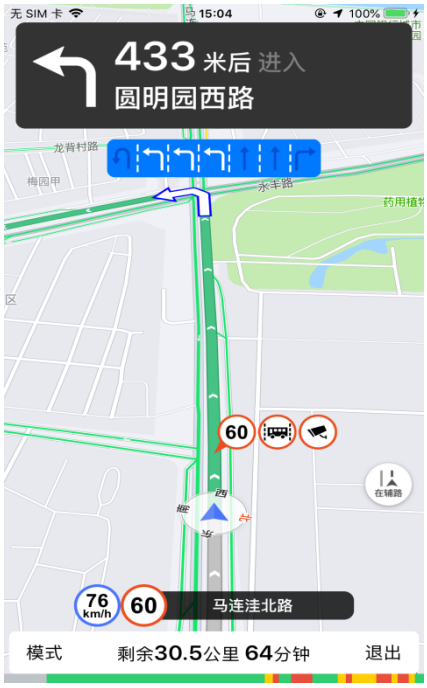
例：2019 年 5 月 9 日北京日夜间模式切换的时间为：

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

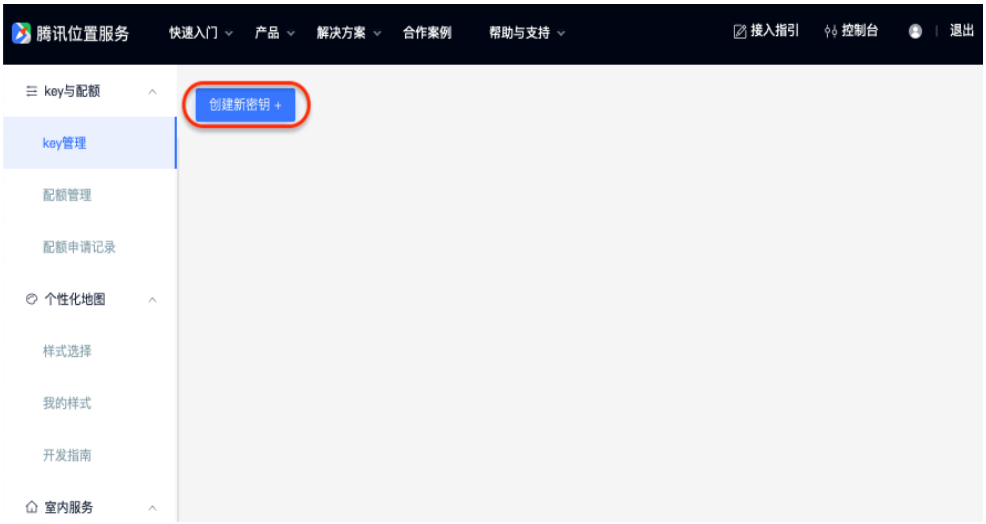
1.6 添加控件

用户可对导航页面添加自己的控件

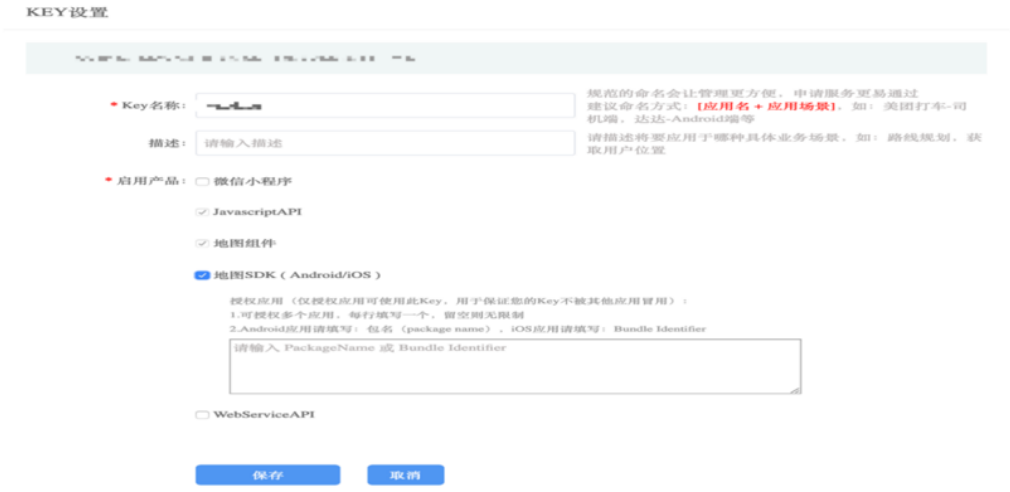
2. 工程配置

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。按照指南下载并安装 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:5.2.2.0'
  // 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.conver
        tToGpsLocation(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,realToMarkerBit
map);

// 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

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。

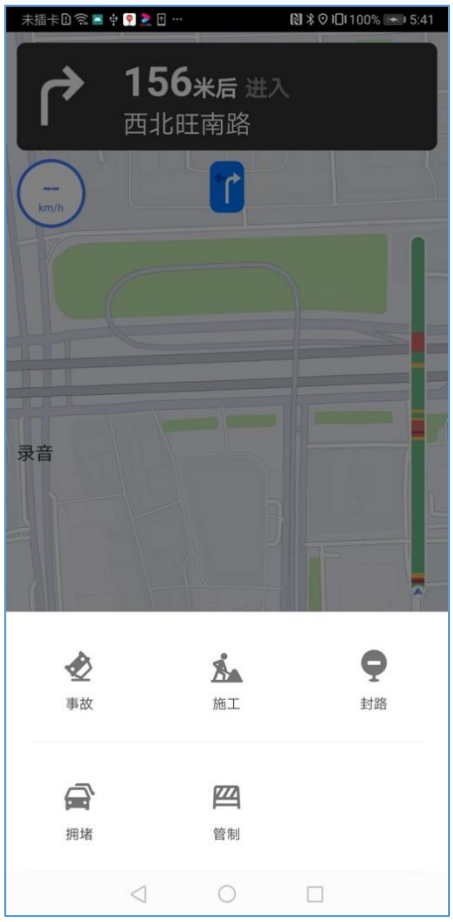
2)获得导航状态事件回调：

```
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。



。

```
// 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


```
// 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(...);
```

4.7 智能定位

智能定位主要是在驾车过程中 GPS 长时间丢失后，利用网络定位结果+导航规划路径来进行定位/导航。使用 4.5 自动灌点功能后，智能定位自动开启。

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
```

2）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);
```