TrackMaster Documentation

This is a project assignment for the *Mobile Development* module.
The application was made by **Mathieu GUÉRIN** (3026954).

Open-source repository: github.com/matiboux/griffith-android-trackmaster (public since after the deadline: May 11, 2020).
Play Store page: TrackMaster.

Table of contents

- The application
- The User Interface
  - Tracking Activity (Tracking)
  - List of Results (ListResults)
  - Results of an entry (Results)
  - About page (About)
- Code documentation
  - Activity Classes
  - Adapters
  - Custom View
  - Structures
  - Helper Classes

The application

The application meets the subject specifications.
It has activities and features for:

- Recording GPS points, and saving those in a `.gpx` file in the external storage.
- Having at least two activities, one being triggered when the recording is stopped.
- Generating statistics on GPS data points: average speed, distance, etc…
- Displaying a graph in a custom view showing the evolution of the speed.

The subject specifications require implementing one additional feature and I've chosen to add a third activity **listing previously recorded data** to open them back.

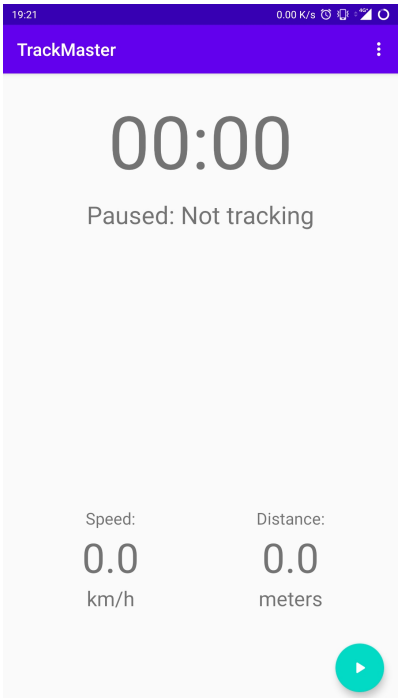
Additionally, based on a suggestion by the professor, the latest speed and the total distance are shown and updated live on the main activity while recording.

Finally, when viewing the results, the user can choose to delete the entry.

The User Interface

When opened, the application shows the activity for tracking and recording.

Tracking Activity (Tracking)



This activity shows a timer, the recording status, the latest recorded speed, and the total recorded distance.
A floating button at the bottom allows the user to start or stop the recording.

When the recording is started, the status changes from “Paused” to “Tracking”, the timer is updated every second, and a new information appears: the number of recorded GPS points. The speed and distance are updated every time a new GPS point is recorded.

The toolbar of this activity has a menu with two items:

- One takes the user to the list of previously recorded data.
- The other takes them to the *About* page.

List of Results (ListResults)

The list of results show a list of all the saved files in the external directory where the application saves its data: `EXTERNAL_DIR + "GPStracks/"`.

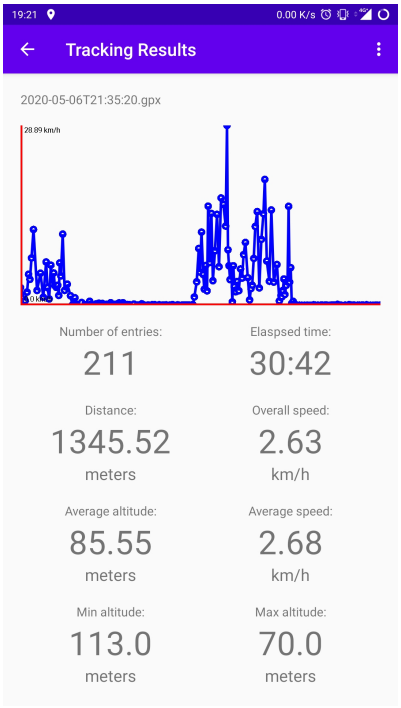
When clicking on a item in the list, statistics and information on the entry will be shown to the user.

Unless the user messes with the files, all of them should be `.gpx` files and should be opened without issues. Otherwise, the app should open results for an empty data set of GPS points.

The toolbar includes a “back” icon to go back to the previous activity.

Additionally, a floating button allows the user to go to the *Tracking* Activity (which usually is the previous activity).

Results of an entry (Results)



This activity shows the name of the entry, a custom view for displaying the graph of speeds, and statistics calculated from the GPS points.

The toolbar includes a “back” icon to go back to the previous activity, as well as a menu with an item for deleting the entry. The action of deleting the contact from the menu will open a confirmation prompt (using an `AlertDialog`). The action can then be confirmed or canceled.

About page (About)

The *About* page is just a simple page, to name the developer, me, and to specify the nature of this project: an assignment at Griffith College, for the *Mobile Development* module.

Code documentation

Activity Classes

Tracking

- `protected void onCreate(Bundle savedInstanceState);`
- `public void onWindowFocusChanged(boolean hasFocus);`

  Updates the bottom padding of `LinearLayout` to give space to the floating button.

- `private void enableTracking();`

  Enables the GPS tracking and the clock.

- `private void enableClock();`

  Called from `enableTracking()`.
  Creates a `Handler` which will run regularly based on `Contants.DELAY_CLOCK` and will update the clock, the number of saved GPS points, the speed, and the distance.

- `public boolean onCreateOptionsMenu(Menu menu);`
- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: opening the *ListResults* or *About* activity.

- `public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults);`

  Handles result from requesting permissions to the user: for GPS location and external storage. Enables the tracking when permissions are granted, as they are requested when the user tries to enables the tracking.

ListResults

- `protected void onCreate(Bundle savedInstanceState);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity.

- `protected void onActivityResult(int requestCode, int resultCode, Intent data);`

  Handles results from other activities, triggers `reloadData()` if necessary.

- `private void reloadData()`

  Reloads the list of files (recorded data).

Results

- `protected void onCreate(Bundle savedInstanceState);`

- `public boolean onCreateOptionsMenu(Menu menu);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity, deleting the file.

About

- `protected void onCreate(Bundle savedInstanceState);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity.

Adapters

ListResultsAdapter

- `public ListContactsAdapter(Context context, List<String> objects);`

  Constructor of the class.
  It defines `objects`, the list of elements that will be in the `ListView`.

- `public View getView(int position, View convertView, ViewGroup parent);`

  Method that will construct each row of the `ListView`.
  Uses `ViewHolder`.

- ```
  private static class ViewHolder {
      TextView txv_filename;
  }
  ```

  Structure element for a single row in the `ListView`.

Custom View

SpeedGraphView

Custom view for the graph of speeds.

- `SpeedGraphView(Context c);`

- `SpeedGraphView(Context c, AttributeSet as);`

- `SpeedGraphView(Context c, AttributeSet as, int default_style);`

- `public void init();`

  Initializer of the view.

- `protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec);`

  Computes the height for the view to maintain a 1:2 ratio.

- `protected void onSizeChanged(int width, int height, int oldWidth, int oldHeight);`

  Invalidates the view on size changed.

- `public void onDraw(Canvas canvas);`

  Draws the graph.
  ```

- `public void setData(List<Pair<Long, Double>> averageSpeeds);`

  Sets the data for drawing the graph.

Structures

Constants

- ```
  public static final int REQUEST_PERMISSIONS = 10;
  public static final int REQUEST_DATA_RELOAD = 20;
  ```

  For requests and intents results.

- ```
  public static final int DELAY_TRACKING = 5000; // 5 sec
  public static final int DELAY_CLOCK = 1000; // 1 sec
  ```

  Delays for GPS tracking and for clock updates.

- `public static final String DIRNAME = "GPStracks/";`

  Directory name in the external storage where the `.gpx` will be saved.

GPXEntry

Structure class for an entry in a GPX file: one GPS point.

- ```
  public final double latitude;
  public final double longitude;
  public final double altitude;
  public final long time;
  ```

- ```
  public GPXEntry(double latitude, double longitude,
      double altitude, long time);
  ```

  Constructor of the class.

Helper Classes

MyLocationListener

Event class which implements the `LocationListener` interface.

- `public final SQLiteDatabase sdb;`
- `public MyLocationListener(Context context);`

  Constructor of the class. It saves the `context`.
- `public void enableTracking();`

  Checks for permissions and enables tracking.
- `public void disableTracking();`

  Disables tracking.
- `private void requestPermissions();`

  Requests for permissions.
- `public void addEntry(Location location);`

  Adds an entry to the saved data.
- `public boolean isTracking();`

  Returns true if we're currently tracking the GPS.
- `public long getRunningTimeMillis();`

  Returns the running time in milliseconds.
- `public GPXFile getGpxFile();`

  Returns the GPXFile object.
- `public int getSavedEntries();`

  Returns the number of saved entries.
- `public double getLatestSpeed();`

  Returns the latest speed (between the latest two saved entries) for live data.

- `public double getTotalMeters();`

  Returns the total distance in meters for live data.

- **`LocationListener` interface**

  - `public void onLocationChanged(Location location);`

    Adds a new entry.

  - `public void onProviderEnabled(String provider);`

    Checks for permissions and adds the last known location as an entry.

  - `public void onProviderDisabled(String provider);`

    Displays an error message.

GPXFile

This class is used to manage a GPX file.

- `public GPXFile(File file);`

  Constructor of the class.

- `public String getName();`

  Returns the name of the file.
  Recommended format example: `2020-05-10T23:59:59.gpx`

- `public void createFile();`

  Creates the parent directories and the file itself with an empty GPX file template.

- `public void addEntry(Location location);`

  Adds an entry to the file and saves it.

- `public GPXData getData();`

  Reads the file and returns a GPXData object.

- **Static methods**
    - `public static List<String> getFileList(File dir);`

      Returns the list of files in a directory.
- **Constants: Templates for writing to the file**
- `private final String NEW_ENTRIES_ANCHOR;`

  Anchor where the script will add the next new entries.

- `private final String NEW_FILE_CONTENT;`

  File structure for an GPX file without entries.
  Contains one *NEW_ENTRIES_ANCHOR*.

- `private final String ENTRY_TEMPLATE;`

  Template for new entries to be added before the anchor.

GPXData

This class is used to compute statistics on GPX data.

- `public GPXData(List<GPXEntry> gpxEntries);`

  Constructor of the class.

- `private void processData();`

  Called from `GPXData()`.
  Computes all the statistics that will be displayed in the application.

- `public List<GPXEntry> getEntries();`

  Returns the list of entries in the GPX file as `GPXEntry` objects.

- `public int getSize();`

  Returns the numbers of entries in the dataset.

- `public long getElapsedSeconds();`

  Returns the recorded time in seconds.

- `public double getTotalMeters();`

  Returns the total distance in meters.

- `public double getOverallSpeed();`

  Returns the overall calculated speed from the total distance and time.

- `public List<Pair<Long, Double>> getSpeedsList();`

  Returns the list of speed between each GPS point.

- `public double getAverageSpeed();`

  Returns the average speed between all recorded speeds.

- `public double getMinAltitude();`

  Returns the minimum altitude.

- `public double getMaxAltitude();`

  Returns the maximum altitude.

- `public double getAverageAltitude();`

  Returns the average altitude.

- **Static methods**
    - `public static double computeDistance(GPXEntry A, GPXEntry B);`

      Returns the distance in meters between `A` and `B`.

    - `public static double computeSpeed(GPXEntry A, GPXEntry B, double distance);`

      Returns the speed in meters per seconds between `A` and `B` for the given `distance` between the two points.

    - `public static double roundDecimals(double value, int nbDecimals);`

      Rounds `value` at `nbDecimals` decimals.