UNIVERSITÉ DE GENÈVE

Bachelor of Arts Thesis

SkyFreeGPS - a GPS Receiver Simulator

University of Geneva

Faculty of Economics and Social Sciences

Information Systems and Communication Department

Professor : Dimitri Konstantas

Supervisor: Michel Deriaz

Author : Alina Burca

Table of content

1. Introduction

Today we are witnessing an expansion of the GPS world in our daily life. We are now beginning to integrate more and more GPS receivers in different kinds of gadgets like mobile telephones, watches, PDAs, compact cameras, etc. In case this term is not familiar to you, a GPS receiver is a device that determines, among other things, the exact position in latitude and longitude coordinates (with an accuracy of about 15 meters), the speed, the altitude etc. The GPS idea started, like many other great things (e.g. Internet), as an army project, but ended up by becoming open to civilian use. Therefore, the GPS applicability fields are expanding rapidly and, in the near future, we expect to find more and more GPS oriented applications available, whether they will be freeware or not.

If you use an application that requires GPS data on a mobile phone, a GPS receiver is necessary, whether it is incorporated in the phone or not. This also applies to software programmers who are developing a GPS based application. At some point they need to test how their application behaves given the data it gets from the GPS receiver. However, with a GPS receiver hardware, all tests must be performed outside, with good satellite visibility, leading to a tedious and time consuming process: installing the application on a mobile telephone, getting a valid fix for the GPS receiver and connecting it to the telephone via Bluetooth (if it is an external GPS receiver). All these preparations are not always evident, especially in the first phases of development when one prefers to test the application directly on the development support - the PC. This task proves to be even more difficult if the test demands a certain speed of movement or specific places to be in. All this could be easily avoided by using a *Bluetooth GPS simulator*, allowing it to directly test the application on the PC without having to step outside. That means time and energy saving.

This is where SkyFreeGPS comes in, as a handy tool to be employed. SkyFreeGPS is a GPS receiver simulator. In others words, it will generate GPS data in the same format used by an actual GPS receiver hardware, only without really being where it says it is. The user of the application has control over what kind of data he or she would like to get from the GPS receiver, based on his/hers specific needs.

Let's take the example of a software programmer that would like to develop a GPS based application that works on mobile phones. More precisely, this application will have to register the position of radars and alert users when approaching these radars, by the use of a mobile phone and, of course, of a GPS receiver. Therefore, this application will need GPS data for two of its features: for tagging the radars and for locating already defined radars. Tagging is the operation of pinning a content (data, image etc.) to a specific location.

This is how SkyFreeGPS would change the life of this developer: let’s say that he is testing the first feature, that of registering a tag at the exact location where the user told the application he has encountered a radar. Without the simulator, the developer would have to go outside, where the GPS receiver can get satellites signals and transmit the location to the application. It would be the actual location where the developer is conducting the test, and maybe not the location he would want to test. Instead, he can simply specify any location using SkyFreeGPS. Either he knows the latitude and longitude values of this location and enters them manually, or he chooses the location on a map with the use of a pointer.

Now let’s say that the developer wants to see if the application will alert the user when passing near by the radar he has just defined. Instead of getting in the car and driving along radars, risking a speed ticket or, even worse, an accident because he was watching the mobile’s screen and not the road, he can simulate the situation using SkyFreeGPS. All he needs to do is to define on the map a location just before the radar and one just after it, giving also the desired travel speed between the two points. Now he can let SkyFreeGPS do the computations and start transmitting GPS data while he can observe how his application reacts: does it alert of the radar in time or not, does it ignores the radar totally etc.?

What if the developer has already defined some radars using SkyFreeGPS simulator and would like to keep their location for future tests? Under SkyFreeGPS he would just have to mark them on the map as points of interest and save them all in a file related to that map. For more precision SkyFreeGPS can display the coordinates of the pointer, so that our developer can make sure the radar is marked exactly where he wants it to be.

To continue with our example, let’s imagine that the developer has created a chain of GPS data using SkyFreeGPS and has observed a failure in his application. He will then have to fix the problem and reproduce the same context in order to test if really he has resolved it or not. Knowing this, he can already save the GPS data in a log file that he could use later to reproduce the same context.

To summarize, this is what our developer could do with SkyFreeGPS:

- generate GPS data in real time;
- use a map and the pointer for a more user friendly definition of GPS data;
- define and save points of interest (POIs);
- define points on the map and let the application compute the intermediary locations based on the speed between these points;
- display the Latitude and Longitude coordinates on the map;
- save to log files GPS data he generates, for future use.

Being a virtual GPS receiver, SkyFreeGPS doesn't require clear sky conditions to operate, hence the name of the application makes reference to the words *sky free*.

The original idea for this application came from Michel Deriaz, assistant at the *Information System and Commutation Department* at the University of Geneva. It is the first open source application of this kind developed in Java 2 Platform, Micro Edition (J2ME) to be using bitmaps and the stylus. The application is designed as a useful tool to be employed within the GPS based developers community for mobile phones.

2. Overview

2.1. The GPS world

2.1.1 General notions reminder

The *Global Navigation Satellite Systems or GNSS* uses satellite positioning techniques to provide users with accurate and timely navigation information. GNNS serves as an essential source of exact positions and velocities, but also as a source of precise time.

Some examples of current GNSS systems:

- NAVSTAR in the United States (the most developed, most complete and most operational one), also known as GPS being generally the first one available,
- Galileo in Europe,
- GLONASS in Russia.

The *Global Positioning System* or *GPS* is a subset of a global navigation satellite systems. Developed and managed by the American government, the GPS system comprises of a constellation of more than 24 satellites (MEO) distributed in such a manner that at least six satellites should be visible to a GPS receiver situated almost anywhere on the surface of the Earth.

The GPS satellites system would be useless to us without a *GPS* receiver. This is a special radio receiver designed to calculate the location based on the satellite signal transmissions. GPS receivers don't have to transmit anything to the satellites and are not visible to the them, which makes it unlimited for the number of GPS receivers to be using the system at any time.

2.1.2 GPS receivers in summary

Usually, GPS receivers consist of: an antenna set to the frequencies transmitted by the satellites and a central unit and of a highly stable clock (often a crystal oscillator based). They can also include a screen to display the information on the position and speed.

A receiver is often described by his number of channels, which means the number of satellites it can see simultaneously (typically between twelve and twenty channels).

Today's GPS receivers present themselves in a variety of forms: from car, telephone or the watch integrated devices to navigation dedicated devices. The receivers can communicate with other devices through several ways: serial connection, USB or Bluetooth.

GPS receiver communication is defined within the *National Marine Electronics Association - NMEA* specification. *NMEA* is a standard communication protocol with many types of sentences for all kinds of devices, not only GPS receivers. SkyFreeGPS uses two of them: GGA - *fix information* and RMC - *recommended minimum data for GPS*.

2.1.3 GGA and RMC NMEA sentences in summary

All NMEA standard sentences start with the letter '$', followed by the description of the sentence: two letters that describe the device using the sentence type, in our case GP standing for GPS receiver, followed by RMC or GGA.

NMEA $GPGGA sentence is comprised of essential fix data which provide 3D location and accuracy data. Below is an explanation of the sentence structure.

$GPGGA,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O<CR><LF>

Field	Description	Value example	Explanation
$GPGGA	NMEA String Identifier. Always the same value	$GPGGA	GPS data, GGA sentence
A	UTC of position: hours, minutes, seconds	083525.129	08:35:25 UTC
B	Latitude in degrees, minutes and thousands of minutes	5125.7397	51 deg 25.7397 '
C	North or South - N,S possible values	N	North
D	Longitude in degrees, minutes and thousands of minutes	00239.8817	2 deg 39.8817 '
E	East or West - E,W possible values	W	West

Field	Description	Value example	Explanation
F	GPS quality indicator, possible values: 0 = invalid 1 = GPS fix (SPS) 2 = DGPS fix 3 = PPS fix 4 = Real Time Kinematic 5 = Float RTK 6 = estimated (dead reckoning) (2.3 feature) 7 = Manual input mode 8 = Simulation mode	1	GPS fix (SPS)
G	Number of satellites in use	08	8 tracked satellites
H	Horizontal Dilution of Precision	01.1	HDOP
I	Antenna Altitude above mean sea level	50.7	
J	Unit of measure of field I	M	Meters
K	Geoidal separation = difference between the WGS-84 Earth ellipsoid and mean sea level	51.6	
L	Unit of measure of field K	M	Meters
M	Age of differential GPS data. Time in seconds since the last type 1 or 9 update	Empty field	-
N	Differential Reference Station ID number	Empty field	-
O	Checksum. Always starts with '*' followed by two hex digits	*42	
<CR><LF>	Carriage return, line feed characters		

Table 1. **NMEA standard GGA sentence structure**

All fields are separated by a comma. By combining the values given above as an example we would obtain a GGA sentence looking like this:

```
$GPGGA,083525.129,5125.7397,N,00239.8817,W,1,08,01.1,50.7,M,51.6,M,,*42
```

The checksum at the end represents an 8 bit exclusive OR of all the characters between the '$' and the '*'.

NMEA $GPRMC sentences repeat some of the information above but also add some new data. RMC comprises essential GPS position, velocity and time data. For more detailed information see the following structure.

$GPRMC,A,B,C,D,E,F,G,H,I,J,K,L<CR><LF>

Field	Description	Value example	Explanation
$GPRMC	NMEA String Identifier. Always the same value	$GPRMC	GPS data, RMC sentence
A	UTC of position: hours, minutes, seconds	083525.129	08:35:25 UTC
B	Status: A = Active,V =Void	A	active
C	Latitude in degrees, minutes and thousands of minutes	5125.7397	51 deg 25.7397 '
D	North or South - N,S possible values	N	North
E	Longitude in degrees, minutes and thousands of minutes	00239.8817	2 deg 39.8817 '
F	East or West - E,W possible values	W	West
G	Speed over the ground in knots	010.3	10.3 knots
H	Track angle in degrees	142.10	142.10º
I	Date	140707	14 June 2007
J	Magnetic Variation	Empty field	
K	East or West - E,W possible values	W	West
L	Checksum. Always begins with '*' followed by two hex digits	*6F	
<CR><LF>	Carriage return, line feed characters		

Table 2. **NMEA standard RMC sentence structure**

All the values from the above table form a RMC sentence like the following:

```
$GPRMC,083525.129,A,5125.7397,N,00239.8817,W,0.00,142.10,140705,,W,*6F
```

2.2. State of the art

Currently there are a lot of applications allowing to test GPS receivers, and also some products allowing to simulate a virtual GPS receiver. Some examples are stated below. However none of these applications are being developed in J2ME, thus not running under Sun Java Wireless Toolkit Emulator, and more importantly they are not freeware. The only similar J2ME applications found are very basic and do not include all these features.

Skylab GPS Simulator - allows replaying GPS log files, multiplexing GPS receivers, transforming the interface of the GPS receiver. Furthermore the Skylab GPS Simulator contains a full featured interoperable OGC WMS client for receiving maps from a standardized WMS server. Within a map the user can navigate (zoom, pan) and select his favourite layers provided by the WMS server. Skylab GPS Simulator generates GGA, GSA and RMC sentences.

Official web site: http://www.skylab-mobilesystems.com/en/products/gps_sim.html

Avangardo GPS generator std 2.2.4 – a GPS device emulator that generates NMEA sentences from different types of data: the user chooses a map, the starting point, a title and the speed; the user chooses a NMEA file. The generator can write the NMEA sentences to a file or send them to another software through a COM port. Generated NMEA sentences can be used by a mapping, navigation or other GPS enabled software. It comes in two versions: standard and professional.

Avangardo's official website: http://avangardo.com/gps/gpsgen/

Virtual GPS 1.33 – application which simulates a GPS receiver unit connected to the system. It supports various NMEA sentences, so it can be used with any GPS mapping software. VGPS operates in two modes: simulator mode and file mode. In simulator mode the latitude and longitude coordinates are periodically incremented with a defined step. In file mode the GPS data is loaded from a text file which contains NMEA 0183 sentences. GPS data is transmitted periodically to a serial port (UDP ports are also supported).

Official web site: http://www.zylsoft.com/vgps.htm

ZylGPSSimulator - a Delphi GPS receiver simulator component. It creates a virtual serial port, converts the position parameters in NMEA 0183 format and writes them to the virtual port. The user can set parameters like latitude, longitude, altitude, speed, heading and transmit them to the port.[*]

Official web site: http://www.zylsoft.com/gpssim.htm

After spending quite some time doing research, I was not able to find a freeware or an open source product developed in J2ME similar to SkyFreeGPS, with the following exceptions:

[*] All the informations about these applications are gathered from the indicated websites, I did not purchase these applications myself

- *GPSSim* (open source) - an application developed at the university of Geneva by Michel Deriaz for similar reasons: testing of a GPS based application - FoxyTag. I have integrated the GPSSim under the *Keyboard Mode* of the SyFreeGPS simulator.
- *BlueGPS* (open source) - a very basic J2ME application that transmits a series of less then twenty NMEA sentences specified in the code of the application.

  Source code at: http://www.digitalmobilemap.com/
- The *External Event Generator of the Wireless Toolkit emulator* (freeware) - a tool build up for testing reasons that can provide location, orientation and elevation data to a MIDlet running in the WTK emulator. The EEG can be opened from the *MIDlet* menu of the emulator's window, by choosing the option External Events. EEG acts like a telephone's internal GPS receiver, giving information about the device's position, as well as access to a databases of known landmarks stored on the device.

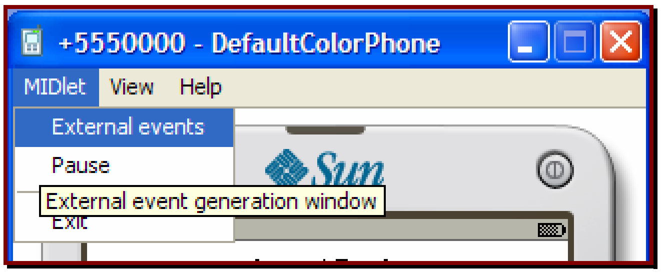

+5550000 - DefaultColorPhone
MIDlet
View
Help
External events
Pause
External event generation window
Exit
Sun

Figure 1. How to open the EEG of a MIDlet emulator

3. Application description

3.1. Overview

SkyFreeGPS is an open source software developed in Java 2 Platform, Micro Edition (J2ME) for simulating a Bluetooth GPS receiver hardware that generates two types of standard NMEA sentences. More precisely, the application allows to generate and manage RMC and GGA NMEA GPS sentences and send them through a Bluetooth connection[1]. The simulator can transmit fix or moving positions in real time, or positions in accordance with a predefined track.

SkyFreeGPS runs on the Sun Java Wireless Toolkit and is available in English.

3.1.1 Main features

The main features of the application include:

- generation of Bluetooth GPS streams by emitting GGA and RMC sentences at a frequency of one second;
- use of the keyboard or of a stylus to set up track points;
- visualisation of calibrated maps imported from files for browsing and setting up track points;
- definition of track points in real time or through computations based on a specified route;
- visualisation of defined track paths on the map;
- visualisation and management (setting up, exporting, importing) of Points of Interest (POIs);
- display of Latitude and Longitude coordinates of a specific location selected on a map;
- export of generated Bluetooth sentences to NMEA log file;
- track import from file.

[1] For more information about a Bluetooth GPS receiver device or about the NMEA standard, please refer to the corresponding paragraphs later in this document

3.1.2 Why are these features important?

Use of the keyboard or of a stylus to set up route points

A basic mode to designate track points is by pressing one specific key that corresponds to one of the following possible directions: up, down, left and right in order to indicate each next location. This use case is practical when the simulation starts from an initial location and the actual track is of less importance. However, in some cases, a track that takes place in specific conditions must be simulated. For these cases the use of a map can be mandatory and therefore the keyboard can no longer be sufficient. A pointer makes it much more easier to set up track points, making it faster and more user-friendly.

Map visualisation for browsing and setting up track points

As mentioned before, apart from being more intuitive, the use of a map is sometimes required. SkyFreeGPS displays external calibrated maps, by reading the information regarding the map from two files: one .png file containing the image of the map and one .txt file containing the calibration data. The map is then visualised on the telephone's screen an can be dragged with a stylus for browsing.

Generating GPS sentences through real time defined route points or by calculated track points and visualisation of the defined track paths on the map

In some menus, the user can decide in real time what GPS position should be transferred next. This option is fast and doesn't require much preparation except for an initial starting position or the loading of a map. If the user wants to simulate passing through a specific route with a different speed at different steps of the way it would be difficult to do it in real time. In this case he/she can establish a route by marking several waypoints on the map and their corresponding speed, starting from which the application will compute the series of sentences to be sent via Bluetooth. The track is displayed on the screen as the user points to each strategic position. Also the speed can be changed along the route for each waypoint.

Display of Latitude and Longitude coordinates of a specific point selected on a map

When importing a map the user can have a precise interest for a specific point on a map. To help locate this exact point SkyFreeGPS will display the latitude and longitude coordinates if the user clicks on the map while pressing down a key.

Set up of Points of Interest

Some testing settings involve using a location of specific interest to the test (for example the location of a radar or any other kind of POI). These points can be marked on the map and stored in a file for later use, spearing the user of the trouble of locating them each time on the map.

Export of generated Bluetooth sentences to file using NMEA format / Import of NMEA format files

The sequence of sentences sent by the GPS receiver simulator is stored in a log array and can be exported to a file. This may be very handy in case the user found the sequence of a certain importance and would like to use it again, either with SkyFreeGPS or with another application.

For SkyFreeGPS external use, the conversion of the GPS log files into .GPX format can be easily performed with the help of a lot of freeware SW tools available on the Internet[2].

To sum up, the application allows several practical modes for generating track points:

- in real time on a map by pointing out the current position – **Real Time Mode**;
- by calculating the points on the basis a predefined track by specifying the speed - **Track Mode** ;
- in real time simply by navigating up, down, left and right starting from an initial point - **Keyboard Mode**;
- by using a previously generated log file – **LogFile Mode**.

GPS sentences transfer mode	Real time versus non-real time computations	Use of a map	Use of the Keyboard	Use of the Stylus
Real Time Mode	Real time	YES	NO	YES
Track Mode	Non-real time	YES	NO	YES
Keyboard Mode	Real time	NO	YES	NO
LogFile Mode	Non-real time	NO	-	-

Table 3. **Characteristics of transfer modes**

[2] Some examples are given under the Useful Links paragraph

SkyFreeGPS can be used in the Sun's Wireless Toolkit (WTK) emulator. For more information about how to install and use the application, please refer to the *User Guide* paragraph.

3.2. User Guide

3.2.1 Application requirements

In order to use SkyFreeGPS please consider the following requirements:

- JAVA 2 Platform, Micro Edition - J2ME;
- Sun's Wireless Toolkit for CLDC with a Bluetooth and touch screen enabled device;
- a calibrated map[3].

SkyFreeGPS runs in Sun's JAVA Wireless Toolkit emulator requiring a Bluetooth and pointer supporting device. The application is intended to be used on a personal computer for testing a GPS based application during the development process, although it could also be used on a device that meats the specified requirements. In the course of this project the application has only been used in WTK as a pointer enabled device was not available.

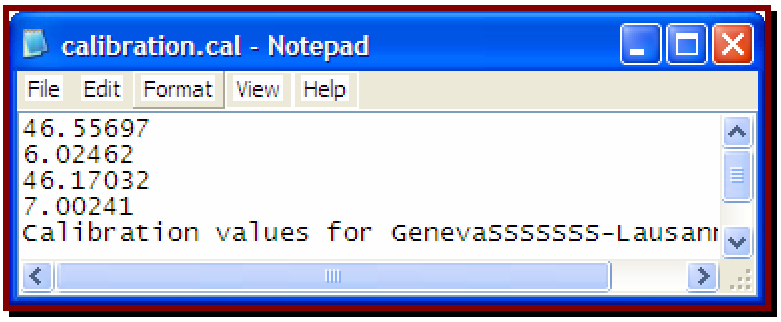
For the use of its map enabled features, SkyFreeGPS also requires a valid calibrated map in the form of two files: one containing the image of the map and one the calibration data.

The image should be stored in a .PNG file, whilst the calibration data must be stored as text under any file extension. SkyFreeGPS will read the first four lines of the file, assigning each line to the next following values in the given order:

- latitude for the most up left corner pixel of the image,
- longitude for the most up left corner pixel of the image,
- latitude for the most down right corner pixel of the image,
- longitude for the most down right corner pixel of the image.

Latitude and longitude should be specified in decimal degrees, as presented in the next figure.

[3] This could be optional in case the map enabled features of the application are not being used



```
46.55697
6.02462
46.17032
7.00241
Calibration values for GenevaSSSSSSS-Lausan
```

Figure 2. Example of map calibration data

There is no restriction to the content of the file following the first four lines, so feel free to add any useful information regarding the map. A file with less than four lines will not be accepted, nor a file in which the first four lines don't contain double type values. However trying to use a file that doesn't meet the previous requirements should not interrupt the application.

The data calibration file name doesn't have to be the same as the image file name. Thus you could have an image file named "*Switzerland.png*" and a corresponding calibration file named "*calibration.txt*".

3.2.2 Set up SkyFreeGPS for use

Using SkyFreeGPS with WTK

For running the application, a new project must be created in WTK using one of the two classic available options: *Create project from JAD, JAR file* or *New Project* under the *File* menu. The MIDlet of the application is the SkyFreeGPS file located in the folder with the same name.

API selection must meet at minimum JSR 82, JSR 75 and JSR 179, with MIDP 2.1 profile and CDLC 1.1 configuration.

Enabling pointer use on any WTK emulator's device

Not all devices of the WTK emulator support pointer driven applications. To be able to use SkyFreeGPS on any device the touch screen facility should be activated. In order to do this you must find the properties file of the selected device. It is usually located under the folder **WTK2.5.2\wtklib\devices**. Choose the folder with the device's name and open the *.properties* file using a basic text editor like *Notepad*. Locate the **touch_screen** parameter and change its value to true. The line should look like this:

touch_screen=true

Save the changes and close the file. The changes will take effect the next time WTK is launched. It is recommended not to change the parameters of to many devices as this could interfere with testing of other applications. Best to chose one device for running SkyFreeGPS and stick to it, preferably one of the default emulator's devices which don't correspond to any particular real device model.

Using maps

Maps must be accessible to the emulator's device. Therefore all files related to maps should be saved in the device's file system. This is usually located under the folder **j2mewtk\2.5.2\appdb\"devices_name"\filesystem\root1**. Once the folder located, it is recommended to create a shortcut on your desktop for faster access in the future, especially if planning to use the map modes and to add new maps all along.

Note that in order to be used in SkyFreeGPS, maps must be calibrated. Therefore, at all times, the two mentioned files are required: the one with the image and the one with the calibration data. There are plenty of tools permitting to calibrate a map. For further references please consult the useful links provided at the end of this document.

3.2.3 Using SkyFreeGPS

The main menu of the application contains the following options: **Select Map, Real Time Mode, Track Mode, Keyboard Mode, Log File Mode, Help and Exit.**

Before explaining in more details all of these menus, here are some specifications regarding the sentences build by SkyFreeGPS for a better understanding of how the application works.

Upon launching, the application will try to open a Bluetooth connection. As soon as a client is connected, SkyFreeGPS will start sending streams with a one per second frequency. These streams contain two lines: one for the GGA sentence and the second for the RMC sentence. These sentences can be empty or not, depending on the status of the application. Under the *main menu*, *select map menu* and *help menu* the application can only build empty sentences. Under the other menus SkyFreeGPS will either send blank or valid sentences based on the user's actions. This is explained in more detail in the following paragraphs regarding these menus.

A blank sentence contains only the current UTC time information and no details about location or velocity or altitude etc. Here is an example of an empty stream:

```
"$GPRMC,132715,V,,,,,,,10032008,003.1,W*65"
"$GPGGA,132715,,,0,00,,,,,,,*41"
```

3.2.3.1 Select Map

Use this option to select or change the map to be used in the map based modes (*Real Time* and *Keyboard*). Three files can be selected under this menu:

- the image of the map,
- the image calibration data,
- the POIs of the map.

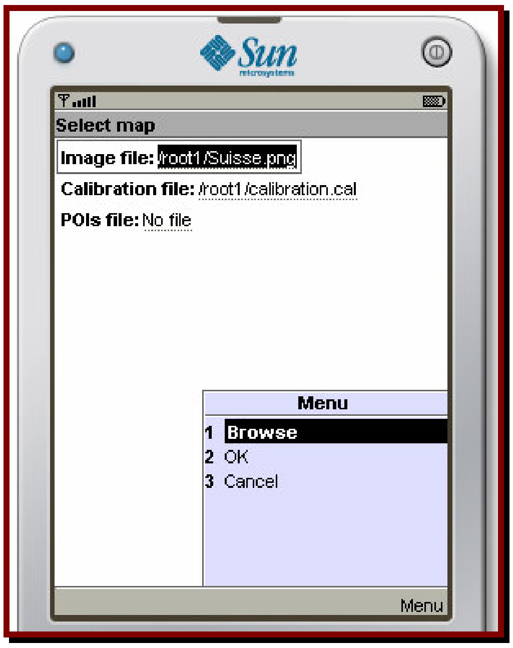

Select map

Image file: /root1/Suisse.png

Calibration file: /root1/calibration.cal

POIs file: No file

Figure 3. *Select map* screen

The first two files are mandatory for using a map, so a "*No File"* value for the first to items can not be used for displaying a map. "Select map" will not read the selected files for displaying the map on the screen. This will be done by a map based option of the menu.

The first time the application is used on a device no file is selected. Once a file has been selected the information will be saved and used next time the application is re-launched. This will save you the trouble of selecting a map each time you run the application.

Browse : opens a new window for browsing the file system folder. To select a file, navigate to one of the three mentioned items and select *Browse* from the menu. Several confirmations messages will be launched by the device emulator[4]. Once you have selected the file, click *Ok* to return to the *Select Map* screen. Repeat the operation for each file you wish to select.

Ok : saves the file selection. Once the files have been selected, click the *Ok* menu option to save the file names and locations.

[4] The confirmations will no longer be needed once the application is signed.

Cancel : cancel the selection process. Select this option of the menu to cancel the current selection and return to the previous selection.

Ok and *Cancel* options will end the Select Map mode. The application will return to the calling screen: the main menu or a map based mode.

Recommendation : use the extension ".cal" for the data calibration file for an easier identification of this type of files.

3.2.3.2 Real Time Mode

Use this option to set up in real time the locations to be sent through the NMEA sentences. Locations can be selected on a map. Real time means that each second SkyFreeGPS will use the current indicated location to build Bluetooth streams.

If the map size is larger than the screen size, drag the map by holding down the '7' key. Without pressing the '7' key, SkyFreeGPS will treat the mouse (pointer) dragging as the track to be sent via Bluetooth. The map can also be moved by using the up, down, left and right device keys.

Press '1' and click or drag the mouse/pointer on the map to display locations.

Press the '3' key and click on the map to set up a point of interest. The POI will be marked on the map and added to the list of POIs of the current map.

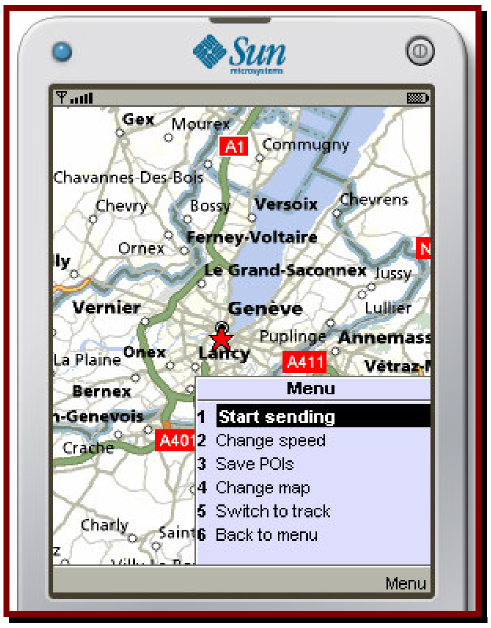


Figure 4. Real time mode screen

Start sending : starts sending valid NMEA sentences using the current position. When setting up the streams, SkyFreeGPS is using the current position and speed. To indicate a position click the mouse on the map or drag the mouse on the map to follow a path. If no location is indicated at the time SkyFreeGPS is building the Bluetooth stream, the last indicated location will be used. That means that if you click once on the map and pause for more than one second, the same location will be sent until your next click on the map.

Stop sending : starts sending blank NMEA sentences. Empty sentences will not be stored to the LogFile list. You can always go back to sending valid sentences by selecting *Start sending* again.

Export LogFile : exports the list of valid NMEA sentences that has been sent via Bluetooth. From the beginning of the use of this run mode, each valid stream is added to a vector. Choosing this option will display a new window for selecting the export file name.

> *Export* : saves the list in the selected file. If the file already exists the content of the file will be replaced, if not the file will be created.
>
> *Browse* : allows the selection of an existing file or the creation of a new file to be selected[5].
>
> *Use last name* : replaces the content of the field with the name of the last used file. Use this option if you don't want to export every time to a new file.
>
> *Use map name* : replaces the content of the field with the name of the map image by changing the extension. For example if you export a track for the map image stored in the file "*Geneva.png"*, SkyFreeGPS will propose a log file named "Geneva.log".
>
> *Cancel* : returns to the browse map window without exporting the track to a file.

Change speed : changes the speed of the current position. A new window will be displayed. In this window, change the speed and click *Ok* to go back to the map window or click *Cancel* to return to the map without changing the speed.

Change map : changes the displayed map. This option opens the *Select Map* window. When selecting this option the log file vector is emptied.

Track Mode : switches to *Track Mode* using the same map. No information is passed on the track mode except for the map and the last used speed. The real time log file vector will be lost.

Save POIs : saves the defined POIs to the specified file. A new window will be opened for specifying the file name.

> *Save* : saves the POIs list in the selected file. If the file already exists the content of the file will be replaced, if not the file will be created.

[5] There is no restriction to the extension used for the log file.

Browse : allows the selection of an existing or newly created file.

Cancel : returns to the previous window without exporting the POIs to file.

Important! The POIs file is not automatically saved/updated when the map is no longer displayed either because the map is changed or because the user quits the Track Mode.

Back to menu : returns to the Main Menu.

3.2.3.3 Track Mode

When needing a specific route that must occur in precise conditions in terms of positions, speed and realistically correct values, use the *Track Mode*. From any starting point *Real Time Mode* can simulate covering any distance in just one second, regardless of the selected speed. In other words there is no correlation between distance and velocity, as each location is selected by the user in real time, independently of the previous one. This is no longer the case with the *Track Mode*. First you need to set up a track and then SkyFreeGPS will compute each location to be sent, based on speed and heading.

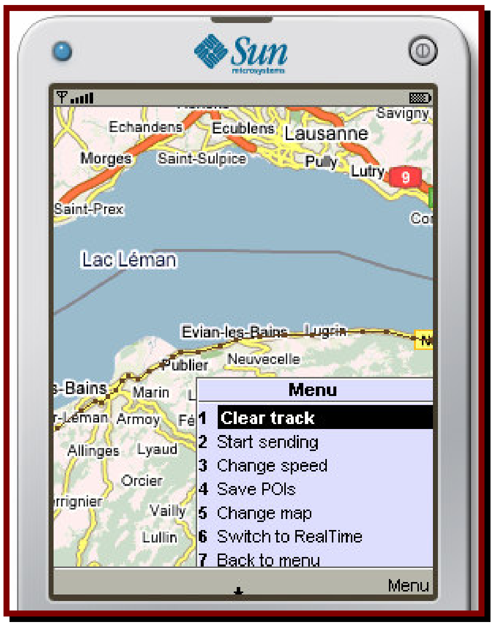


Figure 5. Track mode screen

This is a map based mode as well, which means a calibrated map must be selected before using this feature. Same specifications apply as in the *Real Time Mode* for displaying and moving the map, as well as for displaying the coordinates of a location on the map, or for defining POIs.

Click on the map to set up the key locations that define the track. SkyFreeGPS will display the track on the map by drawing a line between the last clicked location and the previous one. In addition, each selected track point is marked by a small rectangle for an easier identification.

Important! Change the speed before setting up the next track location if you want a different velocity from that point on.

Clear track : cancels the previously defined track. It clears the track from the map and also from the track vector. Once the track is cleared the 'Start sending' option can no longer be used.

Start sending : computes the points to be sent and stores them in a vector. The final track points to be sent are determined based on the drawn track and the speed between each two track points. Once the calculations finished it starts sending valid NMEA sentences using the computed values, beginning with the first selected location. Once the end of the computed positions vector has been reached, SkyFreeGPS returns to sending blank sentences.

Stop sending : returns to sending blank NMEA sentences.

Restart sending : once the sending of a calculated track has been interrupted by choosing *Stop sending*, the calculated vector is not lost. By choosing this option the application will restart sending valid stream using the positions from the beginning of the vector.

Export LogFile : saves/updates the log file of the track. See specifications above.

Change speed : changes the speed for all track points starting from the next selected one. In the new opened window, change the speed and select *Ok* option from the menu to save the change or *Cancel* to ignore it and return to the previous screen.

Change map : select a new map to be displayed on the screen in the *Select Map* window.

Real Time : switches to the *Real Time Mode* window.

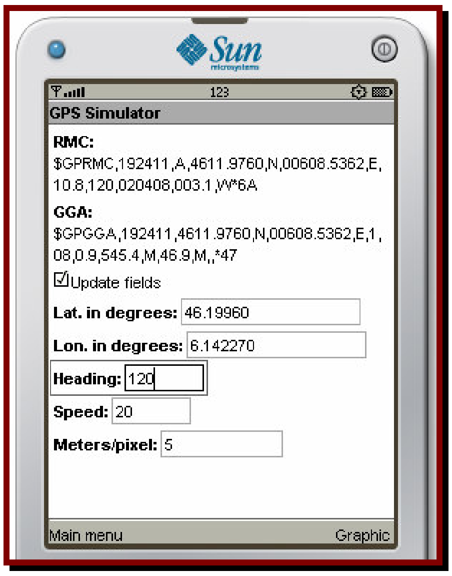
Save POIs : saves/updates the map's POI file. See specifications above.

Back to menu : returns to the Main Menu.

3.2.3.4 Keyboard Mode

This mode doesn't use neither a map nor the pointer. Furthermore, it doesn't send empty sentences as we have seen for the previous two modes. The principle is the following: the application builds the NMEA sentences based on the parameters chosen by the user. These parameters refer to location (latitude and longitude), speed and heading.

There are to ways of changing the position: either by typing it on the keyboard or by using the up, down, left and right keys. These two possibilities correspond to two different windows available under this menu - a text and a graphic window.



GPS Simulator

RMC:
$GPRMC,192411,A,4611.9760,N,00608.5362,E,10.8,120,020408,003.1,W*6A

GGA:
$GPGGA,192411,4611.9760,N,00608.5362,E,1,08,0.9,545.4,M,46.9,M,,*47

☑Update fields

Lat. in degrees: 46.19960

Lon. in degrees: 6.142270

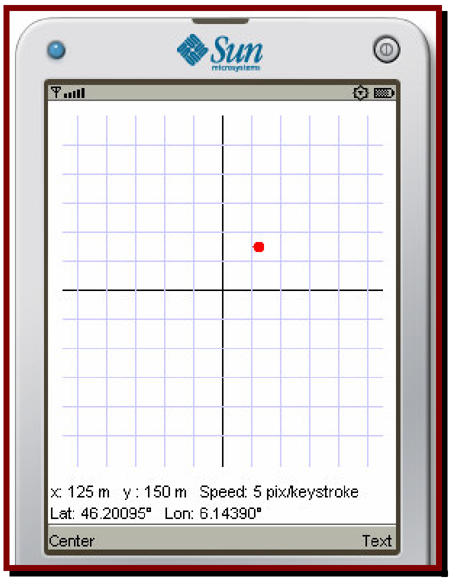
Heading: 120

Speed: 20

Meters/pixel: 5

Main menu Graphic

Figure 6. Keyboard mode - Text screen



x: 125 m y : 150 m Speed: 5 pix/keystroke
Lat: 46.20095° Lon: 6.14390°

Center Text

Figure 7. Keyboard mode - Graphic screen

Text window : allows to change the coordinates of the current position and all other specified parameters. The check field Update fields, will determine the display of the two NMEA sentences, when selected. Change the values of the fields and notice the change in the NMEA sentences. The field *Meters/pixel* refers to the *Graphic window*. To change to the graphic mode select *Graphic*. To return to the main menu select *Main menu*.

Graphic window: permits to change the only current position. The red point represents the current position. To change this position use the up, down, left and right keys. To position the current red point to the centre of the to axes select *Center*. To change the speed at which the red point moves : press the key 1 to increase the speed with one unit or press the key 3 to decrease it with one unit.

The *graphic window* will not change the values of the other parameters. To return to the previous window select *Text*. You can only return to the main menu from the *text window*.

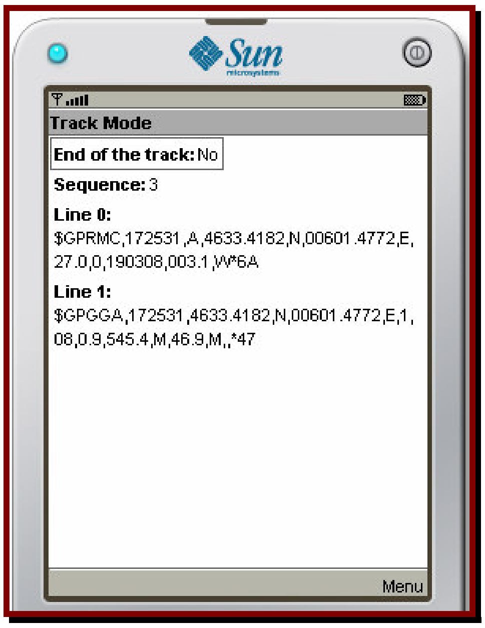
3.2.3.5 Log File Mode

SkyFreeGPS can read previously saved log files and replicate the sentences from the file. When the end of the file is reached the application sends empty sentences. Upon selection of this menu SkyFreeGPS is loading a window for selecting the log file to be used:

Browse : selects a file from the devices file system.

OK : confirms the selection and launches the Log File Mode screen.

Cancel : returns to the main menu without saving the selection of the log file.



Track Mode

End of the track: No

Sequence: 3

Line 0:
```
$GPRMC,172531,A,4633.4182,N,00601.4772,E,
27.0,0,190308,003.1,W*6A
```

Line 1:
```
$GPGGA,172531,4633.4182,N,00601.4772,E,1,
08,0.9,545.4,M,46.9,M,,*47
```

Figure 8. Sending data from a log file

Change track : returns to the selection window mentioned previously, for selecting another log file to be used.

Back to menu : returns to the main menu.

3.2.3.6 Help

Explains the previous menus. Select an item to display its content. Click *Back* to return to the previous screen.

3.2.3.7 Exit

Select Exit to close the application. Some of the values will be saved and used next time the application is run on the same device, like for example: the selected files regarding the map, the file used for exporting a track.

3.2.4 Conclusion Quick steps to start using de application

1. Create a new project in WTK for SkyFreeGPS, either from scratch (if you're using the source code) or from the JAD/JAR file.

2. Activate the use of the pointer on the preferred emulator's device by changing the touch_screen parameter to true in the properties file.

3. Save at least one calibrated map in the device's file system.

4. First open the support application SkyFreeGPS and allow Bluetooth connection.

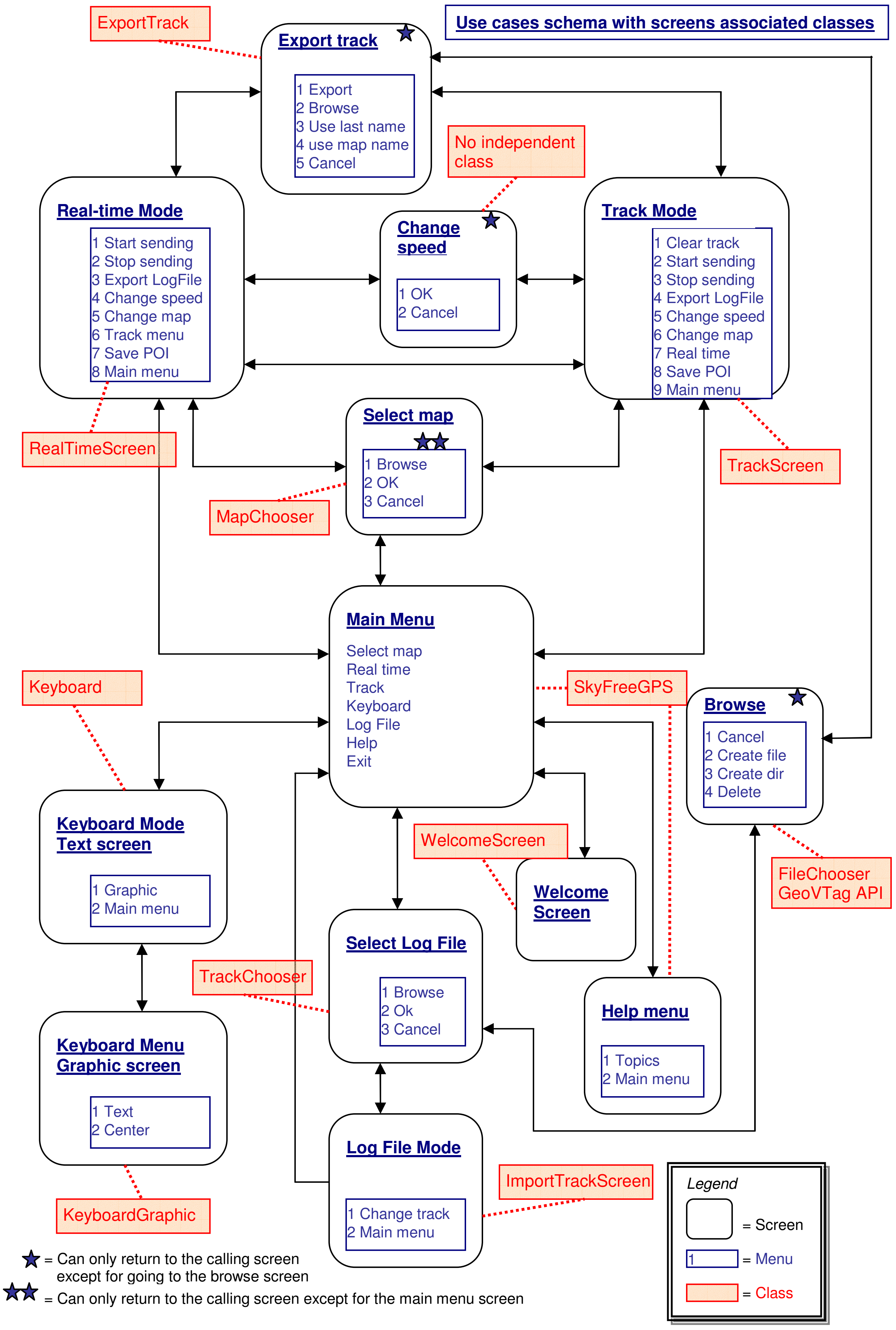
5. Open the application that requires GPS sentences. Connect your application to SkyFreeGPS.

6. Select one of the four available running modes of SkyFreeGPS and start sending valid GPS sentences. Watch the behaviour of the tested application under different GPS track scenarios.

4. Technical specifications

4.1. Classes Description

Below is a schema of the use cases that illustrates how a user can navigate between screens and what are the classes employed to display these screens:


Use cases schema with screens associated classes
ExportTrack
Export track
1 Export
2 Browse
3 Use last name
4 use map name
5 Cancel
No independent class
Real-time Mode
1 Start sending
2 Stop sending
3 Export LogFile
4 Change speed
5 Change map
6 Track menu
7 Save POI
8 Main menu
Change speed
1 OK
2 Cancel
Track Mode
1 Clear track
2 Start sending
3 Stop sending
4 Export LogFile
5 Change speed
6 Change map
7 Real time
8 Save POI
9 Main menu
RealTimeScreen
Select map
1 Browse
2 OK
3 Cancel
TrackScreen
MapChooser
Main Menu
Select map
Real time
Track
Keyboard
Log File
Help
Exit
Keyboard
SkyFreeGPS
Browse
1 Cancel
2 Create file
3 Create dir
4 Delete
Keyboard Mode Text screen
1 Graphic
2 Main menu
WelcomeScreen
Welcome Screen
FileChooser GeoVTag API
Select Log File
1 Browse
2 Ok
3 Cancel
TrackChooser
Help menu
1 Topics
2 Main menu
Keyboard Menu Graphic screen
1 Text
2 Center
Log File Mode
1 Change track
2 Main menu
ImportTrackScreen
KeyboardGraphic
★ = Can only return to the calling screen except for going to the browse screen
★★ = Can only return to the calling screen except for the main menu screen
Legend
= Screen
1 = Menu
= Class

SkyFreeGPS uses some of the classes of the *GeoVTag* API, like:

- *Menu* and *MenuListener* for constructing the menu,
- *PropsRS* to store properties related to running the application,
- *FileIO* and *FileStringItem* for working with files,
- *Tools* for some methods to make data computations,
- *HierarchicText* for the help menu.

All classes directly related with the application are saved in a package called *SkyFreeGPS*.

Params class

The class *Params* is created for storing parameters needed throughout the entire execution, parameters like: the name of the files referencing the map, the image of the map, the current set speed etc. It is used for passing the values of these parameters from a class to another. All objects created by this class are static for ensuring that there is a unique value for each parameter throughout the entire execution of the application. A single object of the class Params is instantiated at the beginning of the application – *params,* and then passed from a class to another.

SkyFreeGPS class

The MIDlet of the application is called SkyFreeGPS. The class is responsible for:

- displaying the logo of the application upon launching,
- handling the main menu,
- handling the help menu,
- storing some values for the next time SkyFreeGPS is used on the same device,
- closing the application.

This class creates an object of the class Menu from the *GeoVTag* API and implements the class *MenuListener* responsible for the menu items, from the same API. It also initialises the parameters by creating a new object of type *Params* and finally opens a Bluetooth connection by creating an object of the class *Connection*.

Upon closing of the application, this class will save values from the *params* object in the PropsRS record store object.

WellcomeScreen class

Creates an image of the logo and displays it on the screen for 2.5 seconds before returning to the display of the main menu.

Connection class

This class manages the Bluetooth connection. It opens a connection, it waits for a client to connect, once a client connected it sends NMEA sentences (empty or valid ones) and waits for a new connection when the client disappears.

MapChooser class

MapChooser is responsible for selecting the files related to the map to be used. For doing this, it creates an object of the class *FileStringItem* (which belongs to the GeoVTag API) for each file to be handled. This class does no verification as to the content of the files. It merely registers the selection of these files. Even "NO_FILE" values are being accepted.

If the command *Ok* has been used and the values of the files have been changed then this class alerts the other classes that the map must be reloaded, by changing the value of *params.calibratedMap* to false. Also the selection of the files is stored in the object *params*.

If the command Cancel has been used then the values of the fields must be restored to the previous ones. They can be retrieved from the *params* object.

A *MapChooser* object is created from the main menu or from the real time menu or from the keyboard menu. The method *exitMapLoader()* is called each time this class returns to another class, regardless of the exit option (*Ok* or *Cancel*). This method makes the choice of the next displayable, based on the menu that called a *MapChooser*. The value of *params.callingDisplayable* specifies exactly the menu/display to return to.

MapMode class

This class is responsible for reading the content of the selected map related files. It is called each time a map based menu is selected : the *real time menu* or the *track menu*.

MapMode first verifies if the files have been selected. If not than displays an alarm and creates a MapChooser in order to allow the user to select the map files.

If the files are being correctly specified than *MapMode* will read the content of the map image file and create the image, will read the calibration values and store them in the params object.

However if a map based menu is re-launched without having changed the map related files, that means that the information already exists so *MapMode* doesn't need to read the files again and can directly display the map based menu.

RealTimeScreen class

Handles the *real time menu*. It displays the map on the screen, it manages the pointer for selecting track points or for moving the map and it changes the value of the speed.

The class RealTimeScreen is never called directly from the main menu. It is always called through a MapMode object, responsible of passing the right map paramenters.

TrackScreen class

This class is responsible for managing the *track menu*. It is very similar to the *RealTimeScreen* class, with some exceptions: it displays the track on the screen, it saves all track points selected by the user in order to calculate which track points to be sent via Bluetooth.

The same specifications are available as the ones for the class *RealTimeScreen.* A *TrackScreen* object is only created in the class *MapMode* and not directly from *SkyFreeGPS* (the main menu).

Keyboard class

Displays the text display of the keyboard menu. This class creates an object of the *KeyboardGraphic* class, making possible to switch between the two *keyboard* menu windows. This class can return to the main menu.

KeyboardGraphic class

Displays the graphic display of the keyboard menu. It can only switch to displaying the text displayable of the keyboard menu, and not to the main menu.

ExportTrack class

This class is responsible for exporting to file the track log that has been created under real time or track menu.

An object of type *ExportTrack* opens a new display for specifying the file to export to and than saves the content of the params.sentSentences to this file. This class is runnable because some operations (like for example creating a file) can not be performed under *CommandAction*, so a thread must be launched from the *CommandAction* method to perform these operations.

ExportPOI class

This class is responsible for exporting to file points of interest defined on the map.

Similar to the *ExportTrack* class, an object of type *ExportPOI* opens a new display for specifying the file to export to and than saves the content of the params.POIs to this file. Same specifications for the file related operations (see *ExportTrack* paragraph).

TrackChooser class

It is very similar to the *MapChosser* class. The differences are that the *TrackChooser* manages the files related to the track to be imported under the *log file menu*. A *TrackChooser* is created only from the *ImportTrack* class and it returns to this one.

ImportTrack class

Verifies if a file has previously been selected by a TrackChooser object. If not, then it creates a TrackChooser object. If yes, it reads the content of the selected file and creates an object of the class ImportTrackScreen which will displays the content of the log file.

ImportTrackScreen class

This class displays the content of the specified log file as it is sent via Bluetooth. It alerts the user when the end of the file has bean reached.

4.2. Implementation choices

During the development phases, choices had to be made between several possible solutions for implementing some features. This paragraph describes below some of these choices and the arguments that balanced in their favour.

***When to open a Bluetooth connection*?**

The considered alternatives were:

- each time the user decides it, from a menu that sets up track points (four menus) *or*
- when the application is launched up until the end of the execution.

At a first glance, the initial option makes more sense: why should SkyFreeGPS send NMEA sentences from the *help menu* for example? Although there is no reason for sending Bluetooth streams from all the menus of the application, the fact that the connection is interrupted each time the user quits a menu forces the tested application to establish the connection again and again. As a consequence this choice was rejected.

This is why empty sentences have been introduced. As a consequence, under the main menu and all other menus that do not set up track points (*select map menu*, *help menu*), SkyFreeGPS simulates a no satellite signal situation, although it does send the current time.

How to display the track on a map?

The considered options were:

- to draw the lines on a second layer each time the map is painted *or*
- to modify the *Image* object of the map in order to include the drawn lines.

The first option consist in storing the selected point tracks in a vector and drawing a line between each two consecutive points, regardless of the fact that the line crosses the screen or not (in the case the map is larger than the screen). Each time the map is dragged the lines are redrawn. This causes a slow down of the application during the dragging of the map. The drawing of a new line is fast, but moving the map is too slow.

The second option consist in drawing a line between each two consecutive lines and updating the original image after each line is drawn. This solution slows down the drawing of a line, on the other hand dragging a map is not disturbed by the size of the track.

How to decide what locations belong to the computed track ?

- follow the path line exactly as it was drawn on the screen *or*
- calculate a location based on speed and heading even though there might be slight differences from what was drawn on the screen.

Using the pixels that marked the trajectory between two track points would be more accurate with regard to what we see on the screen. However the computations are very complex in this case, consequently to much time is needed to perform them. Here is why:

This is how we see a line on the screen and also how this line really looks like:

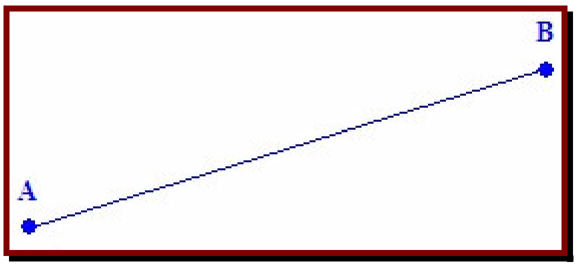


Figure 9. A line between point A and B

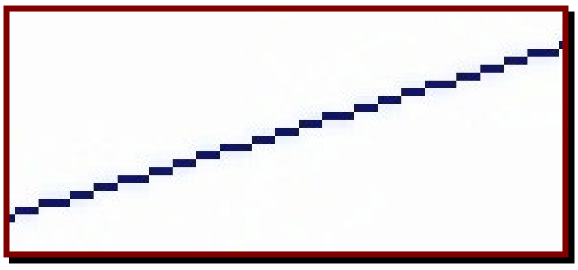


Figure 10. Same line zoomed in

We can find out what are the pixels changed on the screen to draw a line. These pixels are stored in an array following each line of the screen from left to right and from up to bottom. Therefore we know how many pixels the line has in total and what was the time needed to get from point A to point B. As a result we can determine how many pixels we must jump per second in order to transmit a location each second. So far nothing to difficult. The difficulty of the computation comes from the fact that the pixels in the array are not in the correct order and must be rearranged to follow exactly the direction in which the line is drawn (or the track is performed). This must be done in several steps. Without getting in too much detail, the next figure illustrates a part of the problem.

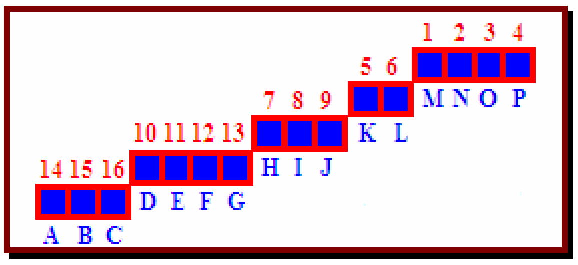


Figure 11. Illustration of the pixels order

The red numbers from 1 to 16 indicate the order in which the pixels are stored in the array and the blue letters from A to P the order in which the track takes place. In this case the computation should change the pixels from the 1 – 16 order to the A - P order. So this isn't the best option to be used.

On the other hand, by computing the intermediary points without considering the way the line has been drawn on the screen we might end up having slightly different locations than what we see on the screen. Nevertheless the differences are acceptable, even less significant than those I have illustrated below to point up the case.

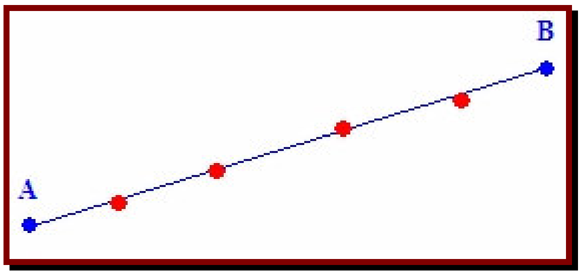


Figure 12. Computed track points versus drawn points

Therefore this second option seemed better than the first one.

4.3. Future application developments

Some of the possible future improvements for the SkyFree GPS application are being listed below:

Bluetooth related improvements

- Adding other types of communication protocols besides NMEA, like for example GARMIN protocol.
- Permitting communication port or TCP socket connection. In this way one can send/receive messages to and from another computer in the network.

- Possibility to send the track log file via Bluetooth.

NMEA related improvements

- Adding an interface for changing other GPS parameters, like the frequency of GPS streams, the number of satellites in use, the altitude etc.

Map related improvements

- Adding more tests when opening the map calibration data file. This could ensure the correct extraction of calibration data with as little restrictions as possible concerning the content of the file.
- Adding the possibility to open an non-calibrated map image. In this case the application should ask for the manual input of the calibration data.
- Displaying the coordinates of the last sent position whenever the map is active on screen.
- Adding a small compass image for emphasising the track angle in degrees whenever the map is active on screen.
- Including a set of calibrated maps with the jar file. These maps should be accessible from the *Chose Map* option. The maps must by copylefted.

Log file and track menu related improvements:

- Allowing the user to visualise the track list before exporting it and also allowing single or multiple selection of the sentences to be exported. Current version exports the entire track without any selection and without showing the exported content on the screen.
- Adding multiple export formats for the log file, like for example the .GPX standard.
- In the track mode (non real time) - adding the option to chose weather the sending of valid GPS sentences should stop with the end of the track or continue from the beginning of the track.
- Possibility to change the colour of the track.

User-friendliness related improvements:

- Developing a completely stylus based application: allowing the stylus to be used in all features of the application, not only for pointing the track.
- Improving the graphical UI of the application.

Some of the above mentioned developments are more easier and faster to implement, while others will take more time and effort.

5. Conclusion

After a first experience with a BA project in the Economics field, which involved more of a research work rather than an innovating one, I decided to look for the opposite this time - a development project. Moreover, I was more interested in a project in which the outcome will be of real use for others rather than an entertaining one. SkyFreeGPS was the perfect choice to make.

The main difficulties I have been confronted with during this project were related to the development phase. Being my first application of this size I had some trouble in putting together all the small MIDlets I have designed separately as a response to different features of the application. But also this one was the most rewarding phase of the entire project, as finding solutions to implementation problems has been a source of motivation for continuing a difficult task.

Even tough there is still place for improvements, the primary objectives were achieved. These objectives were:

- to have an application that simulates a GPS receiver by generating GPS data;
- to allow the user to specify locations trough manual input or through computations done by the application;
- to save to file the generated GPS data for future use;
- to make it easier to define locations by the use of maps and the pointer;
- to allow definition and management of points of interests on maps.

Having said that, here is a reminder of some of the advantages of using SkyFreeGPS:

- Being a simulator, SkyFreeGPS operates indoor without visible GPS satellites, which is very useful inside the buildings where real GPS receivers do not work. This is an ideal software if one needs to use GPS data without having a GPS receiver or GPS signal;
- SkyFreeGPS saves time and energy while performing tests on the application one is developing;
- SkyFreeGPS makes it more comfortable to test a NMEA sentences based application;

- Testing an application with a real GPS receiver can be very dangerous when looking at the small screen of the mobile phone and moving at the same time, regardless whether you walk, cycle or drive. It could prove to be much safer to use a simulator instead.

And here is what makes SkyFreeGPS different than other applications:

- It runs in WTK;
- It is free;
- It is open source;
- It doesn’t need other software or hardware resources than the ones already available to a J2ME developer.

***

6. Useful Links

GPS software and resources

Latitude and longitude finder

http://mapki.com/getLonLat.php

http://www.naffis.com/maphacks/latandlon.html

Distance calculator

http://www.naffis.com/maphacks/distance.html

Map creator from tracks, GPX file converter

http://www.gpsvisualizer.com/

Sun Java Wireless Toolkit location demo with use of the External Events Generator

http://developers.sun.com/mobility/wtk/demos/wtk-lapi.html

Java Language

Java 2 Platform, MicroEdition (J2ME) tutorial for intermediate developers

http://www.scribd.com/doc/425875/J2MEStepbystep

J2ME tips and tutorials

http://www.java-tips.org/java-me-tips/

7. References

Internet sites

NMEA sentence overview

http://www.gpsinformation.org/dale/nmea.htm

JAVA discussions forum

http://www.java-forums.org/java-me/

Publications

J2ME manuals

Wireless Game Development in Java with MIDP 2.0 by Ralph Barbagallo, 2004 Wordware Publishing Inc.

Beginning J2ME From Novice to Professional by Sing Li and Jonathan Knudsen, Apress.com publishing

APIs

GeoVTag

http://cui.unige.ch/~deriazm/Softs/GeoVTag/

8. Special thanks

I would like to thank my supervisor, Michel Deriaz, for suggesting the idea of such an interesting project, for giving me advice and challenging me to find new solutions to various problems. By choosing this project, I had the opportunity to discover several new areas for me: Java 2 Platform, Micro Edition (J2ME) and the GPS sphere. What greater pleasure than to plunge oneself in complete unknown in order to learn new things!

Through constant involvement in different stages of my project, Michel helped me understand the importance of following my initial planning. He showed patience and wisdom supporting me through difficult times, without giving up solutions but rather suggesting ideas and letting me find on my own the way to move further.

I am also very grateful for all the materials Michel provided me with, especially for the source codes of the GeoVTag API, which greatly helped me in understanding the J2ME language and in building up my own application.

I would also like to thank my friend and artist Codruta Cernea for her kind participation in the graphic design of my project. Many thanks also to Horea Burca for his contribution to the design of the website.

I can not forget my colleagues Pedro Velasco and Shaban Shaame for their solidarity and for sharing with me the difficulties involved in developing a new application while doing their own projects.

And finally I would like to thank my husband, Dragos Burca for supporting me throughout the entire project. His comments and recommendations were of great value to the final outcome of the present document.

***