Moving from Configuration Files to an Expressive API in JuPedSim

P. Steinbach, R. Leibold, K. Kratz, M. Chraibi

Institute for Advanced Simulation 7 - Civil Security , Forschungszentrum Jülich

FROM MONOLITHIC XML CONFIGURATION TO OBJECT-ORIENTED PYTHON SCRIPTING

- Prior to JuPedSim v1.0: XML configuration files
- Individual scripts to generate configuration files were written
- Result of 3-day architecture workshop: Have a Python API to give **flexibility** and **creativity** back to researchers

The Old Way

```
<JuPedSim project="JPS-Project" version="0.7">
  <header>
    <seed>{{seed}}</seed>
    <max_sim_time>100</max_sim_time>
    <geometry>../geometry.xml</geometry>
    <show_statistics>true</show_statistics>
    <trajectories format="xml-plain" fps="8">
      <file location="{{location}}" />
    </trajectories>
  </header>
  ...
```

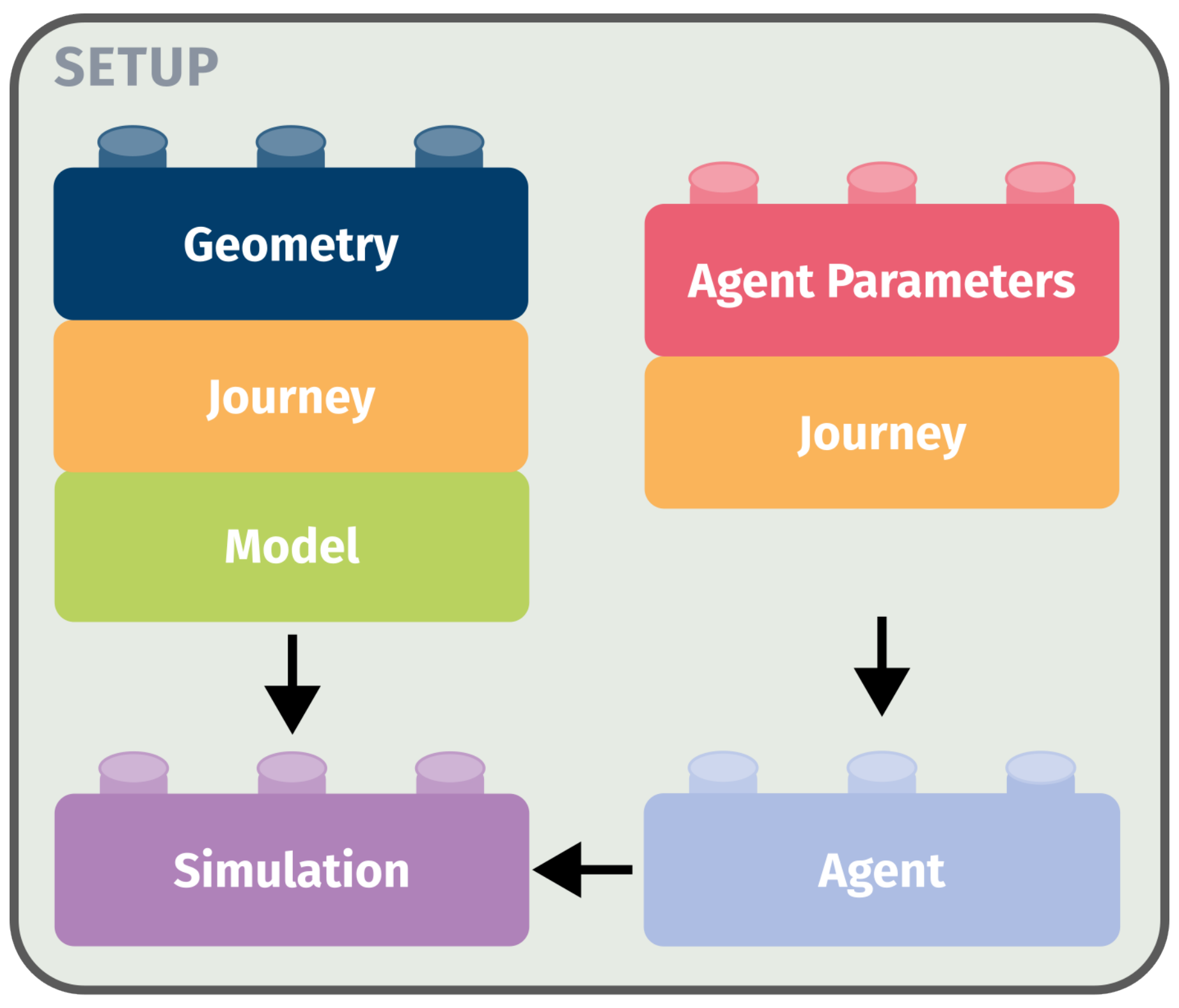
simulation.py

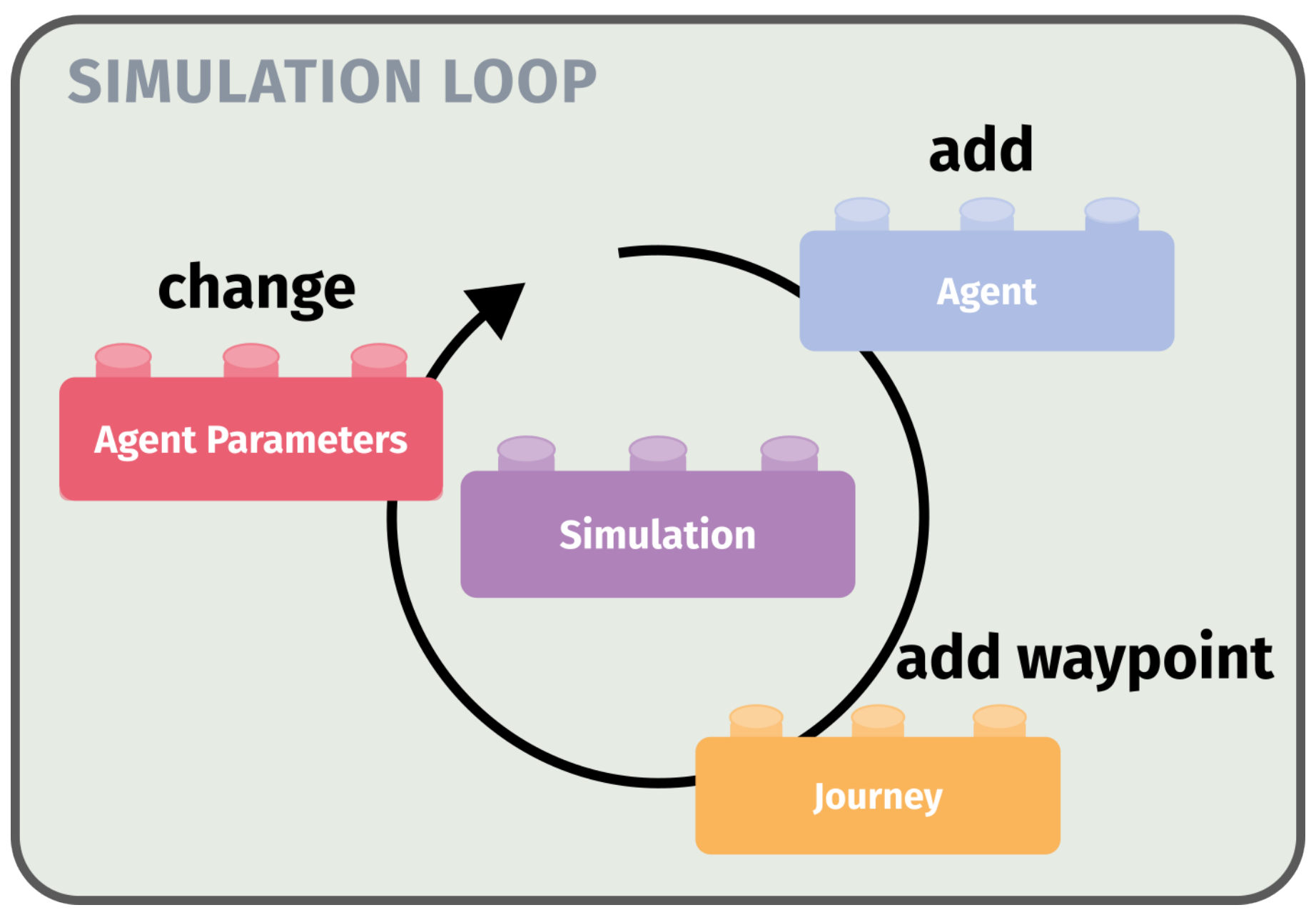
```python
import jupedsim as jps
from shapely import Polygon
area = Polygon([(0, 0), (12, 0), (12, 12), (10, 12),
                (10, 2), (0, 2)])
trajectory_file = "corner.sqlite"
simulation = jps.Simulation(
   model=jps.CollisionFreeSpeedModelV1(),
   geometry=area,
   trajectory_writer=jps.SqliteTrajectoryWriter(
      output_file=pathlib.Path(trajectory_file)),
   dt=0.01)

while simulation.agent_count() > 0:
   simulation.iterate()
```

PHILOSOPHY AND ARCHITECTURE

Composable and Modifiable Building Blocks

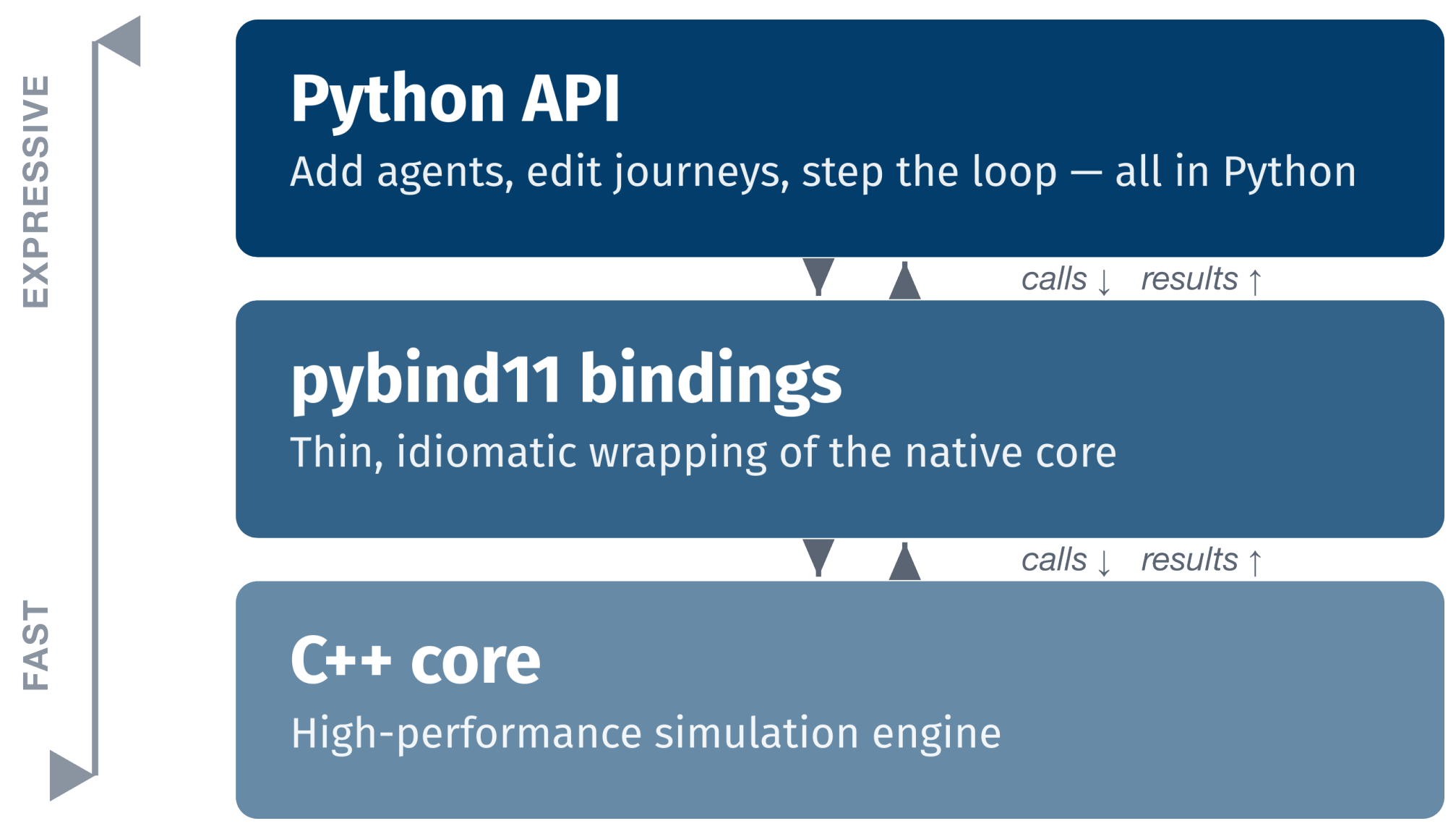
- Geometry, agents, journeys, model and the simulation are independent Python objects.
- Functionality such as adding agents, editing a journey, or propagating the simulation is callable from Python.
- Agent parameters and other variables can be inspected and changed at each iteration step.
- The ability to manipulate core objects at any point provides flexibility.


SETUP
Geometry
Journey
Model
Agent Parameters
Journey
Simulation
Agent

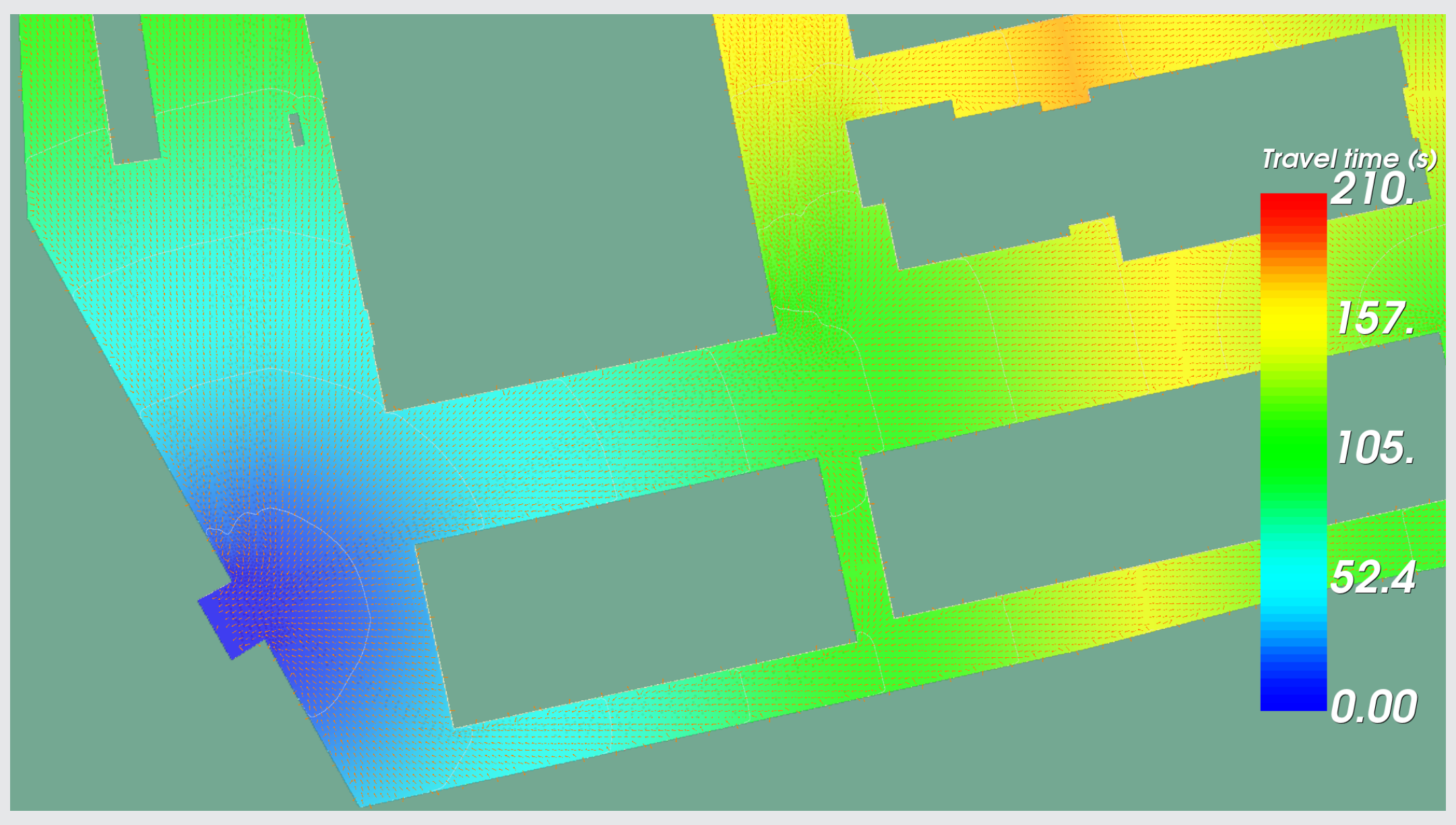

SIMULATION LOOP
add
Agent
change
Agent Parameters
Simulation
add waypoint
Journey

Three Layer Architecture

- No compromise between performance and ease of use.
- Performance-relevant code is written in C++ (by software developers).
- Researchers use the Python API, C++ complexity stays hidden.


EXPRESSIVE
FAST
Python API
Add agents, edit journeys, step the loop — all in Python
calls ↓ results ↑
pybind11 bindings
Thin, idiomatic wrapping of the native core
calls ↓ results ↑
C++ core
High-performance simulation engine

OUTLOOK: ADDING RUST TO THE MIX

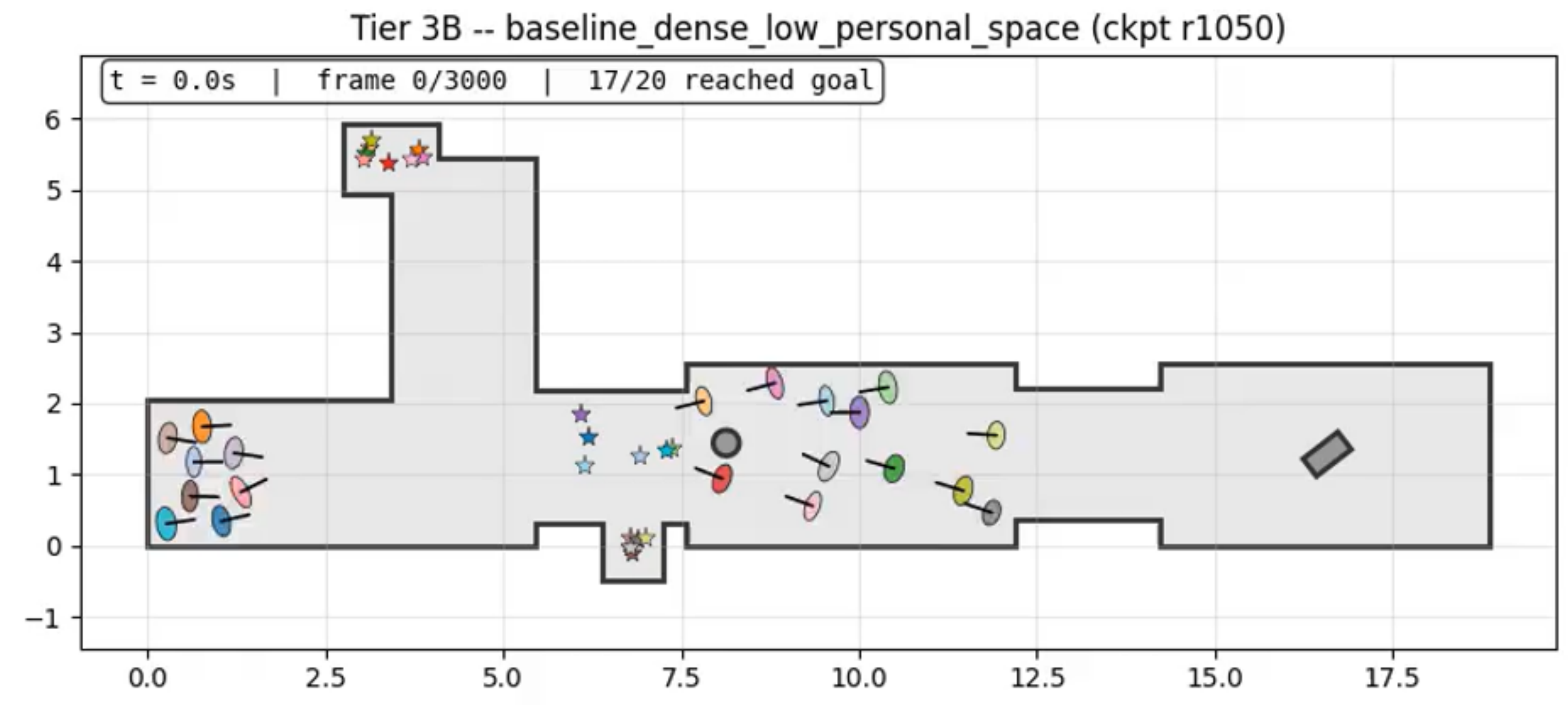


For the JuPedSim 2.0 release, we want to add floor fields again as the default routing algorithm. Similar to JuPedSim, we want to provide a library that serves as basis for cooperation and research projects. Hence, we want to put effort into packaging the floor fields as a separate specialised library. To maximize compatibility and rapid adoption by other simulation codes, we decided to use Rust as a programming language.

USE CASES FOR THE JUPEDSIM PYTHON API

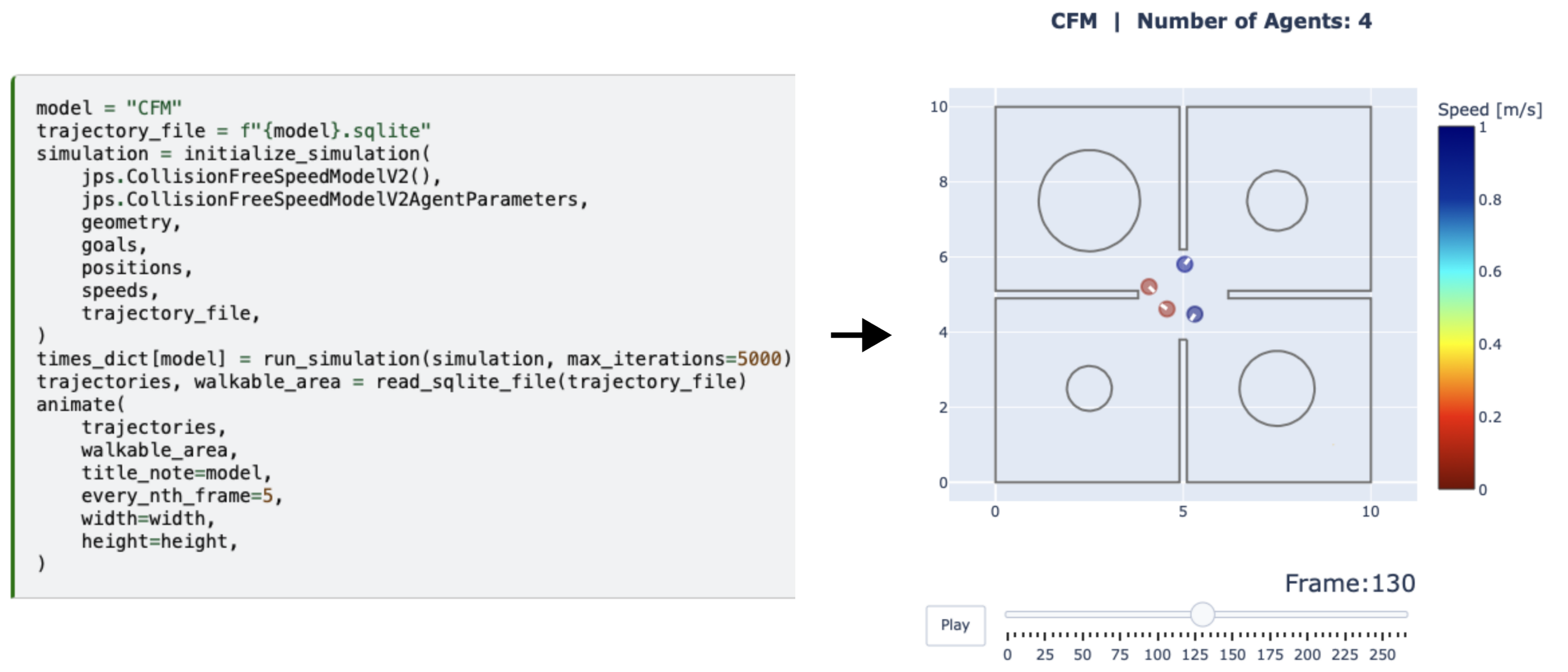
Reinforcement learning-based simulations

Data-driven efforts to model pedestrian behavior have emerged in recent years. Fabian Plum from IAS7 is using reinforcement learning techniques to train models while using JuPedSim as part of his pipeline. Similar efforts from other groups [1]



Interactive Documentation - From Jupyter Notebook to animation

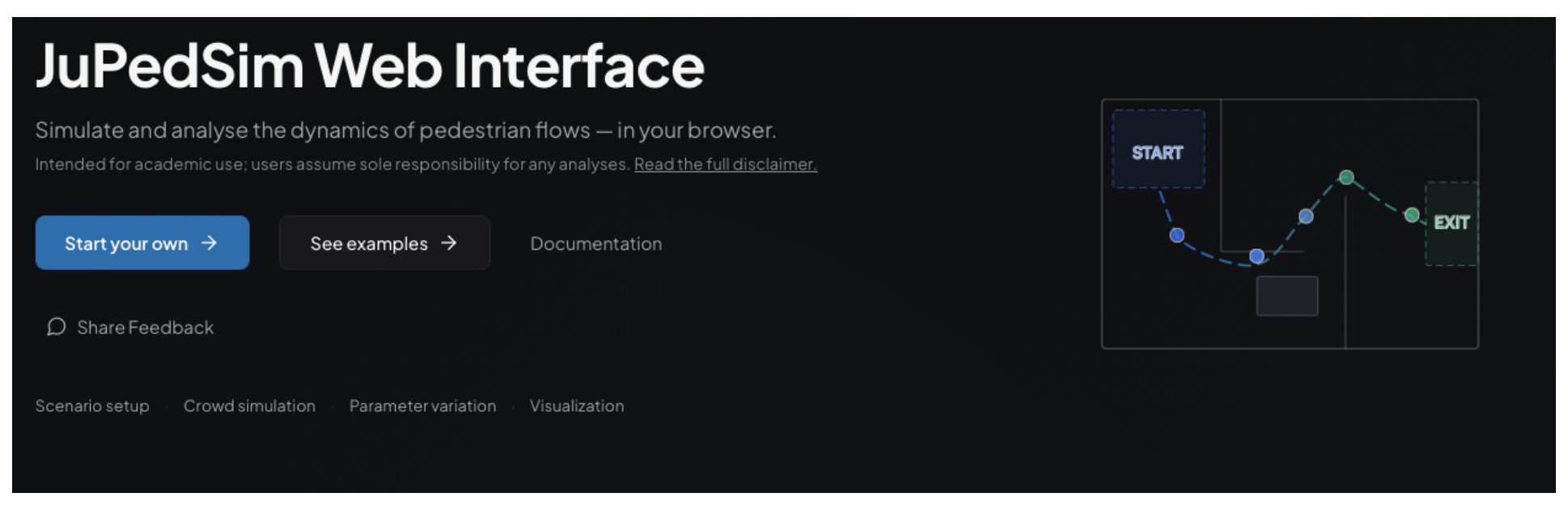
Documentation is now a primary concern and writing it is part of the release plan. We back this up by strong automation: documentation containing code is generated from Jupyter notebooks in GitHub actions for every pull request. All notebooks are available for download with full code from the documentation to allow users to start immediate experimentation.



```python
model = "CFM"
trajectory_file = f"{model}.sqlite"
simulation = initialize_simulation(
    jps.CollisionFreeSpeedModelV2(),
    jps.CollisionFreeSpeedModelV2AgentParameters,
    geometry,
    goals,
    positions,
    speeds,
    trajectory_file,
)
times_dict[model] = run_simulation(simulation, max_iterations=5000)
trajectories, walkable_area = read_sqlite_file(trajectory_file)
animate(
    trajectories,
    walkable_area,
    title_note=model,
    every_nth_frame=5,
    width=width,
    height=height,
)
```

CFM | Number of Agents: 4
Speed [m/s]
1
0.8
0.6
0.4
0.2
0
10
8
6
4
2
0
0
5
10
Frame:130
Play
0 25 50 75 100 125 150 175 200 225 250

Confluence of capability and opportunity - JuPedSim Web App



JuPedSim Web Interface

Simulate and analyse the dynamics of pedestrian flows — in your browser.

Intended for academic use; users assume sole responsibility for any analyses. Read the full disclaimer.

Start your own →
See examples →
Documentation
Share Feedback
Scenario setup
Crowd simulation
Parameter variation
Visualization
START
EXIT

Completion of JuPedSims Python API focused rework coincided with the ongoing AI revolution and opened up unforeseen opportunities for us. With minimal prior knowledge on web development, but a well documented Python API, we managed to quickly build a web app using coding agents.

REFERENCES AND ACKNOWLEDGEMENTS

[1] https://doi.org/10.5281/zenodo.20327414
[2] http://hdl.handle.net/2128/11280
[3] https://pybind11.readthedocs.io/en/stable/index.html
[4] https://doi.org/10.1109/MMSP61759.2024.10743283