Fun with OsmAnd: Tweaking the routing engine

Tweaking the routing engine

- Slippy vs Vector
- What is a rendering engine?
- What is a routing engine?
- The .obf-file
- Parameter-files
- Getting our hands dirty!

This is a highly technical workshop

- If you do not know what XML is…
- If you do not know how to place a file on your Android…
- … run away now!

You’ll need a computer
You’ll need an Android phone
It might work on a *jailbroken* iPhone. Not tested though

Showing maps to users

- OSM = database of dots & lines
- How to show them?
  - Convert them to images!
  - A piece of software does this: the *rendering engine*
- How to distribute these images?
  - Keep all the images in one point and send those to clients
  - Distribute the rendering engine and the data to the clients

Slippy map

- One big computer takes *all* the data and creates lots of images for all zoom levels
- Clients download these tiles when they need them
- Simple
- Lots of space needed
- Static (style changes?)

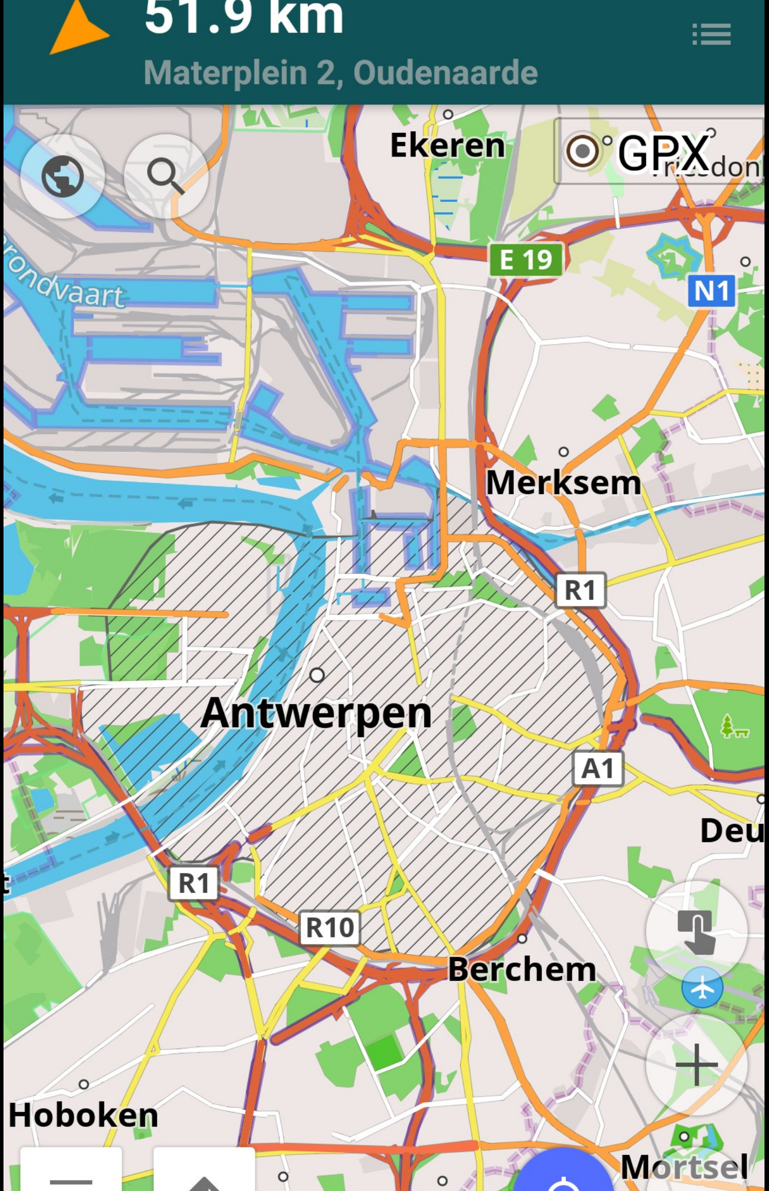
Vector map

- A piece of the database is downloaded
- Image is generated *on the fly*
- Complex
- Less space required
- No connection with internet required

OsmAnd rendering engine

- Complicated piece of software
- Colours and styles are described in `render.xml`
- Read from SD-card/internal memory on startup (if existing)
- Thus: tweakable!

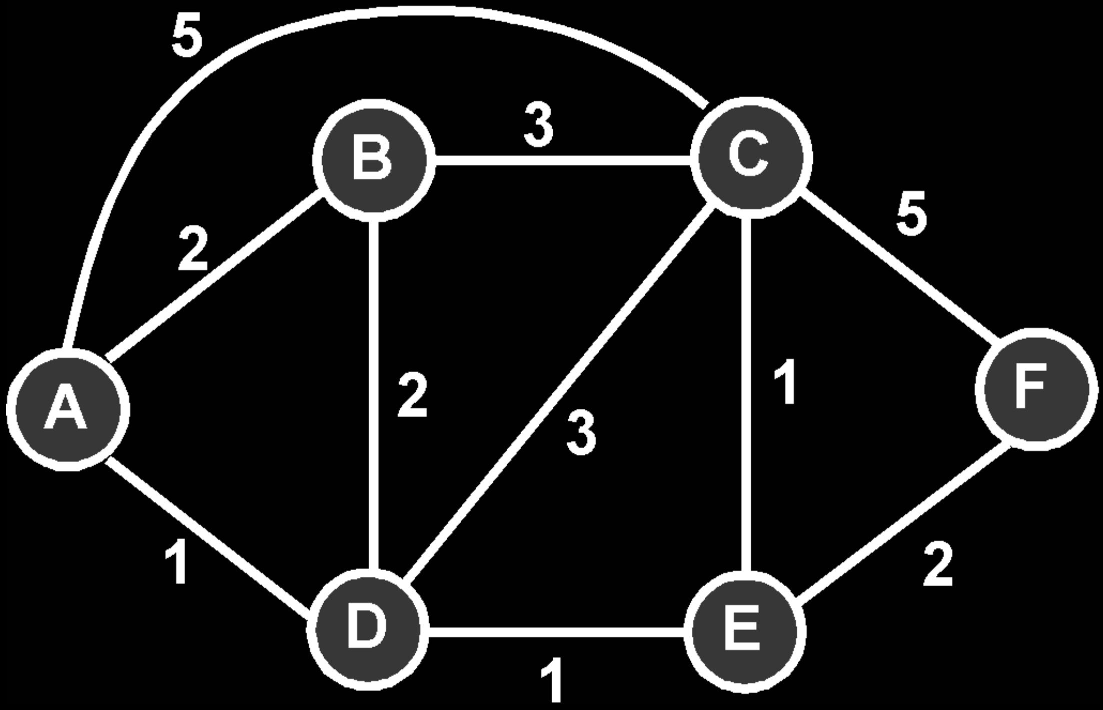
LEZ


51.9 km
Materplein 2, Oudenaarde
GPX
Ekeren
E 19
N1
Merksem
R1
Antwerpen
A1
Deu
R1
R10
Berchem
Hoboken
Mortsel

Routing engine

Calculates the fastest way from A to F

Might try *all* solutions…
… or just a few promising

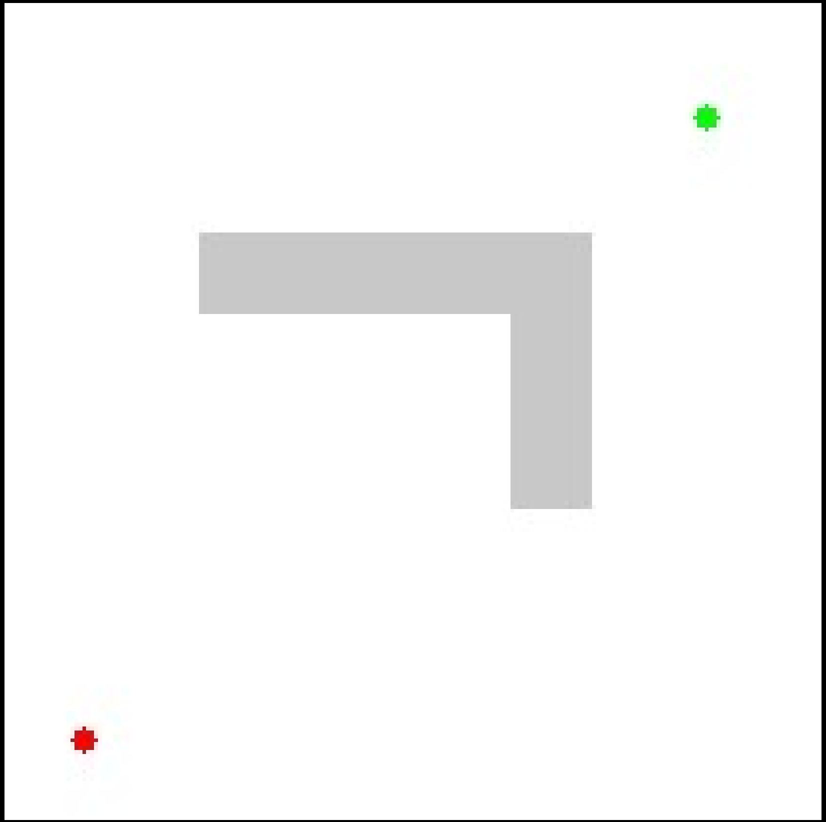


A*

Tries to go to the goal directly

but makes detours

- Slow, but quite accurate
- Can be very slow if
  the last streets are expensive

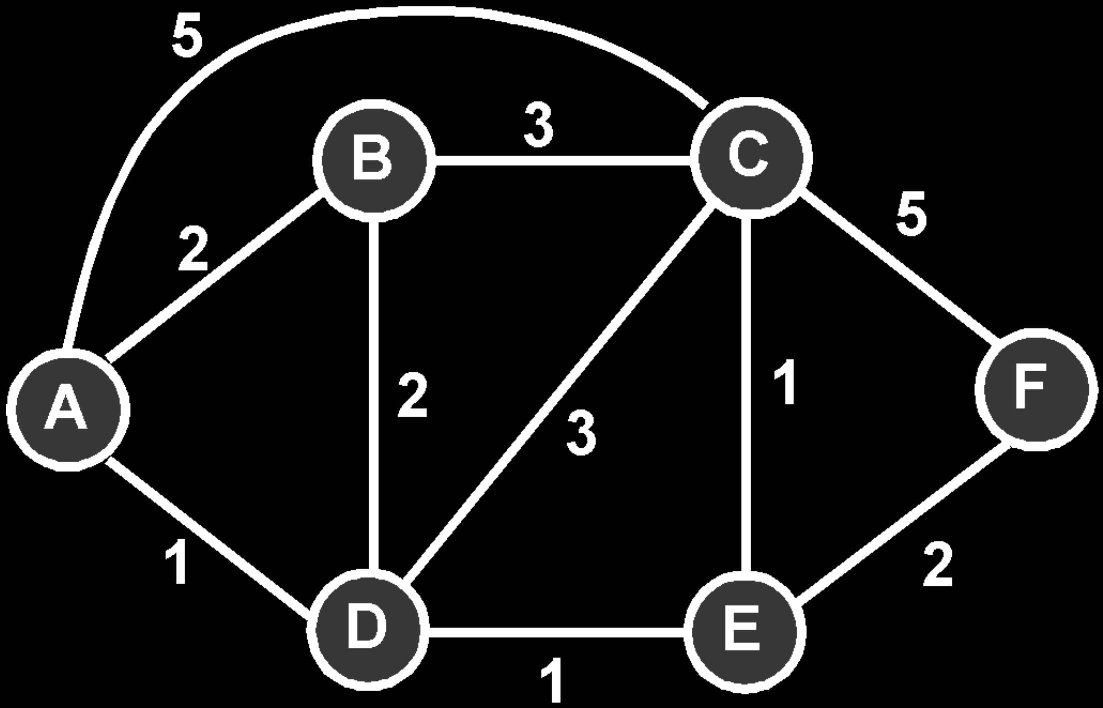


Routing engine

Not every street is equally fast or equally comfortable

Speed and comfort are **subjective**

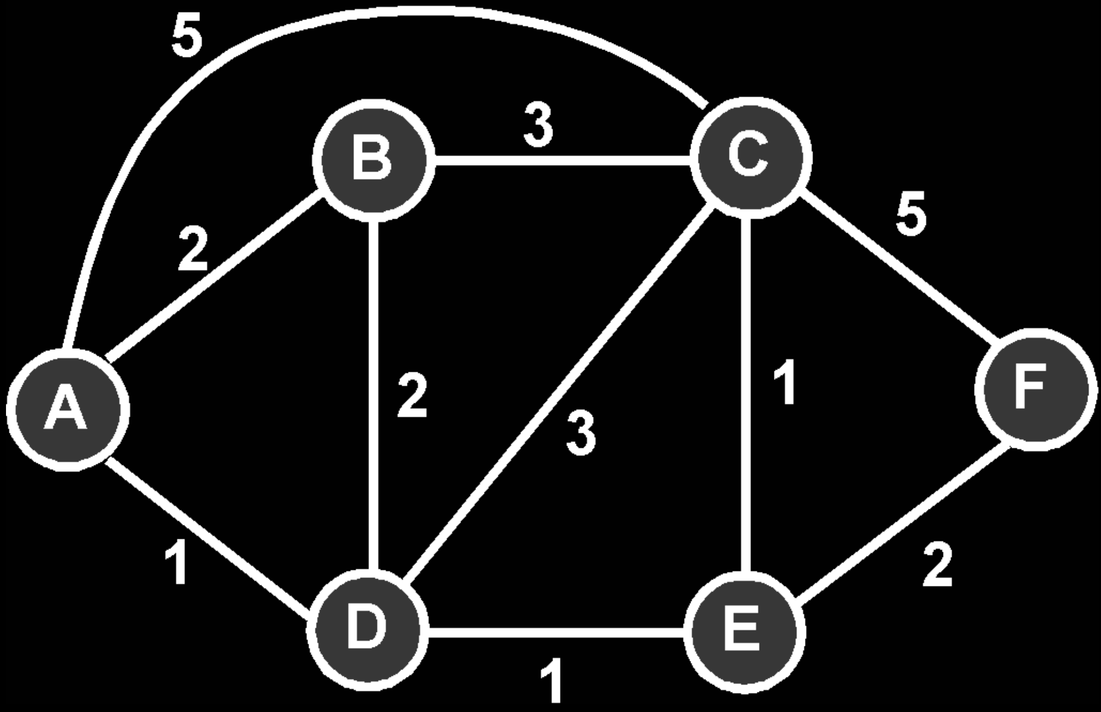
How are these speeds and weights assigned?



Routing engine

OsmAnd assigns speed & weight based on **routing.xml**

We can play with this!

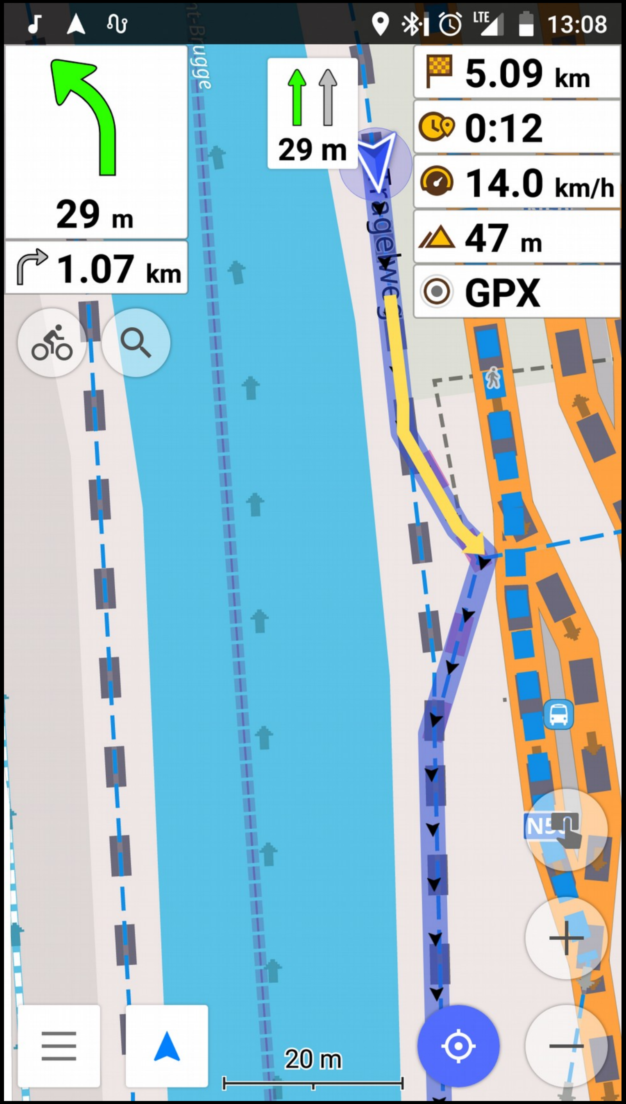


OsmAnd Routing engine

Uses A*

Does weird stuff sometimes
(Small data errors, complex rules)

Just the way it is!
Experiment, use your *fingerspitzengefuhl*



Where does the data come from?

Downloaded by OsmAnd on first boot

- **Belgium.obf**

Belgium.obf

Binary file containing all information of Belgium

Three important parts:

1. Rendering index (I’m here, what should I draw around)
2. Search tree (I’m searching for ‘xyz’)
3. Routing graph (I want to go from *here* to *there*)

Belgium.obf: routing graph

To keep the files small, not everything is included, only relevant ways and tags

Out of scope with the default maps:

- A ‘power line router’ which calculates a route over high voltage lines
- ~~A ski router over ski pistes~~ (will be included from 1th of may)
- A router avoiding (or allowing) streets based on their name (e.g. knippen in Gent)

But we can build our own!

I’ll explain how depending on where we get

Belgium.obf: routing graph

To keep the files small, not everything is included, only relevant tags

In scope:

- Common ‘highways’ as carways, cycleways, bridleways and footways
- Including common tags as maxspeed, surface, part of cycling network, max_height, barrier=*, oneway, ...

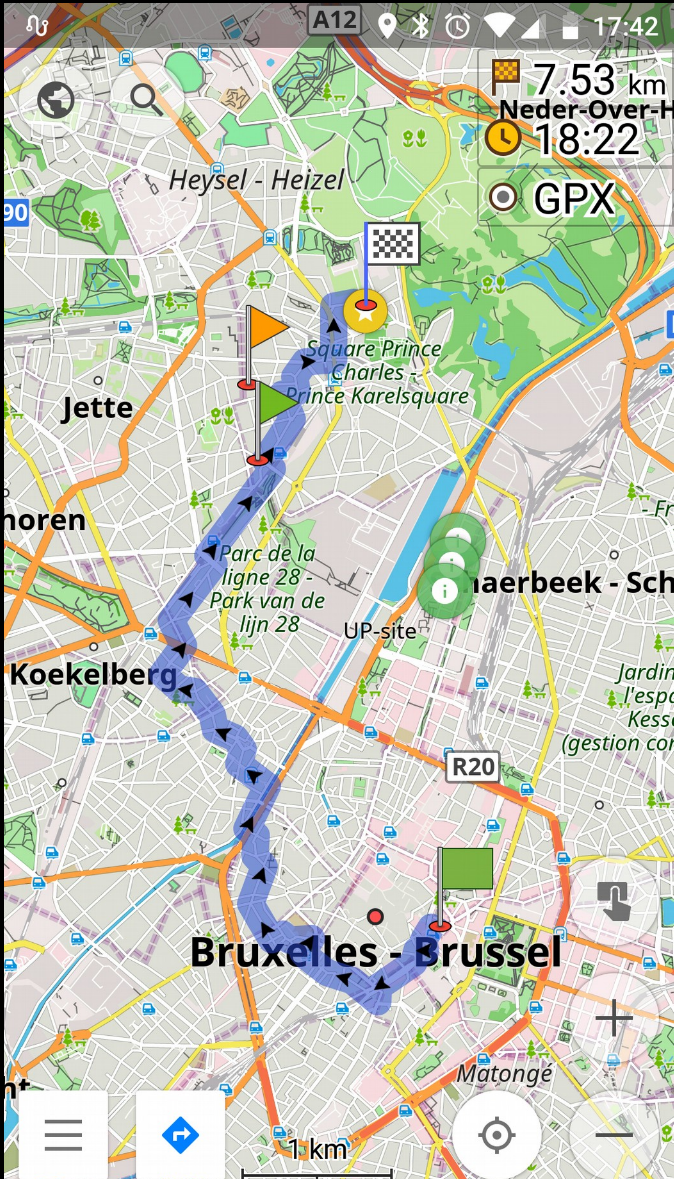
Assigning weights

Done based on ‘routing.xml’

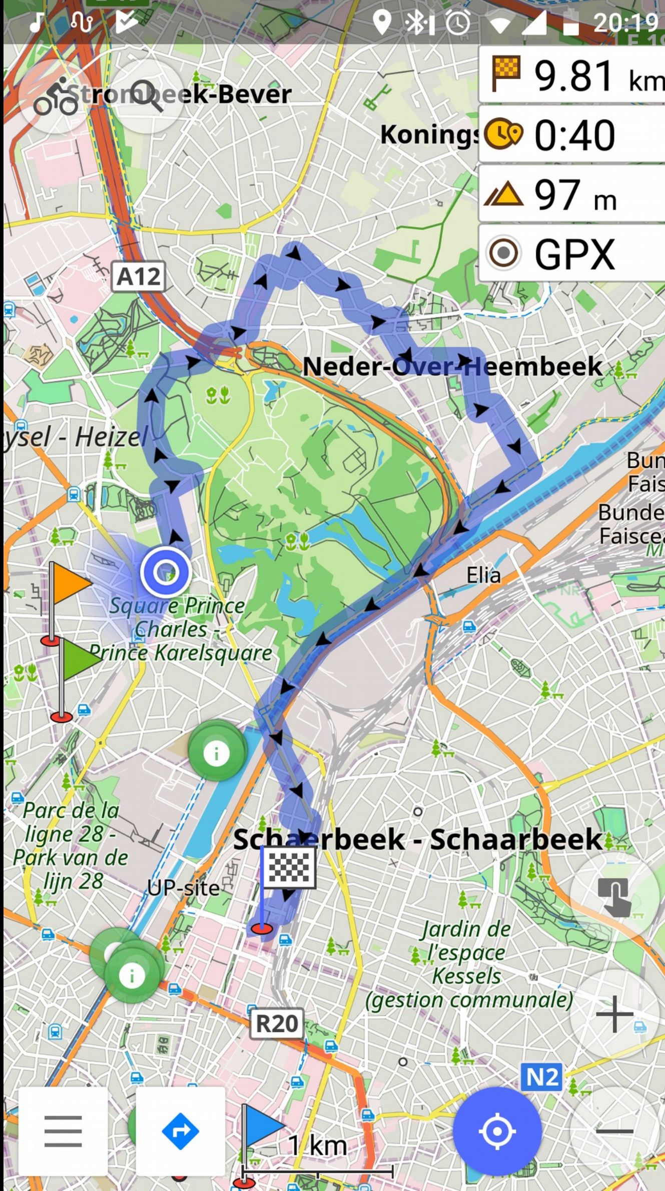
Loaded dynamically on app startup

- We can modify it and load our own!

Routing via unknown surfaces



Routing via unknown surfaces


20:19
9.81 km
0:40
97 m
GPX
Strombeek-Bever
Konings
A12
Neder-Over-Heembeek
ysel - Heizel
Square Prince Charles - Prince Karelsquare
Elia
Parc de la ligne 28 - Park van de lijn 28
UP-site
Schaerbeek - Schaarbeek
Jardin de l'espace Kessels (gestion communale)
R20
N2
1 km

Lets get started

Routing.xml contains several parts

- <routingprofile name=”bicycle”/>
    - <parameter id="avoid_sett" name="Avoid sett roads" description="Heavily prefer concrete and asphalt roads over sett" type="boolean"/>

Lets get started

Routing.xml contains several parts

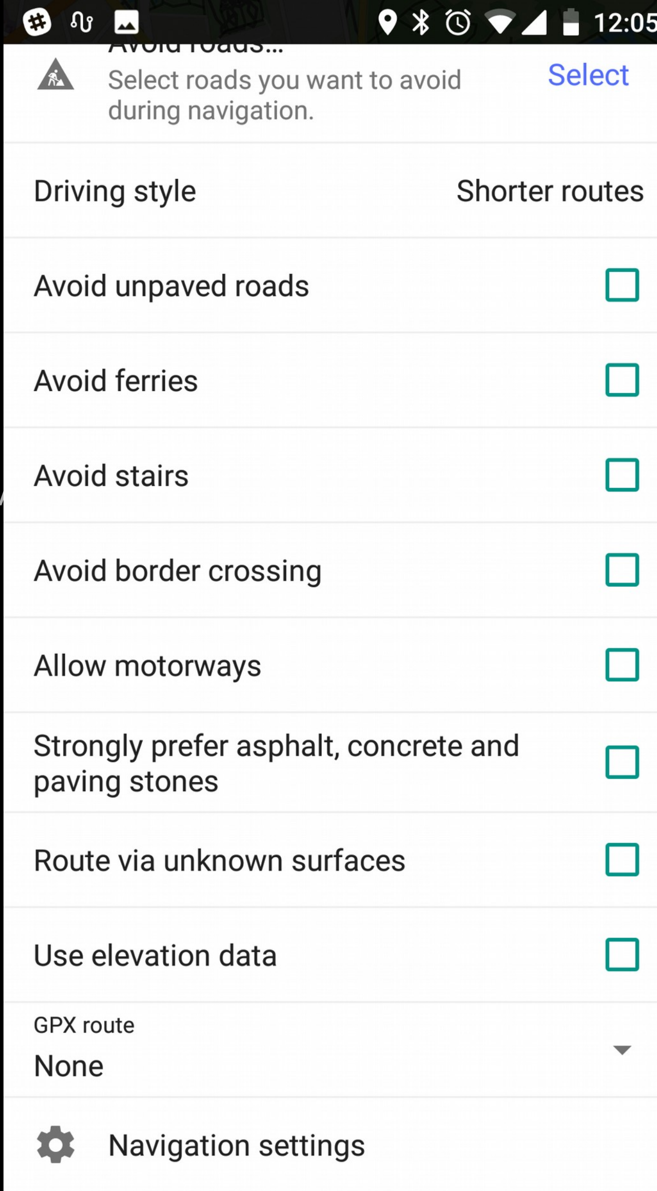
- <routingprofile name=”bicycle”/>
    - <parameter id="avoid_sett" name="Avoid sett roads" description="Heavily prefer concrete and asphalt roads over sett" type="boolean"/>

Lets get started

Routing.xml contains several parts

- <routingprofile name="bicycle"/>
  - <parameter id="avoid_sett" name="Avoid sett roads" description="Heav
    roads over sett" type="boolean"/>

Start with adding an option:
good check to see if your routing profile got loaded


Avoid roads...
Select roads you want to avoid during navigation.
Select

Driving style — Shorter routes

- ☐ Avoid unpaved roads
- ☐ Avoid ferries
- ☐ Avoid stairs
- ☐ Avoid border crossing
- ☐ Allow motorways
- ☐ Strongly prefer asphalt, concrete and paving stones
- ☐ Route via unknown surfaces
- ☐ Use elevation data

GPX route: None

Navigation settings

Lets get started

Routing.xml contains several parts

- <routingprofile name=”bicycle”/>
    - <way attribute=”access”> *Can I enter this road with this vehicle?*

Lets get started

Routing.xml contains several parts

- <routingprofile name=”bicycle”/>
  - <way attribute=”access”> *Can I enter this road with this vehicle?*
    - <select value=”1” t=”highway” v=”cycleway”/> *Ccleways are available*

Lets get started

Routing.xml contains several parts

- <routingprofile name=”bicycle”/>
  - <way attribute=”access”> *Can I enter this road with this vehicle?*
    - <select value=”1” t=”highway” v=”cycleway”/> *Ccleways are available*
    - <select value="-1" t="highway" v="construction"/> *Roads under construction are not*

Lets get started

Routing.xml contains several parts

- <routingprofile name=”bicycle”/>
    - <way attribute=”access”> *Can I enter this road with this vehicle?*
        - <select value=”1” t=”highway” v=”cycleway”/> *Ccleways are available*
        - <select value="-1" t="highway" v="construction"/> *Roads under construction are not*
        - *<if param="allow_motorway"><select value=”1” t=”highway” v=”motorway”</if>*

Parts of a profile

1. access
2. **speed**
3. oneway
4. **priority**
5. obstacle
6. obstacle_time
7. penalty_transition

Speed

How fast will the user drive here? Value in km/hr

Some a little *too* fast

```
<way attribute="speed" type="speed">
    <select value="33" t="highway" v="cycleway"/>
    <if parameter=”avoid_sett”>
        <select value=”5” t=”surface” v=”sett”/>

    </if>
</way>
```

Priority

Niceness of the road

```
<select value="1.35" t="surface" v="asphalt"/>

<select value="0.9" t="cycleway" v="shared_lane"/>

<select value="0.65" t="surface" v="cobblestone"/>
```

The actual weight

The routing time of a segment is the sum of values calculated by these blocks, as the following formula indicates:

distance/minimum(maxDefaultSpeed, speed*priority) + height penalty + obstacle penalties + turn penalties + ...

Where `maxDefaultSpeed` is defined by the XML (in the header of the profile), whereas max_speed can be given explicitly by the road or implicitly by law

Some experiences I had

- A priority between 0 and 1 is more stable then a priority > 1 (chance of not working)
- Tweaking the parameters: lots of experimentation and ‘this feels good’, some *fingerspitzengefuhl* is needed
- Routing definitely needs a refactoring…
- OsmAnd loves pull requests and tweaks (but prepare for some going forth and back)

    - Cobblestone avoidance added
    - Rendering of climbing sites (crags)
    - Ski routing!
    - Documentation

Go forth and experiment!

Get instructions at **pietervdvn.github.io**

Modify it

Load it to your device (/sdcard/Android/data/net.osmand.plus/files/)

> *Actual location can be different on your device!*
>
> *Have a look around*

Kill osmand (recent app overview -> swipe osmand away)

Test routing