Tourio Readme

Notes

Wear interfaces look different on Android Studio wear emulator than on actual Moto 360. This is because Moto 360's resolution is 320 x 290 whereas Android Studio does not provide a 320 x 290 emulator. The layouts should look how they are on the Moto 360.

When downloading from Github, be sure to be on branch "Prud", not "master"!

The first time you trigger either the notification or the wearable app it may take 20-30 seconds to show up as it needs to transfer the image assets to the wearable (in practice this delay won't be noticeable to users as they will only be notified once the data has been sent).

Our wear app users SharedPreferences, to allow users to exit the Transit Activity wear screen to wear gps navigations and be able to go back to the wear app with the same data. The wear app resets the data to open the Start Activity when the user completes a tour and decides to reopen the wear app to do another. However, testing the app on the emulators can be different since it is not the same situation (e.g. testing certain parts but not necessarily finishing a tour during testing). If this is the case in which you are testing without fully completing a tour and pressing the "exit" button which resets the preference, you will need to (i) uncomment line 21 of DispatcherActivity.java under wear so that it will run the next time the program is started, (ii) run the app once so that it can reset the preferences, and (iii) comment out line 21 so that it doesn't always reset the preferences during normal app use.

Pre-requisites

- Android Mobile SDK v16
- Android Wear SDK v20

Getting Started

Download the files from github, or clone the repo onto a directory on your computer.

Compile and install the mobile app onto your mobile device or emulator.
Compile and install the wearable app onto your wear device or emulator.

You can find instructions to set up your emulators and instructions to set up your devices below.

Because our app involves navigation which requires simulation when testing, we have included a step-by-step example of how to do a tour on our app on the emulators.

Running on the Emulator

Setting up Emulators

Use Genymotion for mobile emulator and Android Studio for wear emulator. The wear app was designed for Moto 360 so a Android Wear Round Chin emulator should be used.

Setting up Genymotion and connecting it to the Wear Emulator:
(You only have to do the following once) Download:

- Google Apps APK's. Follow the instructions from the following link: (http://stackoverflow.com/questions/17831990/how-do-you-install-google-frameworks-play-accounts-etc-on-a-genymotion-virt)
- Android Wear APK. (http://www.apkmirror.com/apk/google-inc/android-wear/android-wear-1-1-1-1867902-android-apk-download/)
    - Drag and drop the Android Wear APK onto the home screen of your Genymotion emulator as well. When asked to Flash, Flash the zip file. Restart the emulator

Go to the directory where you have **adb.exe**
(Windows) Users/Your_Name/AppData/Local/Android/sdk/platform-tools
(Mac) Users/Your_Name/Library/Android/sdk/platform-tools
Open up terminal and input the following lines:

- adb devices
- adb -s (id for phone, 192.168.57.102:5555) -d forward tcp:5601 tcp:5601

You can check that devices successfully connected by opening Android Wear app on the mobile emulator and checking that it successfully connected to the wear emulator (you should see that the emulator's status is "Connected")

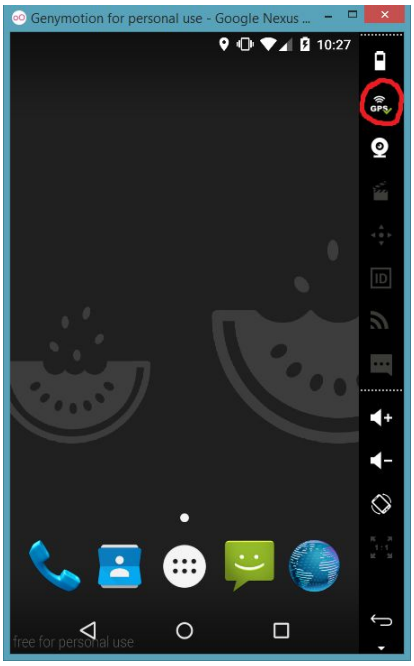
Getting Google Developer's API key

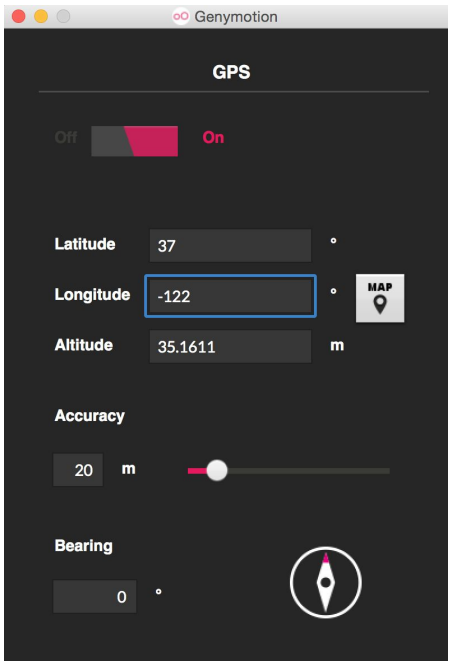
- Follow instructions on https://developers.google.com/console/help/new/ under section "API keys" to obtain own Google Developer's API key (used for accessing data owned by a Google service such as Google Maps or Google Translate)
- When adding a credential, select "API key" and then select "Android key"
- Follow instructions on website under "Android keys" section of "Public API access"
    - Package name for our project should be com.tourio.eklrew.tourio" (we spelled elkrew incorrectly)
- Once you have your API key, we need to replace the existing ones in the mobile and wear manifests so that the app is allowed to use Google Maps under your computer.
    - For both mobile and wear manifests, search for "API_KEY" and replace the value with your own API key.

```
android:label="@string/app_name"
android:theme="@style/Theme.Example" >
<uses-library android:name="com.google.android.maps" />

<activity
    android:name=".TourListActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".DetailTourActivity"
    android:label="@string/title_activity_detail_tour" >
</activity>

<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyB6ZX82AfxgQfqbT-sJ_2X0AqRIwcF5FcA" />

<activity
    android:name=".NavigationBarActivity"
    android:label="@string/title_activity_navigation_bar" >
</activity>
<activity
    android:name=".MyProfileActivity"
    android:label="@string/title_activity_my_profile" >
</activity>
<activity
    android:name=".CreateTourActivity"
```

```
    android:label="@string/app_name" >
</activity>
<activity
    android:name=".ThankYouActivity"
    android:label="@string/app_name" >
</activity>
<activity
    android:name=".NearMapActivity"
    android:label="@string/app_name" >
</activity>

<service
    android:name=".NotifyPhoneService"
    android:label="@string/app_name" >
</service>

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyCNUZiWpEt2AMQY3iOAzCicZKlLL1&JyBg" />
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />

<service
    android:name=".SkipListenerService"
    android:enabled="true"
    android:exported="true" >
</service>

<activity
    android:name=".TourDoneActivity"
    android:label="TourDoneActivity" >
```

Turn GPS on for mobile Genymotion emulator





Put 37 for latitude and -122 for longitude (let's say you are standing at that location). Don't worry about the altitude. Change accuracy to 20 m. In the example to follow, we'll be changing the latitude and longitude as we arrive at different stops.

Running on Real Devices

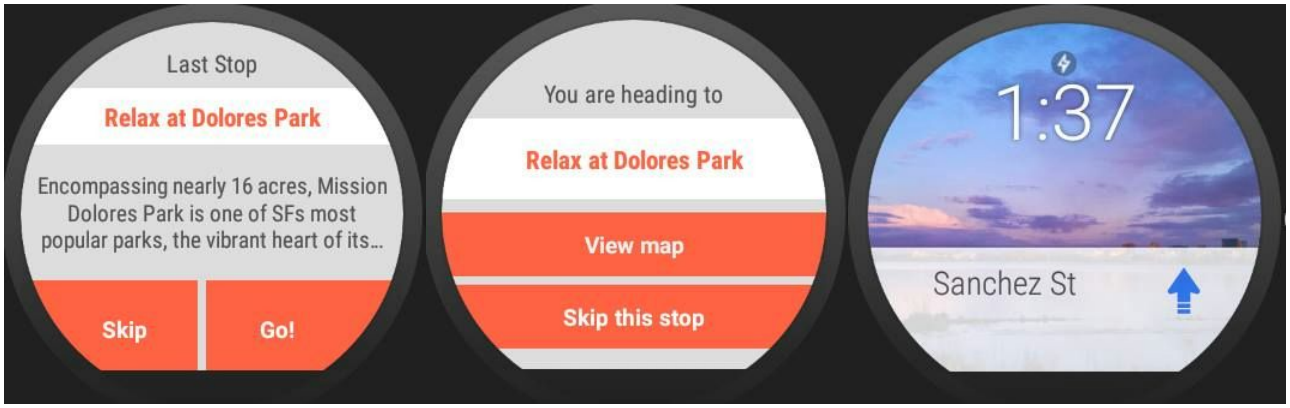
Bluetooth Debugging for Mobile and Wear
This allows you to run the app on a real smartwatch and a real smartphone. Download the app on your android phone from Android Studio. Make sure you have Android Wear on your smartphone. Refer to the Google Documentation for Debugging over Bluetooth.
Link: https://developer.android.com/intl/ja/training/wearables/apps/bt-debugging.html
The main points are:

- Make sure your phone and watch are connected via bluetooth.
- Go to your command line and input the following
    - `adb forward tcp:4444 localabstract:/adb-hub`
    - `adb connect localhost:4444`
- The phone should be connected to the watch. In the Android Wear companion app, you should see the status change to
    - `Host: connected`
    - `Target: connected`

Example of doing a tour on the emulator (step-by-step)

1. Run Tourio (phone app) in Genymotion.
2. Run Tourio (wear app) in the Android Studio wear emulator.
3. Pick a tour in the home screen. (Let's use Scenic Views and Picnic Foods for our example). When you are in the tour's detail page, press START TOUR. You should now have a page telling you that "Your tour is in session! You are free to put your phone away." At this point, the phone is sending data over to the wear.
4. Go to the wear emulator. Open up the app if it did not automatically open up when you ran the code on Android Studio. Press START.
5. You should see the first stop being Brunch at Kitchen Story along with its description. Press anywhere on the stop title or description to expand and read more about the stop (swipe from left to right to go back to the same page). Let's say you want to go to this stop. Press Go! [Remark: from here we'll be navigating with Google Maps, so please make sure you've done everything correctly with the Google Developer's API Key)
6. Press View map. This button will open up the navigation in the phone app. It also brings you to the home screen and you will see navigation notifications.

7. Let's say you've navigated your way to the stop. Go to the phone emulator and input the following (Latitude, Longitude) = (37.764176, -122.430668)[1]. This is the coordinates of the Brunch at Kitchen Story.



8. Once you arrive at the stop, the screen should automatically change to the stop arrival activity as shown below. Press "I'm ready for next stop!"
9. This should lead you to the page below with the Chinese Historical Society of America Museum. Press on the stop title or description to read more of the ellipsized texts.
10. Swipe right to go back to the previous page. Now you've decided to skip this stop. Press Skip. Then, confirm skipping.
11. Now the next stop is Relax at Dolores Park. You pressed on the stop title to read more about the stop and you've decided to go to the stop. Swipe right. Then do the same thing you did to go to Brunch at Kitchen Story. The coordinates for this stop are (Latitude, Longitude) = (37.759773, -122.427063).

12. The app recognizes that you've arrived at Relax at Dolores Park. It also recognizes that this is the last stop in the tour. Press Click to rate!

[1] In the Demo we included an Arrive button to automatically do this for us. In the final deliverable, we remove the button.

13. Now you can rate the tour maker. Once you've decided on the rating, press Next Time! Then you should see a screen that allows you to close the app. You're done!