MATSim

Marcel Rieser
Simunto GmbH

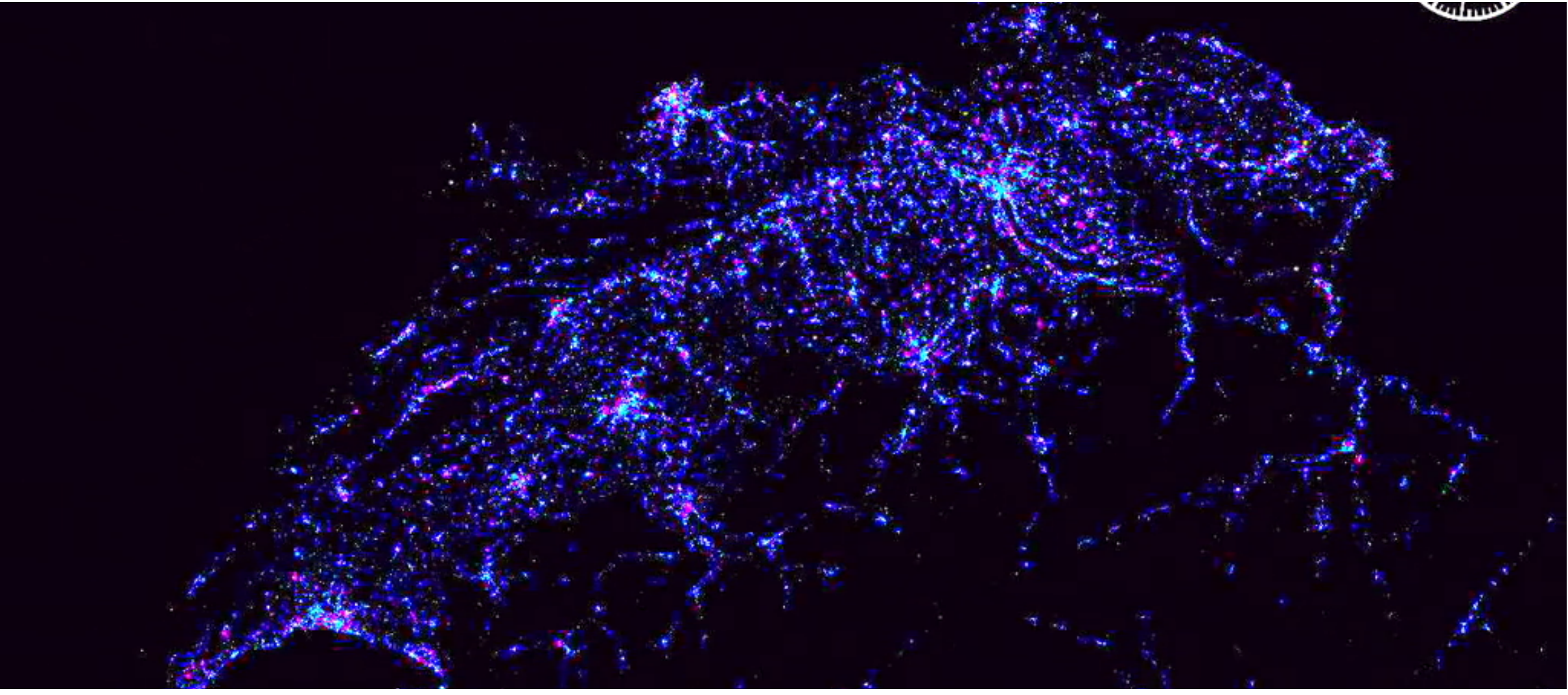
MATSim Training EIfER
July, 2019

MATSim

Multi-**A**gent **T**ransport **Sim**ulation

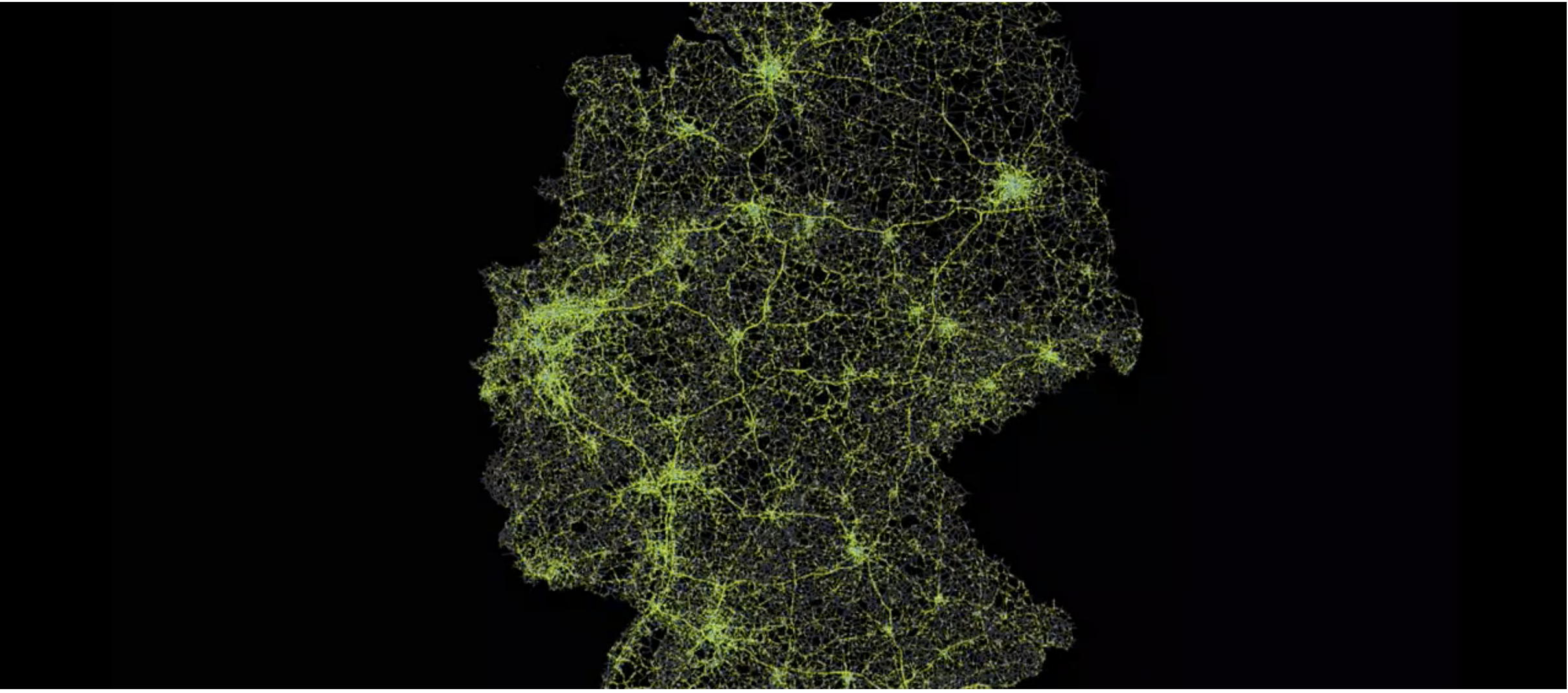
MATSim is an open-source framework for implementing large-scale agent-based transport simulations.

Large Scale



https://vimeo.com/matsim/switzerland-ivt-2018

Large Scale

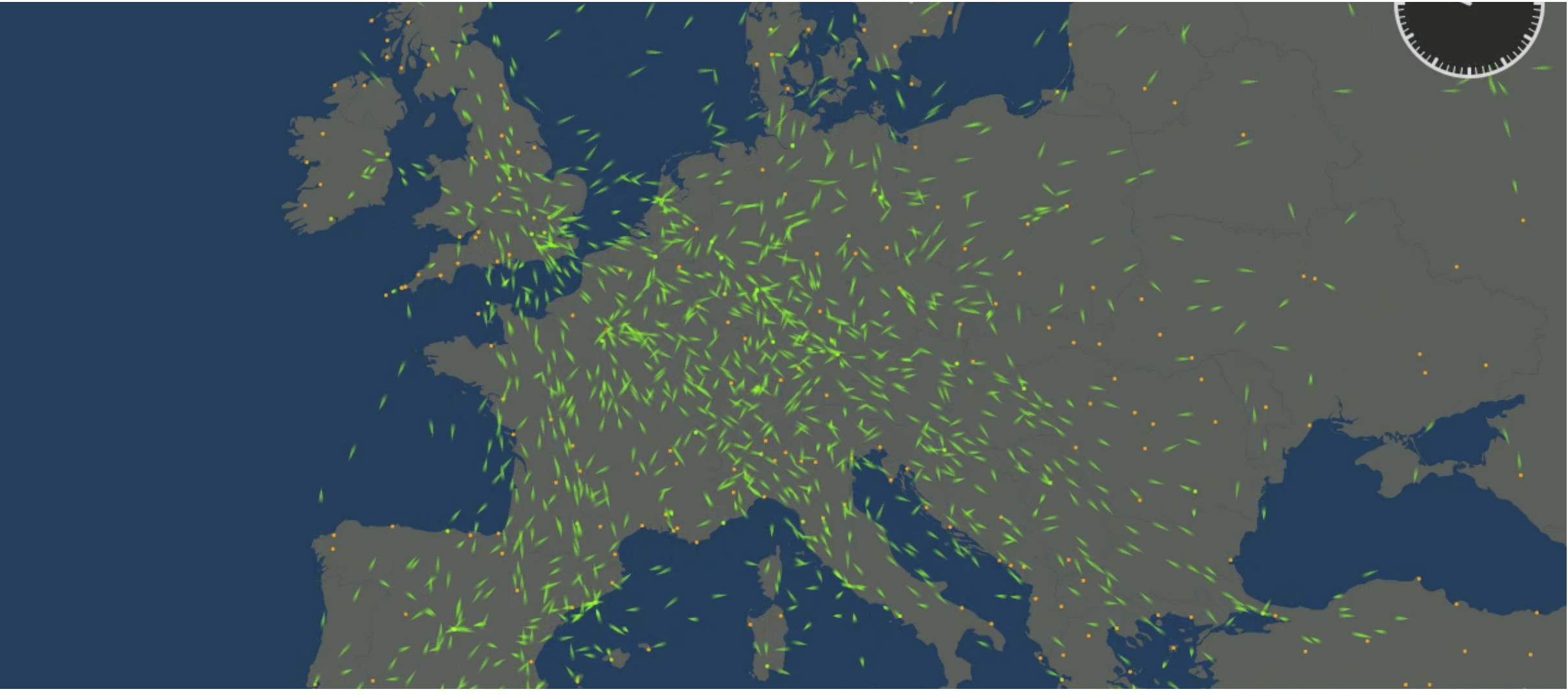


https://www.matsim.org/gallery/germany

Agent-Based

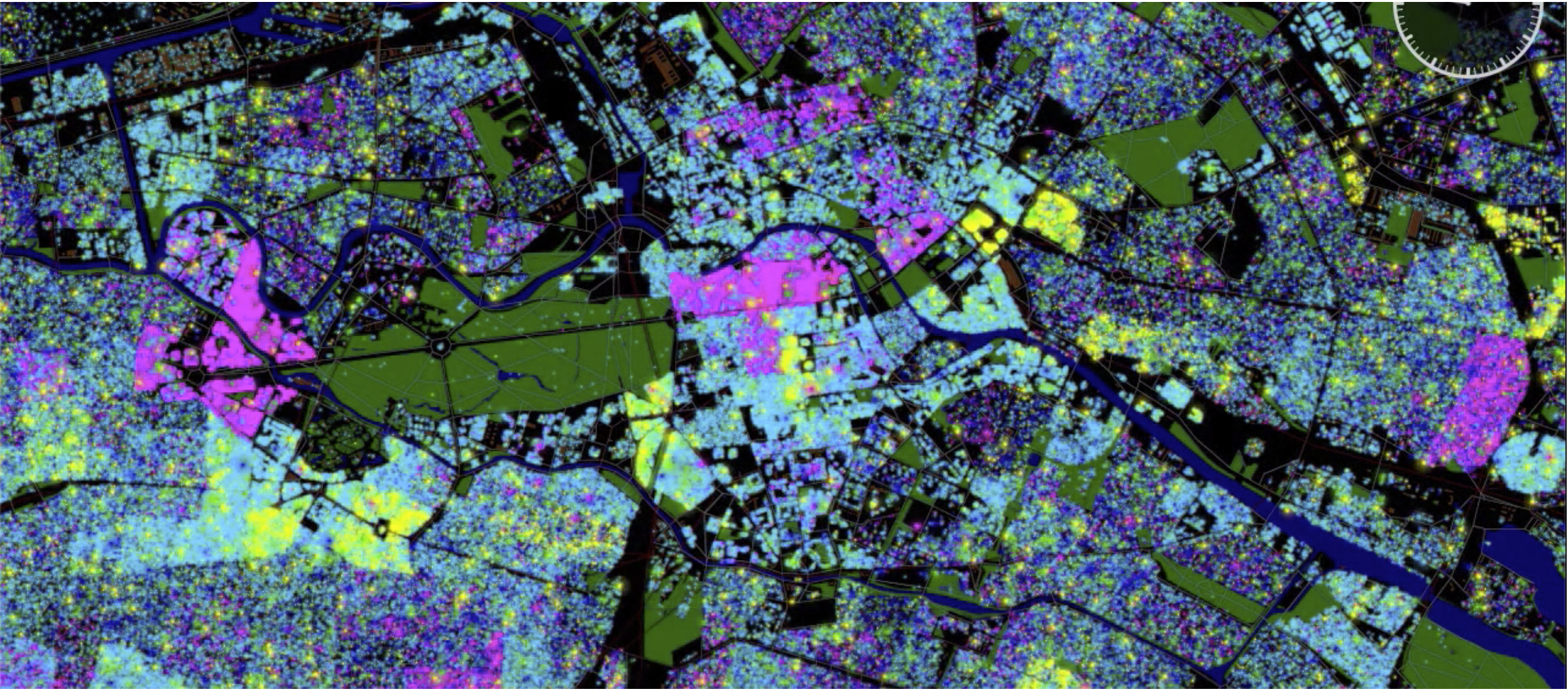
https://www.matsim.org/gallery/poznan

Transport Simulation



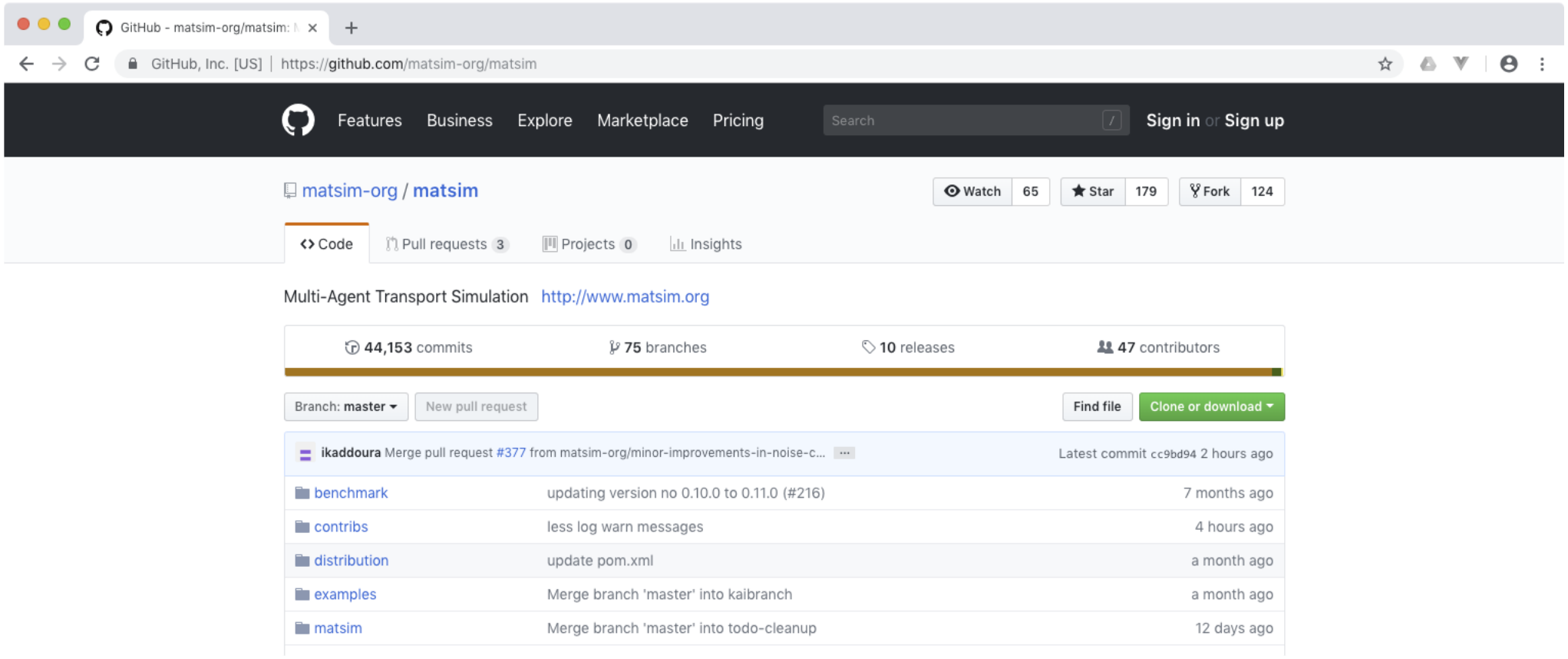
https://www.matsim.org/gallery/air-traffic

Transport **Simulation**



https://www.youtube.com/watch?v=rWTFg1UkZTc

Open-Source



https://github.com/matsim-org/

Who am I?

Marcel Rieser

since 2018: **Simunto GmbH**: Providing tools and support for MATSim

2017-2018: **SBB AG** (Swiss Federal Railway): MATSim Expert

2010-2017: **Senozon AG**: MATSim models, extensions, tools

2006-2009: **Technical University of Berlin**: PhD, Adding Transit to MATSim

2000-2005: **ETH Zurich**: MSc Computer Science

Photography, Travelling, Geocaching, Family

Who are you?

- Who knows a bit about MATSim?
- Who knows about Transport Planning?
- Who has programming skills?
  - What languages?
- Who has data science experience?
- Who is into statistics?
- Who has GIS experience?

This course

- Conceptual understanding of MATSim
- Understand the data requirements of MATSim
- Create a simple MATSim model
- Build and run a simulation of a small region
- Analyze the simulation outcome
- Extend the model with additional functionality

Agenda

	Monday	Tuesday	Wednesday	Thursday
09:00 – 10:30	Introduction, Setup, First Simulation	Population	Events, EventHandlers	EV
11:00 – 12:30	MATSim Model	Population	Analysis	EV
	lunch	lunch	lunch	lunch
13:30 – 15:00	MATSim Controler, API	Public Transport	Model Calibration	Complex Application, Reserve
15:30 – 17:00	Network, QSim	Replanning, Scoring	Emissions	Complex Application, Reserve

Course Material

Course material will be available online:

simunto.com/matsim/tutorials/eifer2019

Introduction to MATSim

— Introduction to MATSim —

History of MATSim

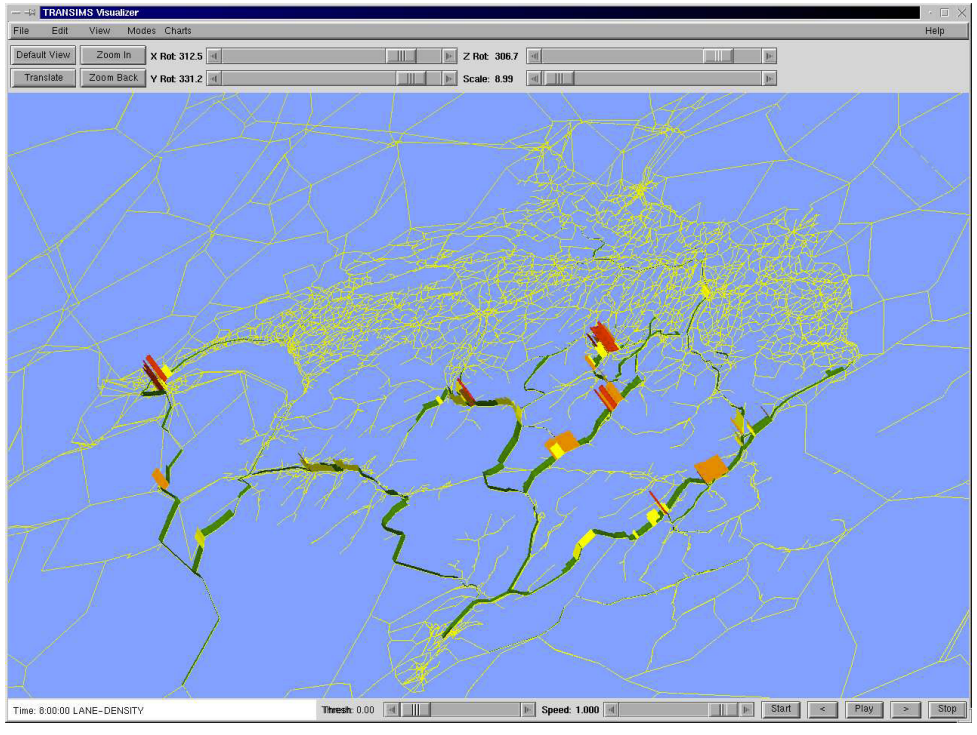
TRANSIMS

A set of integrated tools to simulate regional traffic.

Developed at Los Alamos National Laboratory by a team of researchers including Kai Nagel during the 1990s.

Written in C/C++

Used cellular automata for road traffic modelling.



MATSim in C++

1999: Kai Nagel changes to ETH Zürich

Could not continue to use TRANSIMS due to export restrictions

MATSim was started as lightweight, faster alternative to TRANSIMS

- Written in C/C++
- Uses Queue model instead of cellular automata
- Uses hierarchical XML to store data
- Combination of many small executables
- In-memory object database for fast simulation
- Only simulates private car traffic
- Runs on cluster

MATSim in Java

2004: Kai Nagel changes to TU Berlin
2004: Kay Axhausen (ETH) joins development efforts

2006: Re-implementation in Java
Designed as single-process architecture
(64-bit CPUs, less file I/O)

Since 2007: adding more and more features:

- mode choice
- teleportation
- public transport
- road pricing
- car counts
- emissions
- evacuation
- drt
- ...

MATSim Community

Original development (and use) at ETH Zurich and TU Berlin

2006: First MATSim Seminar (ETHZ + TUB)

2007, 2008: first external users (University of Pretoria, University of Toronto, KIT, ...)

2009: first MATSim User Meeting, MATSim Developer Meeting

Yearly Developer Meetings (2018: 19 participants from 7 institutions)

Yearly User Meetings (2020: Warshaw, Poland)

In planning: MATSim Foundation, MATSim e.V.

— Introduction to MATSim —

MATSim Today

MATSim Core Development

VSP, TU Berlin

Transport Systems Planning and Transport Telematics (Verkehrssystemplanung und Verkehrstelematik)
led by Prof. Dr. Kai Nagel
www.vsp.tu-berlin.de

IVT, ETH Zurich

Institute for Transport Planning and Systems (Institut für Verkehrsplanung und Transportsysteme)
led by Prof. Dr. Kay Axhausen
ivt.ethz.ch

Simunto

Marcel Rieser
simunto.com

MATSim Contributors (besides Core Developers)

FCL (Future Cities Lab, ETH Singapore)

Pieter Fourie, Sergio Ordonez
www.fcl.ethz.ch

University of Pretoria

Johan W. Joubert
www.up.ac.za/en/industrial-and-systems-engineering/

TU Munich

Nico Kühnel
www.msm.bgu.tum.de/nk/

SBB (Swiss Federal Railways)

Personenverkehr - Verkehrsplanung

Senozon

senozon.com

list in no particular order, and likely I forgot to mention some, sorry about that!

MATSim Contributors

By now, over 50 PhD students have contributed to MATSim in Zurich, Berlin and Singapore.
Additional contributors in other institutions.

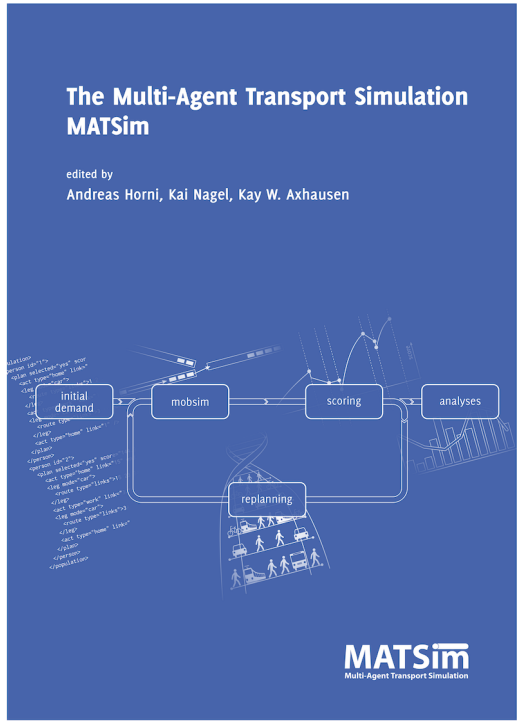
The Java implementation is more than 10 years old,
the original concepts of MATSim are about 20 years old.

MATSim is historically grown, by many contributors.
Several settings and names may seem inconsistent nowadays, but have historical and personal reasons to be so.

MATSim Resources

Learning MATSim

- This course 😉
- MATSim Book matsim.org/the-book
- VSP Tutorial matsim.org/docs/tutorials/general
- Code examples: github.com/matsim-org/matsim-code-examples



Getting Help

- Q&A: matsim.org/faq (don't forget the logfile!)

Stay up to date

- MATSim Website matsim.org
- MATSim Newsletter: matsim.org/docs/#newsletter

— Introduction to MATSim —

Overview of MATSim

MATSim Terminology

Agent-Based Transport Simulation

Agent: A synthetic person, part of a synthetic population.

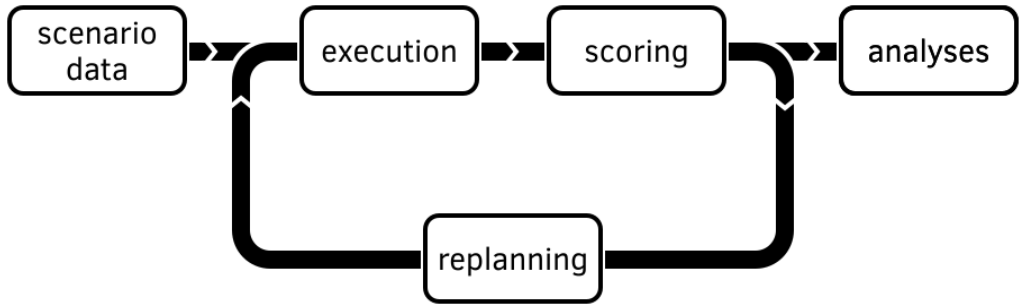
Plan: the intention of an agent, typically for one day.
"going to work at 07:30, go shopping on the way home at 17:00, be home at 17:45".

Each agent needs at least one plan.

Score: utility of a plan after it was simulated.

Scenario, Model: several datasets and parameters describing infrastructure (supply) and demand in a region.
"Scenario" and "model" are often used without differentiation in MATSim's context.

General Design of Simulation System ("MATSim Loop")



scenario data: description of infrastructure (road network, transit schedule, ...) and agents.

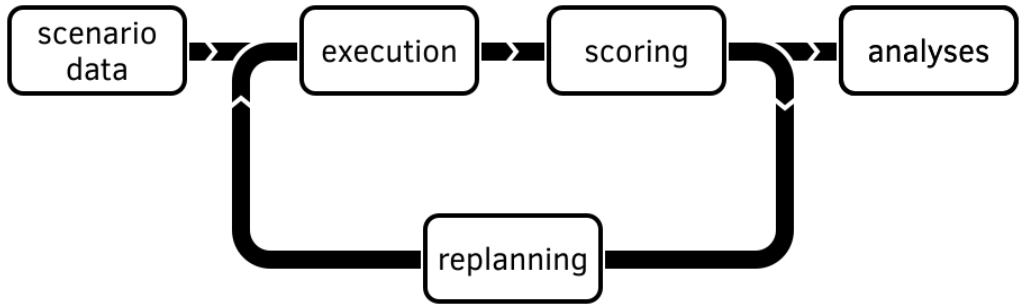
execution: Agents' travel plans are executed in parallel.
A.k.a.: mobsim (mobility simulation), synthetic reality, network loading, traffic flow simulation

scoring: each executed plan obtains a score.

replanning: some agents change plans or create new ones.

analyses: traffic volumes, average speeds, utility changes, emissions, accessibility, ...

General Design of Simulation System



Iterations: execution, scoring and replanning are iteratively performed.

Optimization: each agent tries to optimize its day.

Evolutionary algorithm: each agent has a set of plans ("choice set"), adding new plans (replanning), removing bad ones after a while.

Co-Evolutionary algorithm: If a plan performs good or bad also depends on the plans of all the other agents.

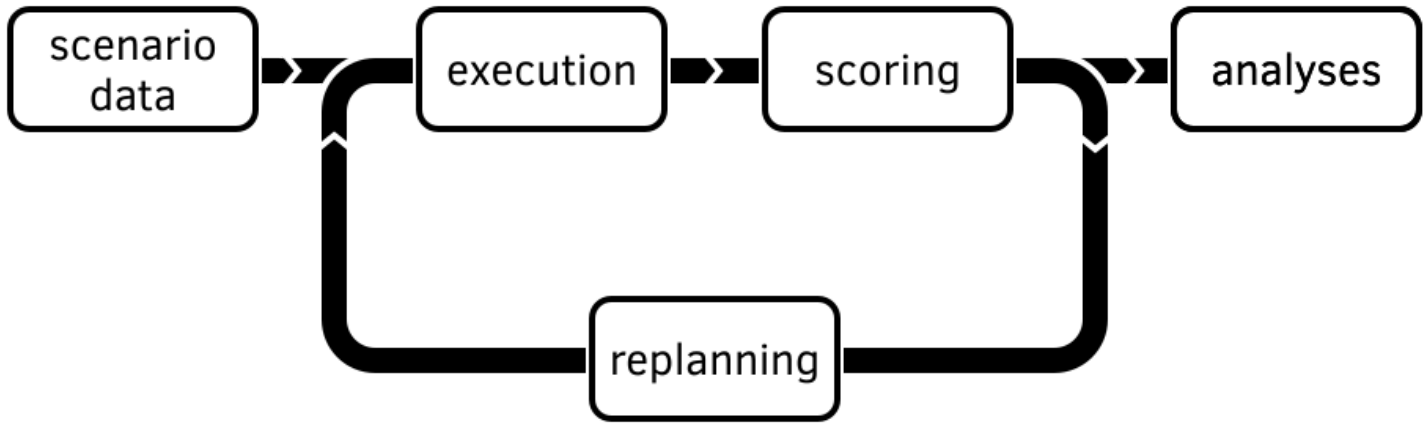
Nash-Equilibrium: Each agent tries to optimize its plan egoistically. (≠ system optimum)

General Design of Simulation System

MATSim is an open-source framework for implementing
large-scale agent-based transport simulations.

MATSim is a travel demand optimization process using a co-evolutionary algorithm.

A closer look



Scenario Data

Minimally required:

- Network (network.xml)
- Demand (population.xml)
- Configuration (config.xml)

Additional data sets:

- Public Transport data (transitSchedule.xml, transitVehicles.xml)
- Facilities (facilities.xml)
- Comparison with count stations (counts.xml)
- Road Pricing (roadpricing.xml)
- Lanes (lanes.xml)
- Signals, Traffic Lights
- ...

Scenario Data: Network

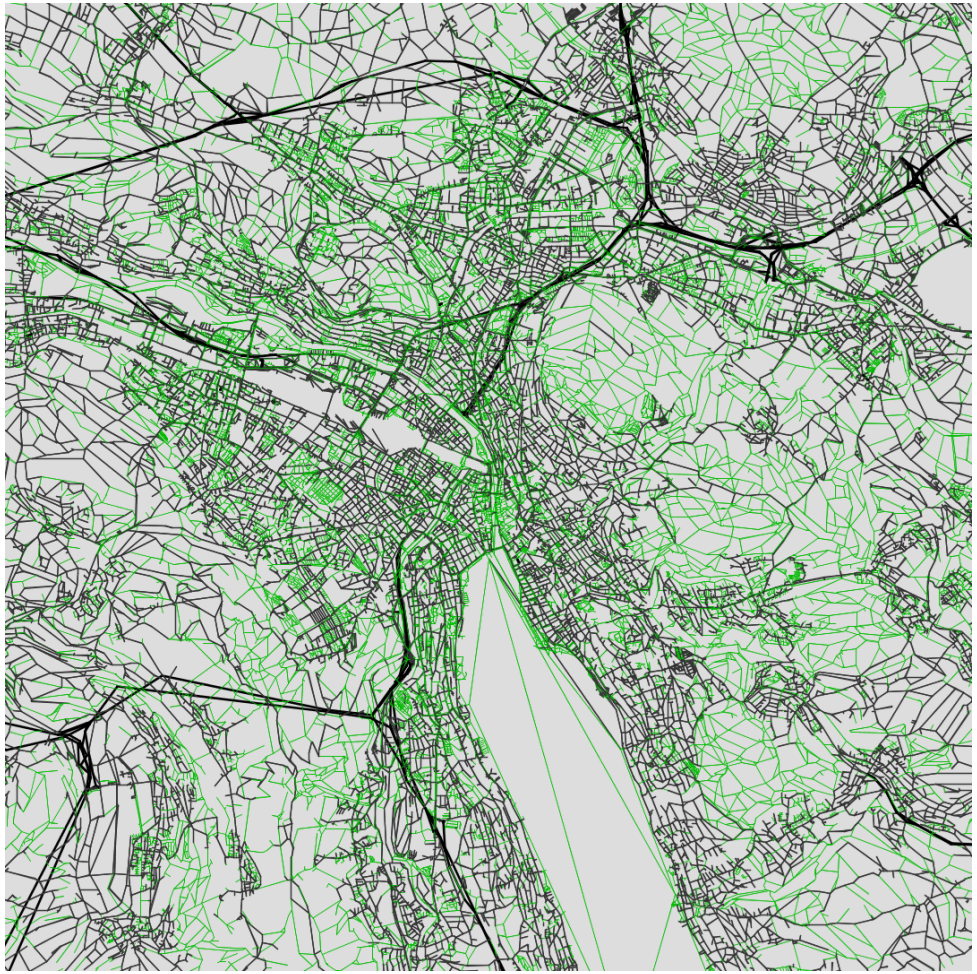
Describes a network of nodes and links along which agents/vehicles can move.

Multimodal: car, walk, bike, bus/train/...,

Attributes for road networks:

- length
- max. speed
- number of lanes
- flow capacity

Links in MATSim are uni-directional.



Network for car (black) and walk-only (green)

Scenario Data: network.xml

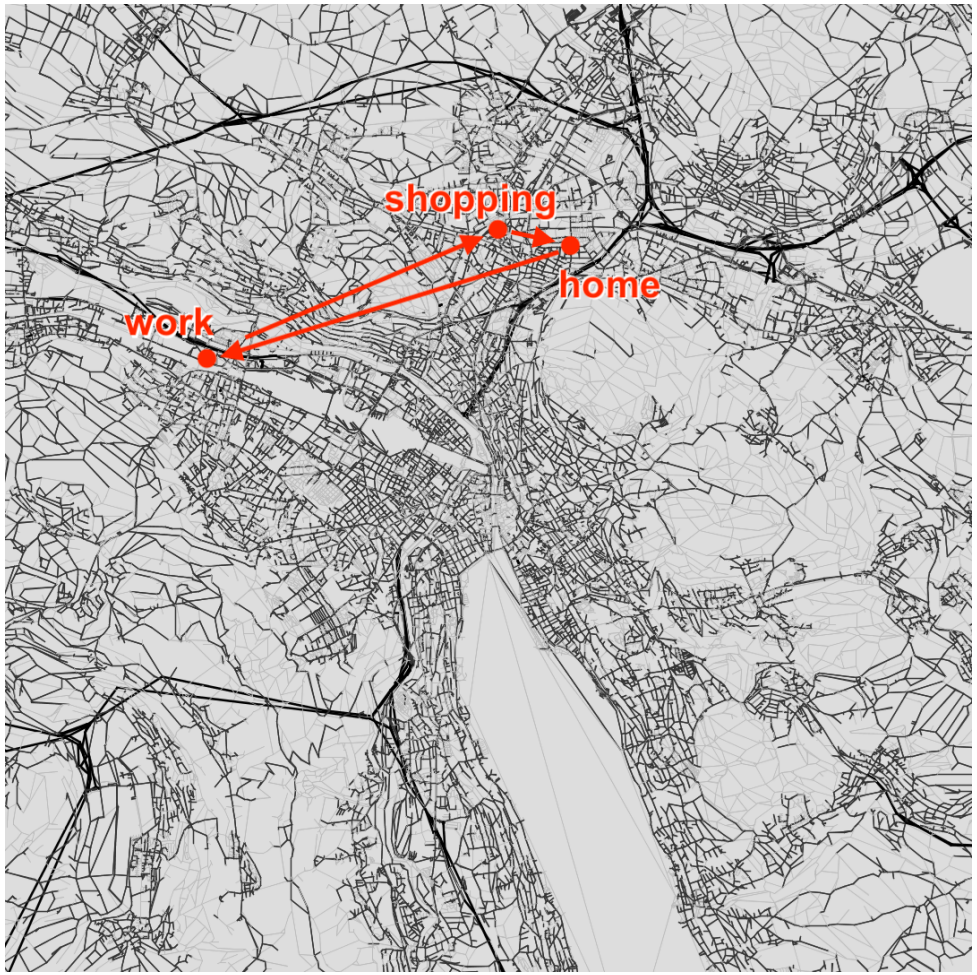
```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE network SYSTEM "http://www.matsim.org/files/dtd/network_v2.dtd">
<network>
  <nodes>
    <node id="1" x="2000" y="1000" />
    <node id="2" x="4000" y="1500" />
    <node id="3" x="3000" y="3000" />
    ...
  </nodes>
  <links capperiod="01:00:00">
    <link id="1" from="1" to="2" length="3000.0" capacity="1800" freespeed="13.88" permlanes="1" modes="car" />
    <link id="2" from="2" to="1" length="3000.0" capacity="1800" freespeed="13.88" permlanes="1" modes="car" />
    <link id="3" from="2" to="3" length="5000.0" capacity="3500" freespeed="22.22" permlanes="2" modes="car" />
    <link id="4" from="3" to="2" length="5000.0" capacity="1800" freespeed="22.22" permlanes="1" modes="car" />
    ...
  </links>
</network>
```

Scenario Data: Demand

Describes the agents and their plans.

Describes:

- Activity types
- Activity durations
- Activity locations
- Trip modes
- Trip routes



A single agent's plan

Scenario Data: population.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <attributes>
      <attribute name="employed" class="java.lang.Boolean" >true</attribute>
    </attributes>
    <plan score="140.00982053869416" selected="yes">
      <activity type="home" link="1112" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" dep_time="07:18:16" trav_time="00:01:40">
        <route type="links" start_link="1112" end_link="2223" trav_time="00:31:40" distance="14403.0">1112 1222 2223</r
      </leg>
      <activity type="work" link="2223" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="2223" end_link="1312" trav_time="00:27:32" distance="12573.0">2223 2313 1312</r
      </leg>
      <activity type="shopping" link="1312" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="1312" end_link="1112" trav_time="00:04:15" distance="3127.0">1312 1211 1112</ro
      </leg>
      <activity type="home" link="1112" x="890.0" y="685.0" />
    </plan>
    <plan ...>
      ...
  </person>
  ...
</population>
```

Scenario Data: population.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <attributes>
      <attribute name="employed" class="java.lang.Boolean" >true</attribute>
    </attributes>
    <plan score="140.00982053869416" selected="yes">
      <activity type="home" link="1112" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" dep_time="07:18:16" trav_time="00:01:40">
        <route type="links" start_link="1112" end_link="2223" trav_time="00:31:40" distance="14403.0">1112 1222 2223</r
      </leg>
      <activity type="work" link="2223" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="2223" end_link="1312" trav_time="00:27:32" distance="12573.0">2223 2313 1312</r
      </leg>
      <activity type="shopping" link="1312" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="1312" end_link="1112" trav_time="00:04:15" distance="3127.0">1312 1211 1112</ro
      </leg>
      <activity type="home" link="1112" x="890.0" y="685.0" />
    </plan>
    <plan ...>
      ...
  </person>
  ...
</population>
```

Scenario Data: population.xml

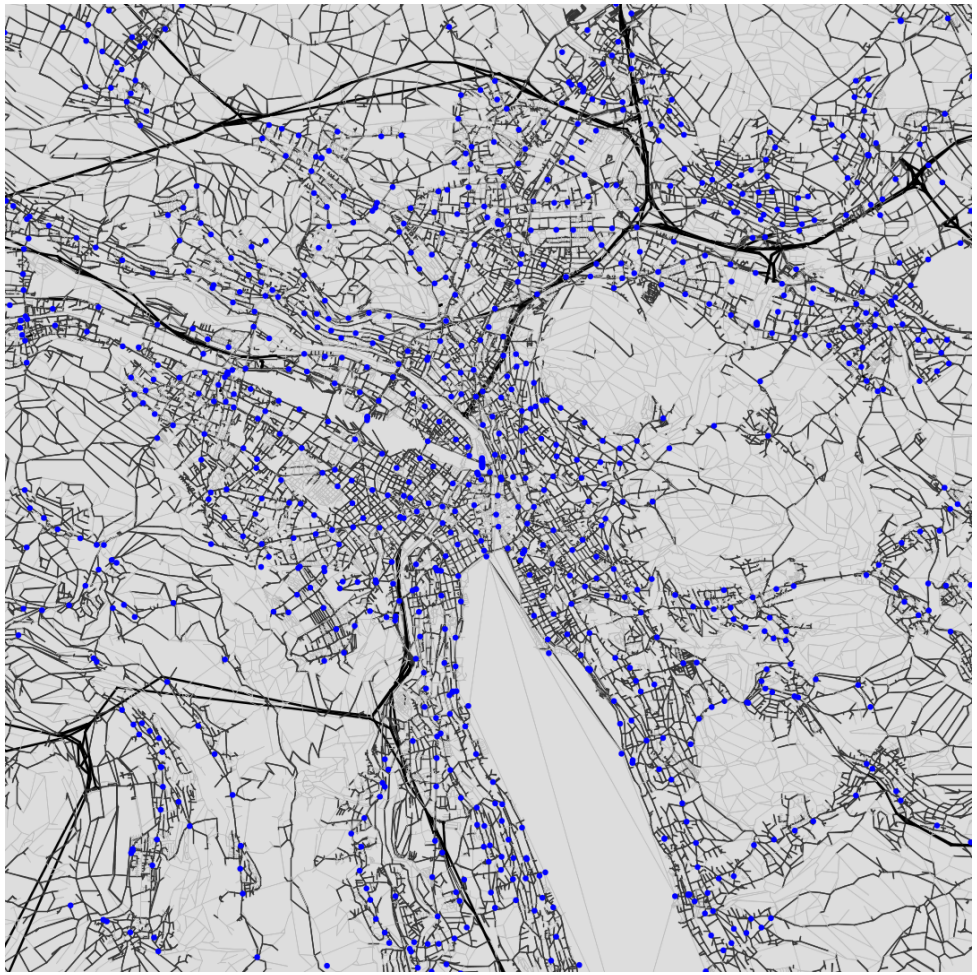
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <attributes>
      <attribute name="employed" class="java.lang.Boolean" >true</attribute>
    </attributes>
    <plan score="140.00982053869416" selected="yes">
      <activity type="home" link="1112" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" dep_time="07:18:16" trav_time="00:01:40">
        <route type="links" start_link="1112" end_link="2223" trav_time="00:31:40" distance="14403.0">1112 1222 2223</r
      </leg>
      <activity type="work" link="2223" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="2223" end_link="1312" trav_time="00:27:32" distance="12573.0">2223 2313 1312</r
      </leg>
      <activity type="shopping" link="1312" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="1312" end_link="1112" trav_time="00:04:15" distance="3127.0">1312 1211 1112</ro
      </leg>
      <activity type="home" link="1112" x="890.0" y="685.0" />
    </plan>
    <plan ...>
      ...
  </person>
  ...
</population>
```

Scenario Data: Public Transport

Describes public transport (pt) services.

Describes:

- Location of pt stop facilities
- PT lines
- Routes of pt lines
- Departure times at stops of routes
- PT vehicle capacities



Public transport stops

Scenario Data: transitSchedule.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE transitSchedule SYSTEM "http://www.matsim.org/files/dtd/transitSchedule_v1.dtd">
<transitSchedule>
  <transitStops>
    <stopFacility id="1" x="1050" y="1050" linkRefId= "11" name="Aadorf" />
    <stopFacility id="2" x="2050" y="2940" linkRefId= "12" name="Beweiler"/>
    ...
  </transitStops>
  <transitLine id="Blue Line">
    <transitRoute id="outbound">
      <transportMode>train</transportMode>
      <routeProfile>
        <stop refId="1" departureOffset="00:00:00"/>
        <stop refId="2" arrivalOffset="00:03:20" departureOffset="00:04:00"/>
        <stop refId="3" arrivalOffset="00:09:00" />
      </routeProfile>
      <route>
        <link refId="11"/>
        <link refId="12"/>
        <link refId="23"/>
        <link refId="33"/>
      </route>
      <departures>
        <departure id="01" departureTime="06:00:00" vehicleRefId="tr_1" />
        <departure id="02" departureTime="06:15:00" vehicleRefId="tr_2" />
        <departure id="03" departureTime="06:30:00" vehicleRefId="tr_1" />
      </departures>
    </transitRoute>
  </transitLine>
</transitSchedule>
```

Execution, MobSim

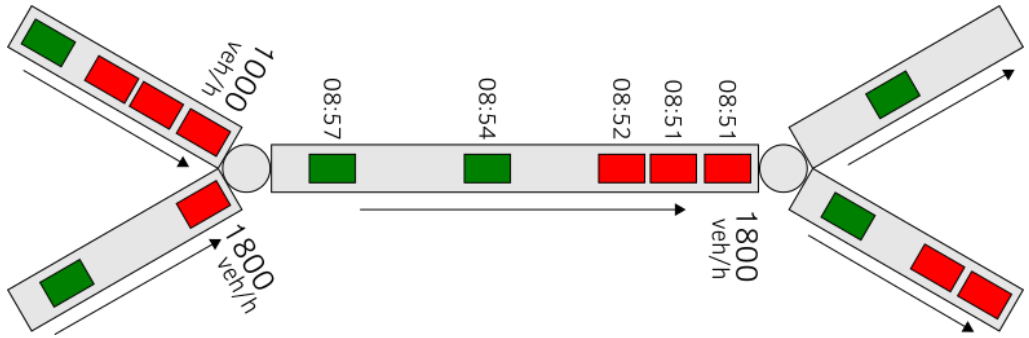
Execute all plans simultaneously in a simulation of the physical world

→ Synthetic Reality

Start at 00:00:00, proceed with time step by step, handle plans:

- agents end activities
- agents depart
- vehicles enter traffic
- vehicles cross intersections
- vehicles get stuck behind other vehicles
- public transport vehicles arrive at stops
- agents arrive
- agents start new activity
- ...

Execution: Queue Model



Simplified modelling of links with Queues → QSim

- First-in first-out (FIFO) queue
- Earliest "out" according to travel time
- "out-flow" corresponding to flow capacity of link
- links can be full (→ traffic jam, spillback)

Execution Result: Events

The mobsim creates **Events** as documentation.

Each event has a **time** and a **type** and can contain arbitrary additional key-value pairs.

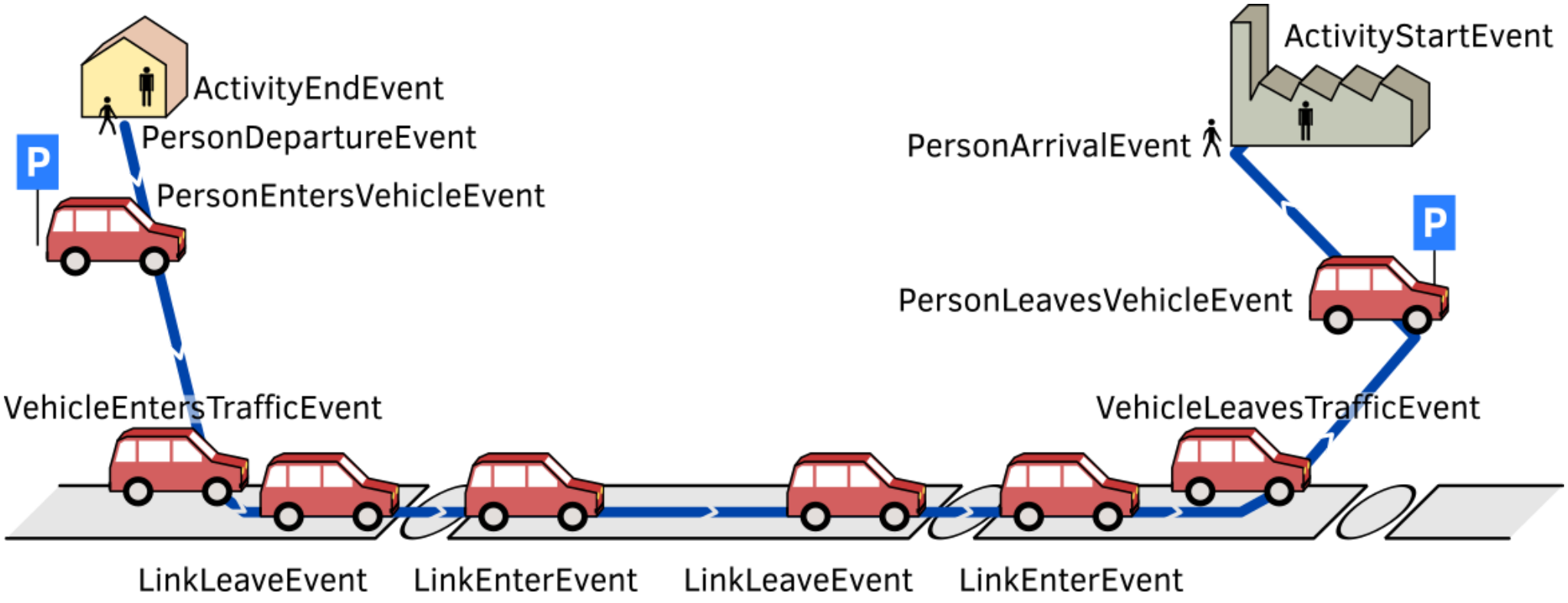
Type of events (non-concluding):

- agent starts / ends activity ("actstart", "actend")
- agent starts / ends a leg ("departure", "arrival")
- vehicle enters / leaves a link ("enter link", "left link")

Large scenarios will create many millions of events.

Events are the main data source for analyses.

Execution Result: Events (a single agent)



Execution Result: Events (a single agent)

```
<event time="25279.0" type="actend" person="188" link="1112" actType="home" />
<event time="25279.0" type="departure" person="188" link="1112" legMode="car" />
<event time="25279.0" type="PersonEntersVehicle" person="188" vehicle="188" />
<event time="25279.0" type="vehicle enters traffic" person="188" link="1112" vehicle="188" networkMode="car" />
<event time="25280.0" type="left link" vehicle="188" link="1112" />
<event time="25280.0" type="entered link" vehicle="188" link="1222" />
<event time="25381.0" type="left link" vehicle="188" link="1222" />
... more enter link / left link events
<event time="25482.0" type="entered link" vehicle="188" link="2324" />
<event time="25582.0" type="vehicle leaves traffic" person="188" link="2324" vehicle="188" networkMode="car" />
<event time="25582.0" type="PersonLeavesVehicle" person="188" vehicle="188" />
<event time="25582.0" type="arrival" person="188" link="2324" legMode="car" />
<event time="25582.0" type="actstart" person="188" link="2324" actType="work" />

<event time="64332.0" type="actend" person="188" link="2324" actType="work" />
<event time="64332.0" type="departure" person="188" link="2324" legMode="car" />
<event time="64332.0" type="PersonEntersVehicle" person="188" vehicle="188" />
<event time="64332.0" type="vehicle enters traffic" person="188" link="2324" vehicle="188" networkMode="car" />
<event time="64333.0" type="left link" vehicle="188" link="2324" />
<event time="64333.0" type="entered link" vehicle="188" link="2423" />
<event time="64434.0" type="left link" vehicle="188" link="2423" />
... more enter link / left link events
<event time="64737.0" type="entered link" vehicle="188" link="1112" />
<event time="64837.0" type="vehicle leaves traffic" person="188" link="1112" vehicle="188" networkMode="car" />
<event time="64837.0" type="PersonLeavesVehicle" person="188" vehicle="188" />
<event time="64837.0" type="arrival" person="188" link="1112" legMode="car" />
<event time="64837.0" type="actstart" person="188" link="1112" actType="home" />
```

events related to vehicles

Execution Result: Events (public transport)

```
<event time="21600.0" type="TransitDriverStarts"
       driverId="pt_tr_1_1" vehicleId="tr_1" transitLineId="Blue Line" transitRouteId="1to3" departureId="01"  />
<event time="21600.0" type="departure" person="pt_tr_1_1" link="11" legMode="car"  />
<event time="21600.0" type="PersonEntersVehicle" person="pt_tr_1_1" vehicle="tr_1"  />
<event time="21600.0" type="vehicle enters traffic" person="pt_tr_1_1" link="11" vehicle="tr_1" networkMode="car"  />
<event time="21600.0" type="VehicleArrivesAtFacility" vehicle="tr_1" facility="1" delay="Infinity"  />
<!-- vehicle is at first stop, a passenger enters -->
<event time="21601.0" type="PersonEntersVehicle" person="298" vehicle="tr_1"  />
<event time="21610.0" type="VehicleDepartsAtFacility" vehicle="tr_1" facility="1" delay="0.0"  />
<event time="21611.0" type="left link" vehicle="tr_1" link="11"  />
<event time="21611.0" type="entered link" vehicle="tr_1" link="12"  />
<event time="21811.0" type="VehicleArrivesAtFacility" vehicle="tr_1" facility="2a" delay="1.0"  />
<!-- vehicle is at second stop, passenger exits -->
<event time="21812.0" type="PersonLeavesVehicle" person="298" vehicle="tr_1"  />
<event time="21821.0" type="VehicleDepartsAtFacility" vehicle="tr_1" facility="2a" delay="-39.0"  />
<event time="21822.0" type="left link" vehicle="tr_1" link="12"  />
<event time="21822.0" type="entered link" vehicle="tr_1" link="23"  />
<event time="22123.0" type="left link" vehicle="tr_1" link="23"  />
<event time="22123.0" type="entered link" vehicle="tr_1" link="33"  />
<event time="22124.0" type="VehicleArrivesAtFacility" vehicle="tr_1" facility="3" delay="-36.0"  />
<!-- vehicle is at third (and last) stop -->
<event time="22124.0" type="VehicleDepartsAtFacility" vehicle="tr_1" facility="3" delay="-36.0"  />
<event time="22124.0" type="vehicle leaves traffic" person="pt_tr_1_1" link="33" vehicle="tr_1" networkMode="car"  />
<event time="22124.0" type="PersonLeavesVehicle" person="pt_tr_1_1" vehicle="tr_1"  />
<event time="22124.0" type="arrival" person="pt_tr_1_1" link="33" legMode="car"  />
```

Execution: Events vs. Plans

Plans are input for the mobsim.

Events are output of the mobsim. (Events ≈ Executed Plans)

Events and Plans often do not match:

- Assumptions to create plans might be wrong
- Interaction with other agents cannot be foreseen

The same plan might create different events when executed multiple times,
due to changes in other agents' plans and resulting different interactions.

Scoring

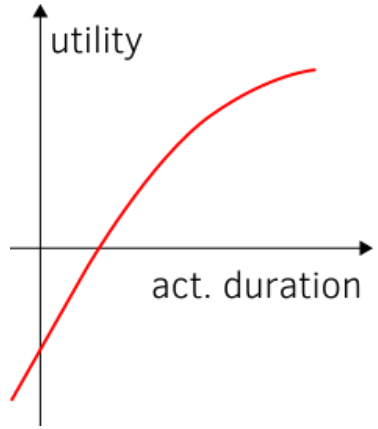
To compare different plans, a score (utility value) is calculated → Scoring Function

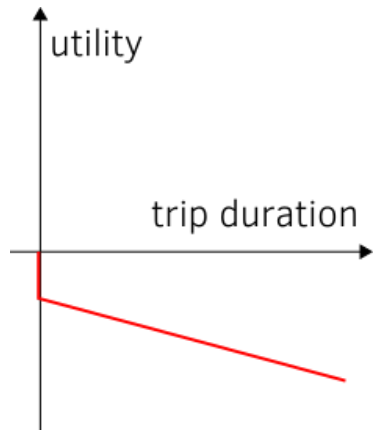
Terms:

- Positive utility for performing activities
- Negative utility for travelling
- Negative utility for monetary costs (tolls, fares, …)
- Negative utility for arriving late, leaving early, …

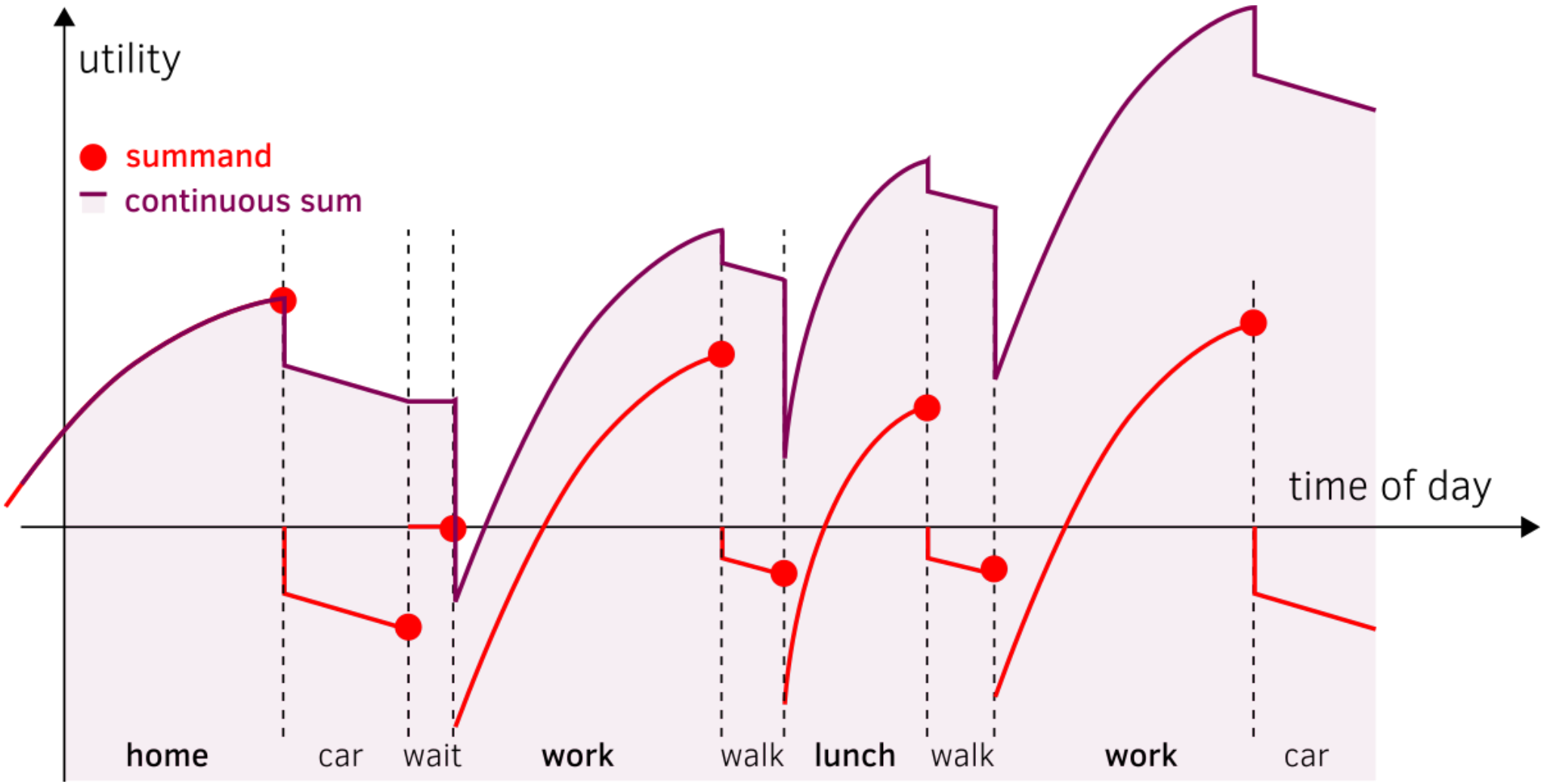
Sum up all terms for each agent over day.

Scoring function ≈ Fitness function (evolutionary computation)
Scoring function ≈ Utility function (economics)





Scoring


utility
summand
continuous sum
time of day
home
car
wait
work
walk
lunch
walk
work
car

Replanning

1. Every agent has one or more plans. → **Choice Set, Gene Pool**
2. One ("selected") plan gets executed and scored.
3. Some agents create new plans (and select them),
   others just select one of existing plans. → **Mutation**
4. Some agents remove plans (usually bad ones). → **Survival of the fittest**

→ **Evolutionary optimization algorithm per agent**
(Plan ≈ Genotype, execution of plan ≈ Phenotype)

Replanning: Plan modification

Standard choice dimensions ("strategies"):

- Departure time choice
- Mode choice
- Route choice

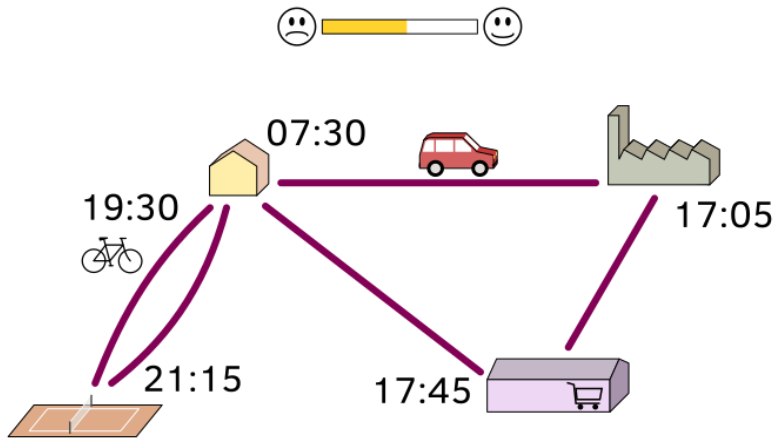
Experimental strategies:

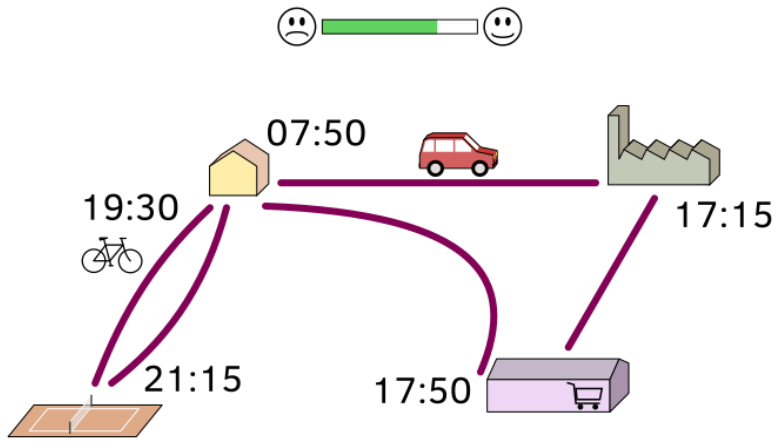
- Secondary activity location choice

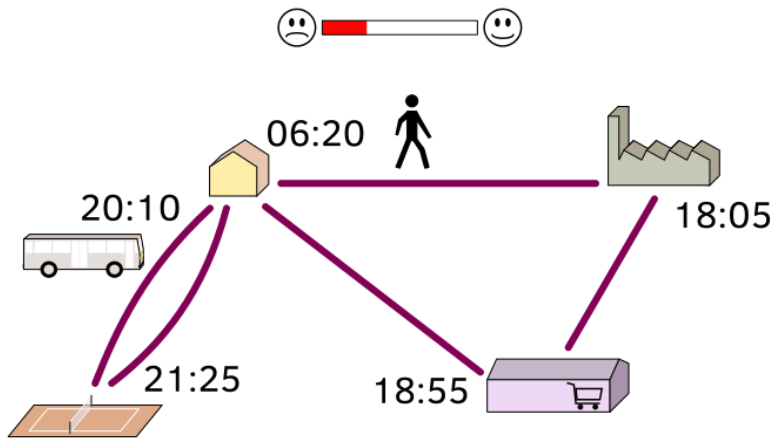
Currently not supported strategies:

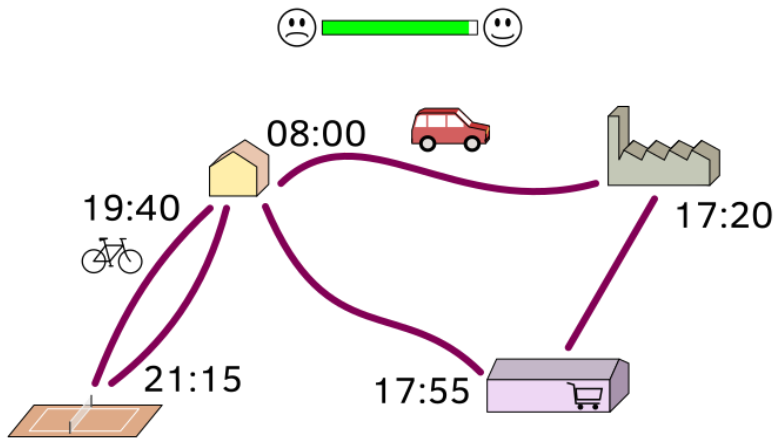
- Activity pattern choice

Replanning: Example









Replanning

- Not every agent should replan in each iteration.

  Otherwise, side effects like fluctuations can happen.

- Replanning strategies can be disabled after a certain iteration.

  Useful to have higher rate of innovation at the start.

- Replanning can globally be disabled after some % of iterations.

  Prevent agents trying out new, likely bad plans. E.g. mode choice can force agents to try rarely used modes.

Analyses

MATSim includes a few analyses out of the box.
(e.g. leg modes, hourly link travel times, hourly link volumes, travel time stats)

Most analyses are very specific and need to be custom implemented.

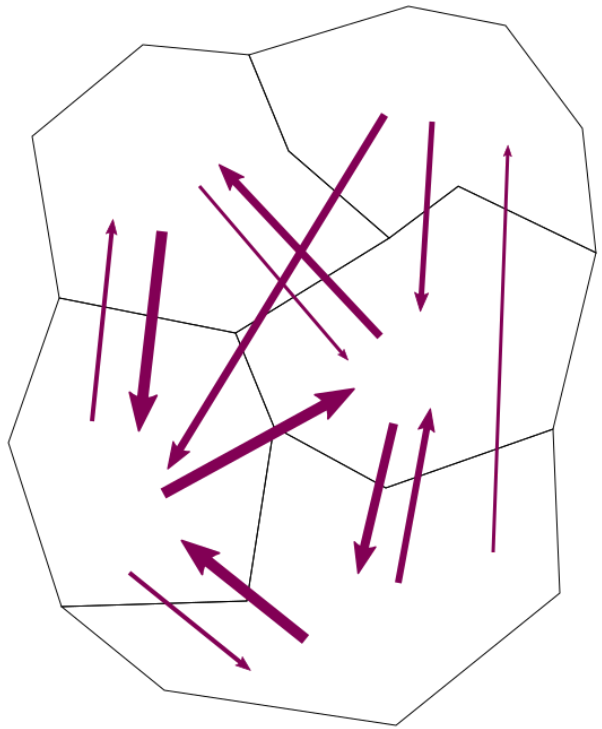
Analyses are typically based on Events.
(e.g. count link-leave-events for each link → link volumes)

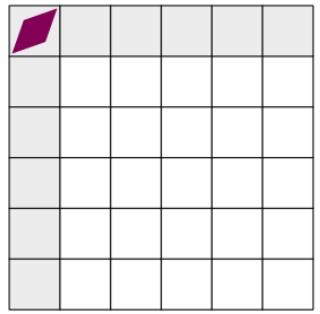
A visualization is also a type of analysis.

Via includes several often-used analyses.

Why Agent-Based Modelling?

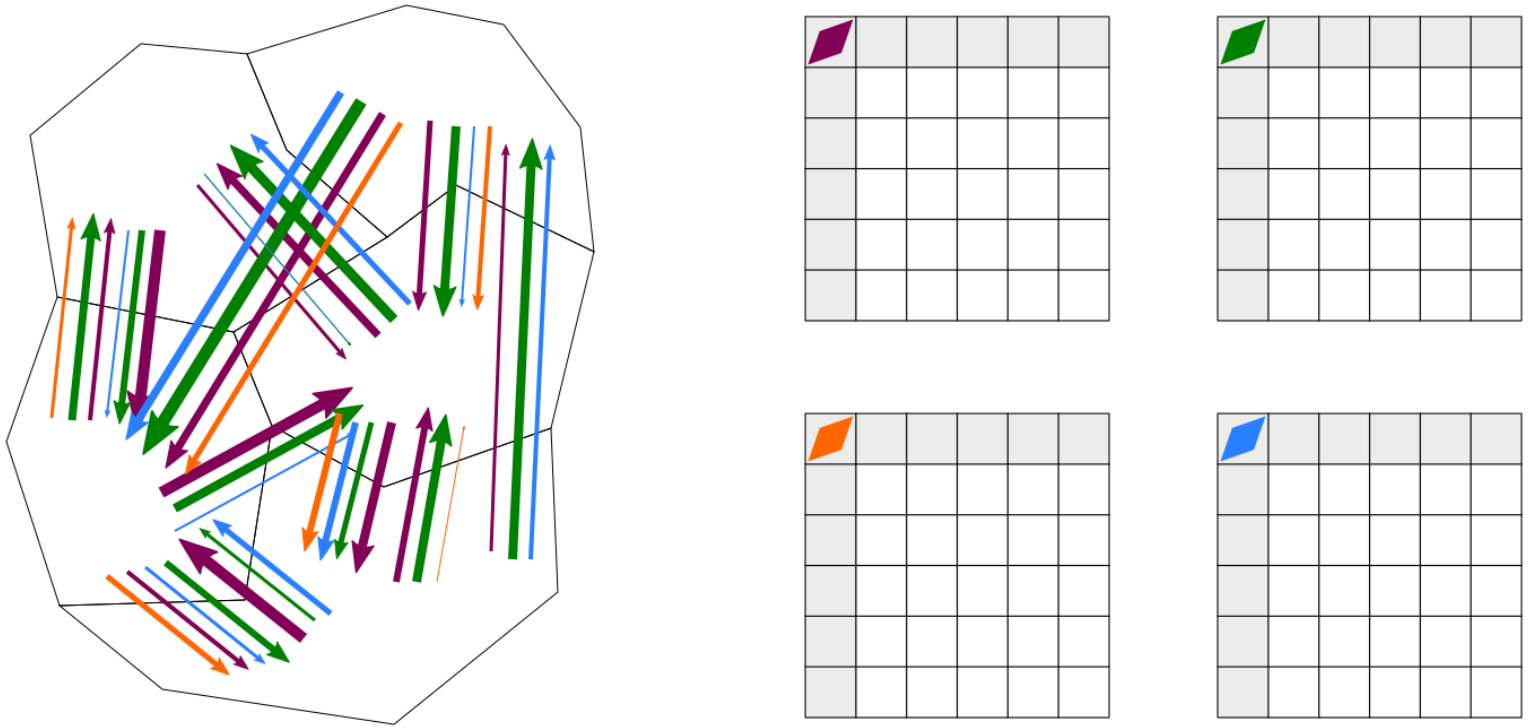
Traditional transport models are based on "aggregated flows between zones",
with demand typically described in OD matrices.





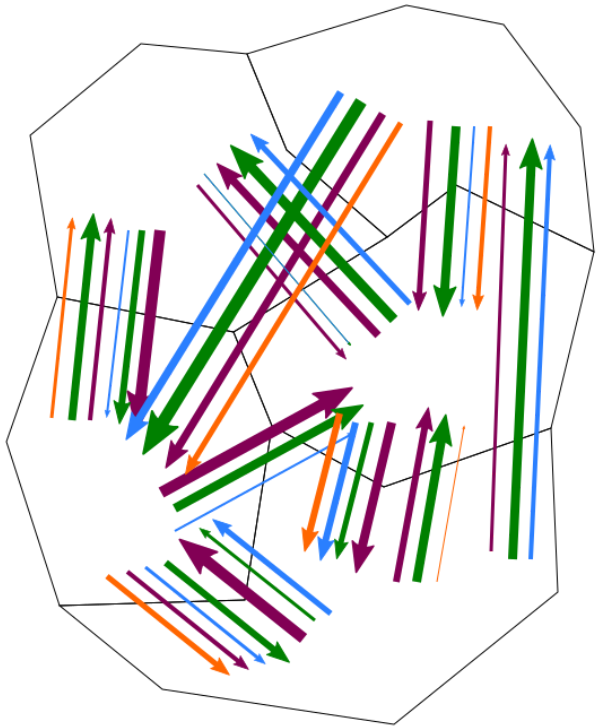
Why Agent-Based Modelling?

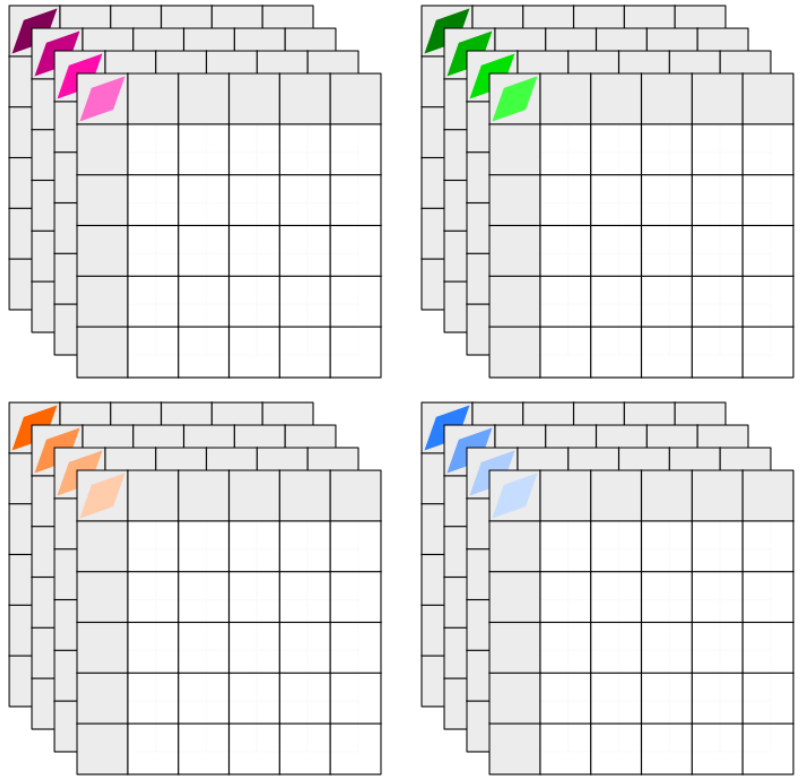
Often, more than 1 matrix is used,
e.g. to describe trips from/to work, from/to school, from/to shopping, and other trips.



Why Agent-Based Modelling?

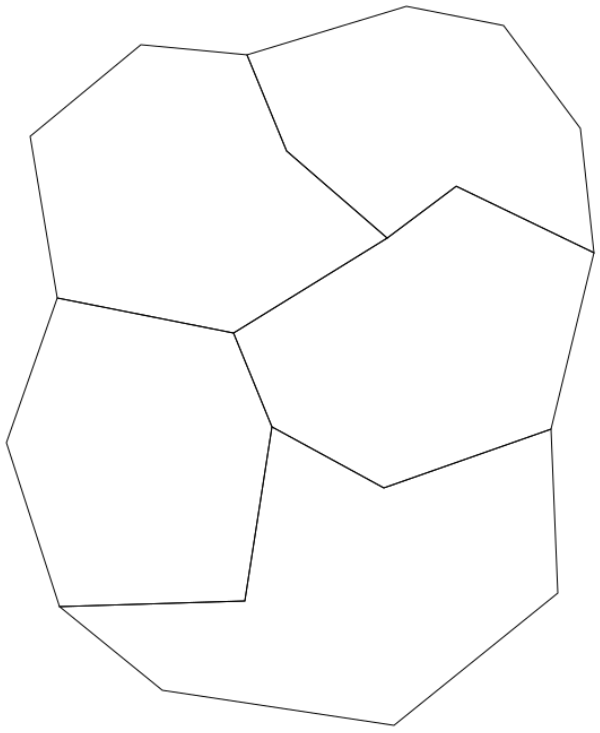
Even more matrices are needed to describe the travel behaviour of different person groups, e.g. aged 0 – 20, male 20 – 60, female 20 – 60, 60+.

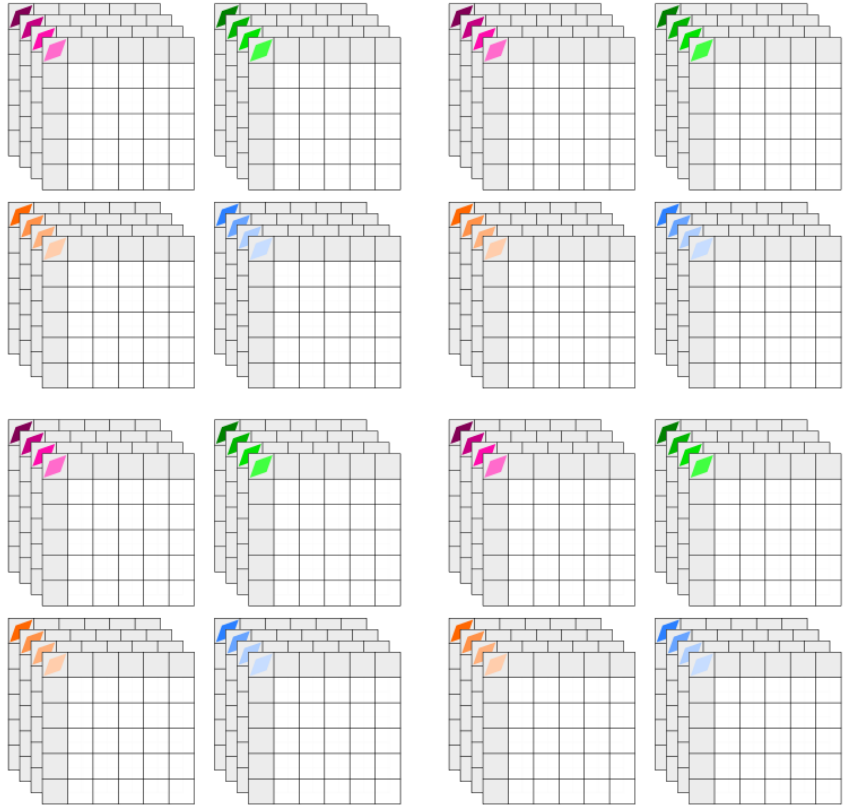




Why Agent-Based Modelling?

If you want to model different times of day, the number of matrices multiply once more.





Why Agent-Based Modelling?

If you build a traditional model with a large number of person groups, a detailed number of different activity groups, multiple modes of transport, and different times of day, you end up with a very large number of matrices.

The more spatially detailed the model is, the more zones the model contains, and the number of matrix cells scales exponentially. Many of these matrix cells may become (close to) 0, leading to unstable numeric operations.

At some point, it is more efficient to list each agent individually, than to build a very large number of matrices where each contains only a handful of trips.

Having agents instead of separate trips helps to maintain physical constraints (can't leave a location before having arrived there, can only take a car if it was parked there before, ...).

Having agents with socio-demographic attributes allows analyzing the behaviour of arbitrary groups of persons after the simulation ran, instead of having to prepare the input data for these groups before the model is calculated.

— Setup —

Some Definitions

Usage types for MATSim

Standalone, JAR-file

- Using the latest release, running it from the command line or from the GUI
- Only MATSim-Main is supported, no extensions
- Suitable to get started

Using the Example-Project, as Maven-Dependency

- Best to use it from within an IDE
- Suitable to use and extend MATSim, run custom analysis
- Recommended for most users

Checking out the Source Code

- Use it in IDE
- Suitable for MATSim Developers

(IDE: Integrated Development Environment, e.g. Eclipse, IntelliJ, NetBeans)

MATSim Parts

MATSim is modular: core + extensions

A MATSim release contains "core" plus some selected extensions.

Additional extensions are available.
Be careful when combining multiple extensions, not all combinations might work.

Terminology:

- **matsim-main**: what makes the standard release.
- **matsim-core**
- **extension**
- **contrib**: an extension whose code is in the same repository as matsim-main.
  No new contribs anymore, new extensions should be separate projects.
- **playgrounds**: code of students and for projects.
  For historic reasons, playgrounds were once part of the same source repository, but should now be moved to separate repositories (e.g. matsim-eth, matsim-vsp)

Set Up

Required software for this week

Java

java.oracle.com, Click on "Java SE" under "Top Downloads"

Please download **Java SE 8u... / JDK**

MATSim

matsim.org/downloads

Download the "Latest stable release" in the Standalone section

IDE

IntelliJ IDEA: Community Edition, jetbrains.com/idea/download/ (recommended)

Eclipse: Eclipse IDE for Java Developers, eclipse.org/downloads/packages

Required software for this week (cnt'd)

GIT

git-scm.com

A Version Control System for source code, required by IntelliJ (and probably Eclipse) to download MATSim sources from GitHub.

Via

simunto.com/via/download

Commercial visualization and analysis tool for MATSim data.

Helpful software for this week (optional)

Text Editor

Notepad++ (Win) notepad-plus-plus.org

BBEdit (Mac) barebones.com/products/bbedit/

Atom (Win, Mac, Linux) atom.io

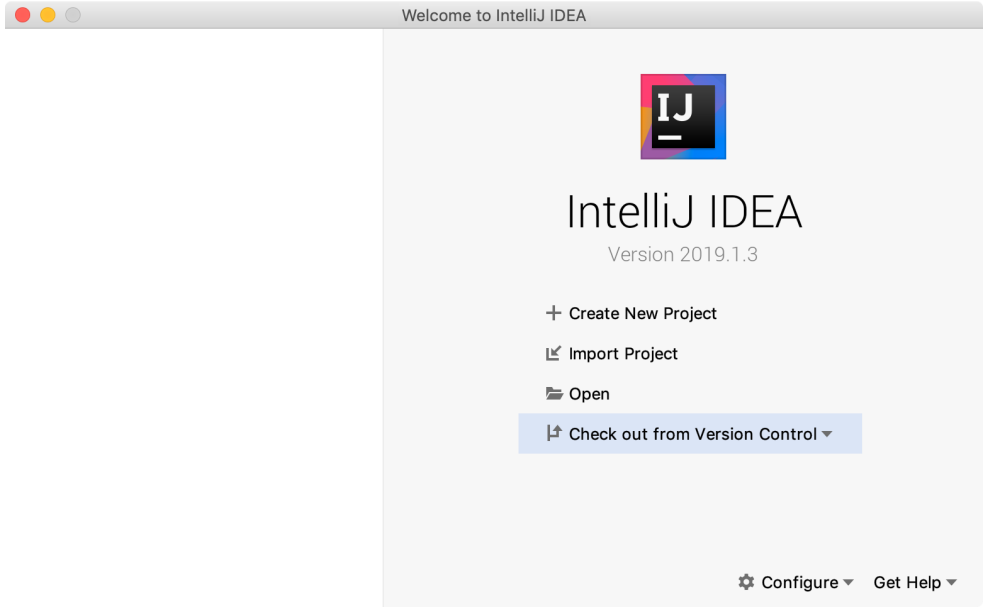
Tools

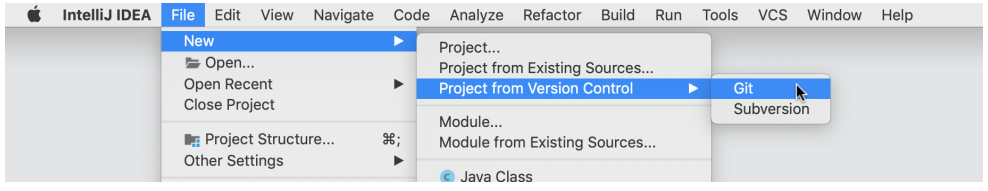
7-Zip (Win) 7-zip.org

JOSM josm.openstreetmap.de

IntelliJ IDEA: Checkout example project

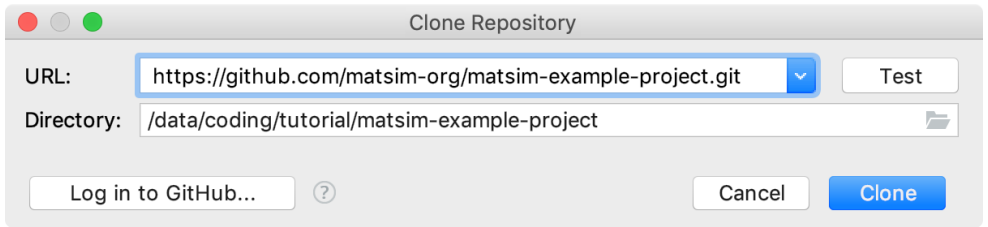
1. Open IntelliJ and create a new project from git version control





IntelliJ IDEA: Checkout example project (continued)

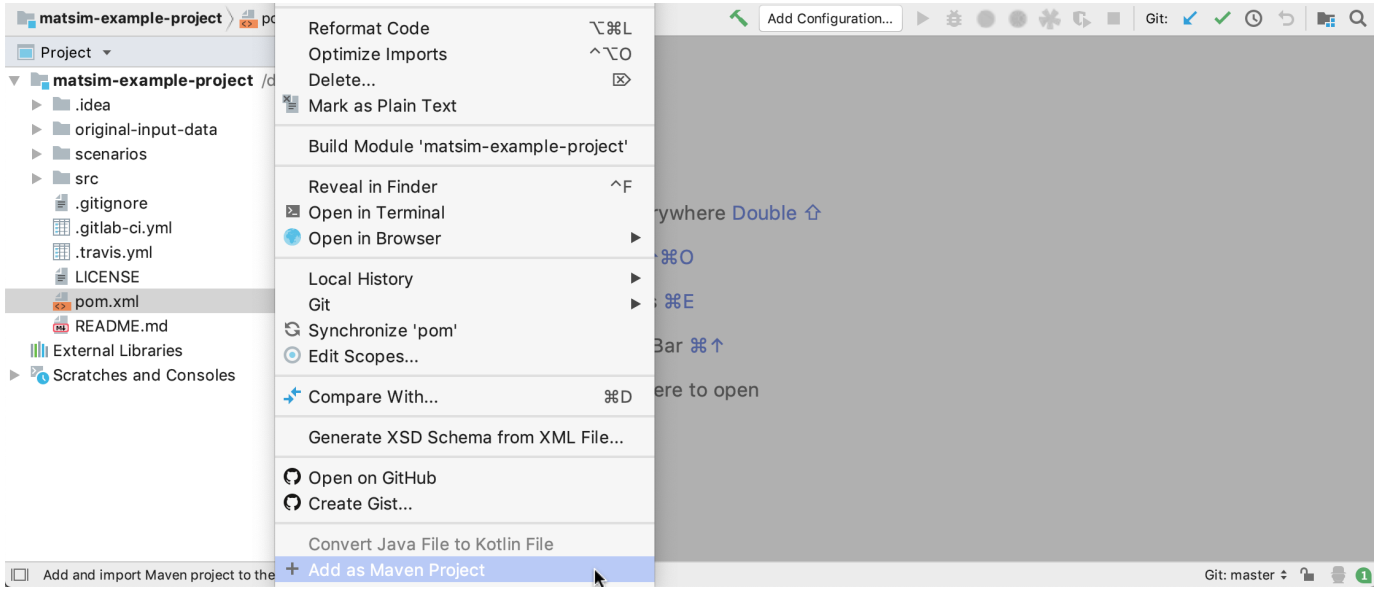
2. Set the URL to `https://github.com/matsim-org/matsim-example-project.git` and set a local directory where the project will be stored.



3. IntelliJ will download the project

IntelliJ IDEA: Checkout example project (continued)

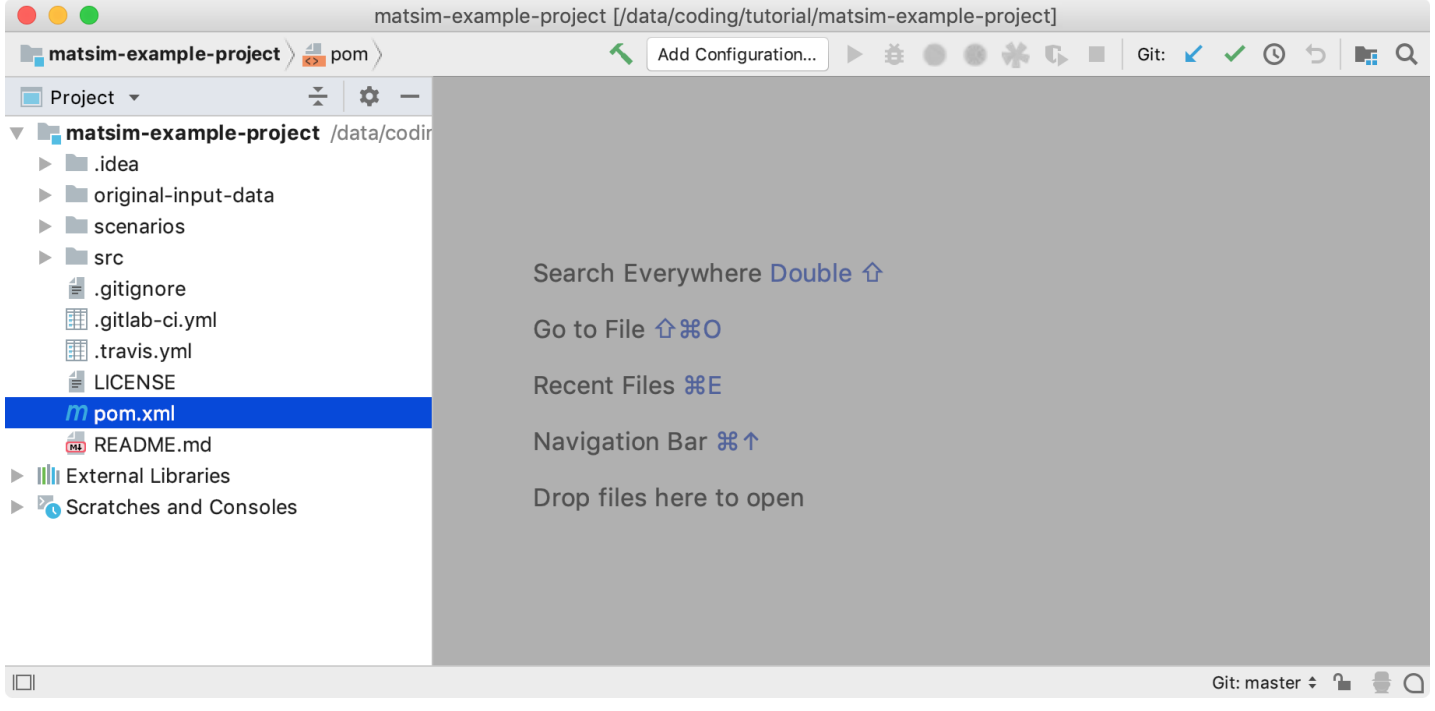
4. Right-click on `pom.xml` and select `Add as Maven Project`



IntelliJ IDEA: Checkout example project (continued)

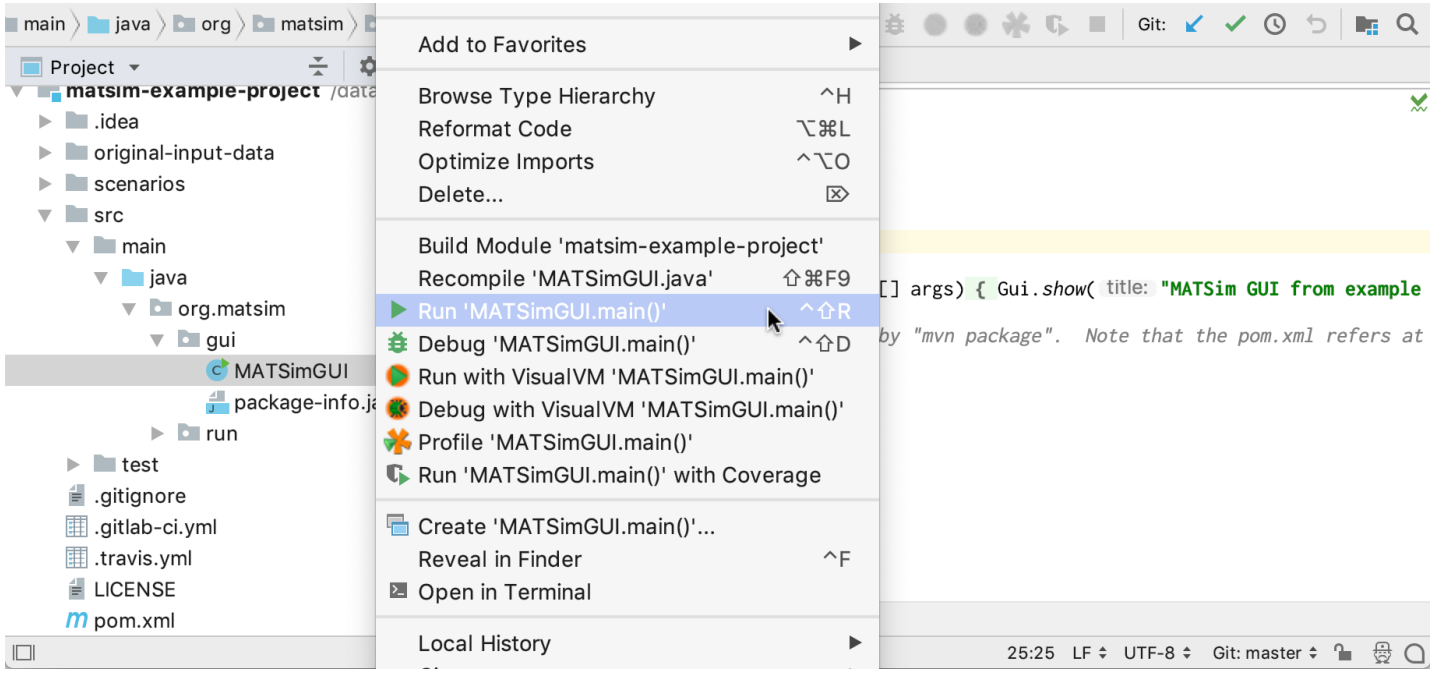
5. IntelliJ will download all required dependencies. This might take several minutes.

After this, the installation is ready.



IntelliJ IDEA: Running the GUI

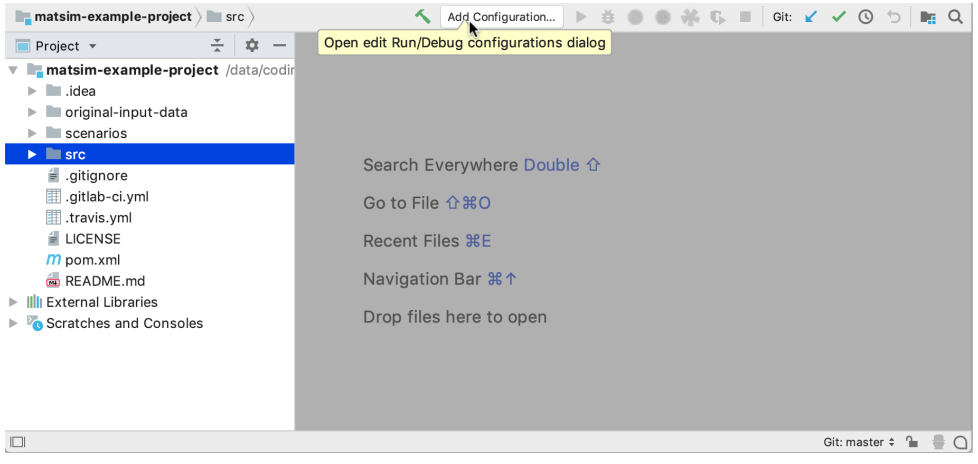
Option 1: Run the class `org.matsim.gui.MATSimGUI`.
Right-click the class and select `Run 'MATSimGui.main()'`.



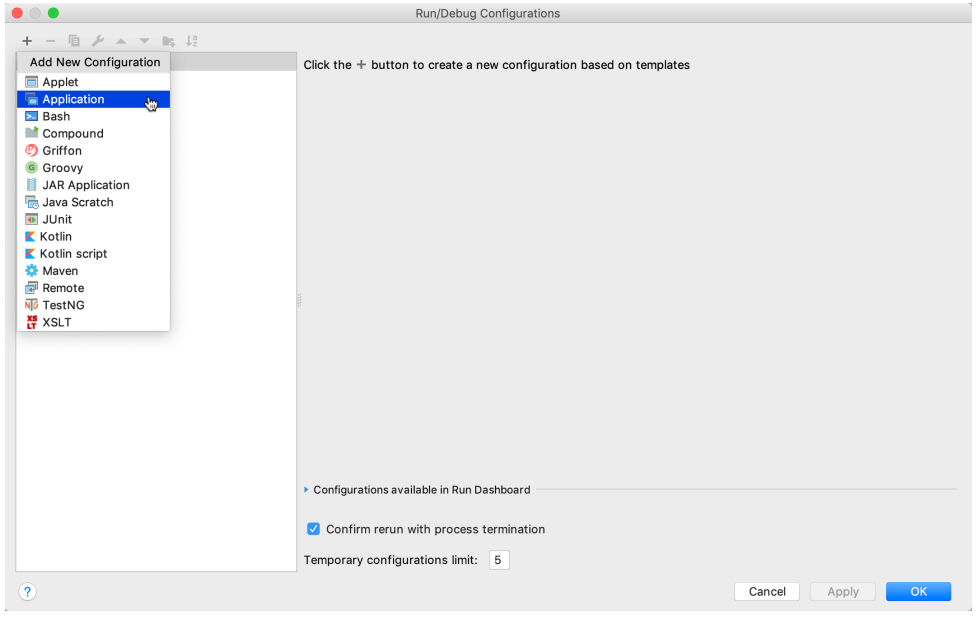
IntelliJ IDEA: Running the GUI

Option 2: Run the class `org.matsim.run.gui.Gui` from the matsim dependency.

Click on `Add Configuration...` in toolbar.



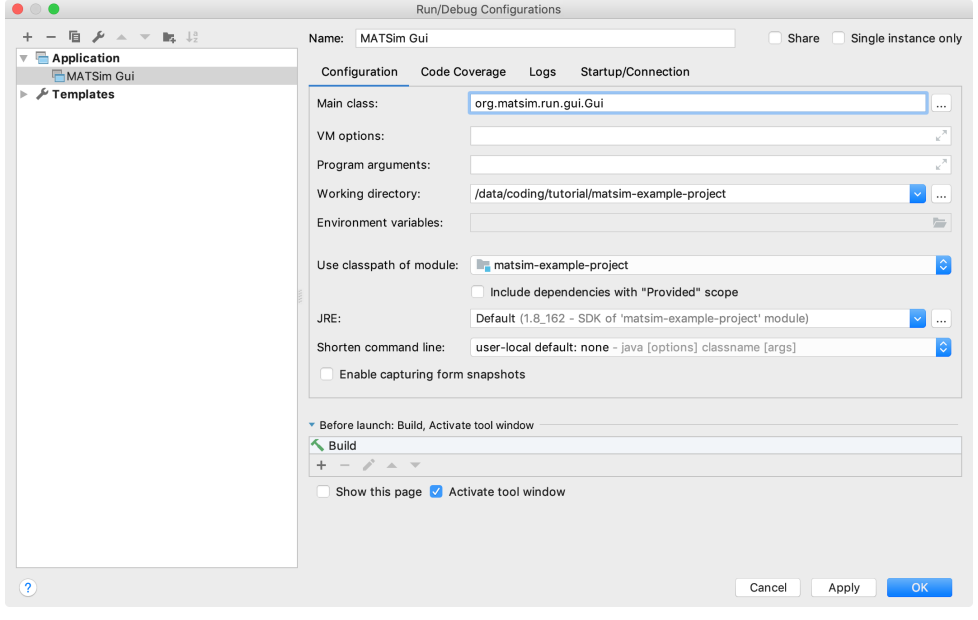
Click on the `+` and select `Application`



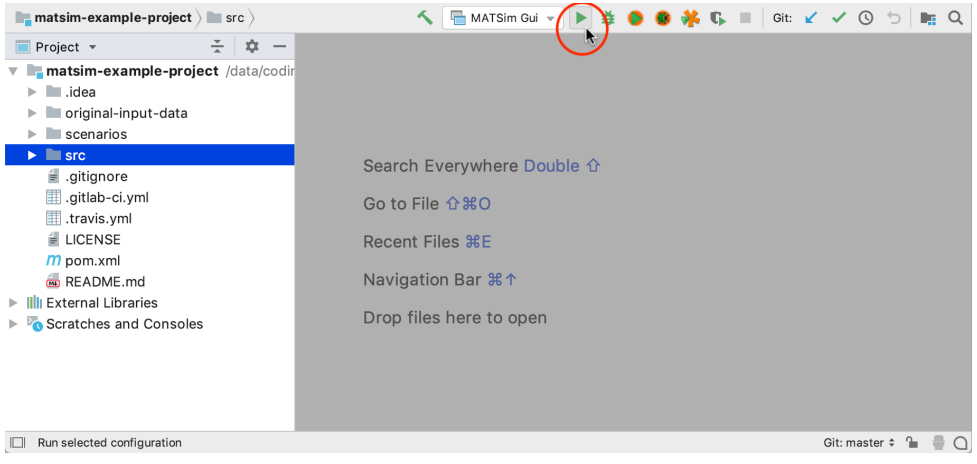
IntelliJ IDEA: Running the GUI

Option 2: (continued)

Set the main class `org.matsim.run.gui.Gui`, then click "OK".



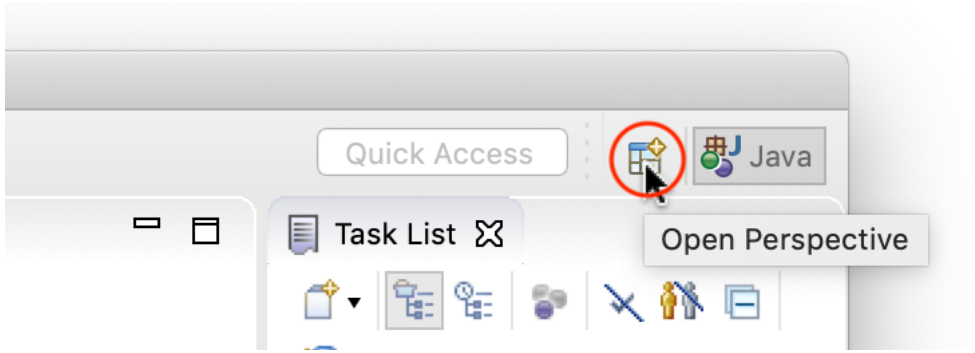
Click on the Run-button in the toolbar.



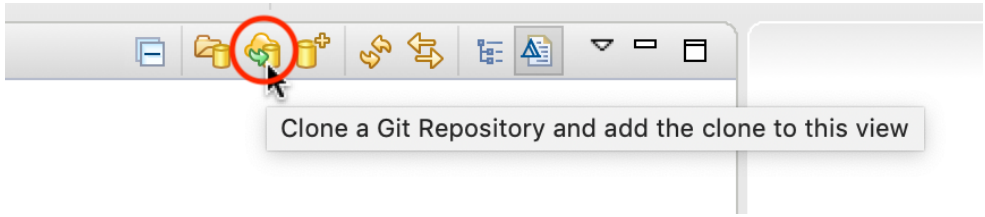
Eclipse: Check out example project

1. Open Eclipse and create a new workspace

2. In the top right corner, click `Open Perspective` and select `Git`

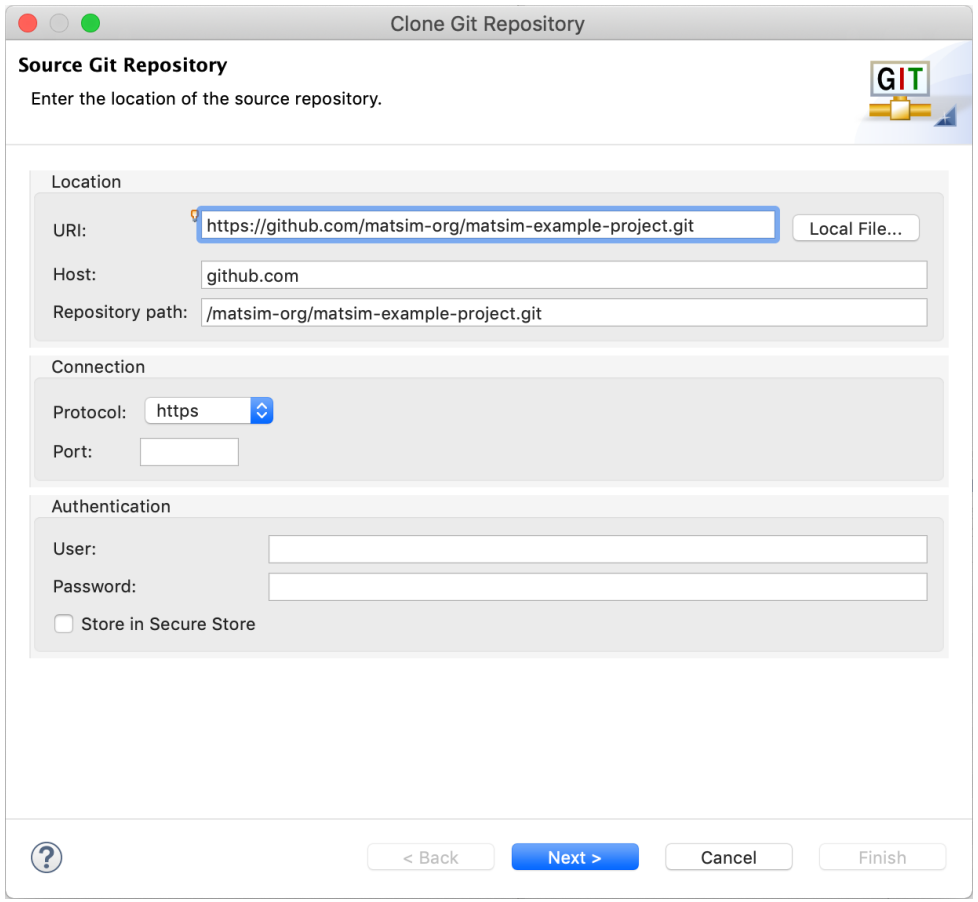


3. In the Git perspective, select `Clone a Git Repository and add the clone to this view`

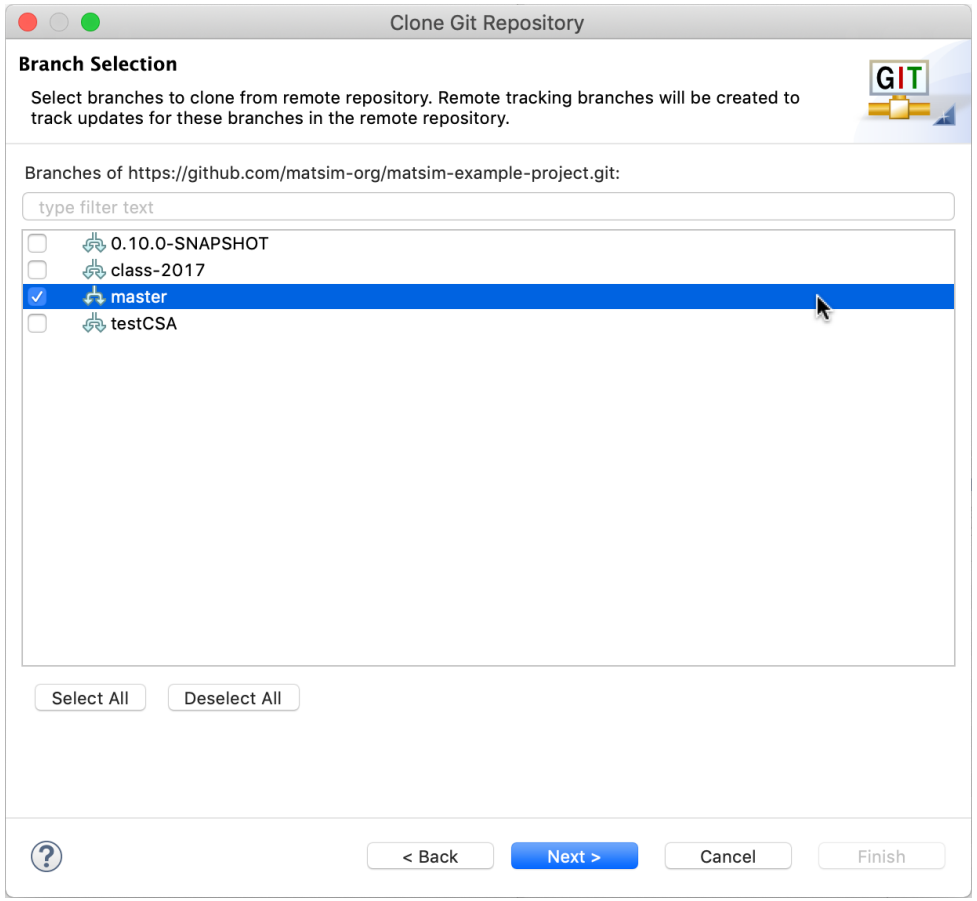


Eclipse: Check out example project (continued)

4. Set the URI to `https://github.com/matsim-org/matsim-example-project.git`

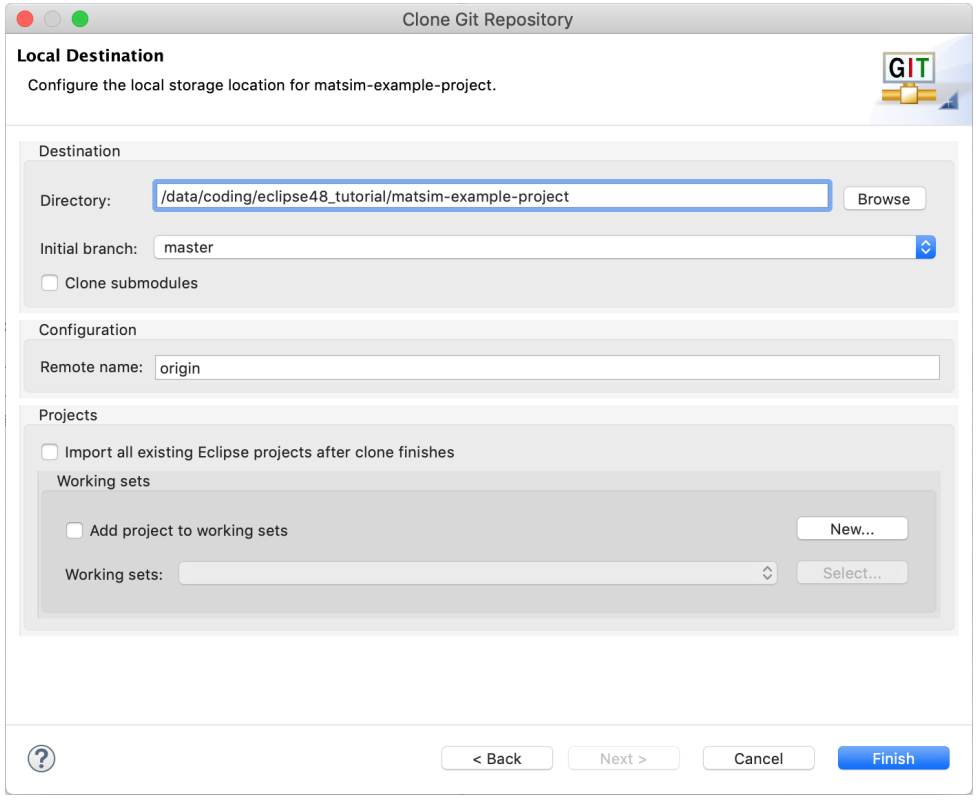


5. Select the `master` branch



Eclipse: Check out example project (continued)

6. Select a destination in your workspace



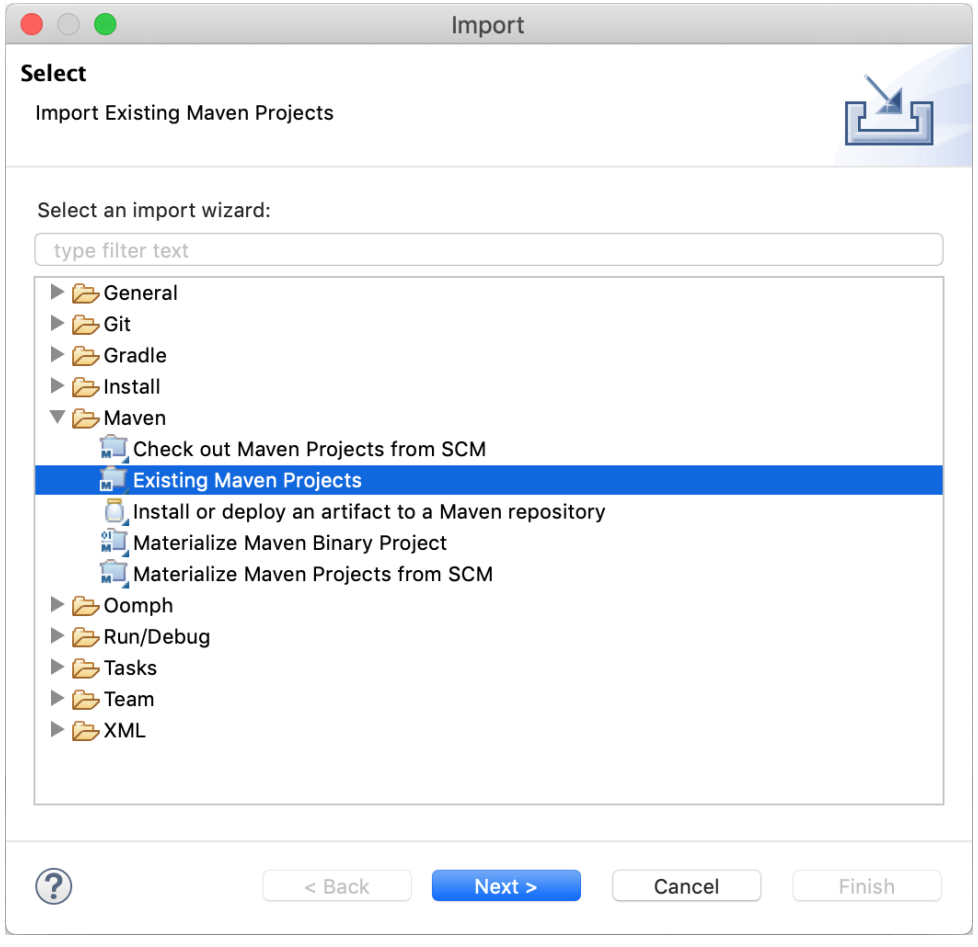
7. Wait for the download to finish

8. Switch back to the Java Perspective

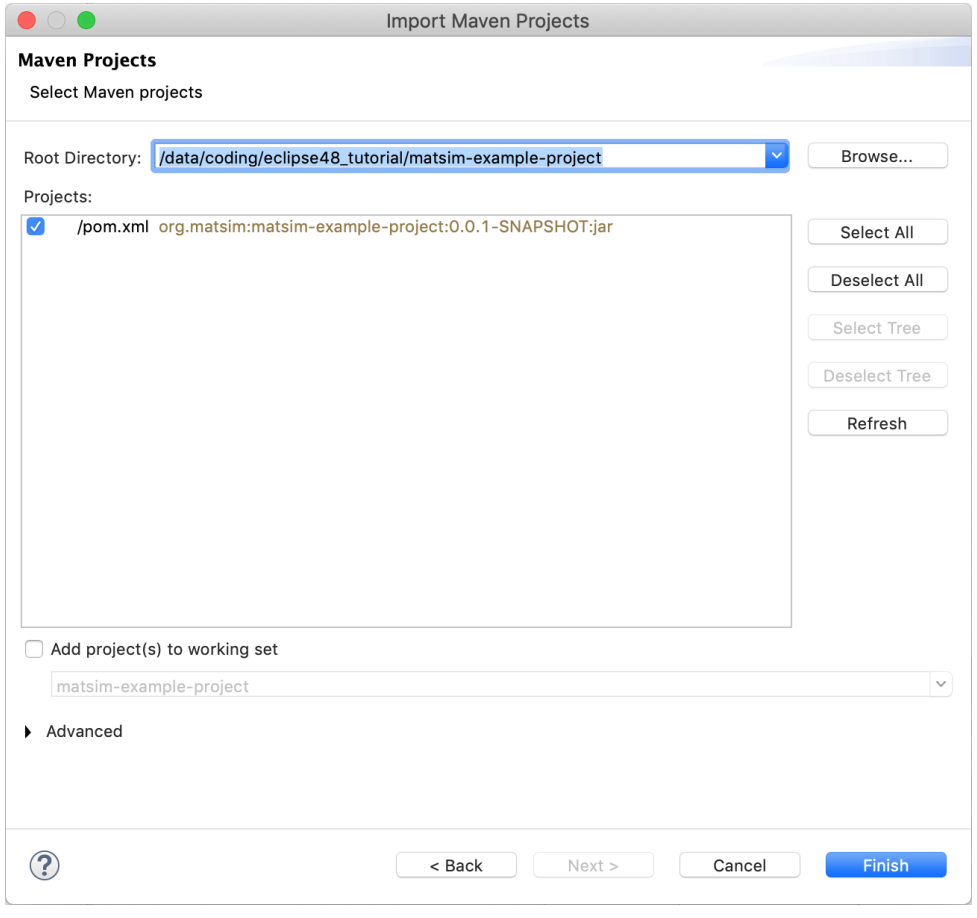
9. Select menu `File` → `Import`

Eclipse: Check out example project (continued)

10. Select Maven → Existing Maven Project



11. Select the previously cloned directory

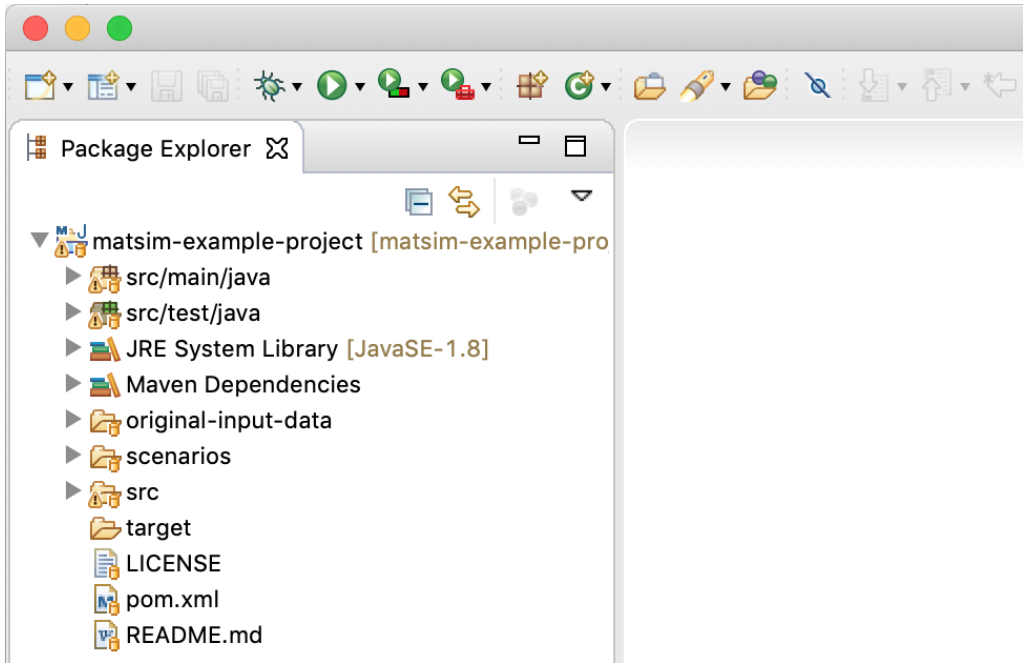


Eclipse: Check out example project (continued)

12. Click `Finish`

All required dependencies will now be downloaded.
This might take several minutes.

After this, the installation is ready.



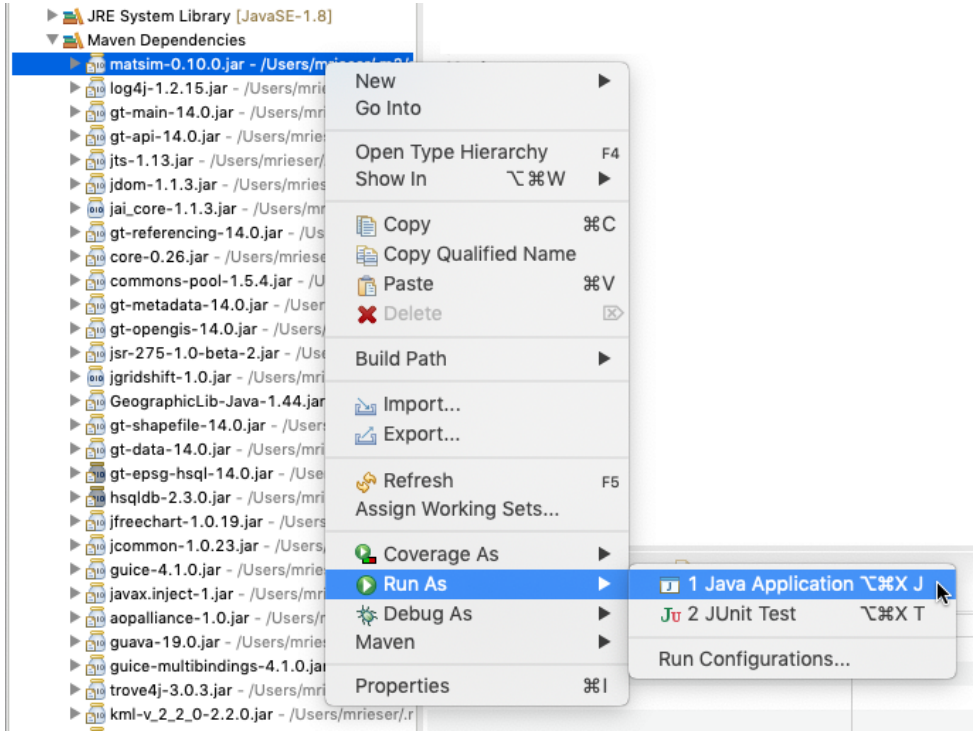
Eclipse: Running the GUI

Option 1: Run the class `org.matsim.gui.MATSimGUI`.
Right-click the class and select `Run As` → `Java Application`.

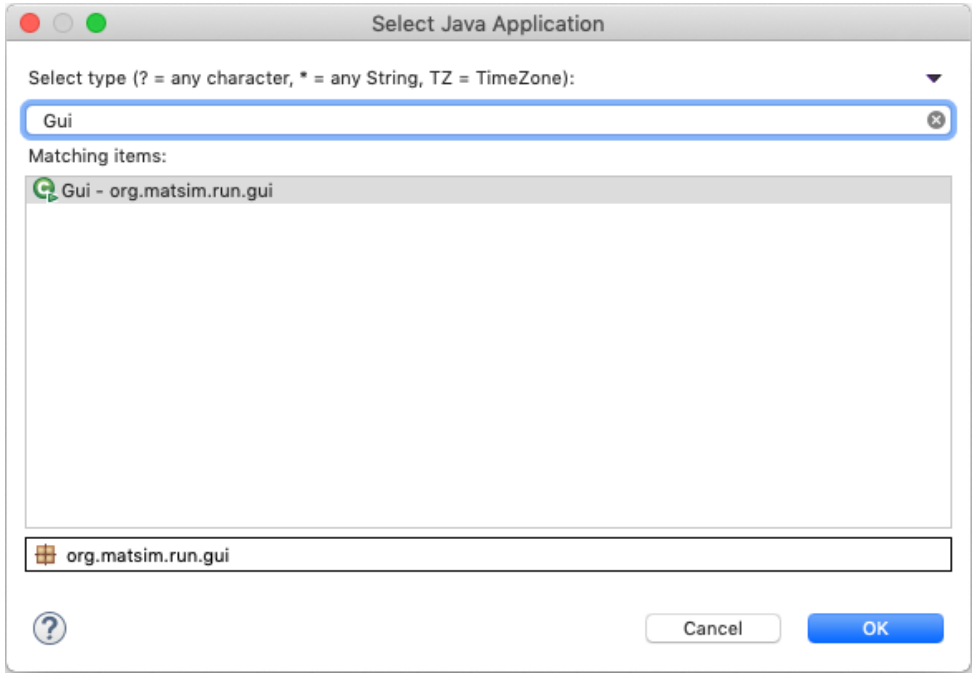
Eclipse: Running the GUI

Option 2: Run the class `org.matsim.run.gui.Gui` from the jar-file.

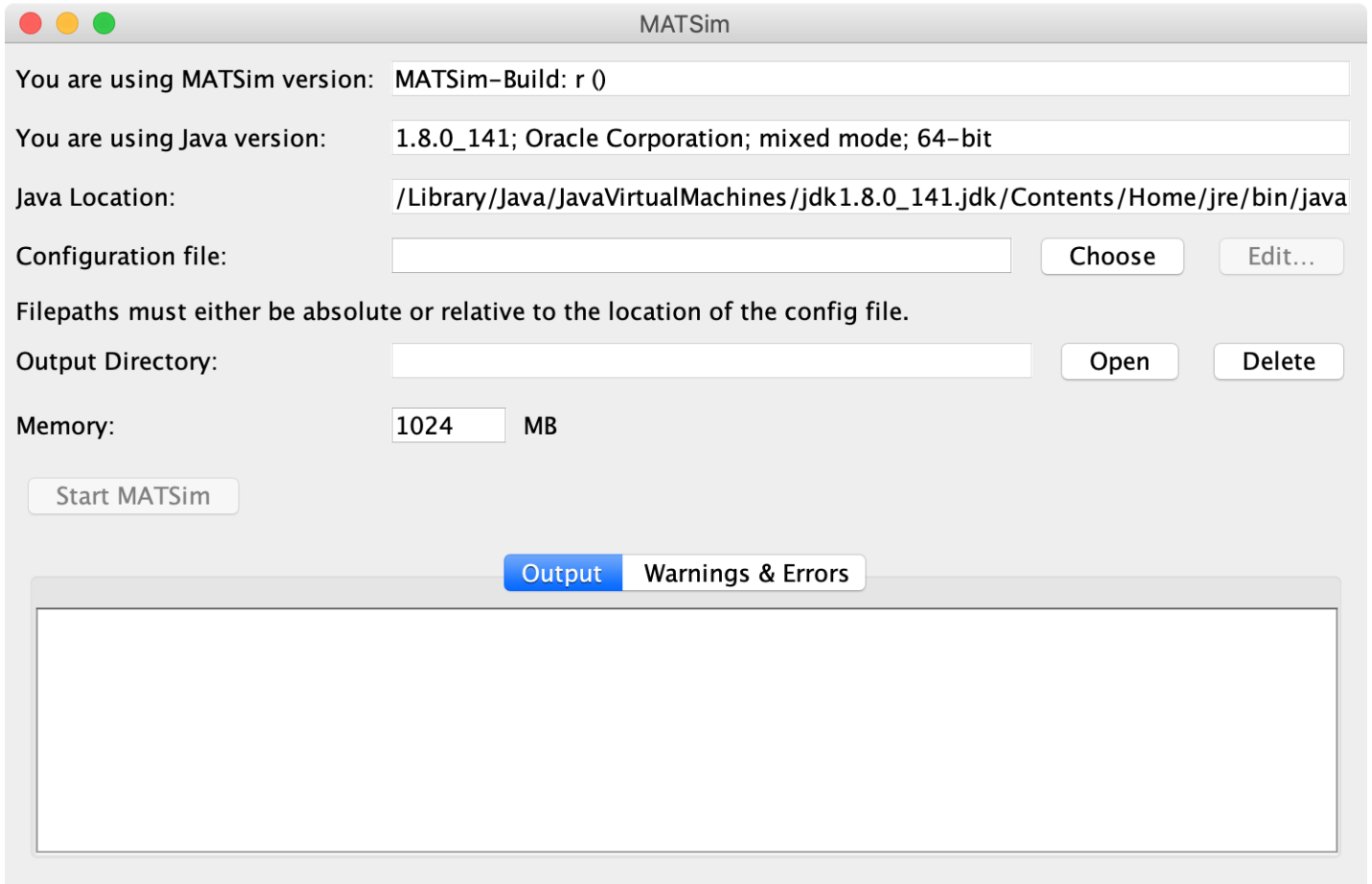
Right-click the matsim-jar and select `Run As` → `Java Application`.



Search for `Gui` and select `org.matsim.run.gui.Gui`.



MATSim GUI


MATSim
You are using MATSim version:
MATSim-Build: r ()
You are using Java version:
1.8.0_141; Oracle Corporation; mixed mode; 64-bit
Java Location:
/Library/Java/JavaVirtualMachines/jdk1.8.0_141.jdk/Contents/Home/jre/bin/java
Configuration file:
Choose
Edit...
Filepaths must either be absolute or relative to the location of the config file.
Output Directory:
Open
Delete
Memory:
1024
MB
Start MATSim
Output
Warnings & Errors

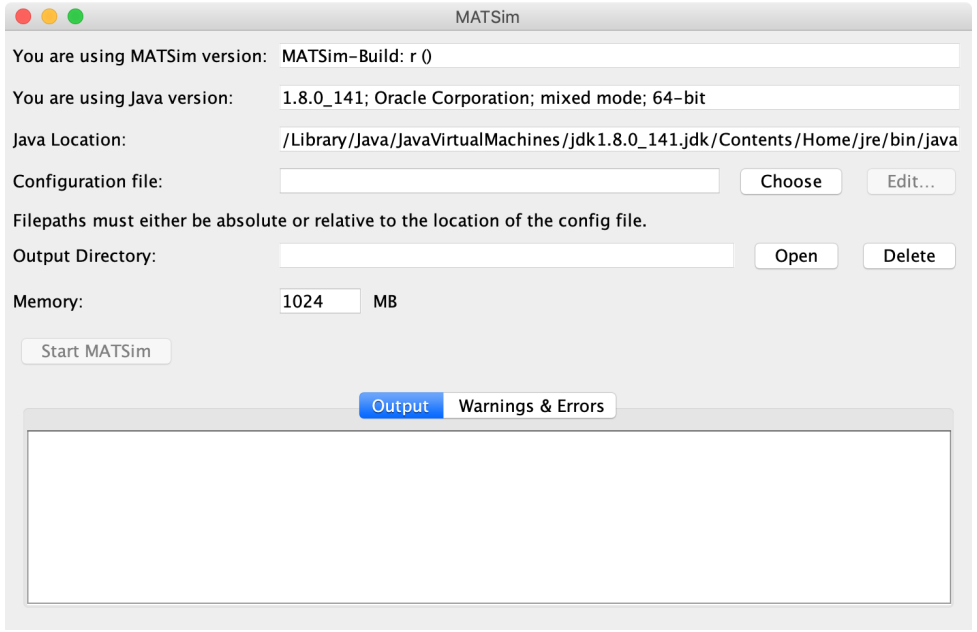
Your First Simulation

Your First Simulation

1. Start the MATSim GUI
   - either from the IDE
   - or by double-clicking `matsim.jar` from the release

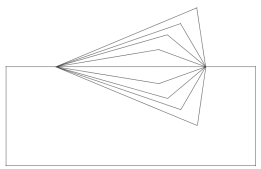
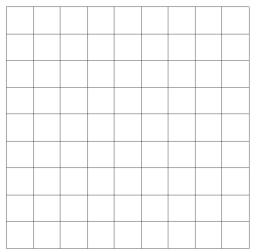
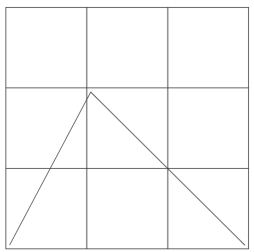
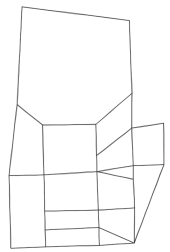
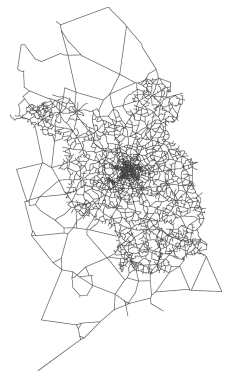
2. Choose a configuration file
   e.g. from the examples directory in the MATSim release

3. Click on "Start MATSim"



Your First Simulation: Example Scenarios

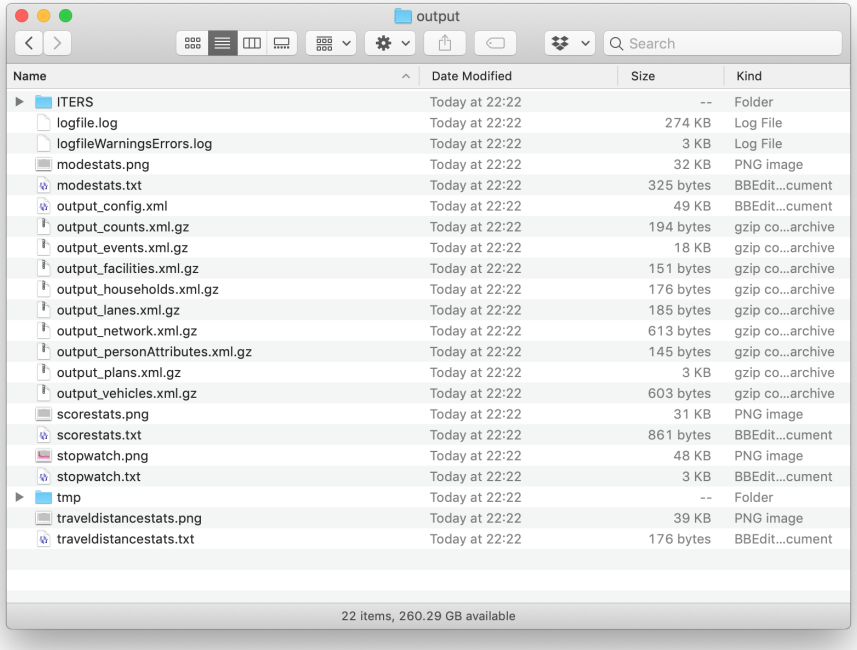
The MATSim-Release comes with several example scenarios:

equil	chessboard	pt-tutorial	siouxfalls-2014	berlin
				

Your First Simulation: Output

After the simulation has finished, have a look at the output directory. It contains:

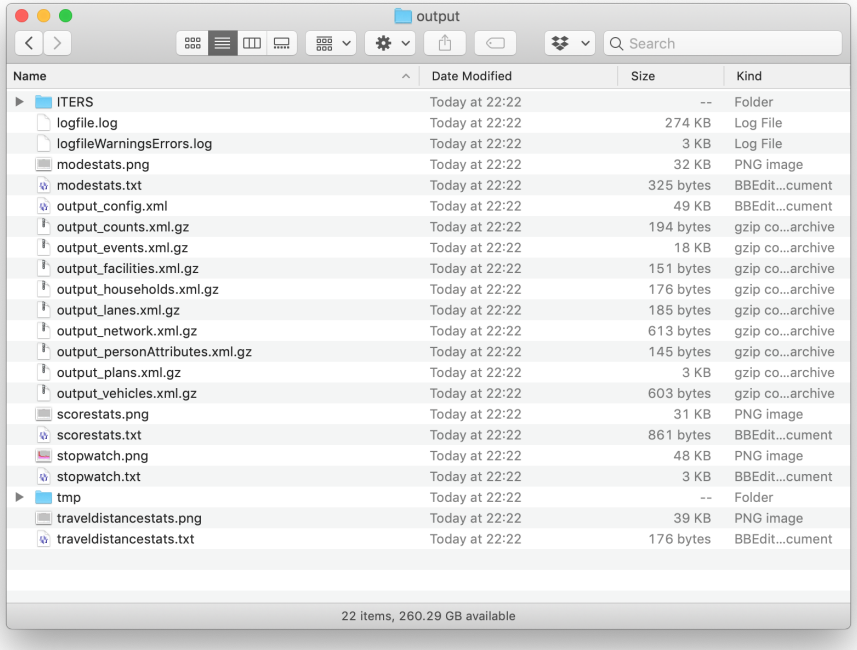
- **log messages:**
  logfile.log, logfileWarningsErrors.log
- **simple statistics:**
  scorestats, modestats, traveldistancestats, stopwatch
- **data files with state at the end of the simulation:**
  output_config, output_network, output_plans, output_events, ...
- **a directory containing data for each iteration**:
  ITERS



Your First Simulation: Output

The `ITERS` directory can usually be ignored.

Only the `output_*.xml.gz` files and the `logfile`s are typically of relevance.



Your First Simulation: Output

Most xml-files are compressed (gzip) as they get pretty large for real-world scenarios.

Some of the xml-files are empty when the feature is not used
(e.g. output_facilities, output_lanes, output_counts).

To "look" at the xml output files, they need to be unzipped (e.g. with 7-Zip).
Some text editors (e.g. BBEdit) support gzipped files and unzip them automatically.

Try this only with small scenarios, most text editors have problems with large files.

For large files (especially output_events), use command-line tools like `zmore` or `zgrep`.

Your First Simulation: Output

Have a look at the output_plans and output_config

- How do they differ from the input?

Check the logfile

- What transport modes are simulated?
- What model split is reported for the last iteration?

Check the other files:

- What does modestats.png show?
- What does scorestats.png show?

Your First Simulation: Output

output_events.xml is the main output.

In most cases, it is too large to load in a text editor.
Best is to process Events in Java.

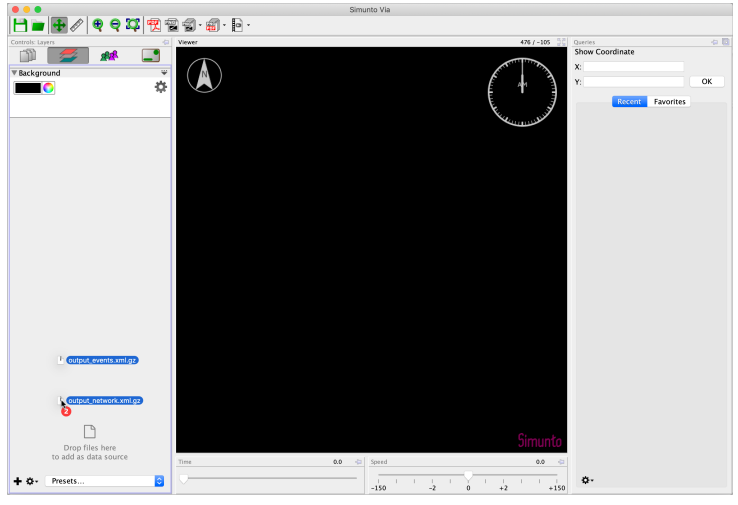
If you really need to look at the XML: use command-line tools like `zmore` or `zgrep`.

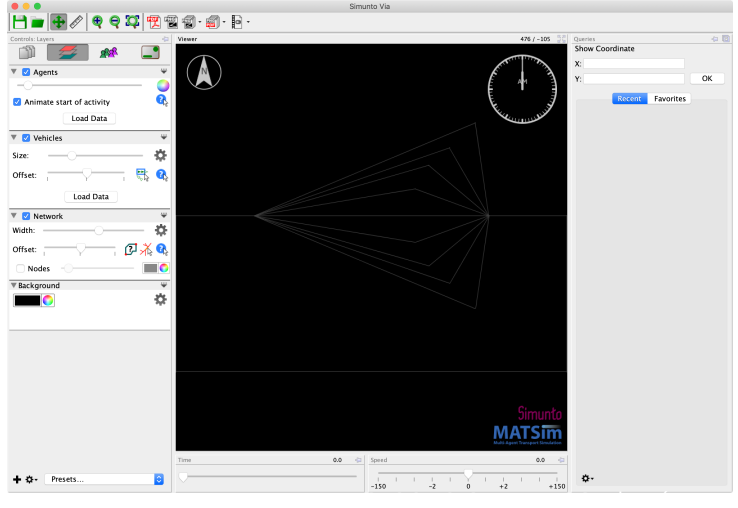
Try to look at output_events.xml.

- Can you find all events of a single agent?
- Do the events match the agent's plan?

Your First Simulation: Visualization

1. Start Via.
2. Drag and drop the files `output_network.xml.gz` and `output_events.xml.gz` to the left side of Via's window.
3. Create the layers `Network`, `Agents` and `Vehicles`.
4. Click on "Load Data".
5. Wait.
6. Change the time or animation speed.





Your First Simulation

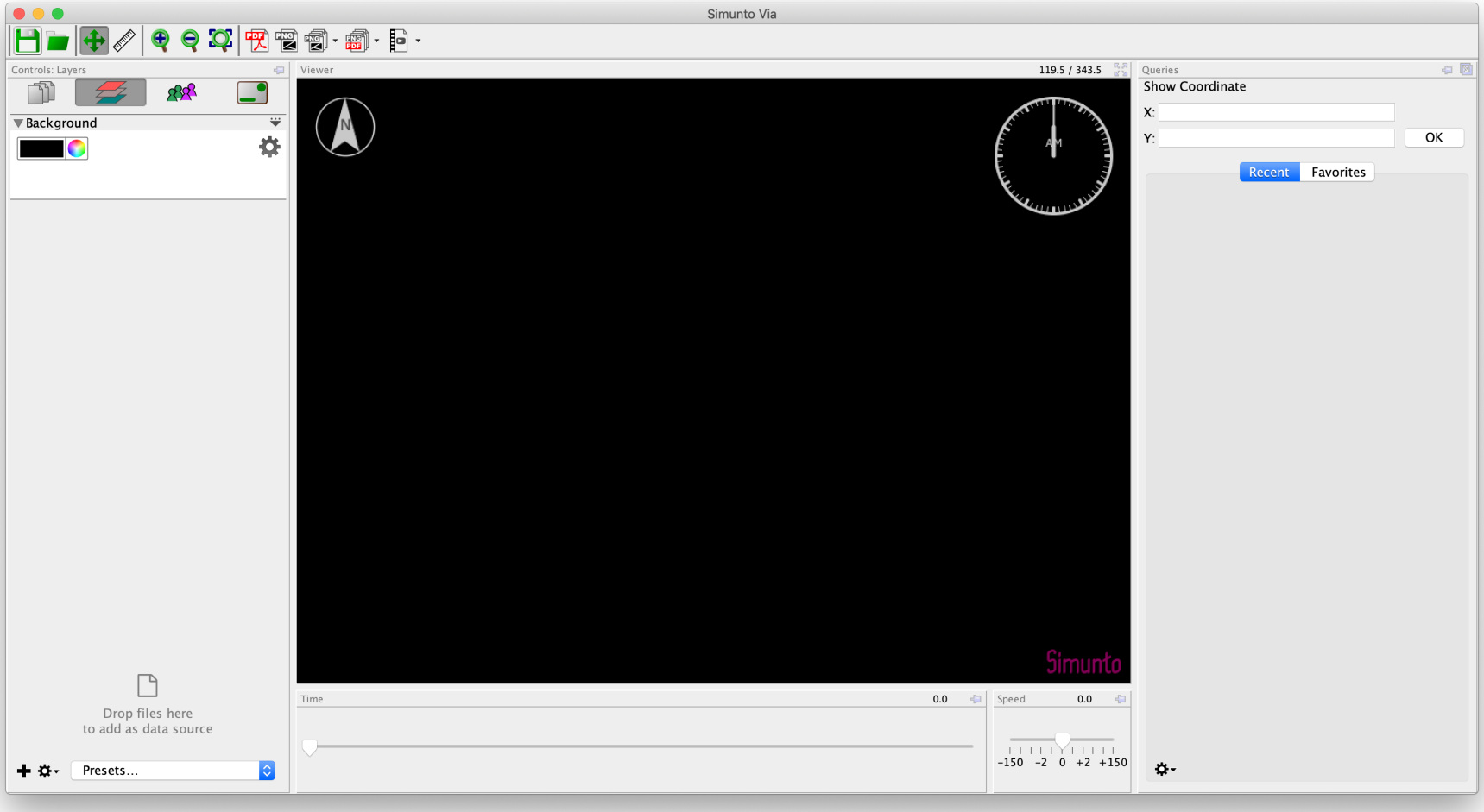
If you want to re-run the simulation, you need to delete (or rename) the output directory.

To visualize the new output, the easiest way is to quit and relaunch Via.
Or remove all layers and data sources and re-add them.

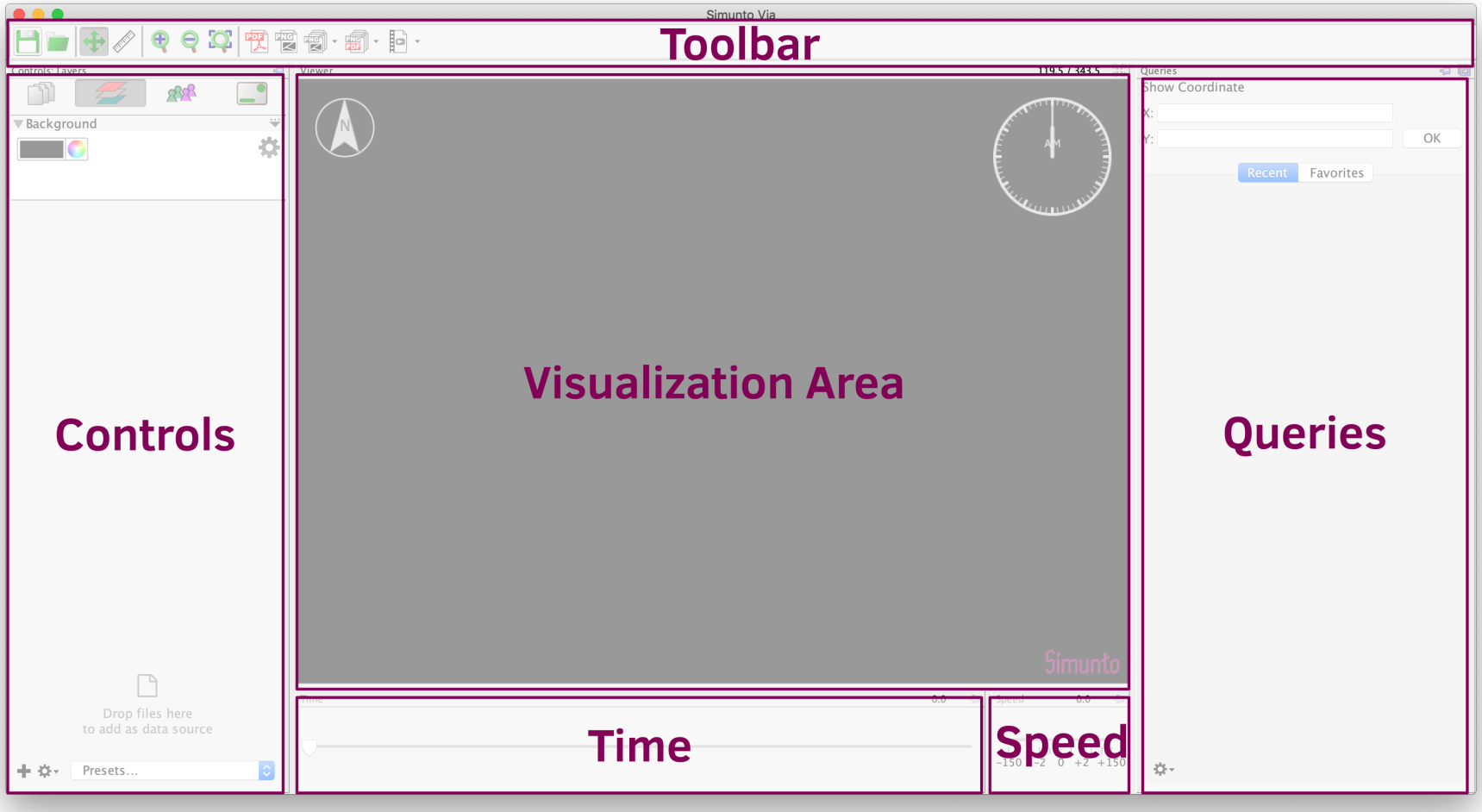
Run a more complex example and visualize it.

Via Crash Course

GUI Overview

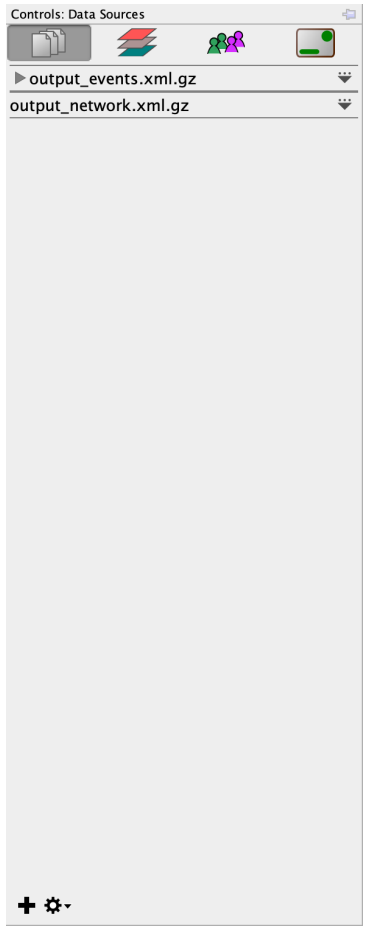


GUI Overview

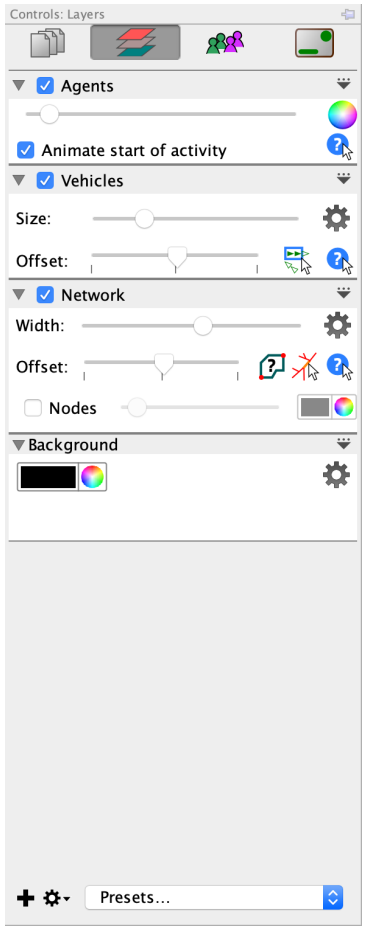


Controls Section

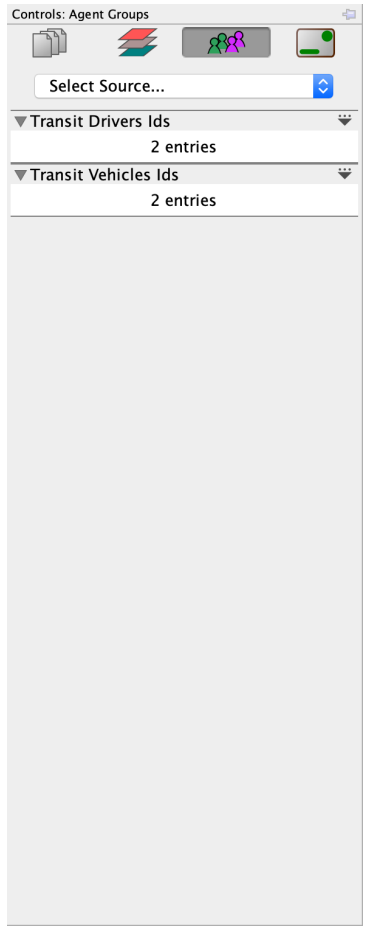
Data Sources



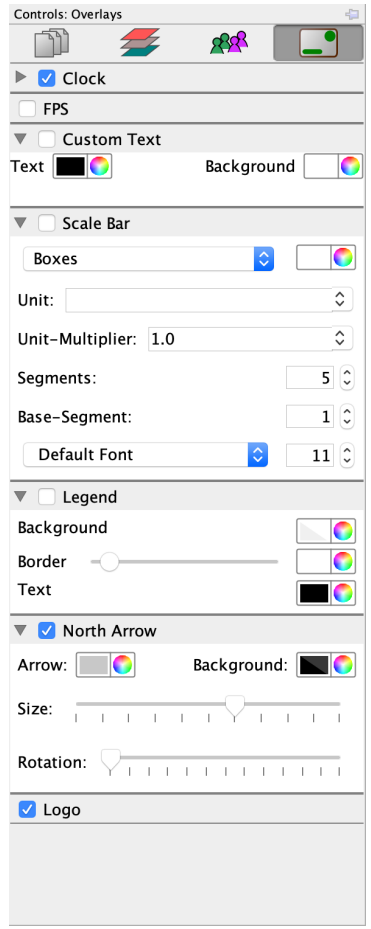
Layers



Agent Groups



Overlays



Adding Data Sources

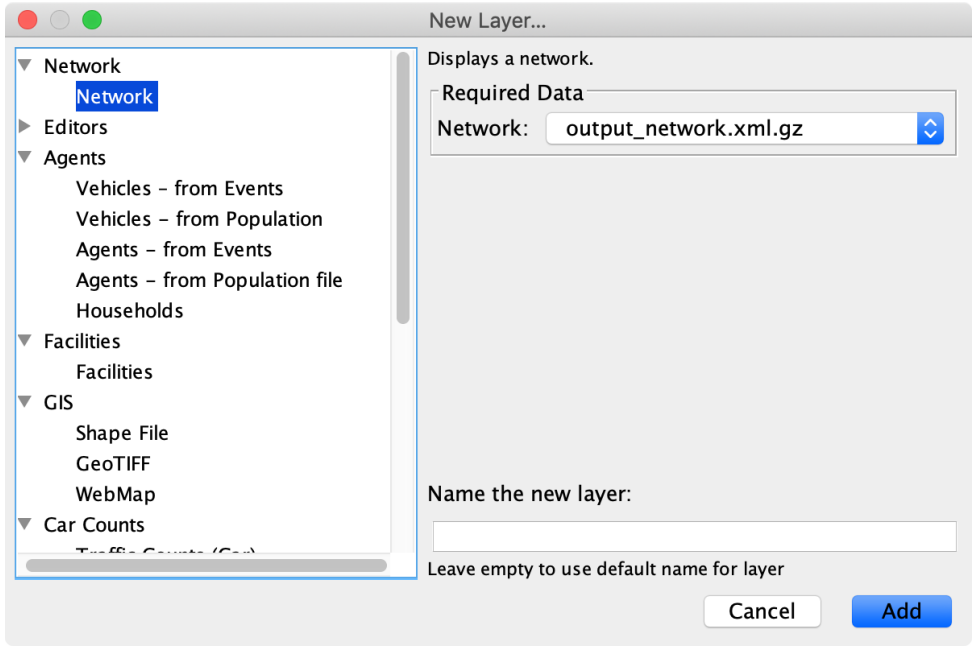
Several variants exist to add data to Via:

- Drag and drop files to Controls section
  (either on Data Sources or Layers).
- Menu `File > Add Data...` (Ctrl-D)
- Switch to "Data Sources" (Ctrl-1) and click on the `+`.

Adding Layers

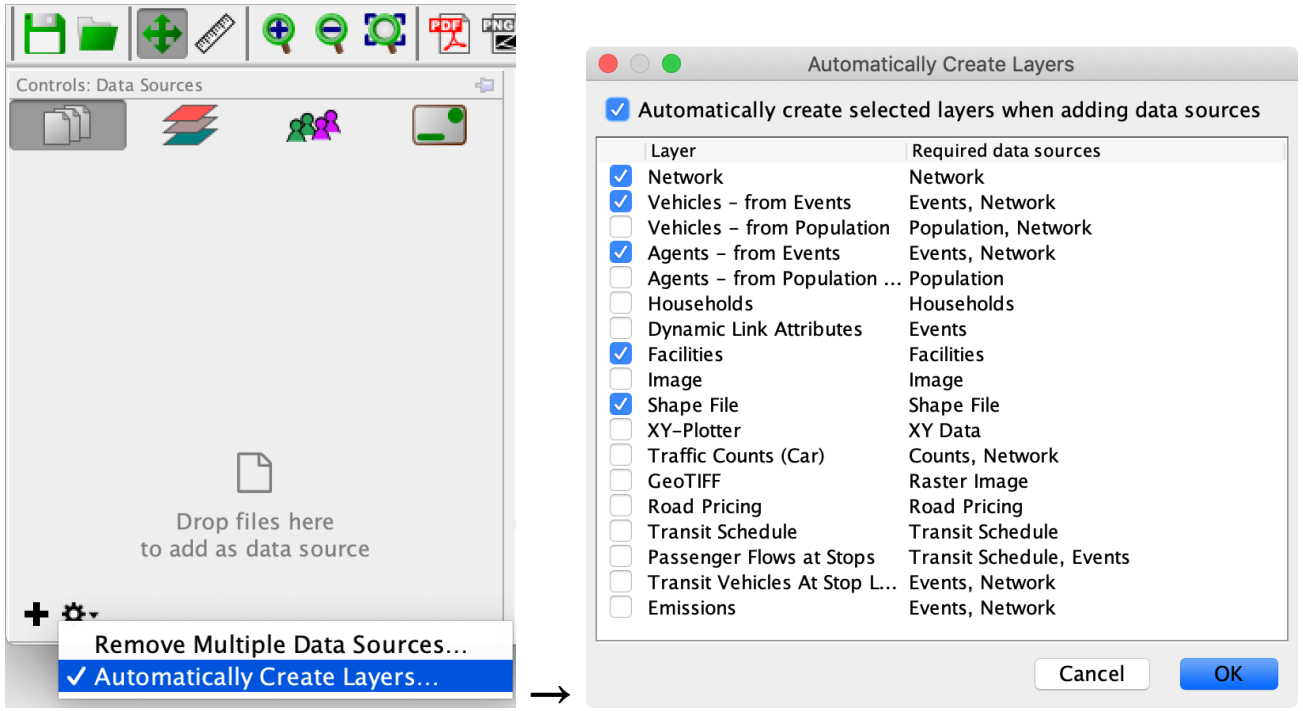
- Switch to "Layers" (Ctrl-2) and click on the `+`.
- Menu `File > Add Layer...` (Ctrl-L)

Select the right data sources and optionally give it a name.



Automatically create Layers

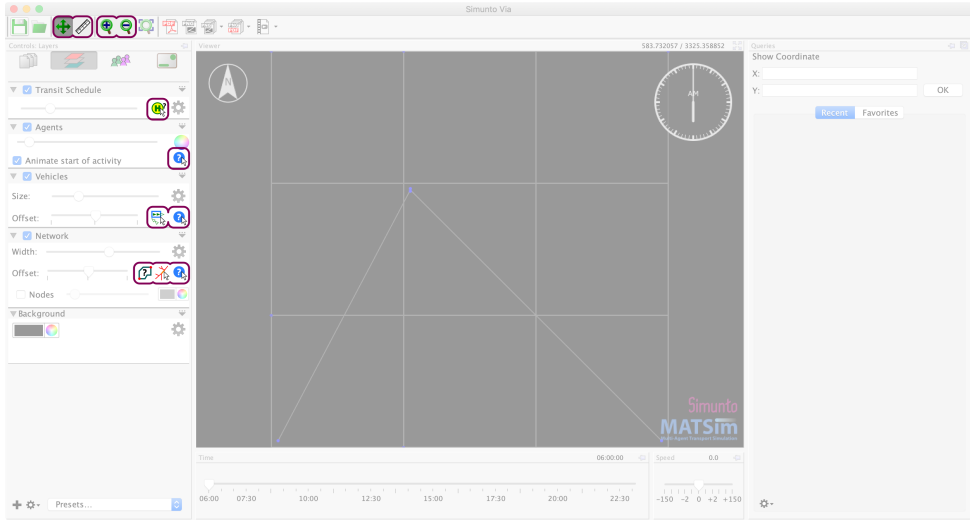
Adding files and layers to Via becomes a repetitive tasks.
Via can be configured to automatically create layers when files are added:



Queries

Layers provide data-specific queries.

Click on a query button to activate it,
then click on an element in the visualization
(e.g. a vehicle, a link, a transit stop, ...).



More about Via

Official Website: simunto.com/via

Video Tutorials and News about Via: simunto.com/news

Via-Manual: docs.simunto.com/via

Using MATSim from Java

Using MATSim from Java

Start with `org.matsim.run.RunMatsim` in the matsim-example-project:

```
public class RunMatsim {

  public static void main(String[] args) {
    String configFilename = args[0];
    Config config = ConfigUtils.loadConfig(configFilename);
    run(config);
  }

  static void run(Config config) {

    // possibly modify config here

    Scenario scenario = ScenarioUtils.loadScenario(config);

    // possibly modify scenario here

    Controler controler = new Controler(scenario);

    // possibly modify controler here

    controler.run();
  }
}
```

slightly modified example.

Using MATSim from Java

The GUI just calls this class to run MATSim, and passes it the selected config file.

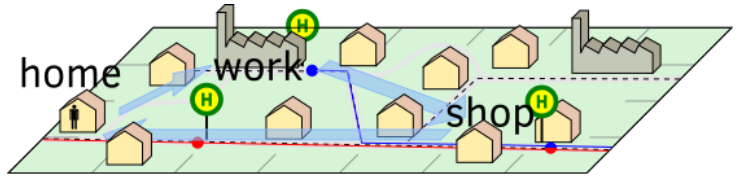
If you make changes to this class, these changes will also be applied when running MATSim with the GUI.

Try running MATSim from this class instead of the GUI.
You need to specify the config file in the run configuration.

MATSim Model

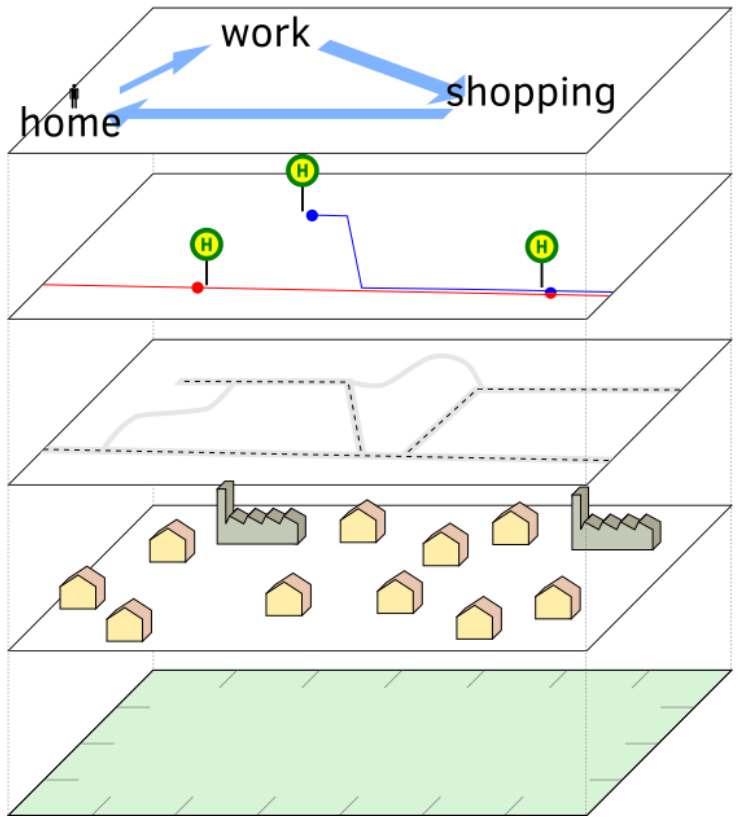
MATSim Model (Scenario Data)

A MATSim Model consists of several input files that specify the region, infrastructure, agents, and behaviour.



Minimally needed input:

- Configuration: config.xml
- Network: network.xml
- Demand (Agents): population.xml



Additional input, depending on required functionality:

- Facilities: facilities.xml
- Public Transport: transitSchedule.xml, transitVehicles.xml
- Lanes, Signal Systems: lanes.xml, signals.xml
- Counts comparison: counts.xml
- ...

MATSim Model: Data Containers

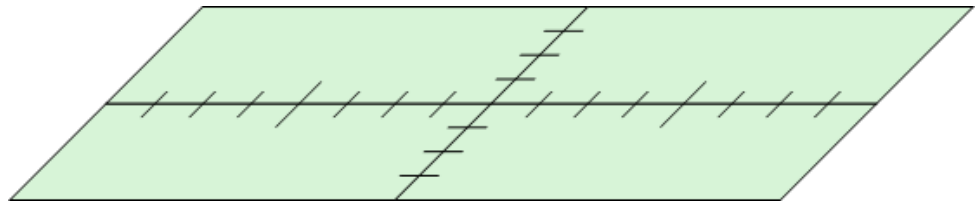
The main data structures like Network, Agents/Population, Facilities, Public Transport etc. are often referred to as "Data Containers".

MATSim Model: Coordinates

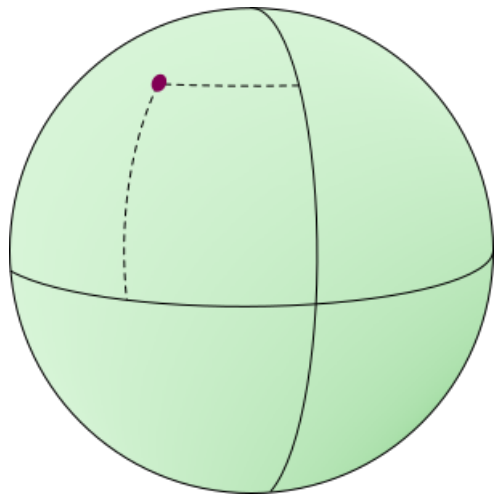
A MATSim model describes spatial objects.

MATSim assumes that coordinates are in

- a euclidean coordinate system
- with **meter** as unit



This allows MATSim to easily calculate distances (Pythagoras).
Many commonly used projections fullfil these criteria.



"GPS Coordinates" (WGS 84) do **not** fullfil these criteria and should not be used with MATSim!

— MATSim Model —

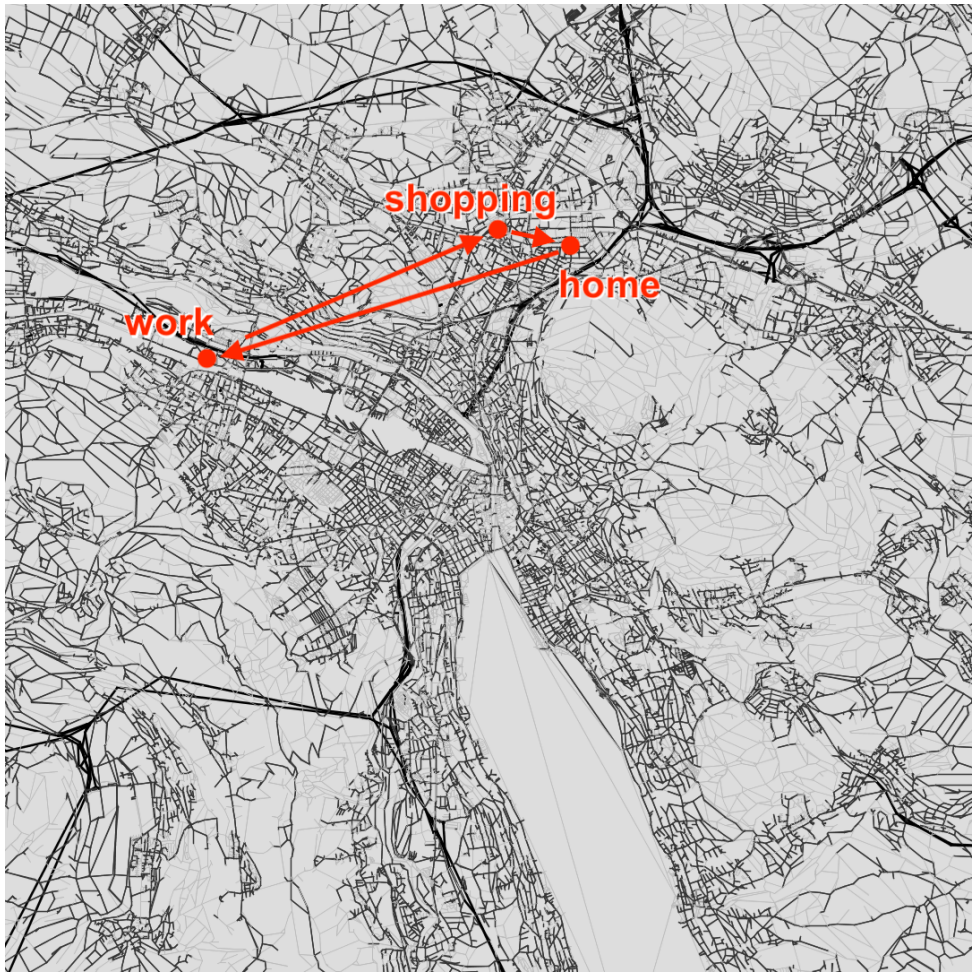
Demand

MATSim Model: Demand

Describes the agents and their plans.

Describes:

- Activity types
- Activity durations
- Activity locations
- Trip modes
- Trip routes



A single agent's plan

MATSim Model: Demand

Each Plan consists of a list of **Activities** with **Legs** in-between.

Each activity has a type, e.g. "work", "shop", "home".
Each leg has a mode, e.g. "car", "walk", "pt".

```
<plan>
  <activity type="home" x="890.0" y="685.0" end_time="07:43:56" />
  <leg mode="car" />
  <activity type="work" x="1802.0" y="2782.0" end_time="17:09:02" />
  <leg mode="car" />
  <activity type="shopping" x="1802.0" y="2782.0" end_time="17:42:02" />
  <leg mode="car" />
  <activity type="home" x="890.0" y="685.0" />
</plan>
```

How would you model a public transport trip which access and egress walk and transfers?

MATSim Model: Legs vs. Trips

MATSim's Legs are stages, not trips.
MATSim does not (yet) know the concept of trips (in its data structures).

To model complex trips, special "interaction" activities are used:

```
<plan>
  <activity type="home" x="890.0" y="685.0" end_time="07:43:56" />
  <leg mode="transit_walk" />
  <activity type="pt_interaction" x="912.0" y="423.0" duration="00:00:00" />
  <leg mode="pt" />
  <activity type="pt_interaction" x="1507.0" y="788.0" duration="00:00:00" />
  <leg mode="pt" />
  <activity type="pt_interaction" x="1834.0" y="2710.0" duration="00:00:00" />
  <leg mode="transit_walk" />
  <activity type="work" x="1802.0" y="2782.0" end_time="17:09:02" />
</plan>
```

The MATSim router automatically create and recognize these interaction activities, so that actually trips, and not stages, get routed. → `TripRouter`

MATSim Model: Legs vs. Trips

It is planned to replace such interaction activities with "waypoints" in the future.

Waypoints automatically have a duration of 0 seconds, will not get scored, and incur a lower overhead than activities.

```
<plan>
  <activity type="home" x="890.0" y="685.0" end_time="07:43:56" />
  <leg mode="transit_walk" />
  <waypoint x="912.0" y="423.0" />
  <leg mode="pt" />
  <waypoint x="1507.0" y="788.0" />
  <leg mode="pt" />
  <waypoint x="1834.0" y="2710.0" />
  <leg mode="transit_walk" />
  <activity type="work" x="1802.0" y="2782.0" end_time="17:09:02" />
</plan>
```

This is currently in development, final version might look and behave differently!

MATSim Model: Activity Types

Activity types can be freely chosen, but must be defined in the configuration (for scoring).
Often used types:

- **home**
- **work**
  sometimes differentiated as work_ft (full time), work_pt (part time)
- **education**
  often as edu1, edu2, edu3; edu_primary, edu_secondary, edu_tertiary; edu, edu_higher
- **shopping**
  sometimes differentiated as shop_small, shop_large, or shop_daily, shop_other
- **leisure**
- **errands**
  e.g. doctor's visit, administrative tasks
- **accompanying**
  e.g. bring kid to / pick-up kid from day care

MATSim Model: Activity Types

Additional activity types might be used for special model features:

- **freight**
  as start and end activity of a fixed freight demand as background traffic
- **service**
  work-related activities outside the own office (e.g. craftsman)
- **cb_***
  "cross-border" activities, activities outside the model region by agents travelling into/out of the modelled region.

MATSim Model: ~~Trip~~ Leg Modes

Leg Modes can be freely chosen, but must be defined in the configuration
(for routing and scoring).
Often used types:

- **car**
- **walk**
- **bike**
  bicycle, not a motorbike
- **pt**
  public transport, transit; could be train, bus, tram, ship, ...
- **transit_walk**, **access_walk**, **egress_walk**
  alternative walk modes, typically used along interaction activities
- **ride**
  use a private car as a passenger, "ride along"

MATSim Model: ~~Trip~~ Leg Modes

Additionally, the following types are regularly used:

- **truck**
  mostly for freight vehicles
- **taxi**
- **drt**
  demand responsive transport, e.g. shared cabs

MATSim defines several of these modes in `org.matsim.api.core.v01.TransportMode`.

MATSim Model: Leg Modes

The simulation can handle legs of different modes differently:

- **network simulation**
  vehicles are moved around the network, taking physical constraints (max speed, flow capacity, ...) into account.
- **teleportation**
  agents disappear at one location, re-appear at a certain time at another location.
- **public transport**
  agents can board a transit vehicle at a stop location, and can alight at another stop.
- **custom**
  MATSim extensions like taxi, drt implement support for their own modes, e.g. to ride as a passanger in a (non-transit) vehicle.

MATSim Model: Leg Modes

How a mode (e.g. "car", "walk") is handled is model-specific.

For the standard types (network, teleportation, pt) it can be configured in config.xml.

MATSim Model: Activity Locations

By default, an activity should have an x- and y-coordinate value:

```
<activity type="home" x="890.0" y="685.0" end_time="..." />
```

Alternatively, an activity could refer to a facility instead:

```
<activity type="home" facility="8224" end_time="..." />
```

MATSim Model: Activity Locations

The mobsim needs to know on what link a car should depart for this activity.

For network-simulated modes (and maybe others), the mobsim needs a link for each activity:

```
<activity type="home" link="3791" end_time="..." />
```

If the link-attribute is missing, the nearest link based on x/y or facility is taken automatically.

MATSim Model: Activity Locations

Activities take place on links.

Many other models use nodes as source/sink of traffic.
MATSim uses links as source/sink of traffic.

QSim has all vehicle interaction (insert/remove/pt-stop) on links close to the downstream node ("to-node").

— MATSim Model —

MATSim Configuration

MATSim Configuration

The configuration specifies MATSim's behaviour.

By default, the configuration file is passed as first (and only) argument when running MATSim.

The `config.xml` is the main "interface" for simple MATSim usage.

This works as long as no MATSim extensions are used.
Many extensions need to be enabled in code, but can then also be configured in the config.xml.

MATSim Configuration

The configuration consists of nested groups with simple key-value pairs ("parameters").

```
- config
  - global
      - numberOfThreads = 2
      - coordinateSystem = EPSG:25832
  - controler
      - firstIteration = 0
      - lastIteration = 100
      - outputDirectory = ./output
      - writeEventsInterval = 10
      - ...
  - network
      - inputNetworkFile = input/network.xml.gz
  - ...
```

MATSim Configuration: XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config SYSTEM "http://www.matsim.org/files/dtd/config_v2.dtd">
<config>

  <module name="global">
    <param name="numberOfThreads" value="2" />
    <param name="coordinateSystem" value="EPSG:25832" />
  </module>

  <module name="controler">
    <param name="firstIteration" value="0" />
    <param name="lastIteration" value="100" />
    <param name="outputDirectory" value="./output/" />
    <param name="writeEventsInterval" value="10" />
  </module>

  <module name="network">
    <param name="inputNetworkFile" value="input/network.xml.gz" />
  </module>

</config>
```

MATSim Configuration: Parameter Sets

Some parameter groups (modules) contain groups themselves: "Parameter Sets":

```
<module name="planCalcScore">
    <parameterset type="scoringParameters" >
        <param name="subpopulation" value="null">

        <parameterset type="activityParams" >
            <param name="activityType" value="home" />
            <param name="typicalDuration" value="10:00:00" />
        </parameterset>
        <parameterset type="activityParams" >
            <param name="activityType" value="work" />
            <param name="typicalDuration" value="8:00:00" />
        </parameterset>

        <parameterset type="modeParams" >
            <param name="mode" value="car" />
            <param name="marginalUtilityOfTraveling_util_hr" value="-6.0" />
        </parameterset>
        <parameterset type="modeParams" > ... </parameterset>
    </parameterset>

    <parameterset type="scoringParameters" >
        <param name="subpopulation" value="freight">
        ...
    </parameterset>
</module>
```

MATSim Configuration: Parameters and Defaults

What parameters and parameter-values are available?

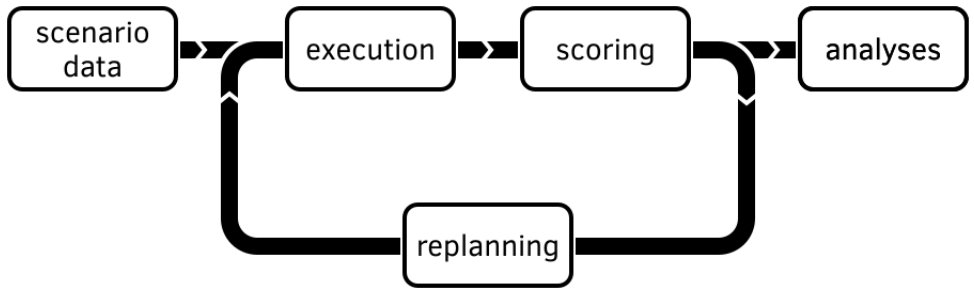
- `output_config.xml` should contain all parameters with their current values.
- The configuration is also written to the logfile when the simulation starts.
- The MATSim GUI can create a "default config"
  (Menu `Tools > Create Default config.xml…`).
- In the code, an empty Config-object could be created and written out, which contains all the defaults and comments:

```
Config config = ConfigUtils.createConfig();
new ConfigWriter(config, ConfigWriter.Verbosity.all).write("defaultConfig.xml");
```

MATSim Configuration

What needs to be configured?

- input files locations
  ("scenario data", "data containers")
- computational/performance settings
  (e.g. number of threads)
- scoring parameters
- replanning strategies
- replanning strategies' behaviour
- analyses/output



MATSim Configuration: Module-Names

Input files locations

`counts`, `facilities`, `households`, `network`, `plans`, `ptCounts`, `transit`, `vehicles`

Computational / Performance settings

`controler`, `global`, `parallelEventHandling`, `qsim`

Scoring parameters

`planCalcScore`

Replanning strategies

`strategy`

Replanning strategies' behaviour

`TimeAllocationMutator` (dep. time choice)
`changeMode` (mode choice)
`planscalcroute` (route choice)
`subtourModeChoice` (mode choice)
`transitRouter` (transit route choice)

Analyses, Output

`counts`, `linkStats`, `ptCounts`, `travelTimeCalculator`

MATSim Configuration

Have a look at the (output_)config.xml

- Where is the output written to?
  → controler, outputDirectory
- Can you figure out which modes are teleported?
  → planscalcroute, parameterset with type "teleportedModeParameters"
  → every mode not otherwise configured (not on network, not as pt)
- What modes are simulated on the network?
  → qsim, mainMode
- How long does the simulation run?
  → qsim, endTime
- How many iterations are run?
  → controler, lastIteration

MATSim Configuration: Useful Parameters

controler.runId
prefixes all output-files with the specified string to better differentiate the different runs.

qsim.endTime
set it to "30:00:00" or a similar value to prevent the simulation taking ages due to agents trying to walk 200 kilometers...

global.numberOfThreads
Set the number of threads to be used by the replanning. Replanning scales nearly linearly.

strategy.maxAgentPlanMemorySize
Controls the size of the agent's choice set. And the memory consumption of the simulation.

qsim.flowCapacityFactor, qsim.storageCapacityFactor
Reduce road capacities if only a sample should be simulated.

MATSim Configuration

Potential misconfiguration problems

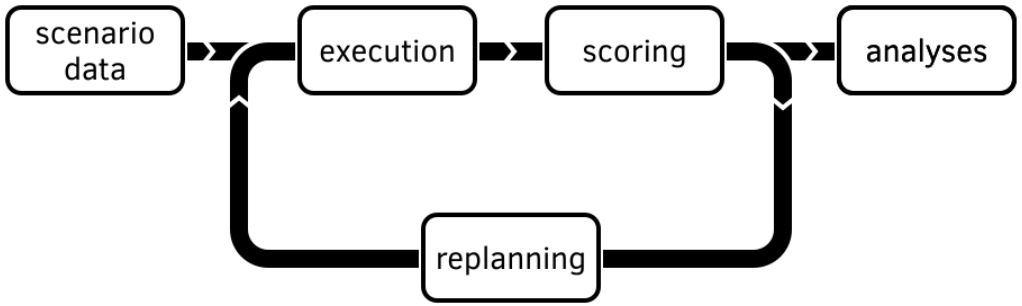
- **mismatched route types**
  "planscalcroute.networkModes" configures for which modes routes on the network are calculated.
  "qsim.mainMode" configures which modes are simulated on the network.

- **unknown transport modes**
  All used transport modes should have scoring parameters.
  This includes the modes listed for mode choice.

- **sample-factors do not match**
  To simulate a 10% sample of the population, qsim.flowCapacityFactory should be set to `0.1` (=10%), but counts.countsScaleFactor should be set to `10` (=1/10%).

MATSim Controler

MATSim Controler



Something has to steer that iteration loop.

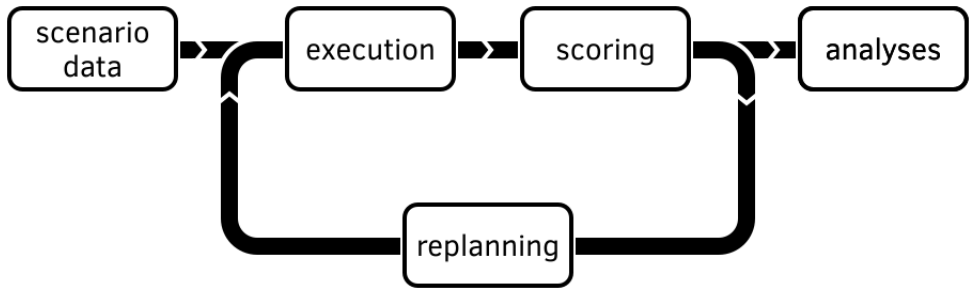
→ Controler [1]

[1] Mis-spelled since its inception.

MATSim Controler

The Controler is responsible for:

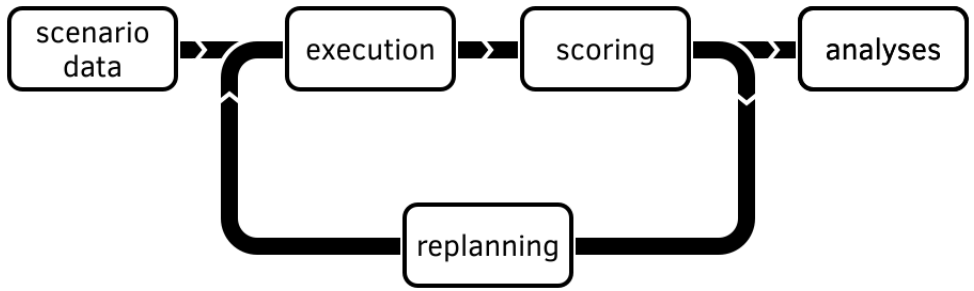
- optionally load scenario
- prepare scenario for simulation
- manage the iterative loop, calling the right "modules" at the right time
  - execution
  - scoring
  - replanning
- write out data at the end (or inbetween)



MATSim Controler

The Controler must be flexible:

- support additional replanning strategies
- support custom scoring functions
- support custom analyses
- support additional mobsim features

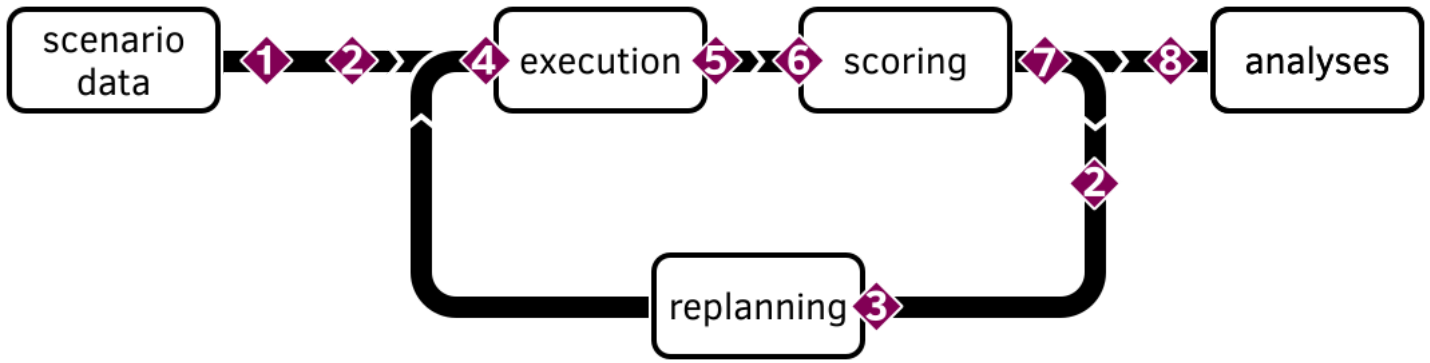


It uses two concepts to achieve this:

- ControlerListener (Extension Points)
- Dependency Injection

MATSim Controler: Extension Points

The Controler provides possibilities to run custom code at certain points in the iterative loop:



1 Startup
2 Iteration Starts
3 Replanning
4 Before Mobsim
5 After Mobsim
6 Scoring
7 Iteration Ends
8 Shutdown

Replanning and Scoring are just specific implementations for the provided extension points.

MATSim Controler: Extension Points

Code to be run at one or more of these extension points are called **ControlerListeners**.

At each extension point, the Controler sends out a **ControlerEvent** that the listeners can react upon.

MATSim Controler: Dependency Injection

The Controler has dependencies. It needs:

- a mobsim
- a list of available replanning strategies
- a list of routing algorithms
- and other things...

These dependencies can be set using "dependency injection".

Think of it as a black box that provides a global lookup table to provide objects based on a class, and optionally on a name.

The concept will become more clear when writing code.

MATSim Controler

To essentially extend or adapt MATSim, one has to (re-)configure the Controler.

This requires writing code in Java.

Using MATSim from Java

Using MATSim from Java

Start with `org.matsim.run.RunMatsim` in the matsim-example-project:

```
public class RunMatsim {

  public static void main(String[] args) {
    String configFilename = args[0];
    Config config = ConfigUtils.loadConfig(configFilename);
    run(config);
  }

  static void run(Config config) {

    // possibly modify config here

    Scenario scenario = ScenarioUtils.loadScenario(config);

    // possibly modify scenario here

    Controler controler = new Controler(scenario);

    // possibly modify controler here

    controler.run();
  }
}
```

slightly modified example.

Controler Listener

Create a new class `MyControlerListener`

```
import org.apache.log4j.Logger;
import org.matsim.core.controler.events.*;
import org.matsim.core.controler.listener.*;

public class MyControlerListener implements StartupListener, ShutdownListener, IterationStartsListener, ScoringListener
  private final static Logger log = Logger.getLogger(MyControlerListener.class);

  @Override
  public void notifyIterationStarts(IterationStartsEvent iterationStartsEvent) {
    log.info("What will happen in iteration " + iterationStartsEvent.getIteration() + "?");
  }

  @Override
  public void notifyScoring(ScoringEvent scoringEvent) {
    log.info("Let's evaluate!");
  }

  @Override
  public void notifyStartup(StartupEvent startupEvent) {
    log.info("We're starting!");
  }

  @Override
  public void notifyShutdown(ShutdownEvent shutdownEvent) {
    log.info("Bye, it's over!");
  }
}
```

Controler Listener

Integrate your ControlerListener into `RunMatsim`

```
public class RunMatsim {

  public static void main(String[] args) {
    String configFilename = args[0];
    Config config = ConfigUtils.loadConfig(configFilename);
    run(config);
  }

  static void run(Config config) {

    // possibly modify config here

    Scenario scenario = ScenarioUtils.loadScenario(config);

    // possibly modify scenario here

    Controler controler = new Controler(scenario);

    // possibly modify controler here
    controler.addControlerListener(new MyControlerListener());

    controler.run();
  }
}
```

Controler Listener

The order in which controler listeners are executed cannot be guaranteed.
If you add multiple listeners for the same event, it is not defined which one will run first.

Run MATSim with your controler listener.
Do you find the log-statements in the output?

Keep `MyControlerListener` and `RunMatsim` as a template for later.
We'll add code to this in later sessions.

MATSim API

MATSim API

```
Config config = ConfigUtils.loadConfig(configFilename);

Scenario scenario = ScenarioUtils.loadScenario(config);

Controler controler = new Controler(scenario);

controler.run();
```

- Config
- Scenario
- Controler

MATSim API: Config

Create or load a config object:

```
Config config1 = ConfigUtils.createConfig();

Config config2 = ConfigUtils.loadConfig("/path/to/config.xml");
```

Create a default config and write it out:

```
Config config = ConfigUtils.createConfig();
new ConfigWriter(config, ConfigWriter.Verbosity.all).write("defaultConfig.xml");
```

MATSim API: Config

`Config` provides access to the different groups and parameters.

Examples:

```
config.controler().setLastIteration(10);
int eventsInterval = config.controler().getWriteEventsInterval();

config.network().setInputFile("/path/to/network.xml");

config.planCalcScore().getScoringParameters(subpopulation).addActivityParams(...);
config.planCalcScore().getScoringParameters(subpopulation).addModeParams(...);
```

MATSim API: Scenario

Create a scenario:

```
Scenario scenario = ScenarioUtils.createScenario(config);
```

Create and load a scenario:

```
Scenario scenario = ScenarioUtils.loadScenario(config);
```

MATSim API: Scenario

`Scenario` provides access to data containers:

```
Network network = scenario.getNetwork();

Population population = scenario.getPopulation();

ActivityFacilities facilities = scenario.getActivityFacilities();

TransitSchedule schedule = scenario.getTransitSchedule();
```

MATSim API: Scenario

By default, a Scenario is immutable.
Its data containers cannot be replaced.

```
scenario.setNetwork(myNetwork); // this will not compile
```

But the content of data containers can be modified.

```
scenario.getNetwork().addLink(myLink); // this works
```

MATSim API: Controler

Create a controler:

```
import org.matsim.core.controler.Controler; // there is another Controler class, use this

Controler controler = new Controler(scenario);
```

Run the simulation:

```
controler.run();
```

MATSim API: Controler

`Controler` provides access to important data structures.

```
Config config = controler.getConfig();
Scenario scenario = controler.getScenario();

EventsManager events = controler.getEvents();

Integer iteration = controler.getIterationNumber(); // might be null in startup or shutdown phase
```

MATSim API: ControlerIO

`ControlerIO` provides standardized access to (output) file paths.
It is available from Controler as well as from ControlerEvents.

```
String filename = controler.getControlerIO().getIterationFilename(iteration, "myOutput.txt");
// write data to 'filename'
```

```
public class MyControlerListener implements IterationEndsListener {

  @Override
  public void notifyIterationEnds(IterationEndsEvent event) {
    String filename = event.getServices().getControlerIO().getIterationFilename(event.getIteration(), "myOutput.txt");
    // e.g. write collected data to 'filename'
  }
  ...
}
```

MATSim API: IOUtils

MATSim provides a helper class `IOUtils` to work with (gzipped) files.
It automatically (de)compresses gzipped files if the filename ends on `.gz`:

```
String filename = "path/to/data.txt";
String filename = "path/to/data.txt.gz";

try (BufferedWriter writer = IOUtils.getBufferedWriter(filename)) {
  writer.write("Hello MATSim!");
} catch (IOException e) {
  log.error("oops!", e);
}
```

Read data:

```
String filename = "path/to/data.txt";
String filename = "path/to/data.txt.gz";

try (BufferedReader reader = IOUtils.getBufferedReader(filename)) {
  String line = reader.readLine();
  log.info(line);
} catch (IOException e) {
  log.error("oops!", e);
}
```

MATSim API: Data Containers

Use Data Containers (network, population, ...) from a scenario,
or use `*Utils`-classes to create them:

```
Scenario scenario = ScenarioUtils.createScenario(ConfigUtils.createConfig());
Network network = scenario.getNetwork();
Population population = scenario.getPopulation();
```

```
Network network = NetworkUtils.createNetwork();
Population population = PopulationUtils.createPopulation(config);
ActivityFacilities facilities = FacilitiesUtils.createActivityFacilities();

// exception from the *Utils-pattern:
TransitSchedule schedule = new TransitScheduleFactoryImpl().createTransitSchedule();
```

MATSim API: Data Containers

Reading and Writing data containers:

```
Scenario scenario = ScenarioUtils.createScenario(ConfigUtils.createConfig());

Network network = scenario.getNetwork();
Population population = scenario.getPopulation();
ActivityFacilities facilities = scenario.getActivityFacilities();
TransitSchedule schedule = scenario.getTransitSchedule();

new MatsimNetworkReader(network).readFile("/path/to/network.xml.gz");
new NetworkWriter(network).write("/path/to/network.xml.gz");

new PopulationReader(scenario).readFile("/path/to/population.xml.gz");
new PopulationWriter(population).write("/path/to/population.xml.gz");

new MatsimFacilitiesReader(scenario).readFile("/path/to/population.xml.gz");
new FacilitiesWriter(facilities).write("/path/to/population.xml.gz");

new TransitScheduleReader(scenario).readFile("/path/to/transitSchedule.xml.gz");
new TransitScheduleWriter(schedule).writeFile("/path/to/transitSchedule.xml.gz");
```

Sadly, the API is not consistent.

MATSim API: Data Containers, Ids

Elements in data containers have `Id`s for identification.

Ids have a type.

```
Id<Link>    linkId;
Id<Node>    nodeId;
Id<Person>  personId;
```

The type must be specified when creating an Id:

```
Id<Person> personId = Id.create("123", Person.class);

Id<Node> nodeId = Id.create("456", Node.class);

Id<MyClass> myId = Id.create("789", MyClass.class);
```

MATSim API: Data Containers, Factories

Use Factories to modify create new elements for data containers. Each data container provides its own factory.

Don't forget to add the created elements to the data container.

```
NetworkFactory netF = network.getFactory();

Node node1 = netF.createNode(Id.create("1", Node.class), new Coord(100, 300));
Node node2 = netF.createNode(Id.create("2", Node.class), new Coord(200, 400));

network.addNode(node1);
network.addNode(node2);

Link link = netF.createLink(Id.create("1", Link.class), node1, node2);
link.setFreespeed(50.0 / 3.6); // 50 km/h in m/s
link.setLength(300); // meter
link.setCapacity(800); // veh / hour
link.setNumberOfLanes(1);
link.setAllowedModes(CollectionUtils.stringToSet("car,bus"));

network.addLink(link);
```

MATSim API: Data Containers, Factories

```
PopulationFactory popF = population.getFactory();

Person person = popF.createPerson(Id.create("12", Person.class));
Plan plan = popF.createPlan();

Activity act1 = popF.createActivityFromCoord("home", new Coord(150, 250));
act1.setEndTime(8 * 3600);
Leg leg = popF.createLeg("car");
Activity act2 = popF.createActivityFromCoord("work", new Coord(750, 650));

plan.addActivity(act1);
plan.addLeg(leg);
plan.addActivity(act2);

person.addPlan(plan);

population.addPerson(person);
```

MATSim API: Data Containers, Attributes

Attributes can often be stored along objects, or externally using the Id for lookup.

```
// "internal"
person.getAttributes().putAttribute("age", 39);
int age = (Integer) person.getAttributes().getAttribute("age");
```

```
// "external", "ObjectAttributes"
String personId = person.getId().toString();
ObjectAttributes attributes = population.getPersonAttributes();
attributes.putAttribute(personId, "age", 39);
int age = (Integer) attributes.getAttribute(personId, "age");
```

MATSim switches more and more to "internal" attributes.

Know where you store your attributes when accessing them.

MATSim API: Time

MATSim internally uses "seconds after midnight".

To convert to and from human-readable time formats, use the `Time` class:

```
double seconds = Time.parseTime("07:12:35");

String time = Time.writeTime(43200);
```

MATSim does not know days, it just keeps counting the hours (seconds, respectively).

After "24:00:00" comes "24:00:01" ... "25:00:00" ...

Creating a Network

MATSim Network

A very simple network in XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE network SYSTEM "http://www.matsim.org/files/dtd/network_v2.dtd">
<network>
  <nodes>
    <node id="1" x="2000" y="1000" />
    <node id="2" x="4000" y="1500" />
    <node id="3" x="3000" y="3000" />
  </nodes>
  <links capperiod="01:00:00">
    <link id="1" from="1" to="2" length="3000.0" capacity="1800" freespeed="13.88" permlanes="1" modes="car" />
    <link id="2" from="2" to="1" length="3000.0" capacity="1800" freespeed="13.88" permlanes="1" modes="car" />
    <link id="3" from="2" to="3" length="5000.0" capacity="3500" freespeed="22.22" permlanes="2" modes="car" />
    <link id="4" from="3" to="2" length="5000.0" capacity="1800" freespeed="22.22" permlanes="1" modes="car" />
  </links>
</network>
```

Coordinates: should be in a euclidean coordinate system with *meter* as unit

Link-Length: in *meter*

Link-Capacity: in *vehicles per hour* (see `capperiod` attribute, should always be set to 1 hour)

Link-Freespeed: in *meter per second*

Links are unidirectional

MATSim Network with Attributes

Nodes and Links can have arbitrary attributes

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE network SYSTEM "http://www.matsim.org/files/dtd/network_v2.dtd">
<network>
  <nodes>
    <node id="1" x="2000" y="1000">
      <attributes>
        <attribute name="signalized" class="java.lang.Boolean">true</attribute>
      </attributes>
    </node>
    ...
  </nodes>
  <links capperiod="01:00:00">
    <link id="1" from="1" to="2" length="3000.0" capacity="1800" freespeed="13.88" permlanes="1" modes="car">
      <attributes>
        <attribute name="name" class="java.lang.String">Main Street</attribute>
        <attribute name="isTunnel" class="java.lang.Boolean">false</attribute>
        <attribute name="category" class="java.lang.Integer">3</attribute>
      </attributes>
    </link>
    ...
  </links>
</network>
```

`signalized`, `name`, `isTunnel`, `category` are exemplary names and have no meaning in MATSim

Creating a network

Write XML by hand

- Only practical for very small test scenarios

Use MATSim API

- Only practical for test scenarios

Convert from existing data

- Use existing tool (GUI) to convert from OSM (OpenStretMap)
- Use MATSim API to create a network based on existing data
- Stable converters exist for OSM
- Experimental converters exist for VISUM, EMME, TeleAtlas.

Creating a network: MATSim API

```
Config config = ConfigUtils.createConfig();
Scenario scenario = ScenarioUtils.createScenario(config);
Network network = scenario.getNetwork();
NetworkFactory factory = net.getFactory();

// create nodes
Node node0 = factory.createNode(Id.createId(0, Node.class), new Coord(2000, 1000));
network.addNode(node0);
Node node1 = factory.createNode(Id.createId(1, Node.class), new Coord(4000, 1500));
network.addNode(node1);

// create links
Link link1 = factory.createLink(Id.createId("0_1", Link.class), node0, node1);
link1.setCapacity(1800.0);
link1.setLength(3000.0);
link1.setFreespeed(13.88);
link1.setNumberOfLanes(1);
link1.setAllowedModes(CollectionUtils.stringToSet("car,bus"));
net.addLink(link1);

// write network
new NetworkWriter(net).write("output/network.xml");
```

matsim-code-examples/src/main/java/tutorial/network/createNetworkExample/RunCreateNetworkExample.java

Creating a network: Converting from OSM

```
CoordinateTransformation wgs84_utm32N = TransformationFactory.getCoordinateTransformation("EPSG:4326", "EPSG:25832");
Network network = NetworkUtils.createNetwork();

OsmNetworkReader reader = new OsmNetworkReader(network, wgs84_utm32N);

reader.setHierarchyLayer(51.9, 9.5, 51.2, 10.3, 3); // 3 = primary roads
reader.setHierarchyLayer(51.6, 9.8, 51.5, 10.0, 8); // 8 = residential roads
reader.setKeepPaths(true);
reader.setMemoryOptimization(true);

reader.parse("/path/to/niedersachsen.osm");

new NetworkWriter(network).write("output/network_detailed.xml.gz");
```

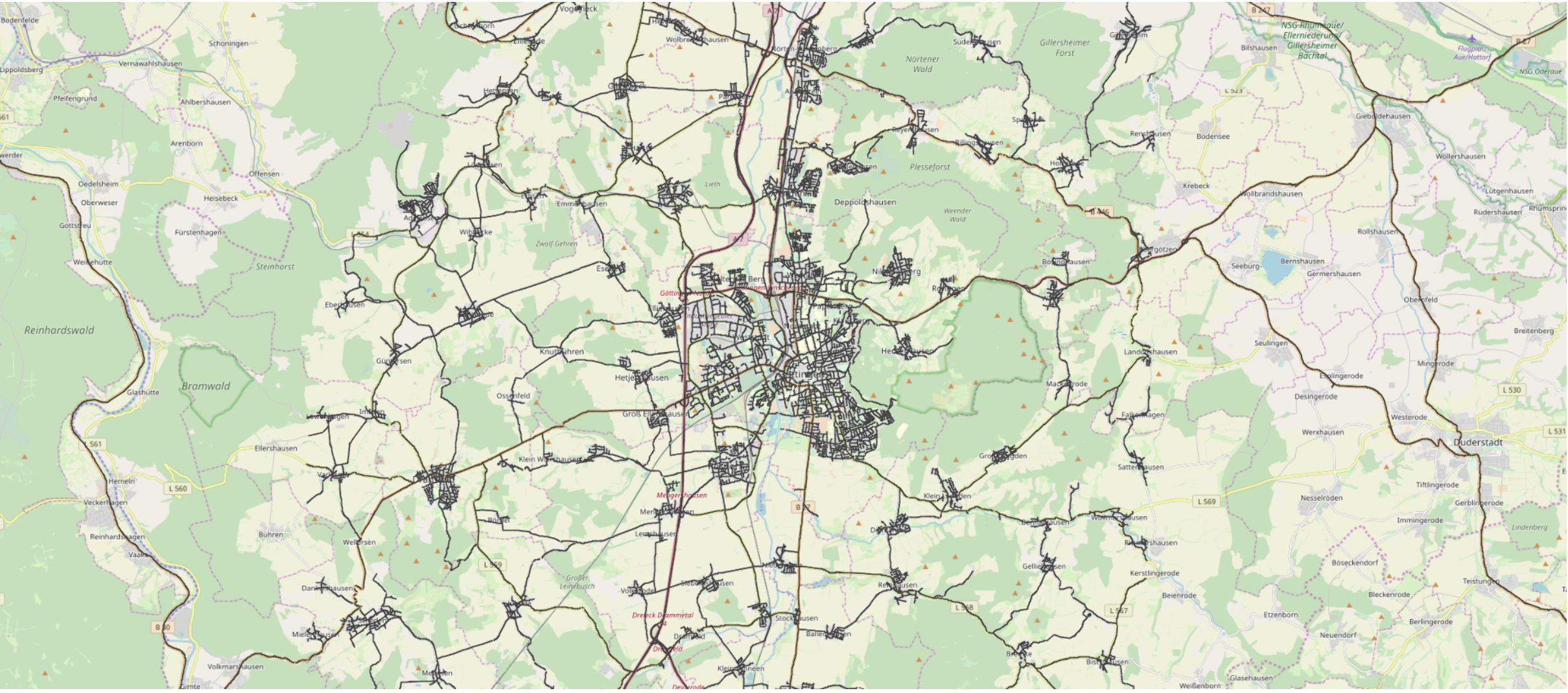
Make sure to chose an appropriate coordinate system for other regions.
If in doubt, `UTM Zone xyN` are a good bet (e.g. `WGS84 / UTM Zone 32N`).
Search for it, e.g. on spatialreference.org

This currently only supports uncompressed OSM-XML files
(***.osm**, no *.osm.bz2 or *.osm.pbf)

Creating a network: Converting from OSM

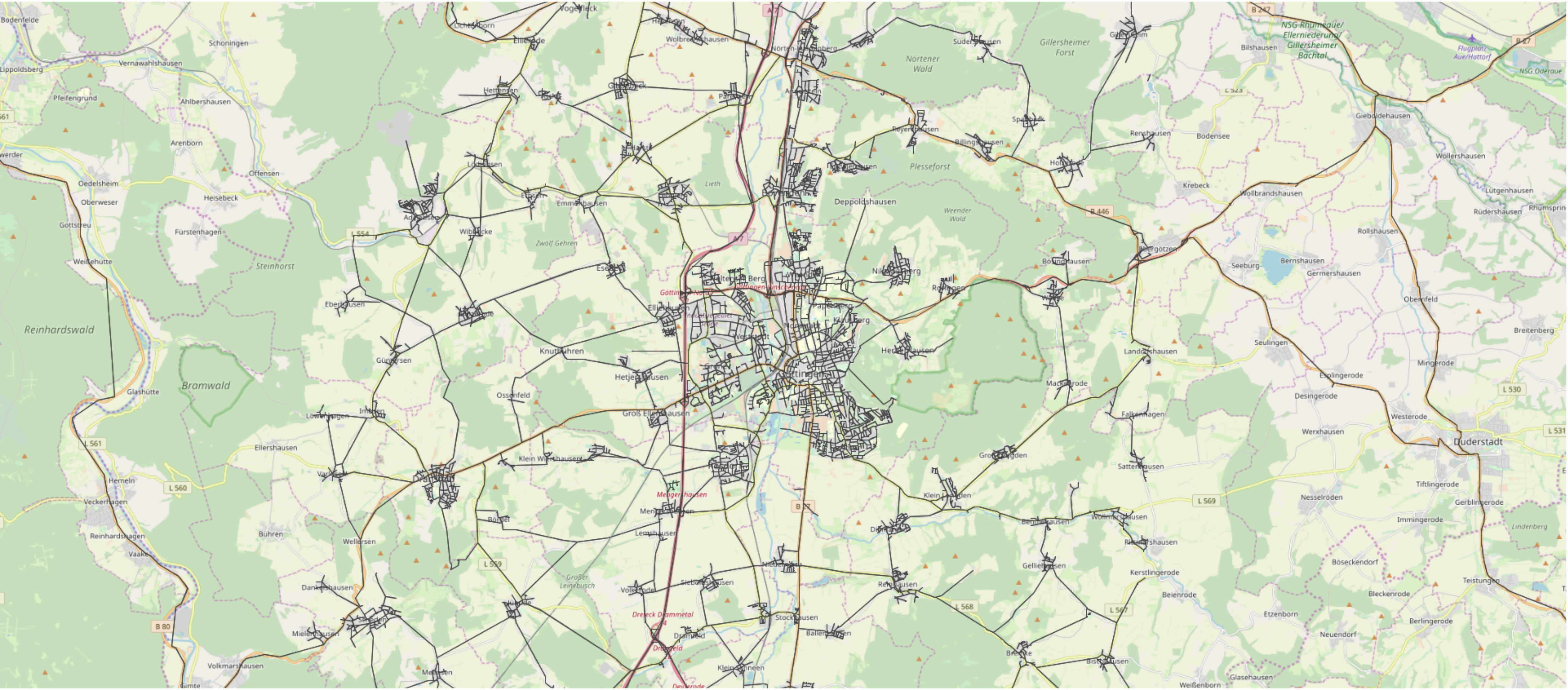
reader.setKeepPaths(true);

Creating a network: Converting from OSM



reader.setKeepPaths(true); Map-Background © OpenStreetMap contributors

Creating a network: Converting from OSM

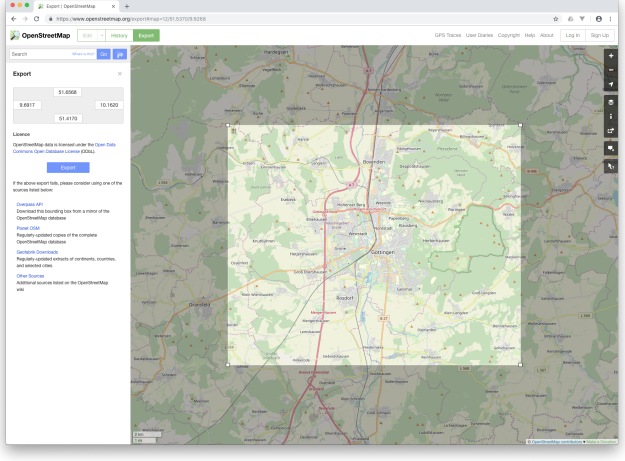


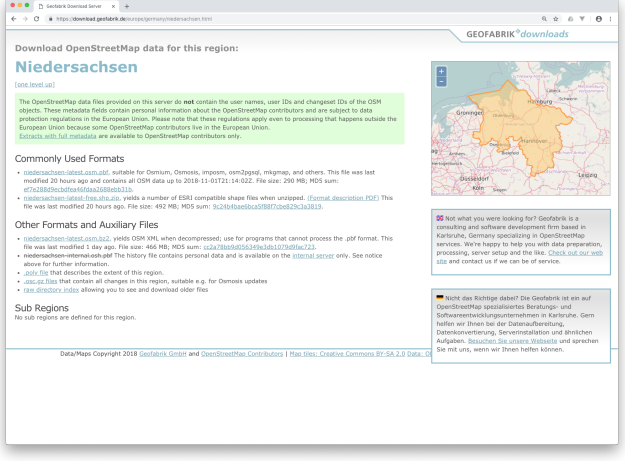
reader.setKeepPaths(false); Map-Background © OpenStreetMap contributors

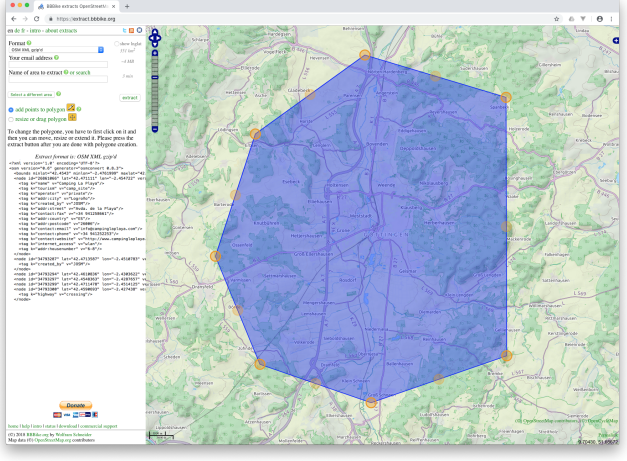
Creating a network: Converting from OSM

Obtaining OSM-data:

- small regions: openstreetmap.org/export
- larger areas: download.geofabrik.de
- custom areas: extract.bbbike.org







Creating a network: Converting other data

Convert VISUM

matsim/matsim/src/main/java/org/matsim/visum/VisumNetworkReader.java
The VISUM network export changes from time to time, so the code might first need to be updated to work correctly.

Convert EMME

matsim-code-examples/src/main/java/tutorial/converter/networkFromEmme/
The EMME format might change from time to time, so the code might first need to be updated to work correctly.

Convert Shp / GIS

There seem to be too many different GIS formats for networks as to be able to provide a converter.

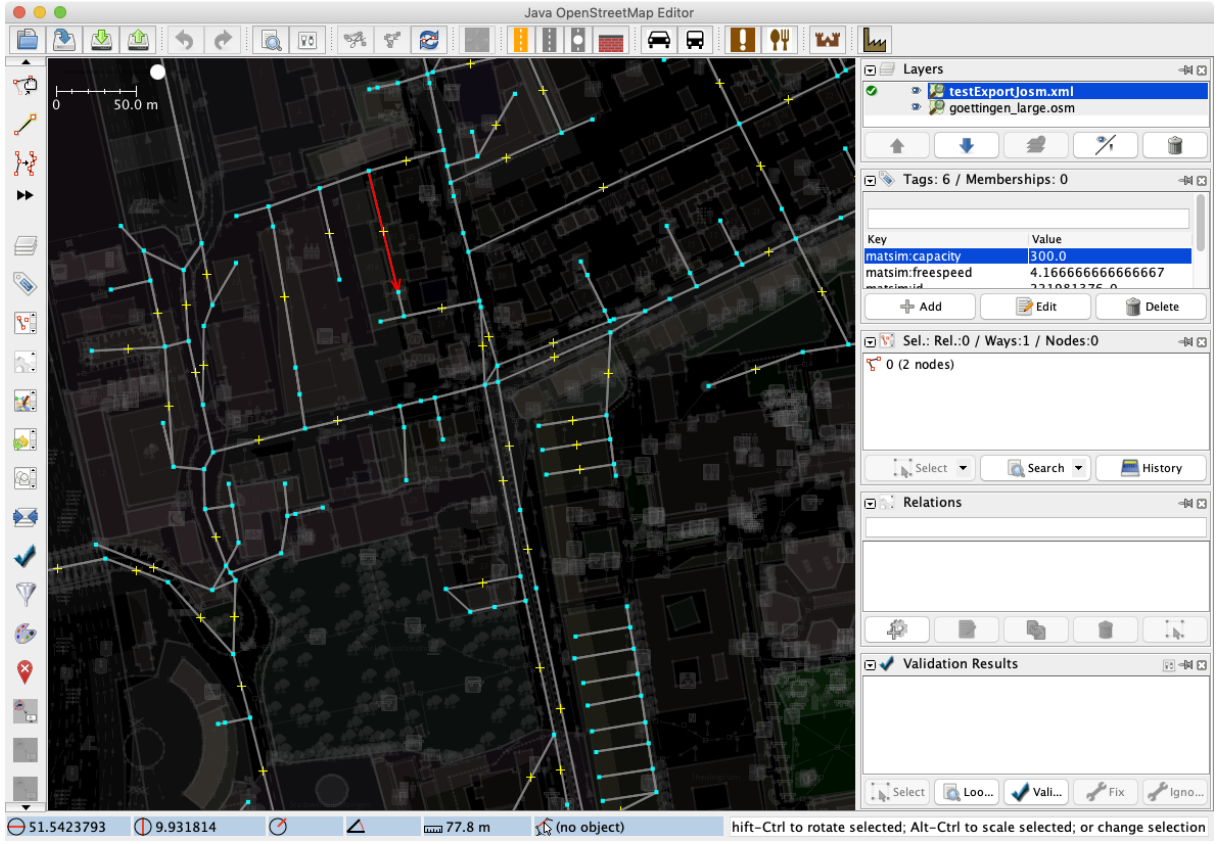
Creating a network: JOSM

JOSM is an editor for OpenStreetMap data.

There is a plugin to directly convert OSM-data to a MATSim network within JOSM.

After a successful installation of the matsim-plugin in JOSM, a new MATSim menu is available with options to convert data.

It can also load and edit existing networks.



Creating a network: pt2matsim

pt2matsim is a tool to convert public transport data.
It also includes an OSM-converter in order to create multimodal networks.

github.com/matsim-org/pt2matsim

```
import org.matsim.pt2matsim.run.Osm2MultimodalNetwork;

Osm2MultimodalNetwork.run("/path/to/goettingen.osm", "output/network.xml.gz", "EPSG:25832");
// alternative methods with more customizations are available
```

It always converts the full osm-file, make sure it covers just the area you need.

Preparing a network for simulation: Network Cleaner

MATSim imposes some restrictions on networks:

- Every node should be reachable by every other node.
- A network should not contain any sources or sinks as nodes.
- A network should consist of a single cluster of nodes and links.

`NetworkCleaner` takes care of this:

```
new NetworkCleaner().run(network);
```

NetworkCleaner detects the largest connected cluster of nodes and links, removes everything else.

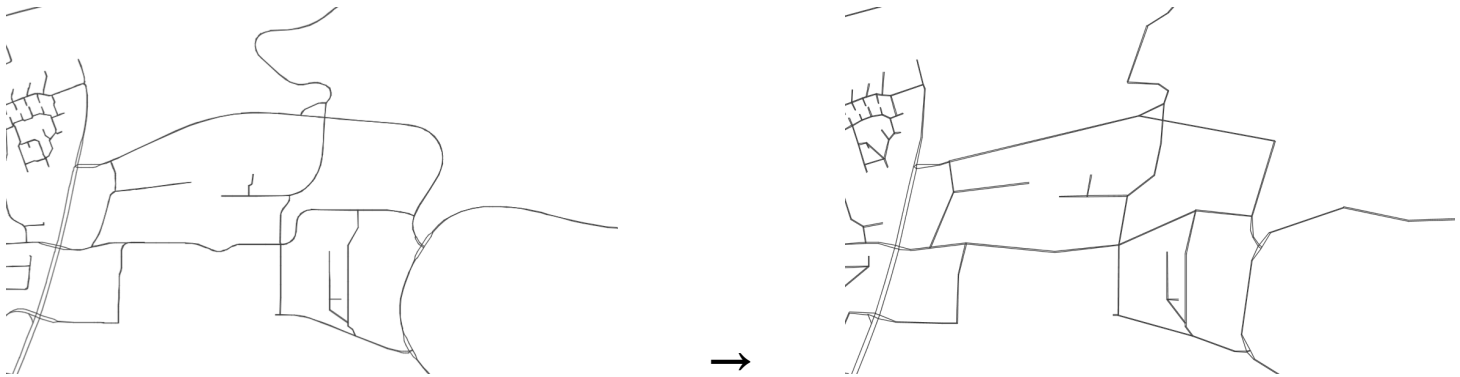
For multimodal networks, use `org.matsim.core.network.algorithms.MultimodalNetworkCleaner`.

Preparing a network for simulation: Network Simplifier

To improve the simulation performance, a network often can be simplified.

```
new NetworkSimplifier().run(network);
```

`NetworkSimplifier` combines multiple links into one if they share the same attribute values (i.e. same speed, flow capacity and number of lanes).



Fewer links often result in faster simulations and fewer events, but might introduce artifacts: fewer u-turn possibilities, first/last link of trip is artificially long, straight lines in visualization, …

Preparing a network for simulation: Network Simplifier

Full network, 127k links

Simplified network, 25k links

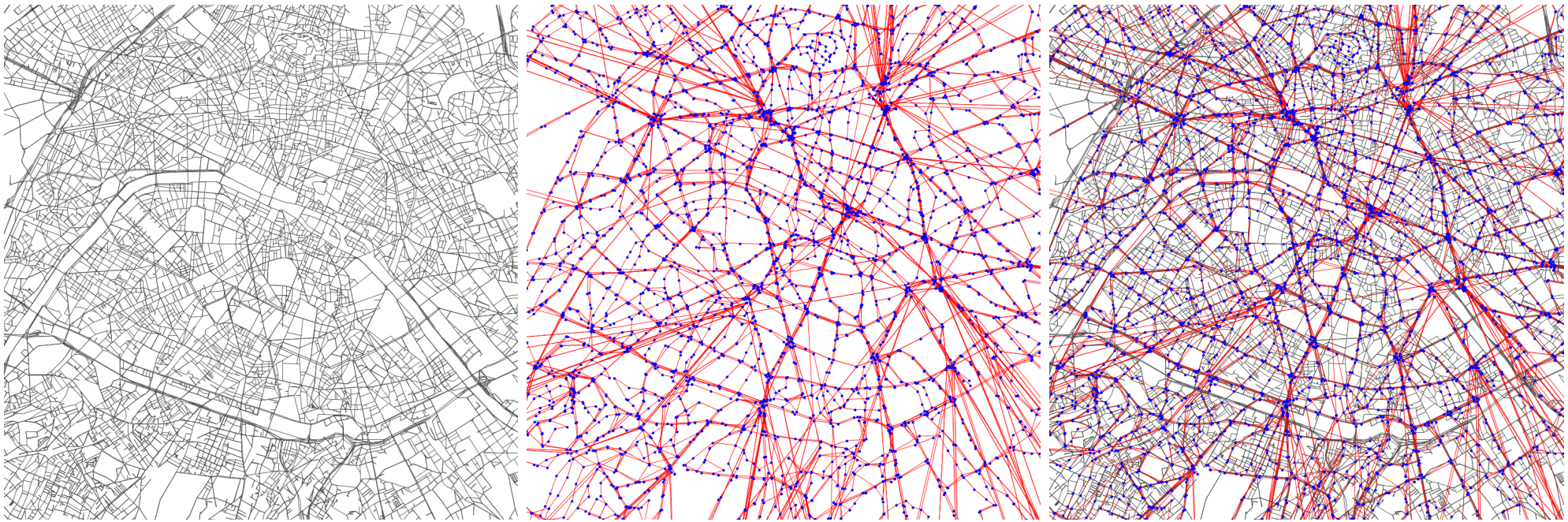
Network Generation: Best Practice

1. convert from OSM with `setKeepPath(true)`
2. run NetworkSimplifier
3. run NetworkCleaner (run it as last before writing the network)

Multi-Modal Networks

Each link has a set of allowed modes. This is currently only used by the router for route-finding. QSim will move the vehicles along whatever links are specified in a vehicle's route.

As vehicle interaction between different modes is often not needed, separate networks for each mode are typically put on top of each other and combined in one file.



Multi-Modal Networks

If different modes share the same links:

- Vehicles block each other by default (FIFO-Queue)
  - OK for car, bus, tram
  - Not OK for car and bicycle

MATSim has an option to enable overtaking on links, using "PassingQ" instead of a FIFO-Queue.

Details about enabling PassingQ: Mixed Traffic

Multi-Modal Networks: Enabling PassingQ

```
<module name="qsim">
  <param name="vehiclesSource" value="modeVehicleTypesFromVehiclesData" />
  <param name="mainMode" value="car,bicycle" />
  <param name="linkDynamics" value="PassingQ" />
  <param name="trafficDynamics" value="queue" />
</module>

<module name="planscalcroute" >
  <param name="networkModes" value="car,bicycle" />
</module>

<module name="travelTimeCalculator" >
  <param name="analyzedModes" value="car,bicycle" />
  <param name="separateModes" value="true" />
</module>
```

Also define a default vehicle type for each main mode:

```
<vehicleType id="car">
  <maximumVelocity meterPerSecond="16.67"/>
  <passengerCarEquivalents pce="1.0"/>
  ...
</vehicleType>
```

```
<vehicleType id="bicycle">
  <maximumVelocity meterPerSecond="4.17"/>
  <passengerCarEquivalents pce="0.25"/>
  ...
</vehicleType>
```

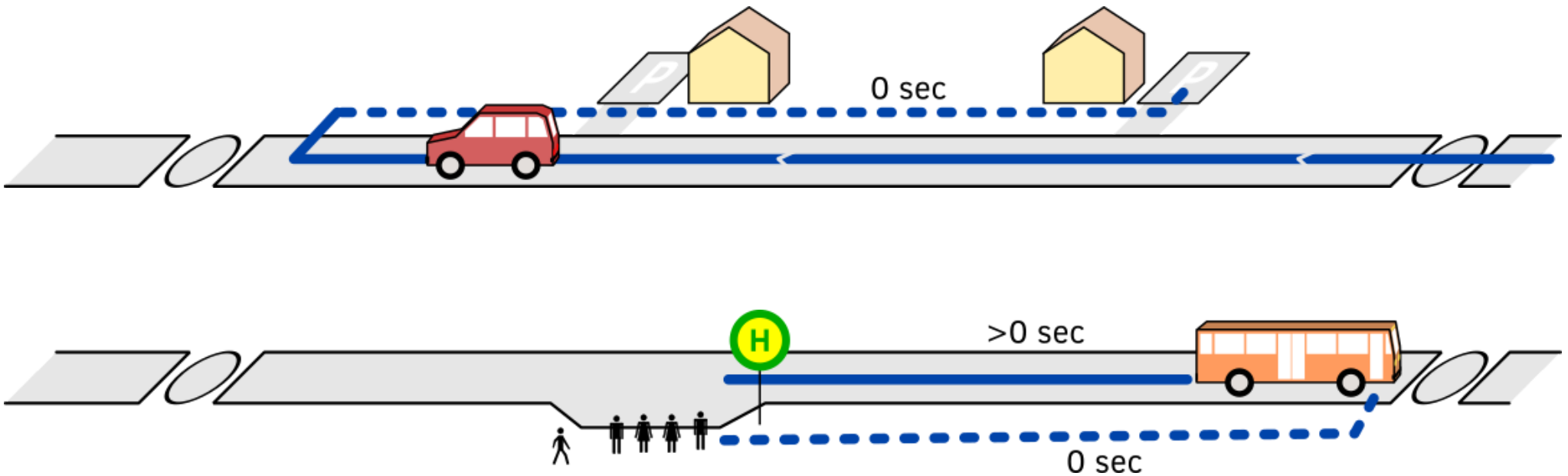
Naturally, links in the network need to support the additional modes as well.

Full details about enabling PassingQ: Mixed Traffic

QSim and the Network

Links are Queues. We can only interact with Queues at the start or the end.
In QSim, all interactions take place near the end of the link (at the "to-node").

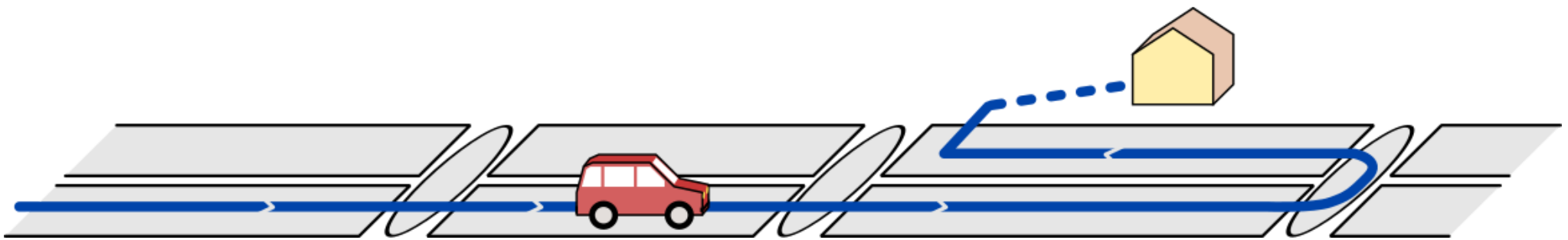
- Cars can depart or arrive only at the end of a link
- Public Transport stops can only be served when the transit vehicle is at the end of a link.



QSim and the Network

Links are uni-directional.

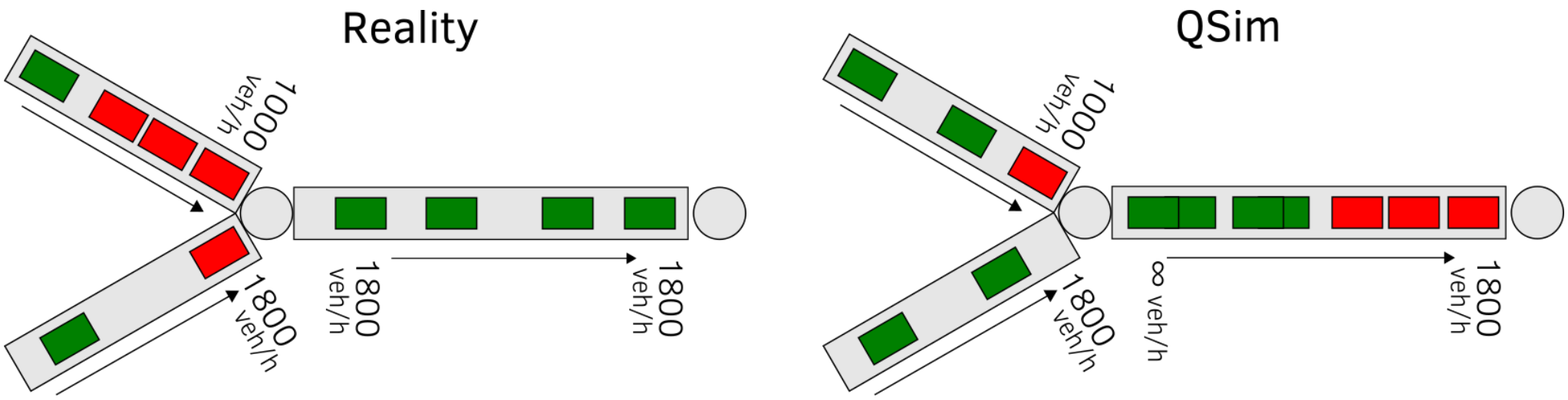
- We need two links for the different directions of a road.
- Vehicles might have to make U-turns to arrive on the desired link.



QSim and the Network

By default, links have unlimited inflow-capacity.
The defined link's capacity acts as outflow-capacity.

- Inflow in a link could be larger than in reality.
  This may result in traffic jams building up too far downstreams.



QSim and the Network

There is an option to enable "backwards travelling holes" or "kinetic waves" which help to a certain extent to mitigate the missing inflow capacity.

```
<module name="qsim">
  <param name="trafficDynamics" value="withHoles" /> <!-- options : queue, withHoles, kinematicWaves  -->
</module>
```

QSim and the Network: Intersection logic

Nodes represent intersections, but have no extent.

For each incoming link, QSim moves vehicles from the up-stream link to the down-stream link

- if there is a vehicle in the up-stream link that can leave the link, and
- if there is space in the down-stream link.

The order in which links are processed in each time step is randomized,
but weighted with the links' flow capacities.

(= Links with a high capacity have a higher probability for having their vehicles moved before vehicles from lower-capacity links.)