Rapid Visualization Development based on Visual Programming

Developing a Visualization Prototyping Language (DAEV)

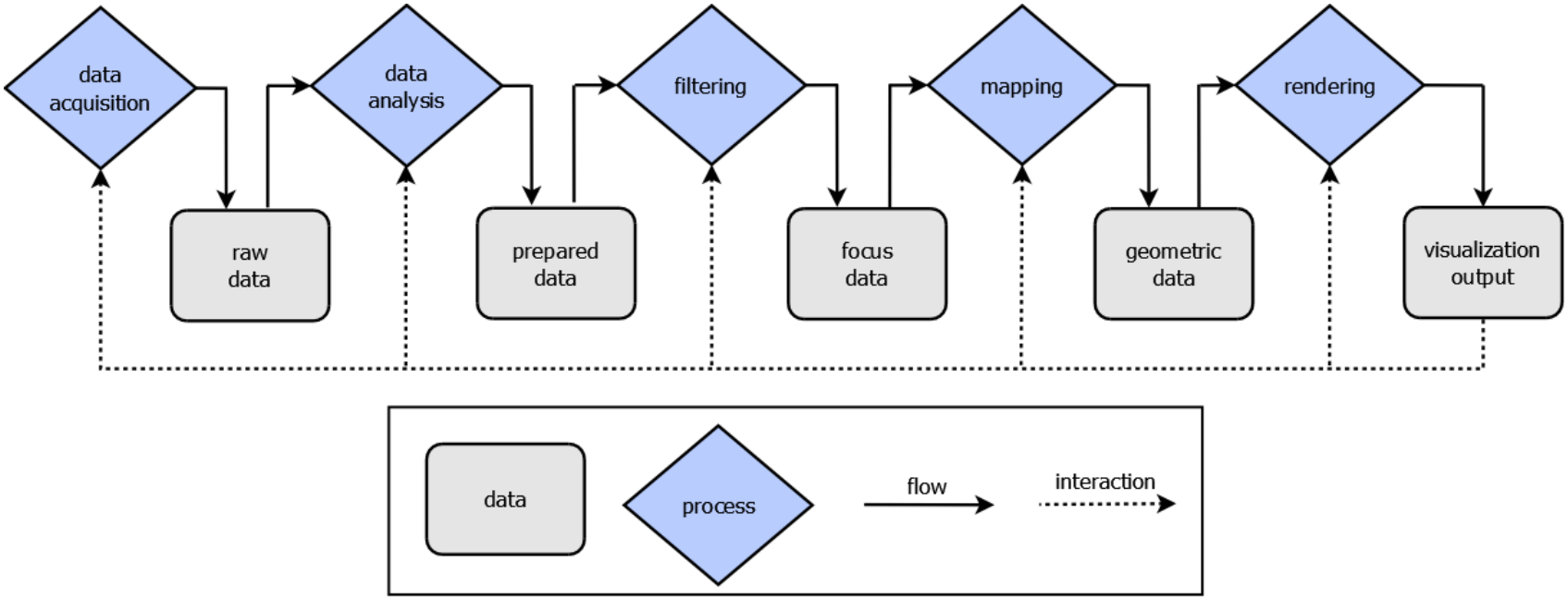
Benedikt Stehno

Goal

To develop a multi (cross) platform rapid visualization prototyping language (***OpenInsightExplorer***)

- easy to use (non programmers)
- extendable
- open source
- automatic parallelization
- supports hardware acceleration (GPU)
- custom data types

Visualization Pipeline



Idea

combine the powers of ...

- ***a modular approach***
- ***visual programming***
- ***dataflow programming***

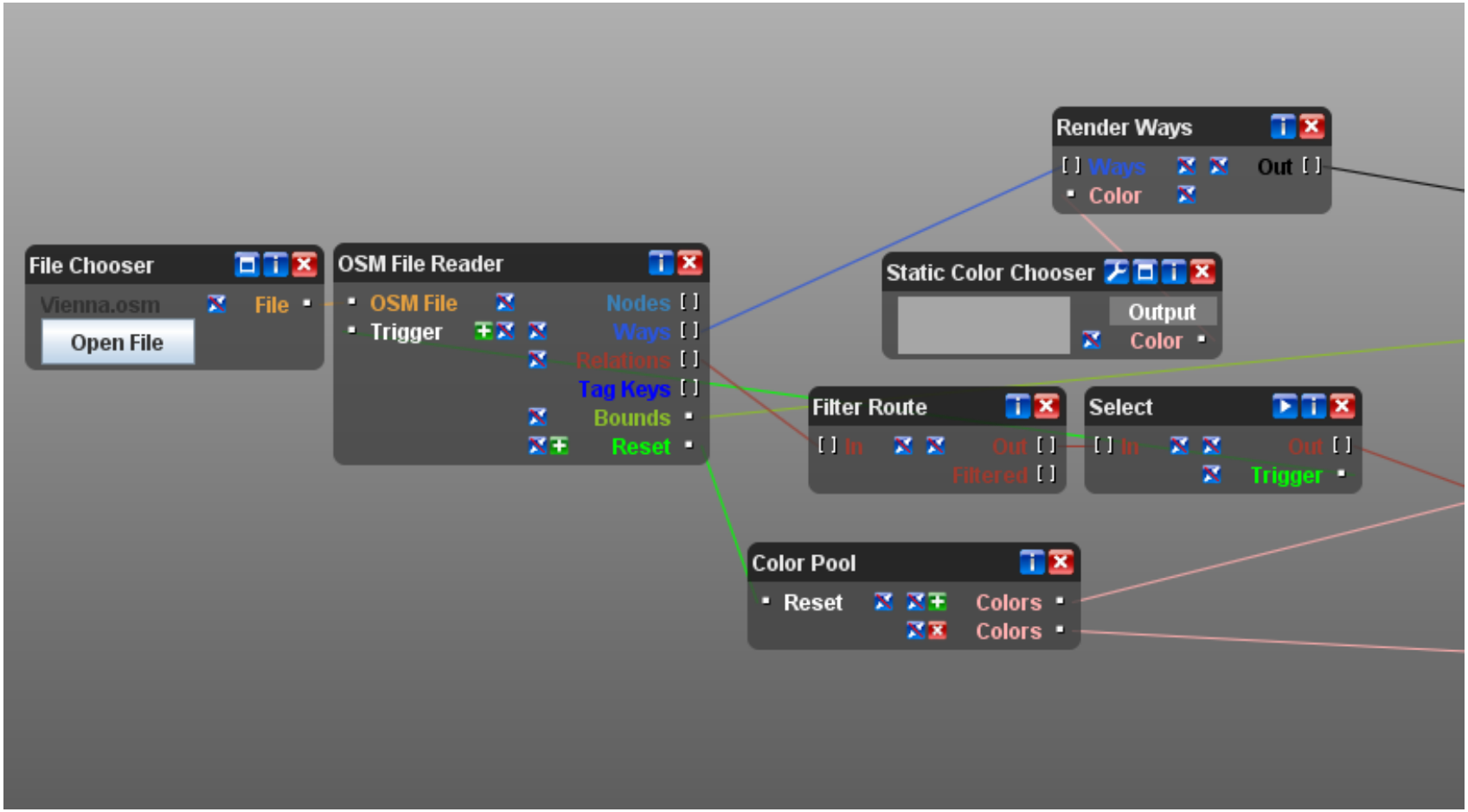
... to a ***Dataflow Visual Programming Language (DFVPL)***

Idea

Each module (or ***patch***)

- represents a stage of the visualization pipeline
- works as an independent ***black box***
- communicates with other modules (over ***ports***)
- is arranged and connected in a visual editor (***visual programming***)
- is executed as soon as data is available (***dataflow execution***)

Programming becomes the task to visually connect a custom pipeline together


Render Ways
Ways
Out
Color
File Chooser
Vienna.osm
File
Open File
OSM File Reader
OSM File
Trigger
Nodes
Ways
Relations
Tag Keys
Bounds
Reset
Static Color Chooser
Output
Color
Filter Route
In
Out
Filtered
Select
In
Out
Trigger
Color Pool
Reset
Colors
Colors

Features

Features of OpenInsightExplorer

- open source and platform independence
- automatic parallelization
- custom data types
- java classes as data type
- data streams
- type-safety
- easy patch / library installation

Features

Unique features of OpenInsightExplorer

- easy to develop modules (***Patch*** interface)
- delegating Patches
- patch GUIs
- ***Growing Ports***
- ***Port Trees***
- ***Generic Ports***

Features

Patch Interface

Only a small Java interface must be implement to write a patch. Similar to the Java Applet interface (***run()***, ***init()***, ***stop()***,...).

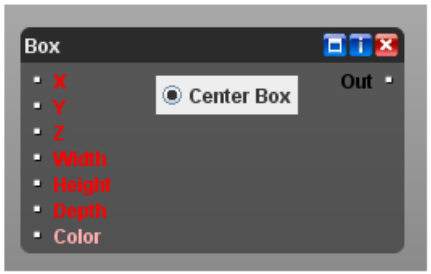
Delegating Patches

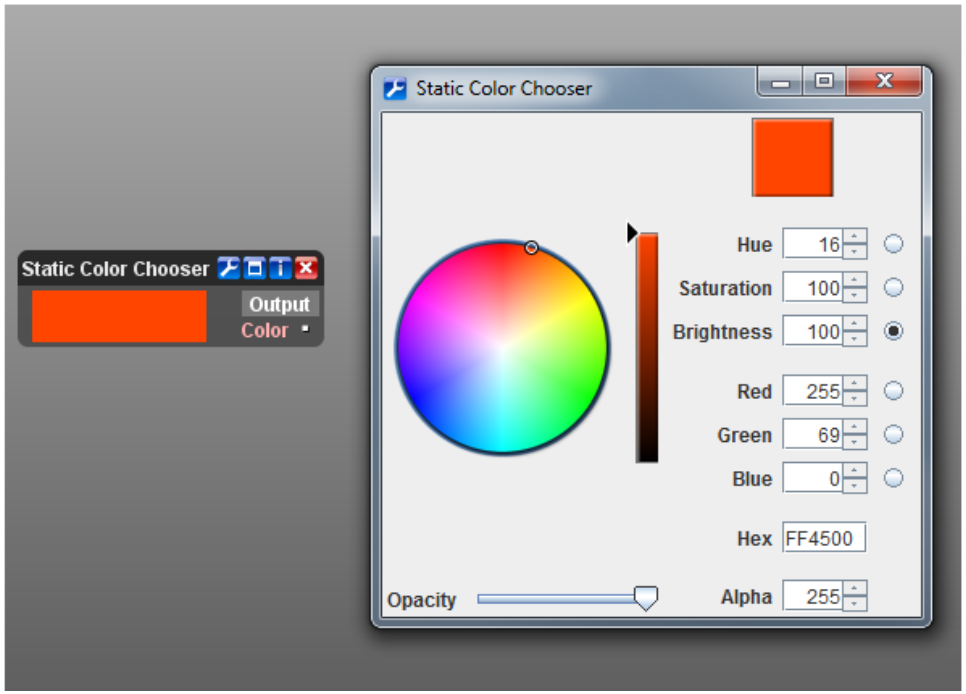
Patches can sent functionality (Methods, ...). Allows to split and combine tasks between patches.

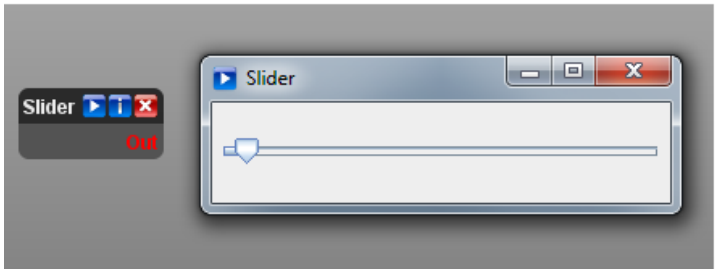
Features

Patch GUIs

Multiple GUIs for a patch possible.







Features

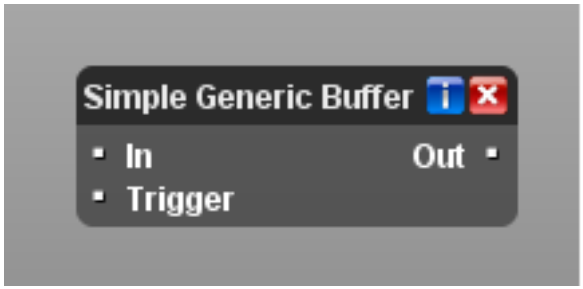
Growing Ports and ***Port Trees***

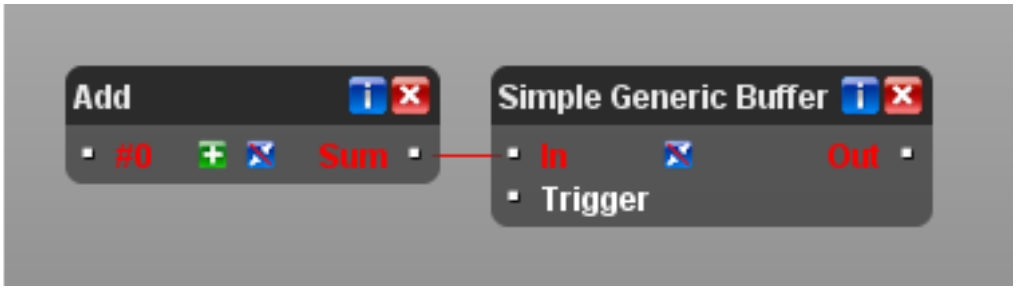
Dynamically add / remove ports to a patch.

Features

Generic Ports

Dynamically adaption to a data type.





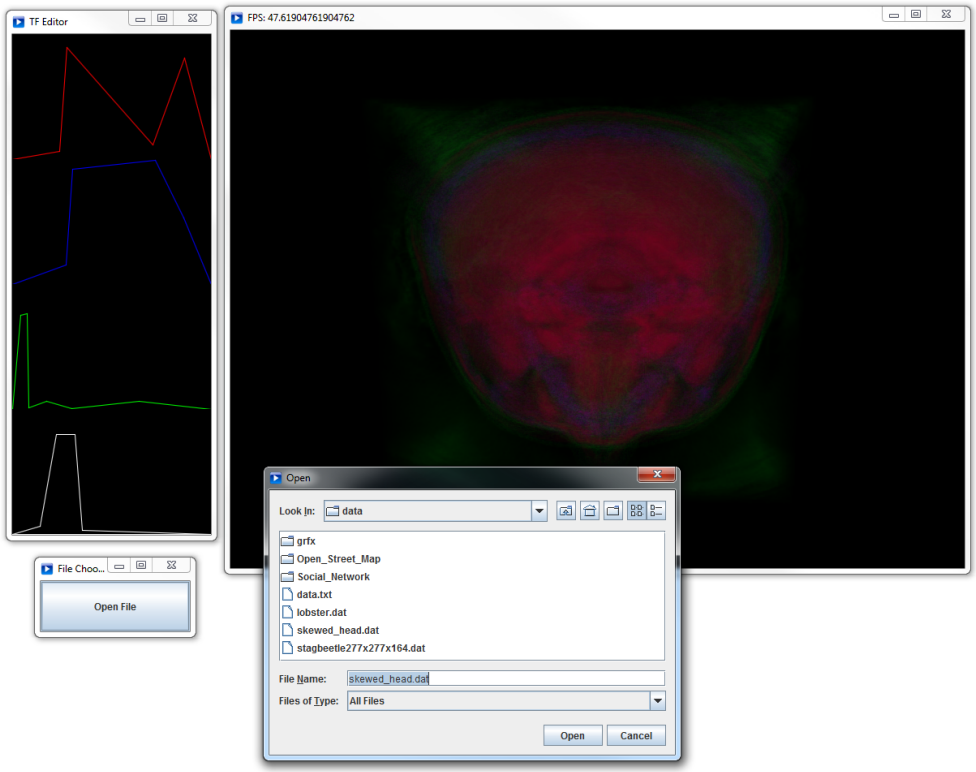
Evaluation

OpenInsightExplorer was evaluated by implementing example visualizations.

- hardware accelerated (GPU) volume renderer
- collection of OpenStreetMap visualizations

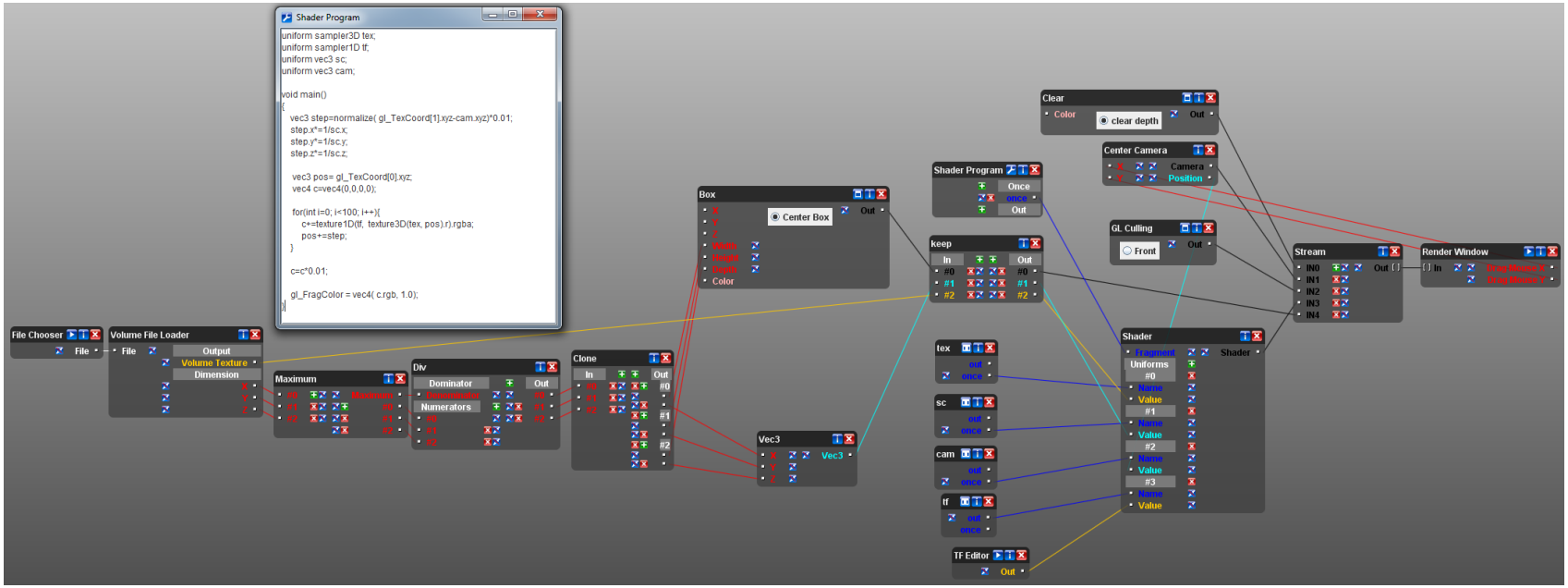
Evaluation

Volume rendering



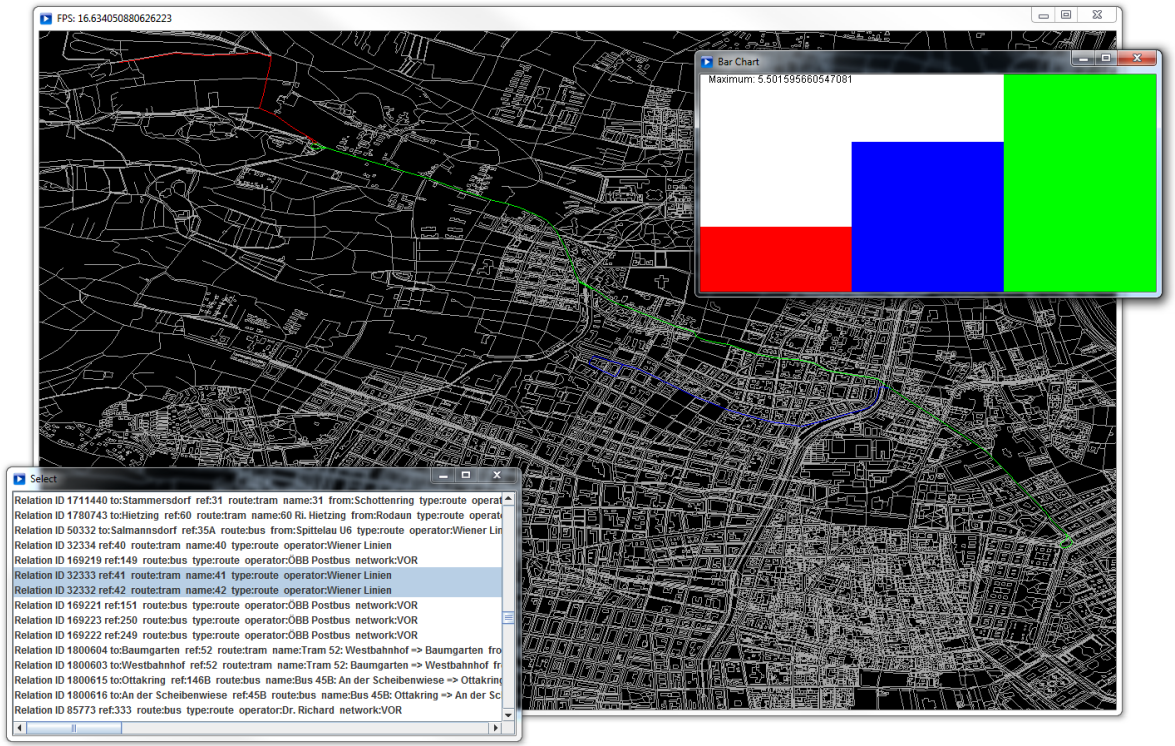
Evaluation

Volume rendering



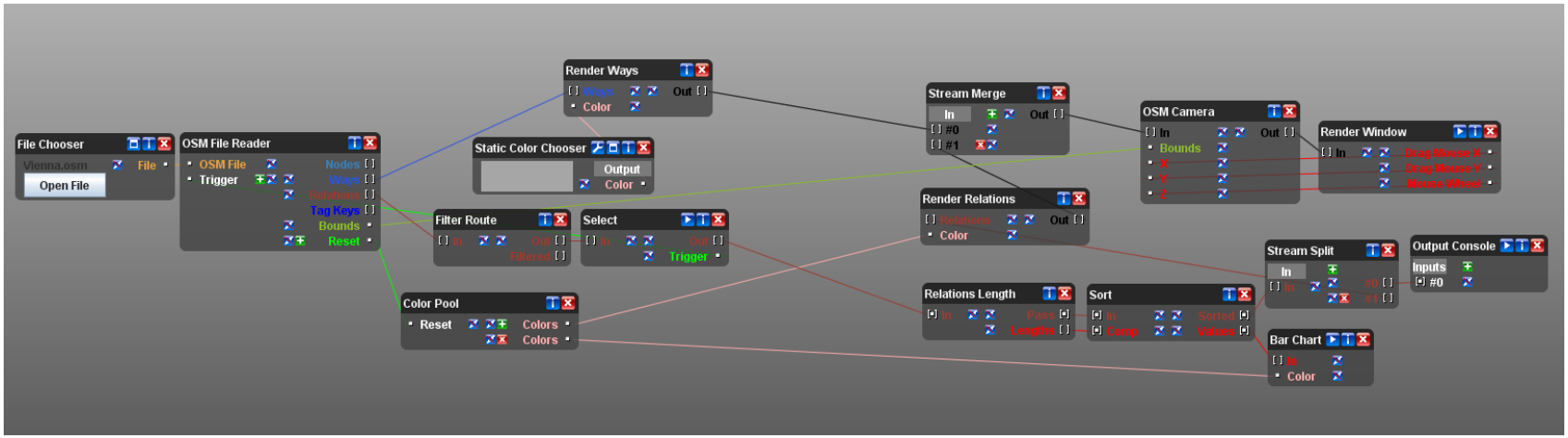
Evaluation

OpenStreetMap



Features

OpenStreetMap



Conclusion

Conclusion

- Still too complex for none programmers
- State-of-the-art DFVPL features are missing (debugging, structured programming, ...)
- ***Arbitrary*** synchronization/execution between patches
- Execution overhead increases fast (only good for prototyping)
- ***Generic Ports*** and ***Growing Ports*** improve the reuse of patches a lot