Open platform Demo instructions

一、The resource file configuration

1. Set the/opt/ directory to read and write permissions chmod u + rw opt/

2. Place the resource file in bnav in the/opt/BaiduMapAuto/bnav/directory (this directory is configurable, please modify the CFG\_DATA\_CONFIG variable in BaiduMapAutoQmlInterface.cpp) Offline Data Update Please follow the bnav/ ReadMe.txt documentation

二、The Linux environment depends on the installation of the library

Install the build tool set installation command:

sudo apt-get install build-essential

After the installation is complete verify gcc/ g ++ version PS: if meet "E: Unable to locate package essential", please update apt-get sudo apt-get update

Install the OpenGL toolset Install the opengl 3 command:

sudo apt-get install freeglut3-dev

Install the opengl es command:

sudo apt-get install libgles2-mesa-dev

Install the freetype update index list

apt-get update

sudo apt-get install libfreetype6-dev

三、Demo project configuration

Create so soft link: enter automap-linux-qml/ libs/ directory, execute sh ln\_sh.sh.

Function module instructions

Development environment: Ubuntu14.04 + Qt 5.7.0, the use of qml development UI, the project name automap-linux-qml.pro

ubuntu environment to build http://agroup.baidu.com/automap\_linux/md/article/152991

Engine use process brief introduction: (Please combine BaiduMapAutoQmlInterface.cpp) by function module is divided into mapView, search, routeGenerator, routeGuide;

Each module must create a new instance, call the corresponding API interface, API see doc;

MapView module

The mapView instance must be created after the SurfaceView is created

1**、initialize the base map**

BDMapViewer\* m\_pMap;  
 m\_pMap = new BDMapViewer(BDMapType::FRONT, DEV\_WINDOW\_WIDTH, DEV\_WINDOW\_HEIGHT);  
Parameters  
[in] type map type such as front, rear left and so on  
[in] width Sub-Surface's width value  
[in] height Sub-Surface's height value  
// set the map display level

m\_pMap-> setMapLevel (14);

// Set the map center point offset: such as the navigation process to set the car under the display or display the intersection when the enlarged view, set the navigation route to the right display

m\_pMap-> setMapOffset (0, -100);

// Set day and night mode

m\_pMap-> setMapTheme (BDMapTheme :: NIGHT\_MODE1);

2, in the drawing thread to 40ms cycle call mapView-> draw ()

m\_pMap.draw ();

**3、Listen listen messages to listen**

m\_pMap->setEventListener(this);

Base map level change

void MyBDMapViewerListener::onMapPositionChanged(const BDMapViewer& object, const common::BDGeoCoord& coord) {  
 ...  
}  
Parameters  
[in] pos The geo-position of the center of BDMapViewer instance

**4、Polygon drawing**

Sample code

// Polygon fixed point definition, coordinates for the National Bureau of Surveying latitude and longitude coordinates, and multiplied by 100000, clockwise direction fixed

std :: vector <common :: BDGeoCoord> geoArr;   
 common::BDGeoCoord tmpCoord1;  
 tmpCoord1.setLongitude(11631326);  
 tmpCoord1.setLatitude(4004173);  
 geoArr.push\_back(tmpCoord1);  
  
 common::BDGeoCoord tmpCoord2;  
 tmpCoord2.setLongitude(11633548);  
 tmpCoord2.setLatitude(4003021);  
 geoArr.push\_back(tmpCoord2);  
  
 common::BDGeoCoord tmpCoord3;  
 tmpCoord3.setLongitude(11662187);  
 tmpCoord3.setLatitude(4006176);  
 geoArr.push\_back(tmpCoord3);  
  
 common::BDGeoCoord tmpCoord4;  
 tmpCoord4.setLongitude(11639759);  
 tmpCoord4.setLatitude(3990877);  
 geoArr.push\_back(tmpCoord4);  
  
 common::BDGeoCoord tmpCoord5;  
 tmpCoord5.setLongitude(11629523);  
 tmpCoord5.setLatitude(3998697);  
 geoArr.push\_back(tmpCoord5);  
// Style definition, rgba, value between 0 and 1

m\_pMap-> setMapRegionStyle (0.0f, 0.6f, 0.1f, 0.3f);

// set the data and start drawing

m\_pMap-> setMapRegionData (geoArr);  
// Clear polygon drawing

m\_pMap-> clearMapRegionData ();

Base map POI click (POI details data need to be further modified)

5, custom basemap annotation

Sample Code - Add a label point

std::vector<HDynamicMapPoint> array;  
HDynamicMapPoint point0, point1, point2;  
//大厦  
point0.name = "百度大厦";  
point0.point.setLongitude(11630218);  
point0.point.setLatitude(4005007);  
point0.tag = "TYPE\_1";  
array.push\_back(point0);  
//奎科  
point1.name = "奎科科技大厦";  
point1.point.setLongitude(11630691);  
point1.point.setLatitude(4004174);  
point1.tag = "TYPE\_1";  
array.push\_back(point1);  
//科技园  
point2.name = "百度科技园";  
point2.point.setLongitude(11627512);  
point2.point.setLatitude(4004396);  
point2.tag = "TYPE\_2";  
array.push\_back(point2);  
  
m\_pMap->setDynamicMapPoints(array);

Sample Code - Set Marker Icon

QImage\* img = new QImage(16, 16, QImage::Format\_RGBA8888);  
int iTWidth = img->width();  
int iTHeight = img->height();  
int iChannel = 4;  
img->fill(QColor(255, 0, 0, 200));  
m\_pMap->setDynamicMapImages("TYPE\_1", (HChar \*)img->bits(), iTWidth, iTHeight, iChannel);  
delete img;

Note: the engine will be deep copy of the picture buffer, so the upper memory to be their own management

Search Module

Retrieve Distinguished Area Retrieval and Peripheral Retrieval, by setting different search filters to distinguish between search module support sug search

1、Initialize the Search module

BDPOISearch\* m\_pSearch;  
m\_pSearch = new BDPOISearch();

2, set the search filter (retrieve the relevant parameter attributes, sug search without setting the filter)

a,initialize Filter

BDPOISearchFilter filter;

b, filter set the region search or peripheral search, the two search for the mutually exclusive, that is, filter.setAddress (address) and filter.setBoundary (pos, 1000) at the same time using only one. Area Search:

BDAddress address;

address.setRegionCode (131);// set the area id to Beijing

filter.setAddress (address);

Peripheral Search: Requires radius greater than 0, unit meter, code example:

  BDGeoCoord pos;

  pos.setLatitude (3672808);

  pos.setLongitude (11920123);

  filter.setBoundary (pos, 1000);

c, flip function

  filter.setPageOption (BDUInt32 curPage, BDUInt32 pageCount);// curPage: current page number, pageCount: number of search results per page

  BDPOISearch :: isLastPage ();// determine whether the last page

3, set the search keyword: filter setKeyword

filter.setKeyword ("Tiananmen");

4, search in the set filter

m\_pSearch->setSearchFilter(filter);

5, initiated search/ sug retrieval

m\_pSearch-> search ();

m\_pSearch-> searchSug ("Ken");// search sug without setting filter

6, set the monitor to obtain the retrieval results callback overload IBDPOISearchListener, and the realization of the virtual function method in the class. Code example:

class BaiduMapAutoInterface: public IBDPOISearchListener

{

    m\_pSearch-> setEventListener (this);

// Get the result of the search

    void onSearchResultUpdate (BDPOISearch \* pSearch, const BDInt32 & status, const BDInt32 & count)

    {// logic code

    }

// Get the result of sug

    void onSearchSugResultUpdate (BDPOISearch \* pSearch)

    {// logic code

    }

}

7, cross-city search results that cross-city search results will return to the destination where the city name, get the results of the RegionCode, re-initiated regional search. The identity of this scenario is: BDPOIInfo :: getUid () is empty. For example: Retrieve "Shanghai KFC" results return, if info.getAddress (). GetRegion () [0] is "", the description returns the city list, the list includes the Shanghai and Shanghai regioncode, and then use regioncode to re-launch the region search.

8, POI Details Search Results Search example http://dev.mapauto.baidu.com/detail?qt=infauto&uid=784961807213499085

9, search results map bubble show

1) initiate a search based on the above steps, wait for a search result notification

2) to obtain a search result list

searchApi-> getResultItemCount (count);

int offset = 0;

std::vector<BDPOIInfo> resultList;  
searchApi->getResultItemList(offset, count, resultList);

3) the coordinates of the search results into vector

std :: vector <BDGeoCoord> coords;

for (int i = 0; i <resultList.size (); ++ i)

{

     coords.push\_back (resultList [i] .getPosition ());

}

4) call the bottom of the interface to add the bubble, the search results for the red bubble, red bubbles marked poi index

  m\_pMap-> addBkgPoiMarker (coords);

10, click on poi on the screen

1) capture the screen click event, get the screen coordinates, call the mapview interface getSelectItemInfo get poiInfo

     QPoint curPoint = event-> pos ();

     BDPOIInfo info;

     m\_pMap-> getSelectItemInfo (curPoint.rx (), curPoint.ry (), info);

2) to achieve two message callback, in the callback function call call poi interface addPoiMarker (common :: BDGeoCoord & coord); call getSelectItemInfo will call the corresponding callback function, selected, the map will pop up the blue bubble

• onMapPoiClicked is the poi point on the base map

• onMapBkgPoiMarkerClicked the poi point for retrieving the red bubbles of the result

Note: If you click on the blank map area is not get poi information, so no bubble display

     void MapViewerListener :: onMapPoiClicked (const BDMapViewer & object, const BDPOIInfo & info)

    {

          map-> addPoiMarker (info.getPosition ());

    }

    void MapViewerListener :: onMapBkgPoiMarkerClicked (const BDMapViewer & object, const BDUInt32 & index)

    { map->addPoiMarker(resultList[index].getPosition());  
 }

11、hidden bubbles

m\_pMap->hideBkgPoiMarker();// hide red bubbles  
m\_pMap->hidePoiMarker();// hide blue bubbles

RouteGenerator Module

1, initialization

BDRouteGenerator \* m\_pRouteGenerator;

m\_pRouteGenerator = new BDRouteGenerator ();

2, set the starting point, the end point, code examples

// 1st set startPosition

    BNACoordinate cur\_pos;

    AMGeoTools :: locate (cur\_pos.iLongitude, cur\_pos.iLatitude);

    BDGeoCoord start\_geo\_coord;

    start\_geo\_coord.setLatitude (cur\_pos.iLatitude \* 100000);

    start\_geo\_coord.setLongitude (cur\_pos.iLongitude \* 100000);

    BDRoutePosition start\_pos;

    start\_pos.setPosition (start\_geo\_coord);

    m\_pRouteGenerator-> setStartPosition (start\_pos);

// 2nd set destPosition

BDGeoCoord dest\_geo\_coord;

dest\_geo\_coord.setLatitude (latitude \* 100000);

dest\_geo\_coord.setLongitude (longitude \* 100000);

BDRoutePosition dest\_pos;

dest\_pos.setPosition (dest\_geo\_coord);

m\_pRouteGenerator-> setDestination (dest\_pos);

3, set the search parameters, set through the routeOption

BDRouteOption routeOption;

std :: vector <std :: string> optionList;

// Get the list of settable parameters

routeOption.getOptionList (optionList);

// set to high speed first

routeOption.setOption ("BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_HIGHWAY", 1);

// set the module for online retrieval  
routeOption.enableOnlineMode(true);  
// set the license plate number, localize the limit line

routeOption.setOption ("BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_CARNUM", 1);

BDCarInfo carinfo ("P261U6", "zin", BDCarTypeEnum :: BD\_CAR\_TYPE\_14L);

m\_pRouteGenerator-> setLocalCarInfo (carinfo);

// set the option to the Generator

m\_pRouteGenerator-> addRouteOption (routeOption);

Operators prefer to use road Description: -BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_RECOMMEND recommended route - BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_HIGHWAY high-speed priority - BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_NO\_TOLL small fee - BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_AVOID\_TRAFFICJAM avoid congestion - BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_CARNUM license limit line eg: The following settings are high-speed priority and avoid congestion at the same time take effect routeOption.setOption ( "BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_HIGHWAY", 1);// High speed priority, the second parameter is 1 when the entry into force, it is not valid when the routeOption.setOption ("BNA\_ROUTEPLAN\_PREFERENCE\_TYPE\_AVOID\_TRAFFICJAM", 1); Note: high-speed priority and less charges are mutually exclusive; license plate limit and other items at the same time; The route can only be used with the license plate limit line; avoid congestion can not be used in conjunction with the recommendation, and other items can be used simultaneously

4, set the way through the point

BDRoutePosition wpt\_pos;  
 BDGeoCoord geo\_coord;  
 geo\_coord.setLatitude( lat );  
 geo\_coord.setLongitude( lon );  
 wpt\_pos.setPosition( geo\_coord);  
 m\_pRouteGenerator->addWaypoint(wpt\_pos);

The next pass before the way to clear the way through：

m\_pRouteGenerator->clearWaypoints();

5, initiate the calculation

m\_pRouteGenerator-> generate ();

6, calculate the results of monitoring and drawing on the map overloaded IBDRouteGeneratorListener, and to achieve the class in the virtual function method onGenerateStatusUpdated (). Code example:

class BaiduMapAutoInterface: public IBDRouteGeneratorListener

{

m\_pRouteGenerator-> setEventListener (this);

// Get the path plan results

void BaiduMapAutoInterface :: onGenerateStatusUpdated (const BDRouteGenerator & generator, const BDRouteGeneratingStatus & status, const BDByte & numOfRoutes)

{

// Get the number of routes returned

    int route\_cnt = numOfRoutes> 3? 3: numOfRoutes;

     BDGeoArea geo\_area;

     BDRouteInfo route\_info;

// Get the route route array to mapview  
 std::vector<baidu::mapauto::navi::route::BDRouteInfo> routeInfos;  
 for (int i = 0; i < route\_cnt; i++)  
 {  
 int index = i + 1;  
 m\_pRouteGenerator->getRouteInfo(i, route\_info);  
 routeInfos.push\_back(route\_info);  
 emit setRoutePlanResultItemData(index, route\_info.getTotalTime(),  
 route\_info.getTotalDistance(), route\_info.getTrafficLight());  
 }  
  
// add directions to mapView

    m\_pMap-> addRoutes (routeInfos);

// Adjust the basemap display scale

    geo\_area = route\_info.getArea ();

    baidu :: mapauto :: common :: BDRectangle rect = {0, 0, 800, 600};

    m\_pMap-> fitGeoAreaToScreen (geo\_area, rect);

   }

}

RouteGuide Module

1, initialization

BDRouteGuide \* m\_pRouteGuide

m\_pRouteGuide = BDRouteGuide :: getInstance ();

m\_pRouteGuide-> setBasicEventListener (this);

m\_pRouteGuide-> setViewEventListener (this);

2, open the navigation or open the simulation navigation (1) to open the navigation:

BDResult result = m\_pRouteGuide-> startRouteGuidance ();

(2) to open the simulation:

BDResult result = m\_pRouteGuide-> startSimulation ();

3, simulation navigation set speed:

float speed = 10;//> 0, speed +; <0, speed-

m\_pRouteGuide-> setSimulationSpeed (10);

4, TBT message processing

void onTurnInfoUpdated(const std::vector<BDTurnInfo>& turnInfoList)  
{  
 BDTurnInfo turn\_info = turnInfoList[0];  
  
// Next road name  
QString next\_road\_name = QString::fromStdString(turn\_info.getNextRoadName());  
  
// Next road distance  
int neext\_road\_distance = turn\_info.getRemainDistance();  
  
// Remain time  
int route\_index = 0;  
BDUInt32 remain\_time;  
m\_pRouteGuide->getRemainTime(route\_index, remain\_time);  
  
// Remain distance  
BDUInt32 remain\_distance;  
m\_pRouteGuide->getRemainDistance(route\_index,remain\_distance);  
}

5, intersection zoom (1) message notification, update the process

void onILSImageViewUpdated (const BDILSImageViewInfo & hInfo) {

// get buffer

     const char \* pBGMap = hInfo.getBackGoundMap (). c\_str ();

     const char \* pArrowMap = hInfo.getArrowMap (). c\_str ();

     unsigned int bgLength = 0;

     unsigned int arrowLength = 0;

     unsigned char \* pbgByteBuf = NULL;

     unsigned char \* parrowByteBuf = NULL;

     m\_pRouteGuide-> getRasterExpandMapImage (pBGMap, 1, & pbgByteBuf, bgLength);

     m\_pRouteGuide-> getRasterExpandMapImage (pArrowMap, 0, & parrowByteBuf, arrowLength);

// load file

     ...

     }

(

2) Close the road Enlargement: Received the following message and put away the enlarged view

void onHideILSImageViewUpdated (bool & status) {

}

6, intelligent scale

// Close autoLevel

BDResult result1 = m\_pRouteGuide-> setMapMemoryScale (BNAScreenCenter, level);

BDResult result2 = m\_pRouteGuide-> setMapAutoLevelStatus (BNAScreenCenter, false);

7, traffic data: drawing beam chart

std :: vector <BDRoadCondition> arrRoadCondition;

float progess

BDResult res1 = m\_pRouteGuide-> getRoadCondition (arrRoadCondition);

BDResult res2 = m\_pRouteGuide-> getCarProgress (progess);

8, parking lot processing logic (1) to confirm receipt of the message

void onDestParkAvailableUpdated (bool & status) {}

(3) When the search results are returned, they are displayed on the map and the text of the parking lot is displayed. (4) When the user clicks on the "parking lot" and determines "stop here" (5) to re-set the starting point for the current point position: GetMapMatchInfo get (6) set the end point for the parking lot position (7) to start the algorithm (8) call switchRoute switch to the algorithm The first data in the results

9, full view mode function (1) into the full range:

m\_pRouteGuide-> enterViewAll (SIMPLE\_LAND);

/ \*\* <SIMPLE\_LAND reference BDRouteGuideViewAllType \*/

(2) Exit full view:

m\_pRouteGuide-> exitViewAll ();

10, cruise mode (1) is not open navigation mode, the number of consecutive geographical changes to a certain threshold and the speed reaches a certain threshold, will be thrown to open the message

onRouteCruiseAvailableUpdated ();

(2) according to the need to call the interface, open the cruise

startRouteCruise ();

(3) Under certain conditions (such as user operating map), you can call the interface to close the cruise

stopRouteCruise ();

11, the main road switch function: (1) the main road switching and blanking are through the message control

onChangeRouteUpdated (BDRouteGuideChangeRouteType resChangeRoute);

Where BDRouteGuideChangeRouteType is defined as follows:

CHANGETO\_MAINROUTE,// switch to main road

CHANGETO\_SALVEROUTE,// switch to slave

CHANGETO\_PARALLELROUTE,// switch to parallel route

HIDECHANGEROUTE,// Hide the switch information

(2) call the interface to complete the road switch

m\_pRouteGuide-> onlineChangeRoute ();

12, get the current road name (basemap non navigation mode use, navigation mode recommended onRoadNameUpdated ()):

m\_pRouteGuide-> getCurRoadNameByPos (longitude, latitude);

Which longitude, latitude format for gcj02 \* 100000

13, in the navigation route refresh

m\_pRouteGuide-> refreshRoute ();

After the refresh is complete, the refresh result is returned in the onRefreshRouteUpdated callback method, the result type description：

typedef enum \_BDRefreshRouteStatusType {  
REFRESH\_ROUTE\_INVALID = -1,/\*\*< invalid\*/  
REFRESH\_ROUTE\_SUCCESS,/\*\*< refresh success\*/  
REFRESH\_ROUTE\_FAILED,/\*\*< refresh failed\*/  
REFRESH\_ROUTE\_NEWROUTE,/\*\*< update new Route\*/  
REFRESH\_ROUTE\_NO\_NEWROUTE,/\*\*< no Route\*/  
REFRESH\_ROUTE\_NET\_TIMEOUT,/\*\*< timeout\*/  
REFRESH\_ROUTE\_OTHER\_ROUTE/\*\*< other route\*/  
}BDRefreshRouteStatusType;

Location Module

The Location module function consists of two parts: 1. Support the status of the LocationManager operations, such as opening, pausing, resuming, stopping, and changing the LocationDriver for simulated navigation. 2. Supports synchronous and asynchronous access to Location-related information.

1, initialize and destroy the Location module:

// initialization

BDLocation \* m\_pLocation;

m\_pLocation = BDLocation :: getInstance ();

m\_pLocation-> setEventListener (listener);

// destroyed

m\_pLocation-> unsetEventListener ();

2, operation LocationManager running status:

m\_pLocation-> start ();// open the Location thread

m\_pLocation-> stop ();// End the Location thread

m\_pLocation-> pause ();// pause the Location thread

m\_pLocation-> resume ();// restore the Location thread

// Switch to NMEA and NaviTrack simulators for simulated navigation. (interval is the sampling time, the unit is us; speed unit is m/ s)

m\_pLocation-> setNmeaSimulator (trackFile, interval, speed));

m\_pLocation-> setNaviTrackSimulator (trackFile, interval, speed));

// switch to arithmetic results for simulated navigation, not yet implemented

m\_pLocation-> setRouteSimulator ();// not yet implemented

// switch the normal navigation mode (currently support passive driver, if different platforms, need to be configured separately, such as ccos driver)

m\_pLocation-> unsetSimulator ();// Currently only passive driver is supported

3, synchronization access Location Related information:

// Synchronize to get the Location structure

BDLocationInfo LocationInfo;

BDResult ret = m\_pLocation-> getLocationInfo (LocationInfo);

if (ret == BDResult :: ERROR) {

// bdlog\_e record the error message

}

4, asynchronous access Location Related information:

// Get the Location structure asynchronously

class LocationApi: public IBDLocationListener {

     BDLocationInfo m\_clLocationInfo;

     void onLocationInfoUpdated (const BDLocationInfo & LocationInfo) {

// cycle trigger

// bdlog\_v records the LocationInfo output data  
 doSomeThingWithLocationInfo(LocationInfo);  
 }  
}  
m\_pLocationApi = new LocationApi();  
m\_pLocation->setEventListener(m\_pLocationApi);  
// onLocationInfoUpdated triggered periodly

MapMatching Module

MapMatching module to achieve the patrol state, navigation mode MapMatching data output function, and inertial data output function. At present, the inertial navigation function is not implemented, only the inertial data output interface is provided.

1, initialization and destruction of MapMatching module:

// initialization

BDMapMatching \* m\_pMapMatching;

m\_pMapMatching = BDMapMatching :: getInstance ();

m\_pMapMatching-> setEventListener (listener);

// destroyed

m\_pMapMatching-> unsetEventListener ();

2, synchronized access to MapMatching related information:

// Synchronize the MapMatching structure

BDMapMatchingInfo MapMatchingInfo;

BDResult ret = m\_pMapMatching-> getMapMatchingInfo (MapMatchingInfo);

if (ret == BDResult :: ERROR) {

// bdlog\_e record the error message

}

4, Asynchronous access to MapMatching related information:

// Get the MapMatching structure asynchronously

class MapMatchingApi: public IBDMapMatchingListener {

    BDMapMatchingInfo m\_MapMatchingInfo;

    void onMapMatchingInfoUpdated (

        const BDMapMatchingInfo & MapMatchingInfo) {

// cycle trigger

// bdlog\_v records MapMatchingInfo output data

        doSomeThingWithMapMatchingInfo (MapMatchingInfo);

    }

}m\_pMapMatchingApi = new MapMatchingApi();  
m\_pMapMatching->setEventListener(m\_pMapMatchingApi);  
// onMapMatchingInfoUpdated triggered periodly