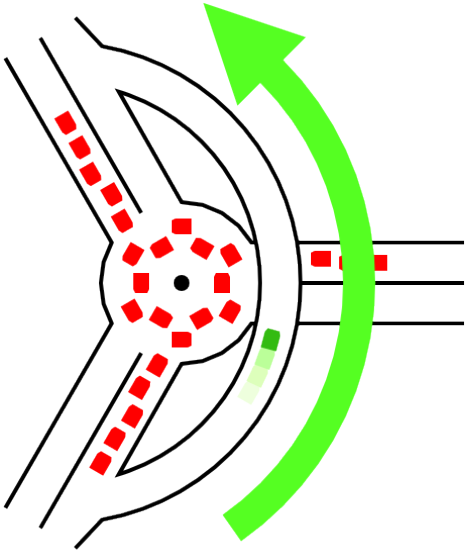
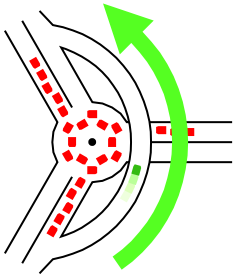


Overpass
API

Present and Future of the OSM data model from the Overpass API perspective

Roland Olbricht

at SOTM 2018 in Milano



Overview

Technology stack of OpenStreetMap

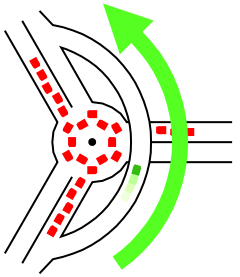
osm.org map, Taginfo, Nominatim, iD, JOSM

Presets, Styles, etc., in general: Processing rules

Tagging schemes

Data model, Database Schemes, File formats

Database engines, XML, Compression



Overview

Technology stack of OpenStreetMap

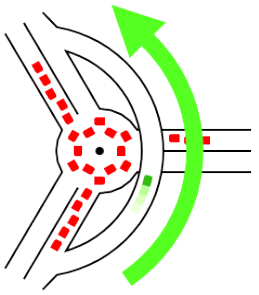
osm.org map, Taginfo, Nominatim, iD, JOSM

Presets, Styles, etc., in general: Processing rules

Tagging schemes

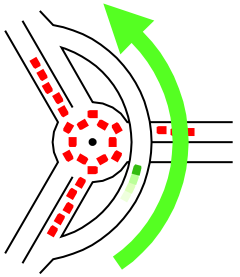
Data model, Database Schemes, File formats

Database engines, XML, Compression



Overpass
API

Geometry for Ways



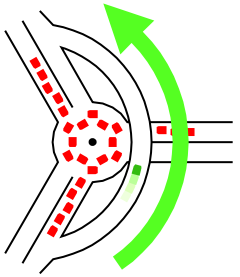
Geometry for Ways

Problem: no geometry on ways

Drawing, filtering (rendering, routing)
all require geometry.

Resolving for Planet.osm is expensive:
- often exceeds available RAM
- more than half of the runtime

Much worse when dealing with full-history OSM data.



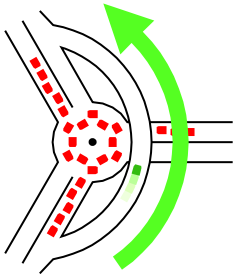
Geometry for Ways

We even do not see geometry changes in version numbers.

„Way 2.3“ people vs. ***Way version renumberers***

vs.

Timestamp people



Geometry for Ways

We even do not see geometry changes in version numbers.

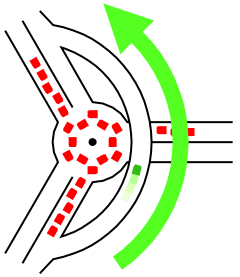
„Way 2.3“ people vs. ***Way version renumberers***

vs.

Timestamp people

(no agreement ever)

„Why cannot I get the *unique* geometry of a way‘s version“?



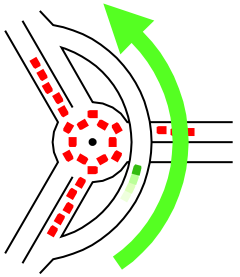
Geometry for Ways

Problem: no geometry on ways

Drawing, filtering (rendering, routing)
all require geometry.

Resolving for Planet.osm is expensive:
- often exceeds available RAM
- more than half of the runtime

Much worse when dealing with full-history OSM data.



Geometry for Ways

Partial solution: *out geom*

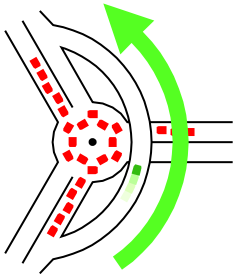
OSM Main DB

↓

Tool that adds geometry

↓

Data consumer



Geometry for Ways

Partial solution: *out geom*

Solved:
- Computing effort for consumers

Not solved:
- File sizes
- Make sense of way history

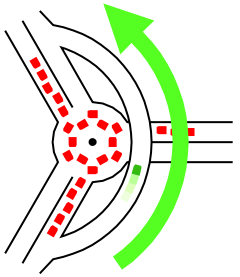
OSM Main DB

↓

Tool that adds geometry

↓

Data consumer



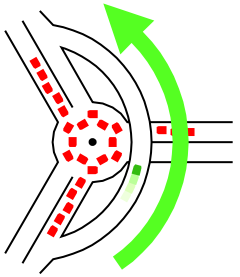
Geometry for Ways

Real solution:
geometry for ways in the Main DB?

Problem: Redundancy
Ways can get inconsistent if node changes without way.
Ways can get inconsistent if adjacent way changes.
~> Remove node id refs from way

Before: Moving the node implicitly moved the way
- good for: intentionally glued ways
- bad for: unintentionally glued ways, filtered editing

After: Moving the node separates the node from the way

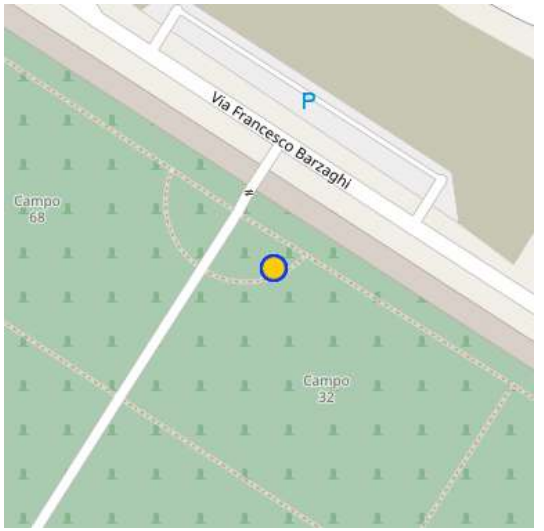


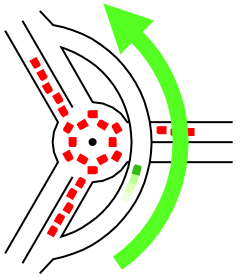
Geometry for Ways

Real solution:
geometry for ways in the Main DB, remove node id refs from way

Problem: Topology
What if multiple nodes are in the exact same place?

Does this happen at all?
- Rarely, but yes (0.01% - 0.05%)
- Almost all cases are errors
  (e.g. nodes 5578163459 – 5578163466)
- similar frequent to turn restrictions





Geometry for Ways

Data model change: Variant 1

Node:

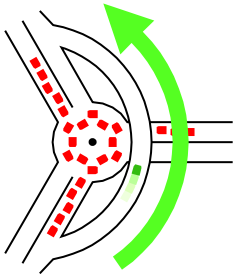
```
1      uint64 id
1       int32 lat
1       int32 lon
0..n  { string k,
          string v }
metadata
```

Way:

```
1      uint64 id
2..n  { int32 lat,
          int32 lon,
          uint64? ref }
0..n  { string k,
          string v }
metadata
```

Relation:

```
1      uint64 id
0..n  { enum type,
          uint64 ref,
          string role }
0..n  { string k,
          string v }
metadata
```



Geometry for Ways

Data model change: Variant 2

Node:

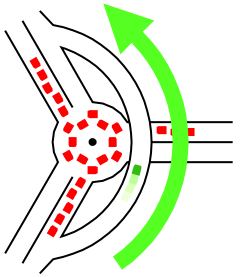
1 uint64 id
1 uint32? loc_id
1 int32 lat
1 int32 lon
0..n { string k,
string v }
metadata

Way:

1 uint64 id
2..n { int32 lat,
int32 lon,
uint32? loc_ref
~~uint64 ref~~ }
0..n { string k,
string v }
metadata

Relation:

1 uint64 id
0..n { enum type,
uint64 ref,
string role }
0..n { string k,
string v }
metadata



Geometry for Ways

Data model change: Variant 3 (force-split ways)

Node:

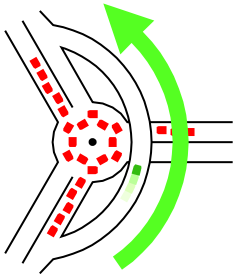
1 uint64 id
1 uint32? loc_id
1 int32 lat
1 int32 lon
0..n { string k,
string v }
metadata

Way:

1 uint64 id
1 uint64? from_loc_ref
1 uint64? to_loc_ref
2..n { int32 lat,
int32 lon
~~uint64 ref~~ }
0..n { string k,
string v }
metadata

Relation:

1 uint64 id
0..n { enum type,
uint64 ref,
string role }
0..n { string k,
string v }
metadata



Geometry for Ways

Data model change: Variant 4

Node:

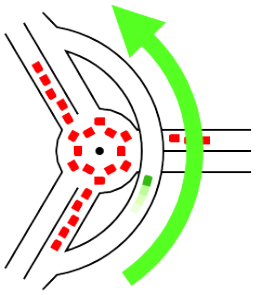
1 uint64 id
0..n { uint64 way_ref,
uint32 pos }
1 int32 lat
1 int32 lon
0..n { string k,
string v }
metadata

Way:

1 uint64 id
2..n { int32 lat,
int32 lon
~~uint64 ref~~ }
0..n { string k,
string v }
metadata

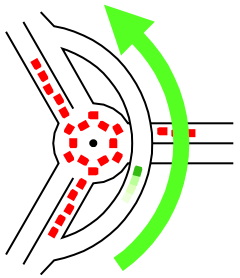
Relation:

1 uint64 id
0..n { enum type,
uint64 ref,
string role }
0..n { string k,
string v }
metadata



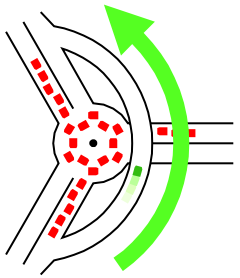
Overpass
API

A Migration Path



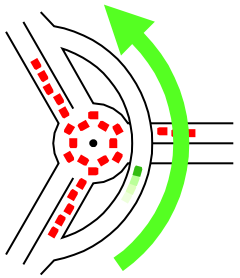
A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	●				
Closed vs. open ways					
Bounding boxes on relations					
...					



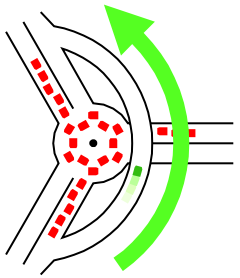
A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	→				
Closed vs. open ways					
Bounding boxes on relations					
...					



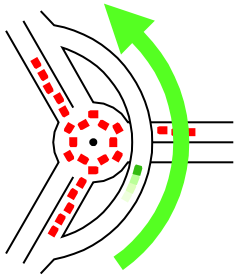
A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	→	→			
Closed vs. open ways					
Bounding boxes on relations					
...					



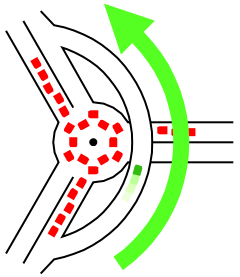
A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	→	→	→	→	
Closed vs. open ways					
Bounding boxes on relations					
...					



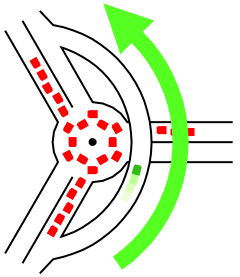
A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	→	→	→	→	→
Closed vs. open ways					
Bounding boxes on relations					
...					



A Migration Plan

	Spec and Proof of concept	Consumers	Read by Editors	Written by Editors	Main API support
Geometry for Ways	→	→	→	→	→
Closed vs. open ways	→	→	→	→	→
Bounding boxes on relations	→	→	→	→	→
...					



A Migration Plan

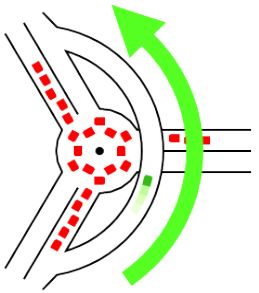
Specify the Format
and Implement Converters

Make Back-To-Back Comparisons
of Consuming Tools

Let Editing Software Read the Format

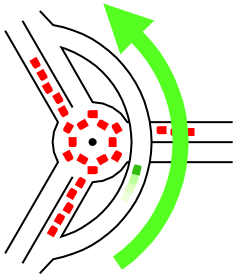
Implement Main API Calls
and Write From Editing Software

Move Old Format to Compatibility Layer



Overpass
API

Other Issues



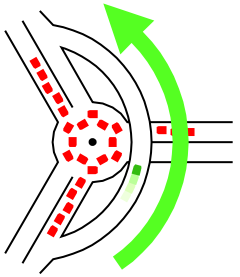
Other Issues

Areas

We can:
- represent any shape
- share boundaries
- edit single elements

We cannot:
- check for self-Intersection
- easily figure out is_in
- easily edit huge areas

Local editing?
Topology?
Coastline approach?

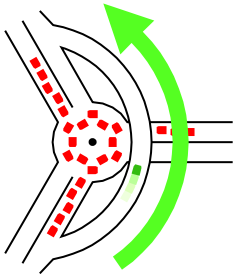


Other Issues

Size of Relations

The typical town center pulls many large relations.

How to present them concise?



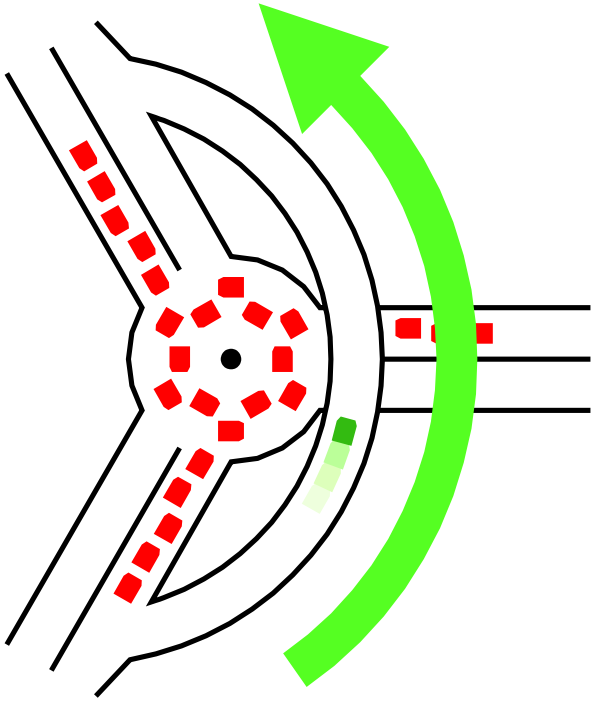
Other Issues

Relations and Way Splitting

Too many versions,
also on other relations

Avoid splitting for Routes?

What about large boundaries?



Thank you for your attention

Back to your questions