General Magic POC

This article should document the findings of the first evaluation of the iOS SDK of General Magic. General Magic is providing a SDK for navigation and we want to find out if the service is an alternative to Mapbox we are currently using.

In a timeboxed environment the target was to check to functionalities of the SDK and if it can fulfill our needs for a navigation SDK. Our current main needs are:

- Get a cycling route with turn by turn navigation instructions based on a list of GPS coordinates
- Back-route the user back to the route to the closest point he has not arrived (no complete new route to the end)
- Draw the route on a map which is an independent view in SwiftUI
- Get the instructions in a way to display them in an independent view
- Get a map style dedicated for cycling

Documentation and examples

General Magic is providing a public available documentation on their website: Maps SDK for iOS | General Magic Unfortunately contains the “Guides“ section mainly screenshots from their examples and how to use the examples. There is no documentation how to use their SDK in code. This makes it very difficult to understand how to use the SDK and what exactly is to do to reach specific things.

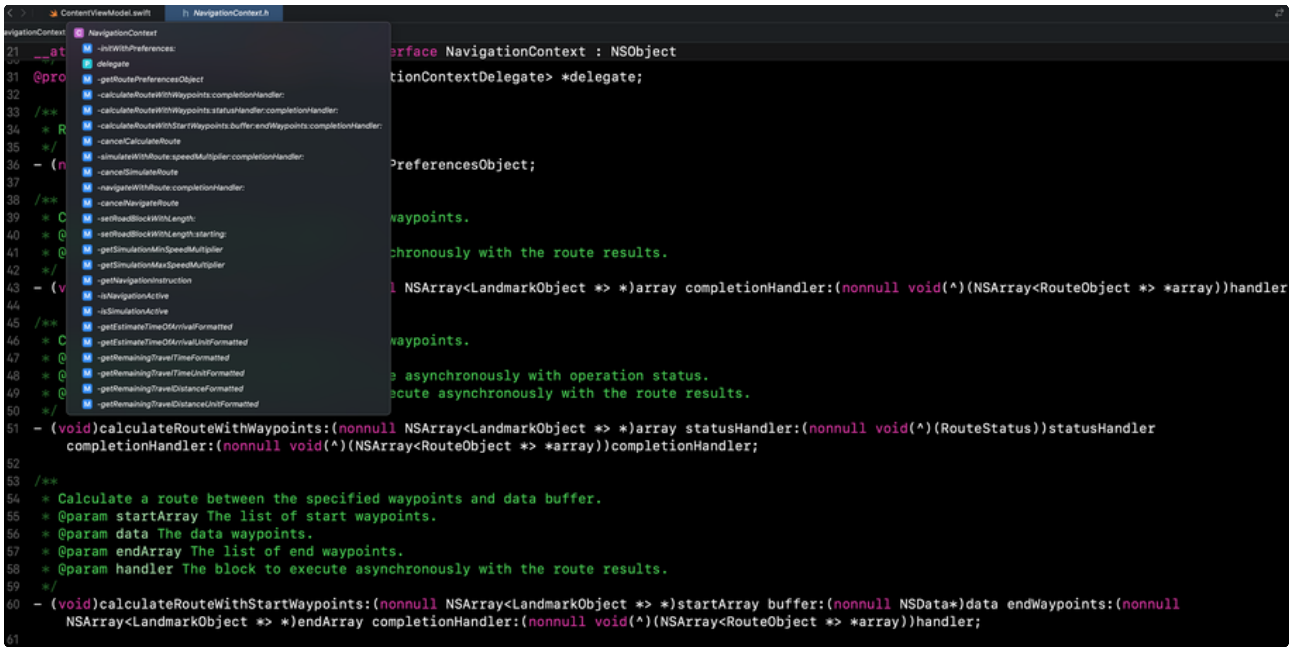
The examples and the SDK itself are only reachable after creating an account. It is at some points not clear what the example is exactly demonstrating and you have to search a bit through the examples to understand what is what showing. The code in the examples is not documented so its hard to understand the responsibilities in the code.

Unfortunately the examples were crashing with the current version of the SDK (**07.01.22.39.4A2FB489**) on the simulator on Intel Macs. So we needed to test the examples on a real device which was a bit cumbersome and took time to recognize it.

SDK integration

The SDK is only downloadable from the General Magic website and there is no dependency management tool integrated which makes it hard to maintain. After dropping the SDK into the project only the API token has to be set and the SDK is ready to use. It is nice that the token is set in code and the way of receiving the token is free (e.g. loading from the backend) in comparison to Mapbox where you have to set the token hard in the info.plist.

The SDK is providing us only Objective-C header files to understand their interface which is pretty hard to read especially in large classes is really hard to get an overview about the functionalities.


ContentViewModel.swift
NavigationContext.h
NavigationContext
-initWithPreferences:
delegate
-getRoutePreferencesObject
-calculateRouteWithWaypoints:completionHandler:
-calculateRouteWithWaypoints:statusHandler:completionHandler:
-calculateRouteWithStartWaypoints:buffer:endWaypoints:completionHandler:
-cancelCalculateRoute
-simulateWithRoute:speedMultiplier:completionHandler:
-cancelSimulateRoute
-navigateWithRoute:completionHandler:
-cancelNavigateRoute
-setRoadBlockWithLength:
-setRoadBlockWithLength:starting:
-getSimulationMinSpeedMultiplier
-getSimulationMaxSpeedMultiplier
-getNavigationInstruction
-isNavigationActive
-isSimulationActive
-getEstimateTimeOfArrivalFormatted
-getEstimateTimeOfArrivalUnitFormatted
-getRemainingTravelTimeFormatted
-getRemainingTravelTimeUnitFormatted
-getRemainingTravelDistanceFormatted
-getRemainingTravelDistanceUnitFormatted

```
__at                                    erface NavigationContext : NSObject
@pro                                    tionContextDelegate> *delegate;

/**
 * R
 */
- (n                                    PreferencesObject;

/**
 * C                                    waypoints.
 * @
 * @                                    chronously with the route results.
 */
- (v                                    l NSArray<LandmarkObject *> *)array completionHandler:(nonnull void(^)(NSArray<RouteObject *> *array))handler

/**
 * C                                    waypoints.
 * @
 * @                                    e asynchronously with operation status.
 * @                                    ecute asynchronously with the route results.
 */
- (void)calculateRouteWithWaypoints:(nonnull NSArray<LandmarkObject *> *)array statusHandler:(nonnull void(^)(RouteStatus))statusHandler
    completionHandler:(nonnull void(^)(NSArray<RouteObject *> *array))completionHandler;

/**
 * Calculate a route between the specified waypoints and data buffer.
 * @param startArray The list of start waypoints.
 * @param data The data waypoints.
 * @param endArray The list of end waypoints.
 * @param handler The block to execute asynchronously with the route results.
 */
- (void)calculateRouteWithStartWaypoints:(nonnull NSArray<LandmarkObject *> *)startArray buffer:(nonnull NSData*)data endWaypoints:(nonnull
    NSArray<LandmarkObject *> *)endArray completionHandler:(nonnull void(^)(NSArray<RouteObject *> *array))handler;
```

Navigation

To check our initial need to get the navigation functionality based on a predefined route a small route based on 5 GPS points was hardcoded. To get a route for the GPS points from the General Magic SDK we need to create a `NavigationContext` which is able to create us a route:

```
navigationContext.calculateRoute(withWaypoints: waypoints, completionHandler: { (results: [RouteObject]) in
    NSLog("Found %d routes.", results.count)

     for route in results {
        if let timeDuration = route.getTimeDistance() {
            let time = timeDuration.getTotalTimeFormatted() + timeDuration.getTotalTimeUnitFormatted()
            let distance = timeDuration.getTotalDistanceFormatted() + timeDuration.getTotalDistanceUnitFormatte

             NSLog("route time:%@, distance:%@", time, distance)
        }
     }
    if let route = results.first {
        self.mapViewController.showRoutes([route], withTraffic: nil, showSummary: false)
    }
})
```

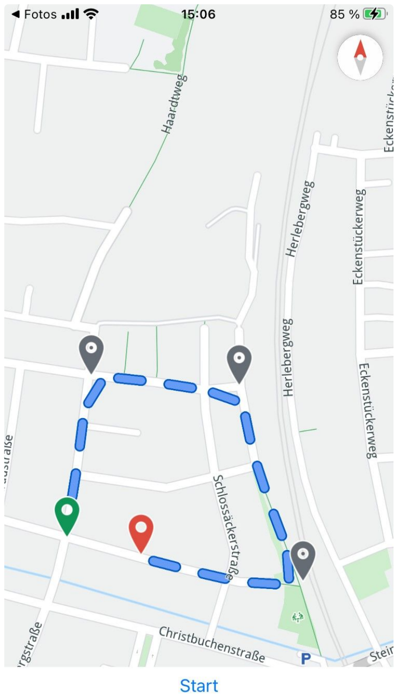
For our example the function is returning exactly on route with a route time of 2 minutes and a length of 700m. To display the route we need a `MapViewController` where we can put the route into.

```
mapViewController = .init()
mapViewController.hideCompass()

DispatchQueue.main.asyncAfter(deadline: .now() + 1) {
    self.mapViewController.startRender()
    self.mapViewController.startFollowingPosition(withAnimationDuration: 1000, zoomLevel: -1, viewAngle: 0) { _ i
 }
```

We need to wait a short time after creating the controller before being able to start the rendering and seeing the map. If we do not do this, the map is not shown at all.

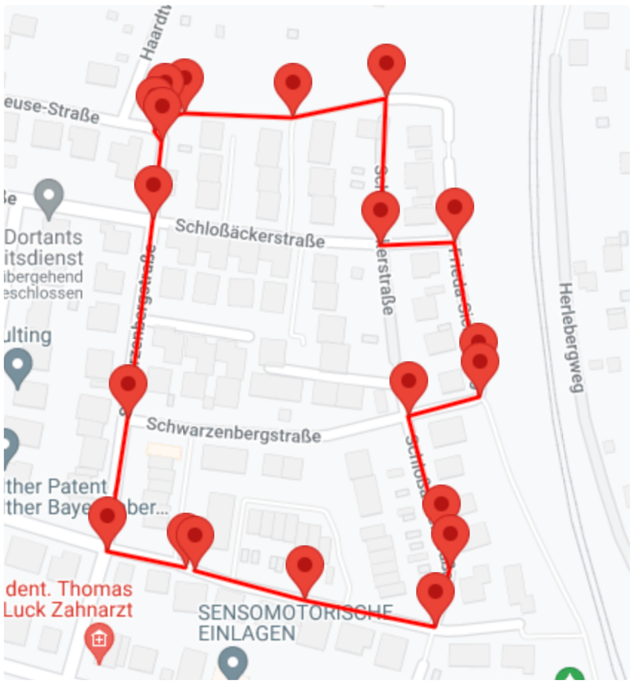
The result of creating the `MapViewController` and putting the route into it is this:



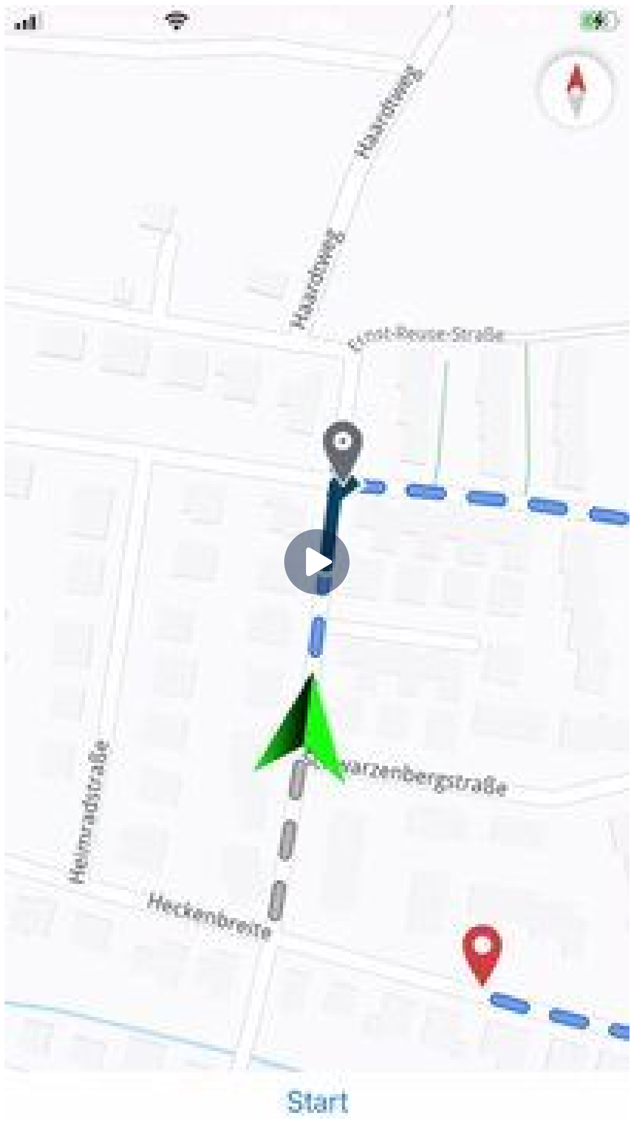
You can see that a route based on the GPS is created and visible. We are now able to start a navigation based on the navigation context:

```
navigationContext.navigate(withRoute: route)
```

Starting at that point we are getting updates about the route progress from the `NavigationContextDelegate` functions like `func navigationContext(_ navigationContext: NavigationContext, navigationInstructionUpdatedForRoute route: RouteObject)`. To simulate a ride we are driving a predefined GPX route which will not exactly follow the route:



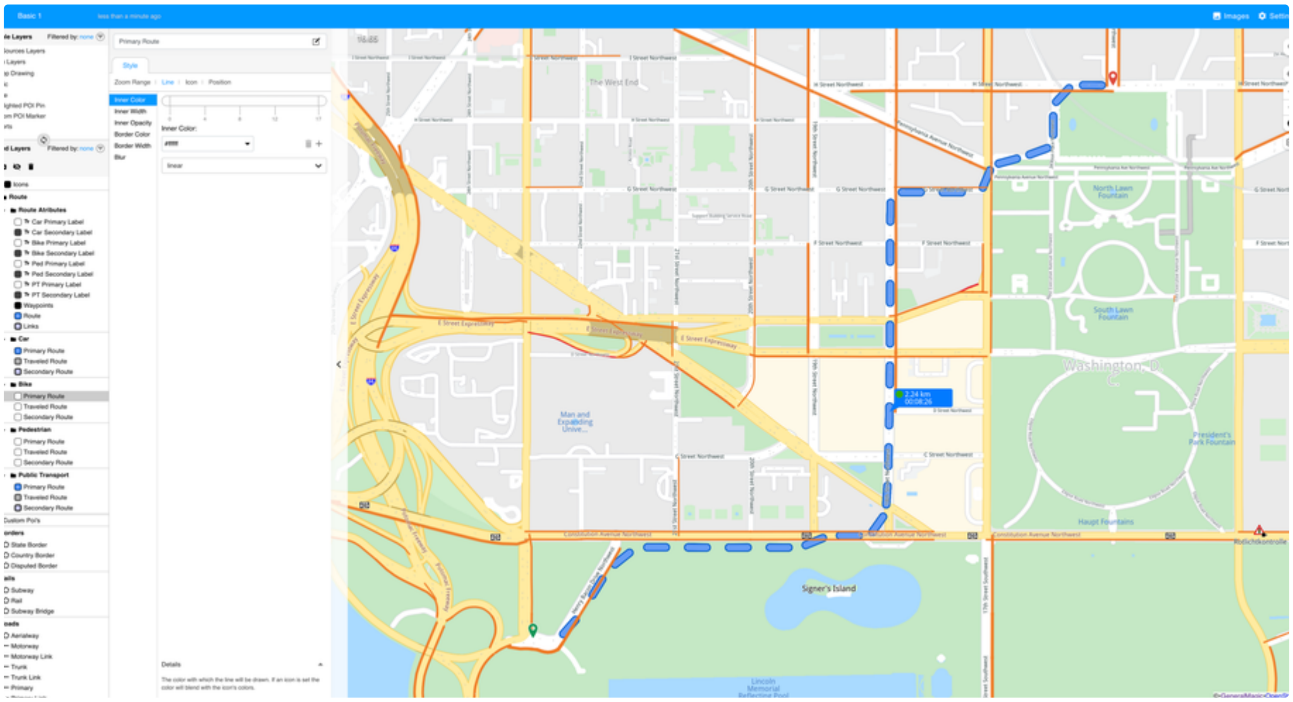
In the following video you can see what is happening in the example app:



When following the original route the already driven part of the route is marked. When leaving the route the SDK seems to get problems. The location puck is rotating around and is not displaying for a few moments the correct position of the user. Afterwards the back routing worked in the first scenario. When leaving the route the second time the map is still showing the user on the original route and jumping fast afterwards again to the correct position where some weird backrouting is happening although the user is already back on the original route.

Map style

General Magic is providing a “Studio“ to customize map styles. The functionality is similar to the one from Mapbox. But General Magic is not providing a default outdoor map style. This we would have to create ourselves. But there seems (at least not intuitively) no possibility to change the dash style of a bike route. Furthermore no functionality to customize the user location indicator was found (neither in the SDK nor in the studio).



Conclusion

The POC shows, that it is basically possible to use General Magic for a bike navigation. But the usage is not as comfortable as it should be. Furthermore there are big doubts about the quality of the SDK and the service (see the rerouting issue shown in the video). We would have to invest much more time to get a complete picture about the whole functionality of the SDK and their pros and cons. From the first insights it is not very convincing.

More to read

You can find the code of the POC in this repository: GitHub - brose-ebike/ios-generalmagic-poc: Example repository for the usage of the General Magic iOS SDK. The API token can be found in 1Password.