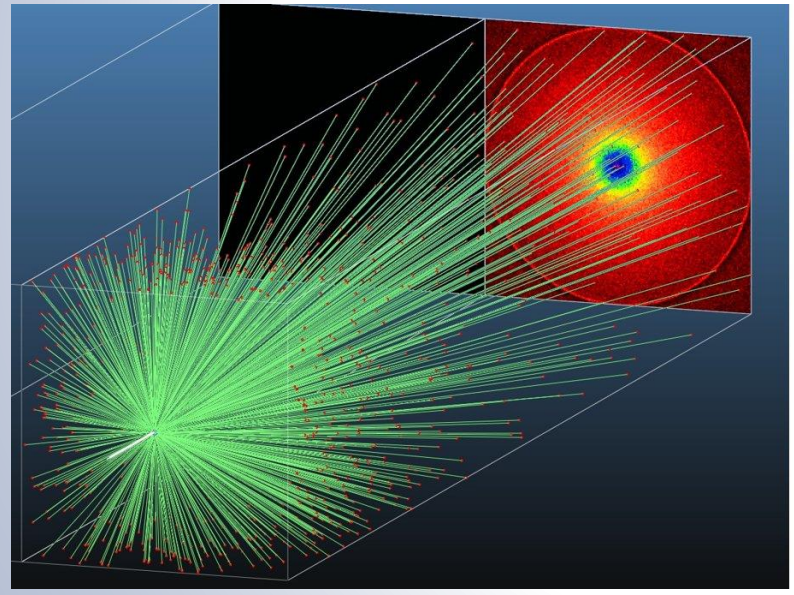
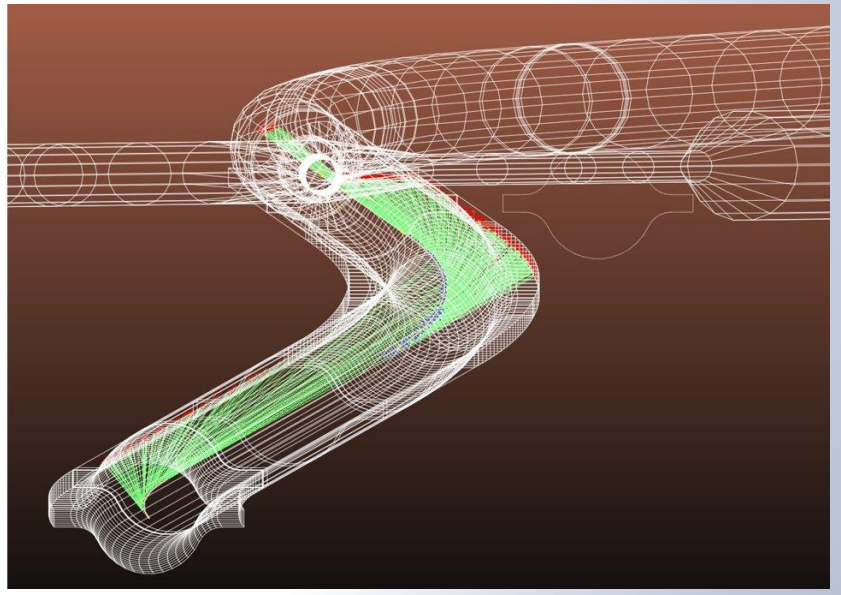
# MolFlow and SynRad development update

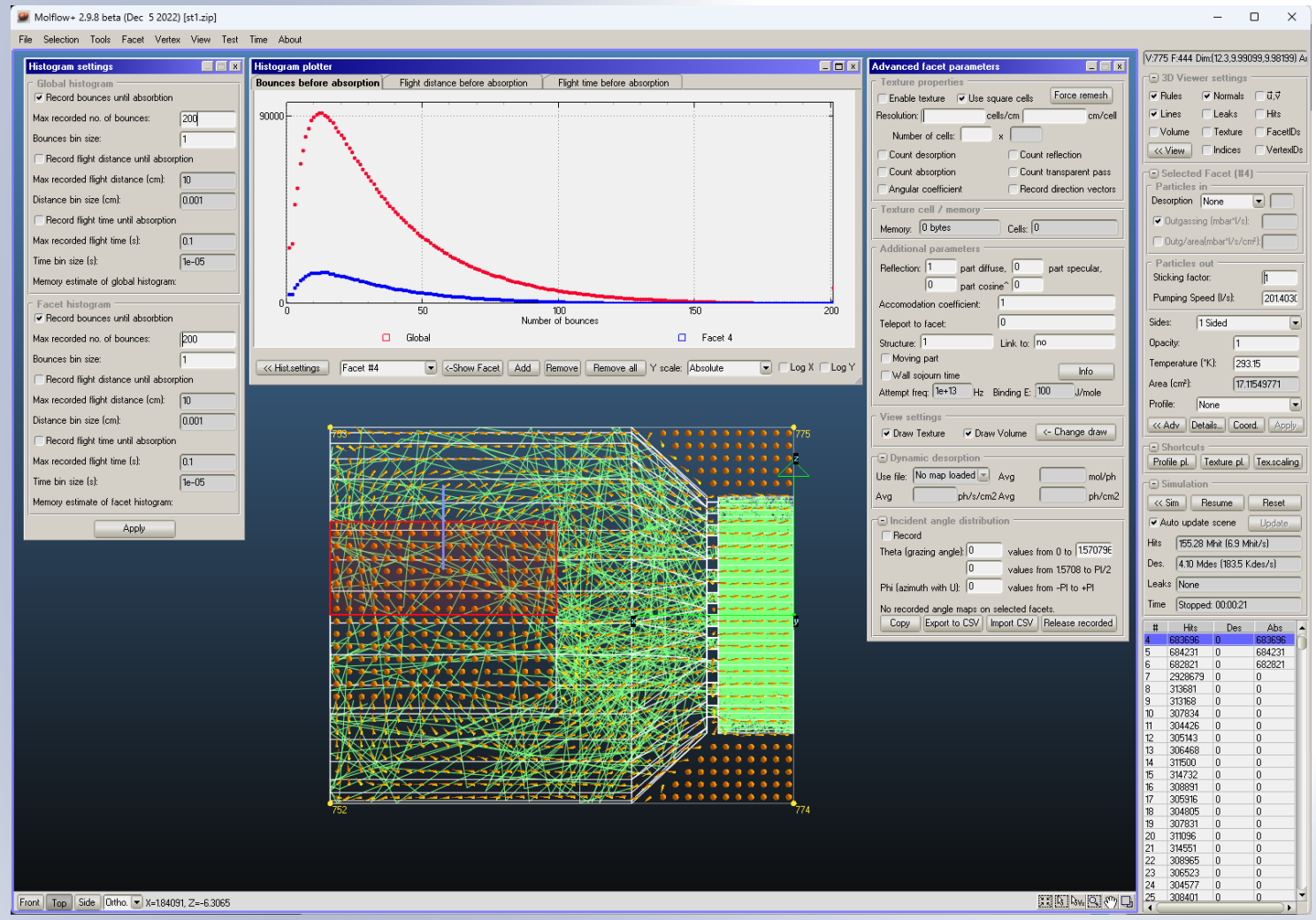




2025-02-25 TE-VSC seminar

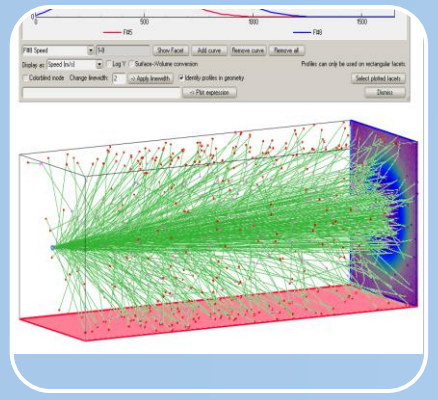
Márton Ady, Petar Trifunović

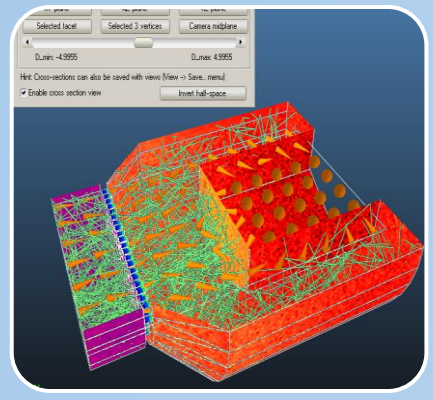
# MolFlow in one slide

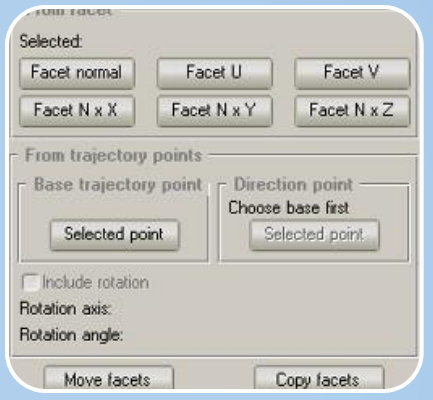


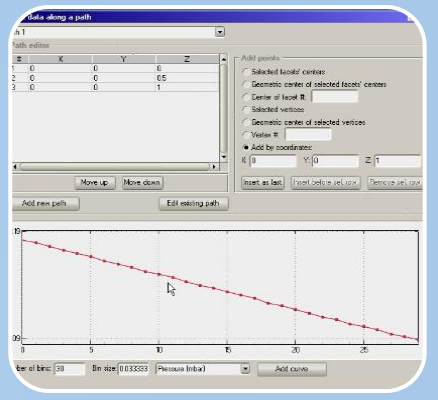

- Test-particle vacuum simulator
- Uses Monte Carlo test particles
- Assumes molecular flow
- Written by Roberto Kersevan in 1990s
- Modernized at ESRF in 2008
- Developed at TE-VSC since 2012
- Approx. 1000 users

# Today: 5 new functions







Physics

Visualization

Contamination

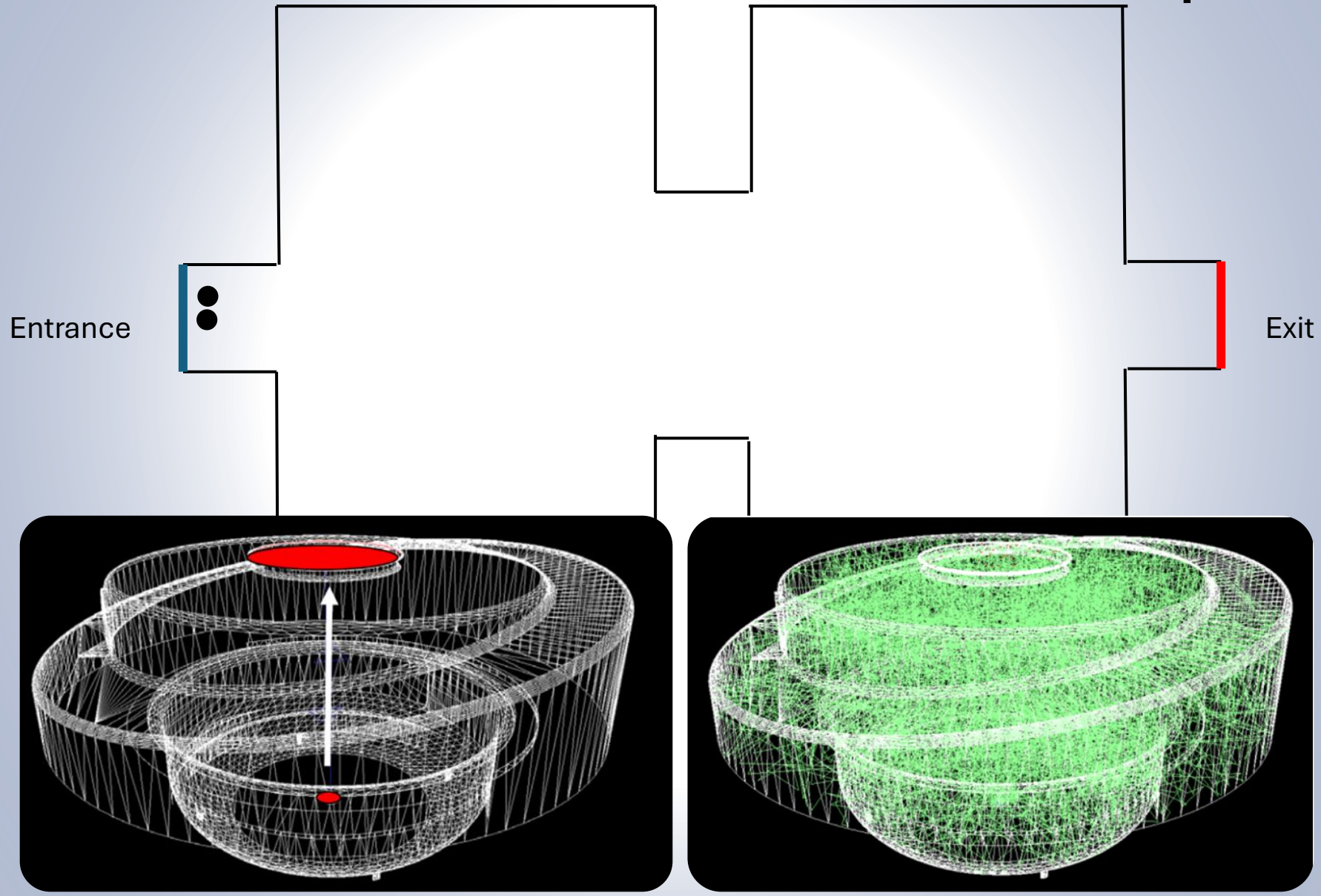
Geometry edit

Data extraction

2024

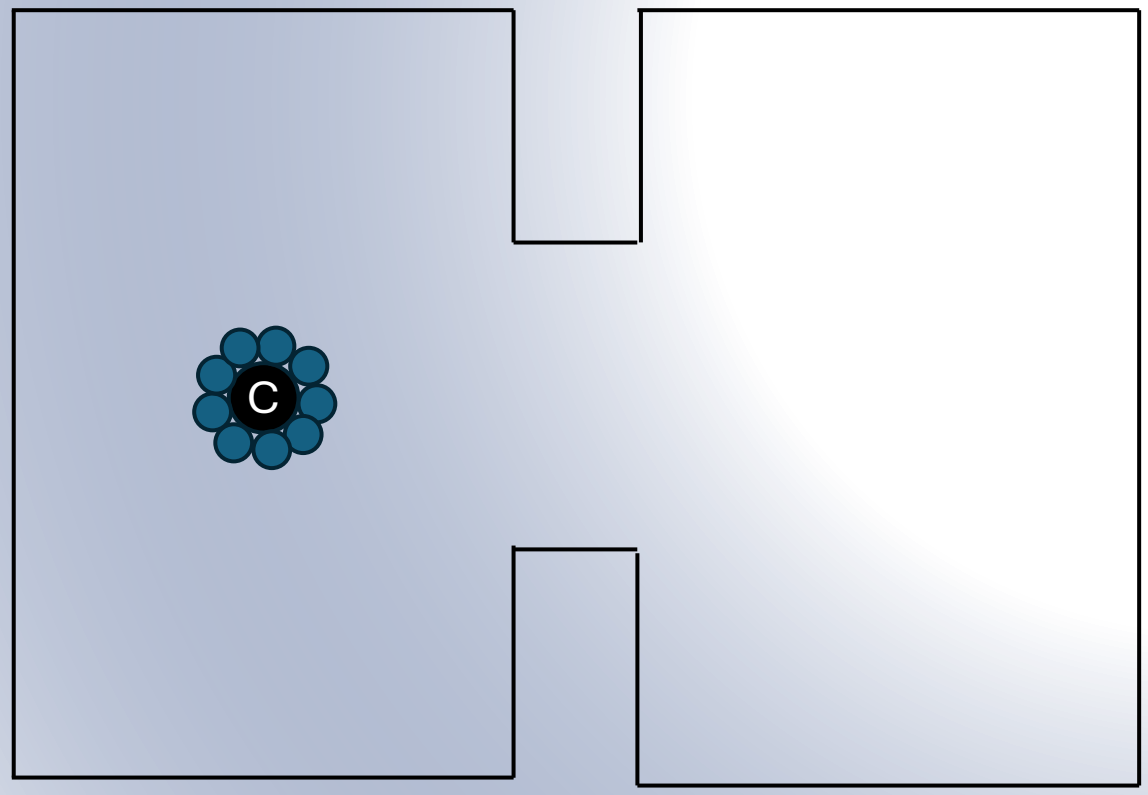
2025

# A Monte Carlo simulation: Transmission probability

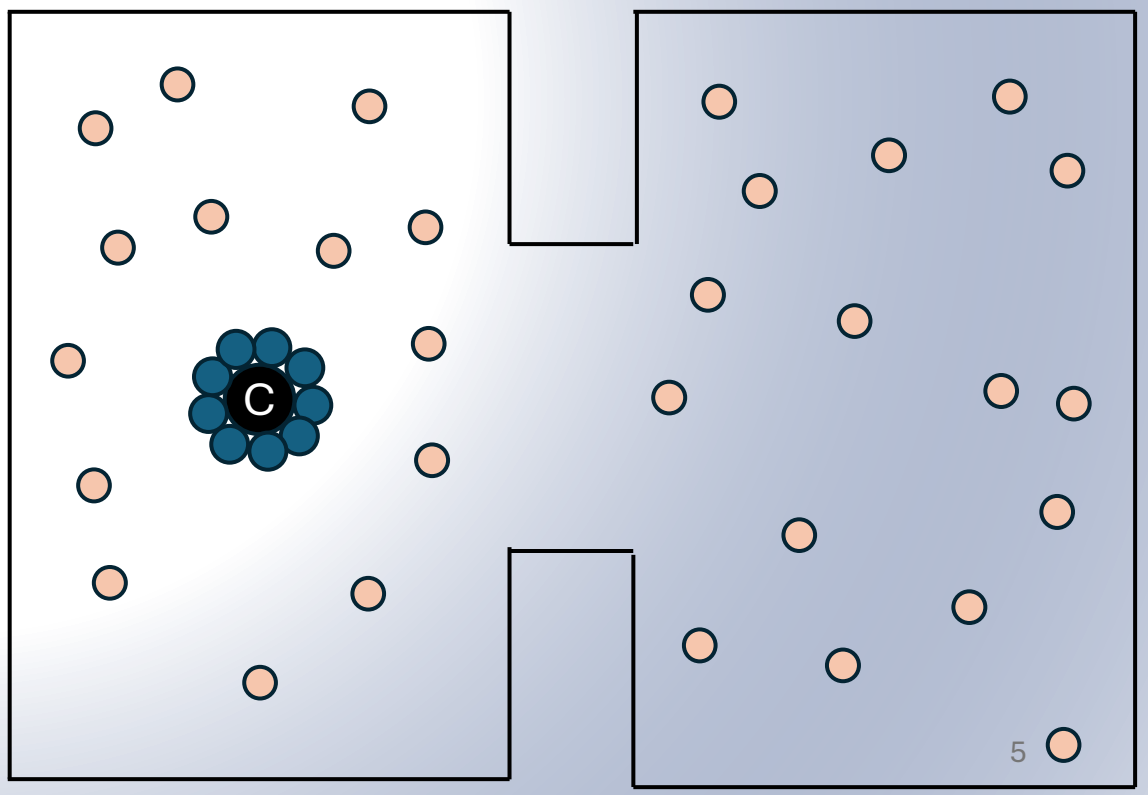


# MolFlow for sputtering simulations

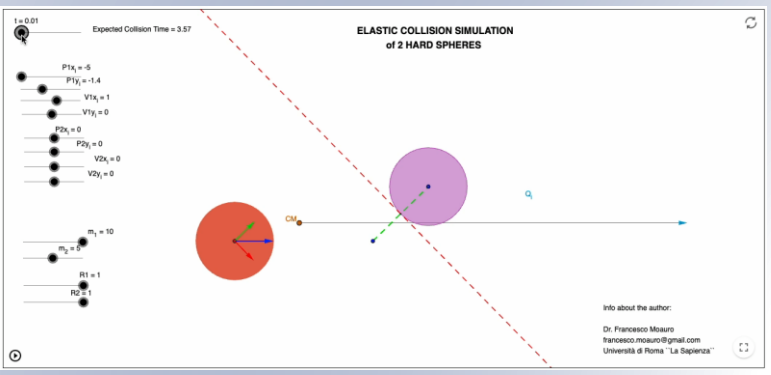
Simple sputtering simulation

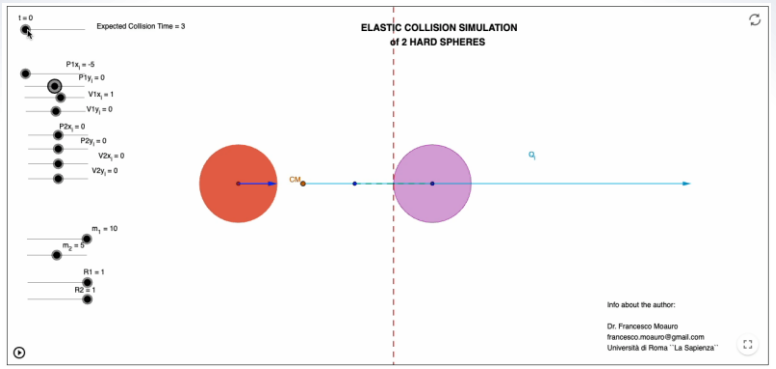


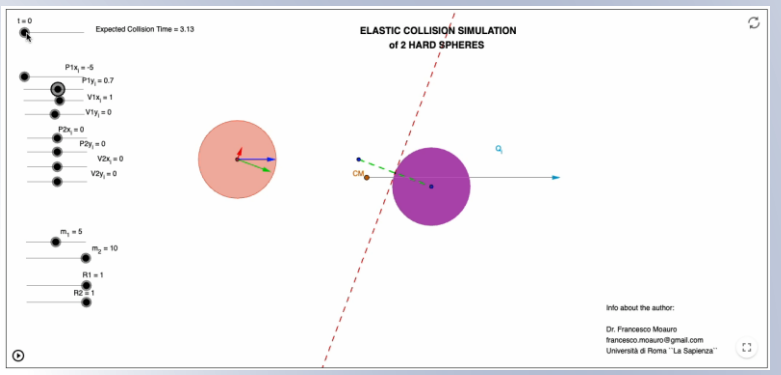

With background gas



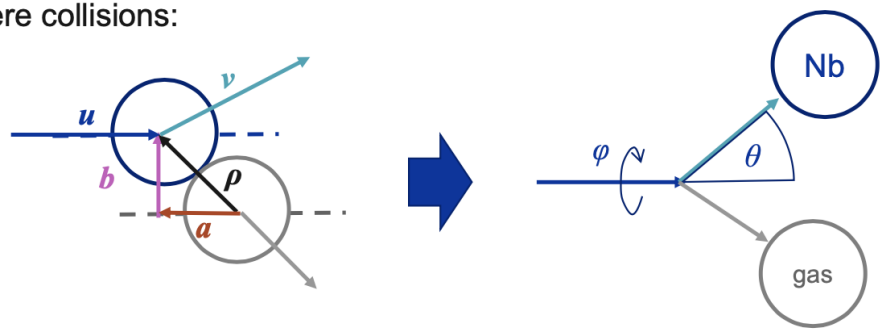
# Collision between two particles



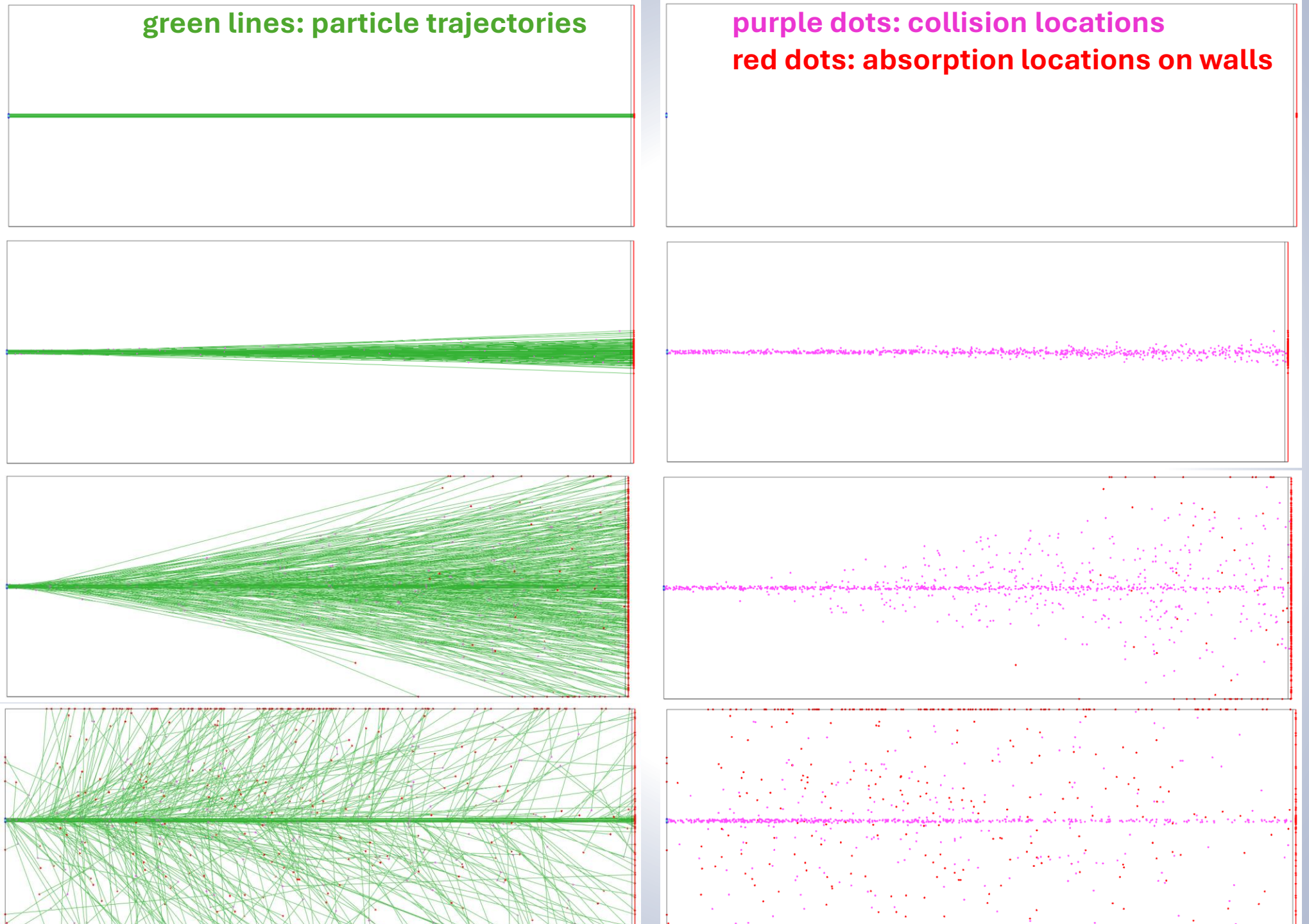

- Describing hard sphere collisions:

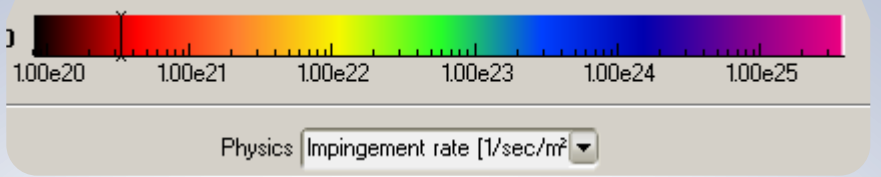


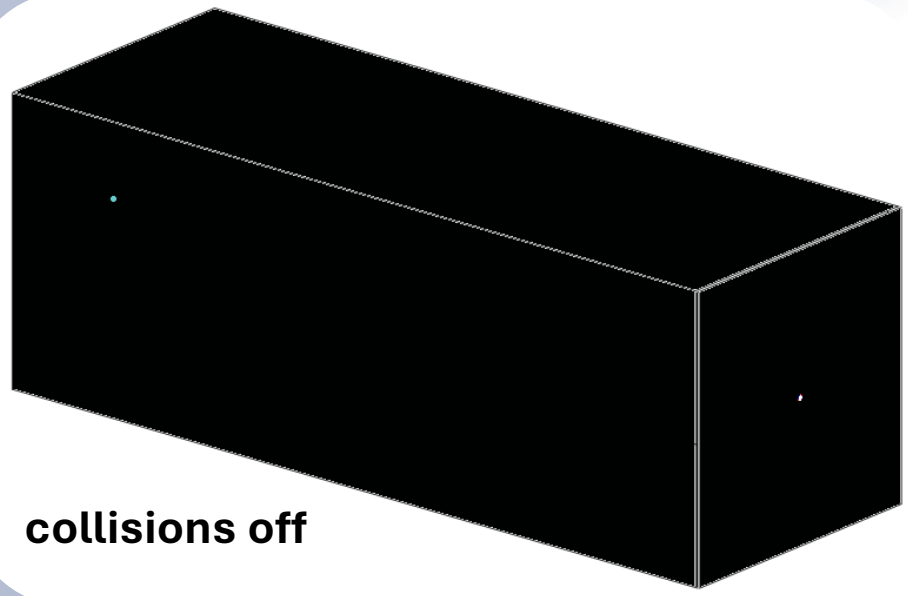

- Azimuthal scattering angle φ has uniform PDF due to cylindrical symmetry
- Polar scattering angle θ based on linear PDF of “impact parameter” b (cf “differential cross sections”)
- New speed v is defined uniquely by θ (b)

$$P(b) = \frac{2b}{\rho^2} \quad \Rightarrow \quad \cot(\theta) = \frac{1}{2ab}\left(\rho^2 \frac{m_{Nb}}{m_{gas}} + b^2 - a^2\right) \quad \Rightarrow \quad v^2 = u^2\left(1 + \frac{m_{Nb}^2 \rho^2}{m_{gas}^2 b^2}\sin^2(\theta)\right)^{-1}$$

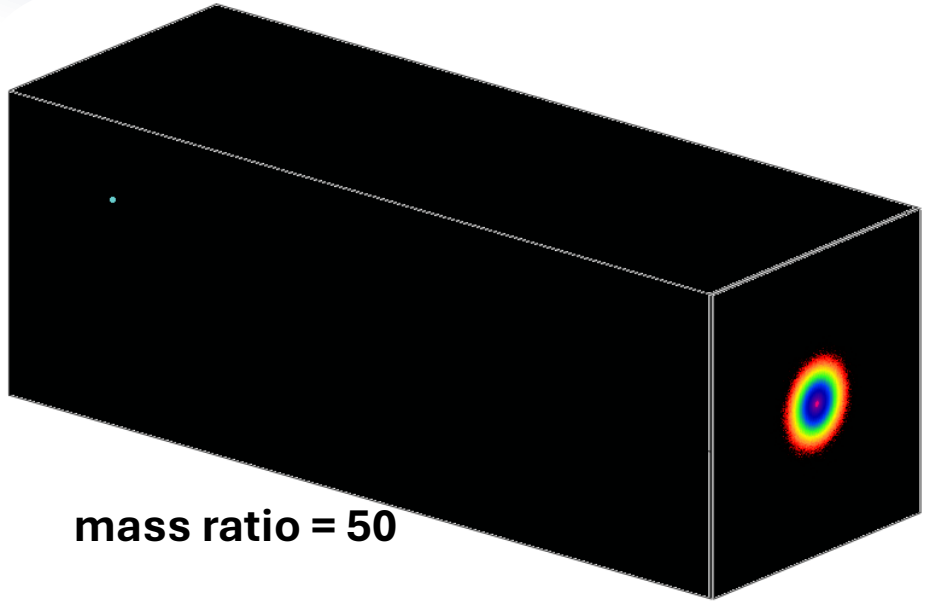
From Fabian Manke


green lines: particle trajectories
purple dots: collision locations
red dots: absorption locations on walls
collisions off
mass ratio = 50
mass ratio = 5
mass ratio = 0.5

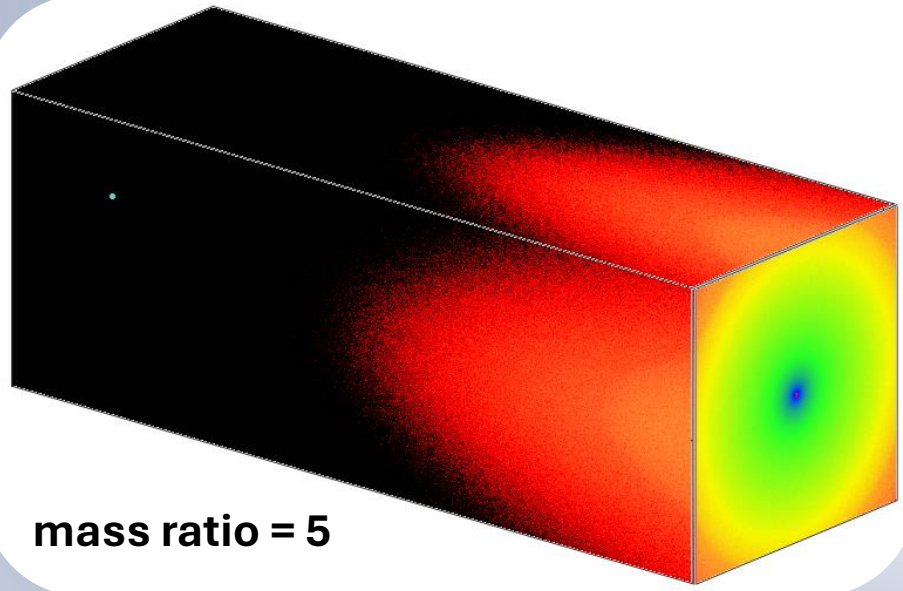


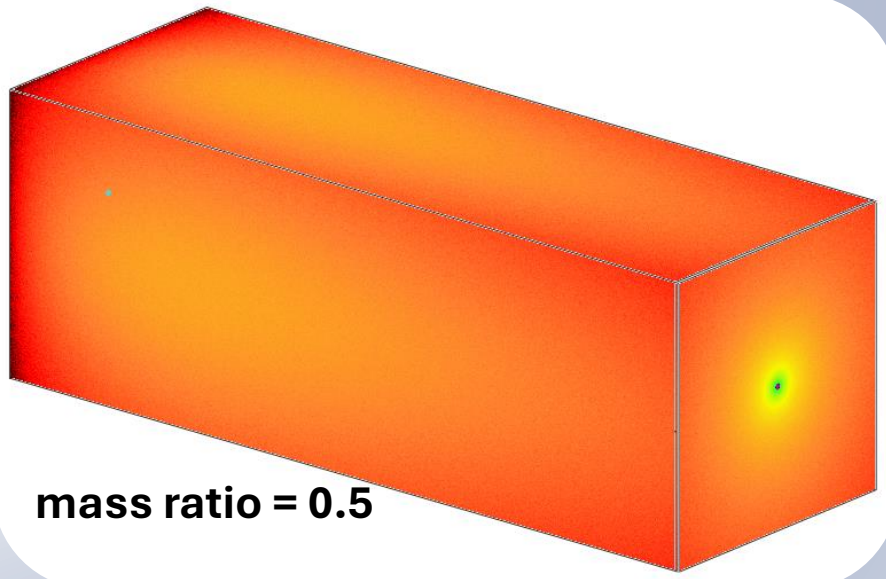
collisions off



mass ratio = 50

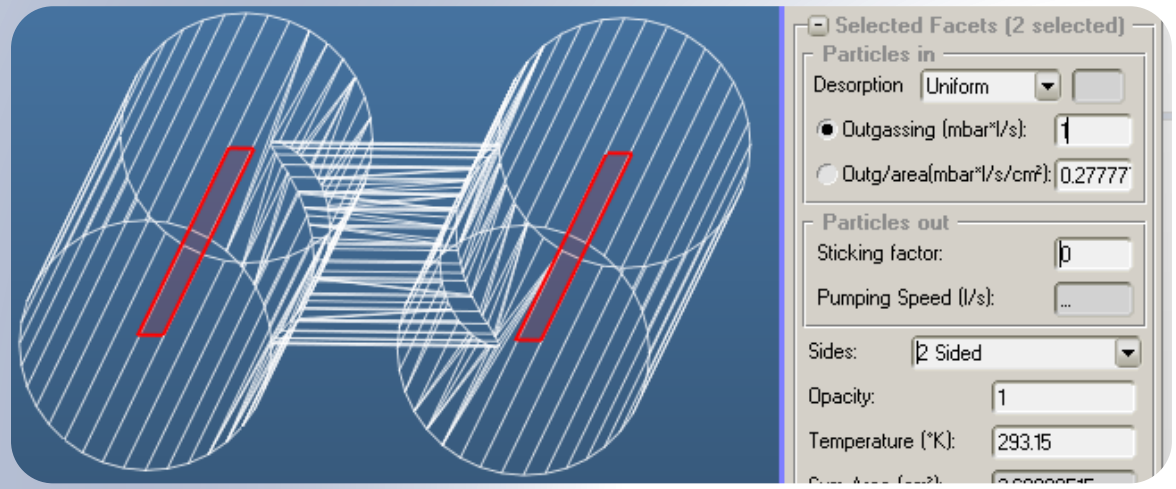


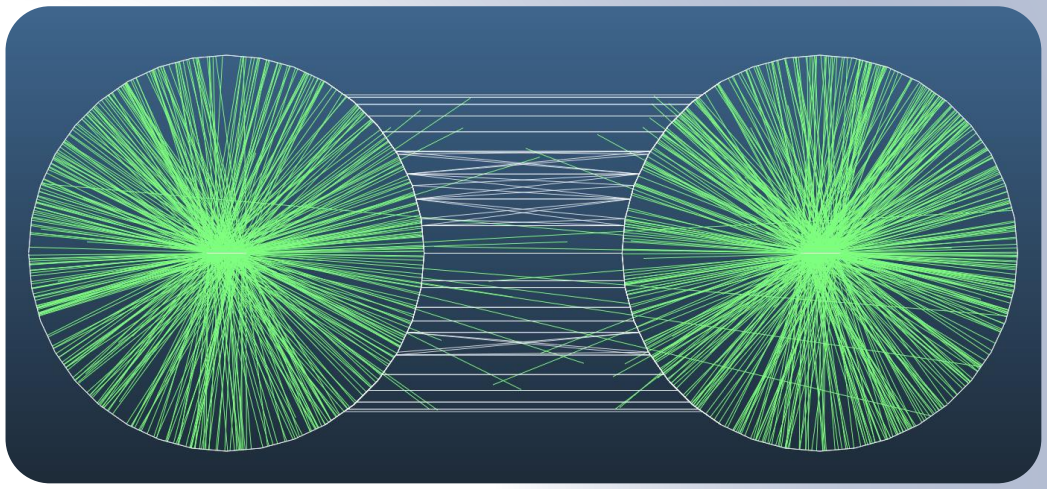
mass ratio = 5

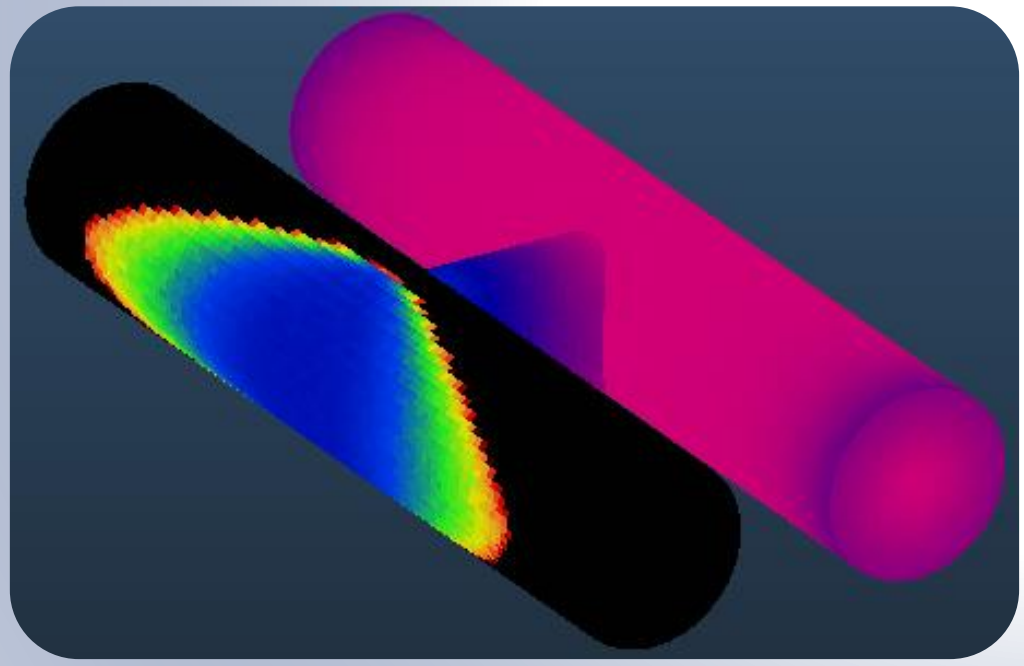


mass ratio = 0.5

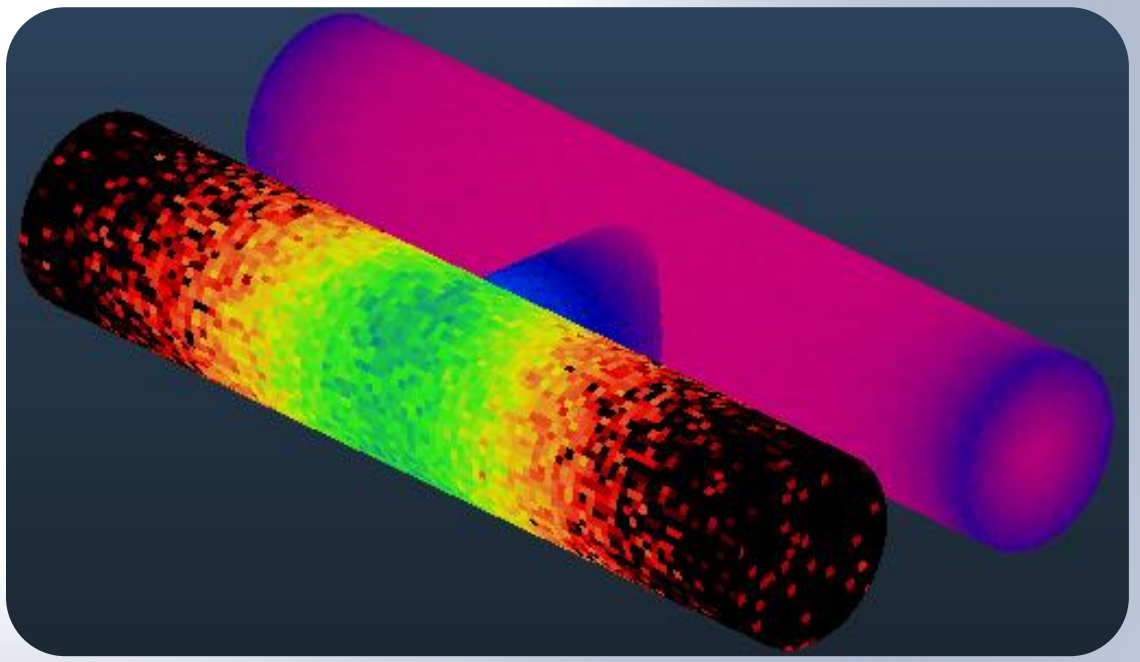
# Sputtering seminar for SCC section



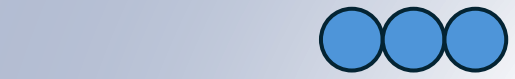


reference – no background



with background gas

# Water vapor sticking



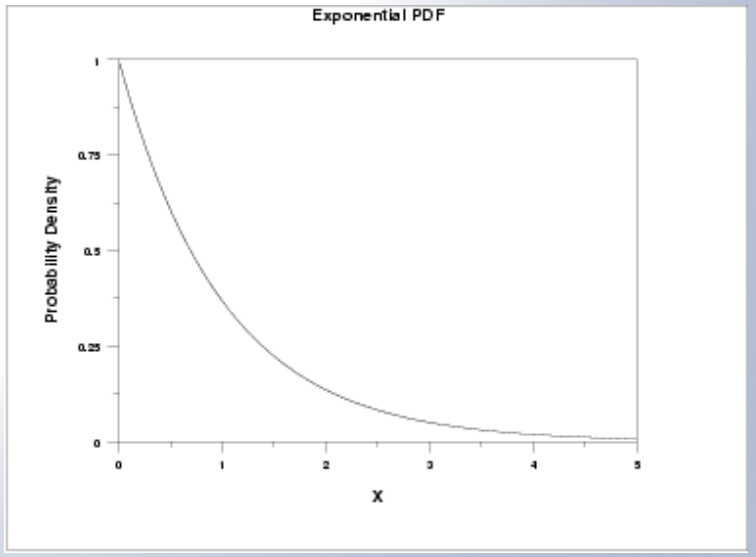
Surface

Probability of sojourn time t:

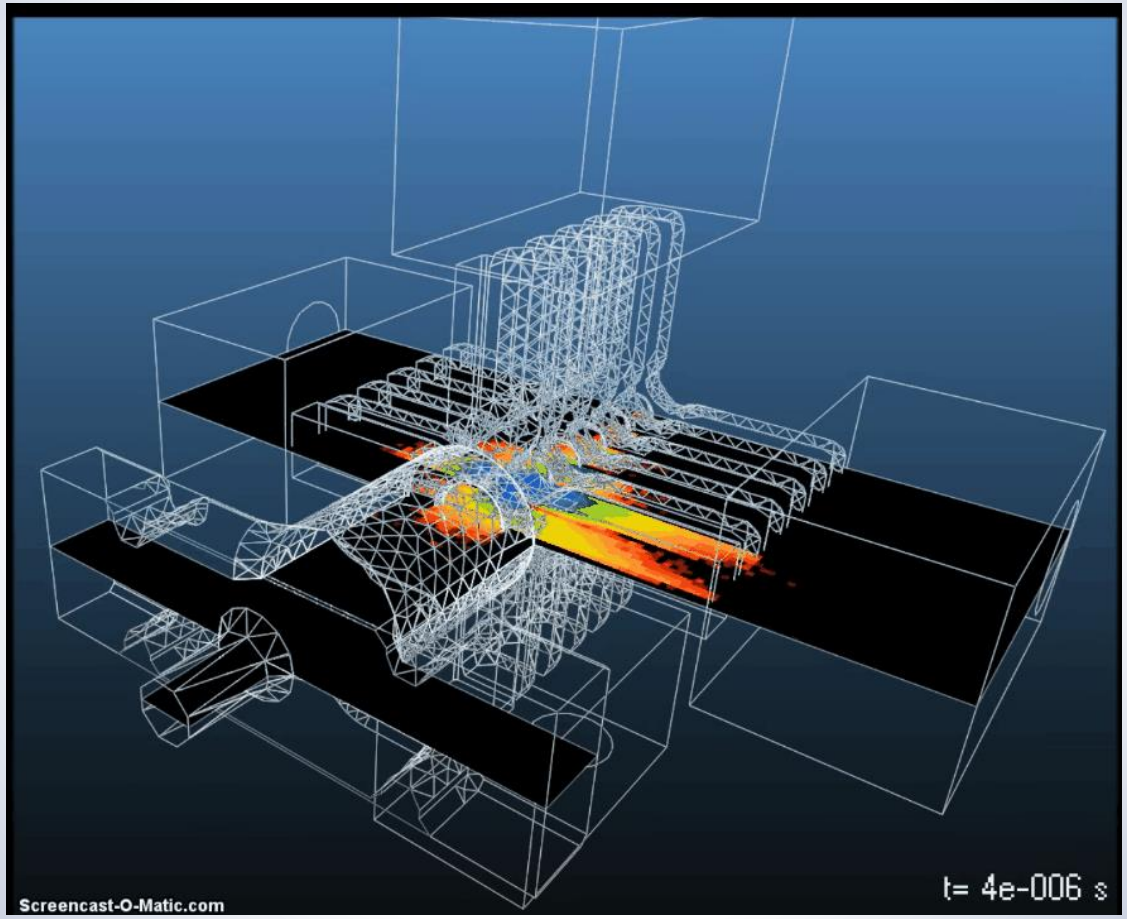
$$p(t) = Afe^{-Aft}$$

where

- f: Molecule's surface oscillation frequency [Hz]
- E: Adsorption energy [J/mole]
- A: Escape probability per oscillation: A=exp(−E/(RT))



# Time-dependent mode in MolFlow

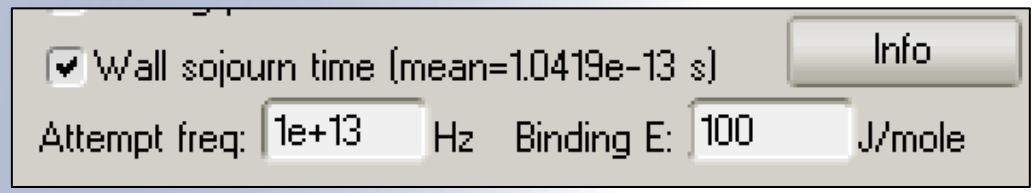



CLIC cavity geometry: Cedric Garion

Problem: No connection between independent time moments

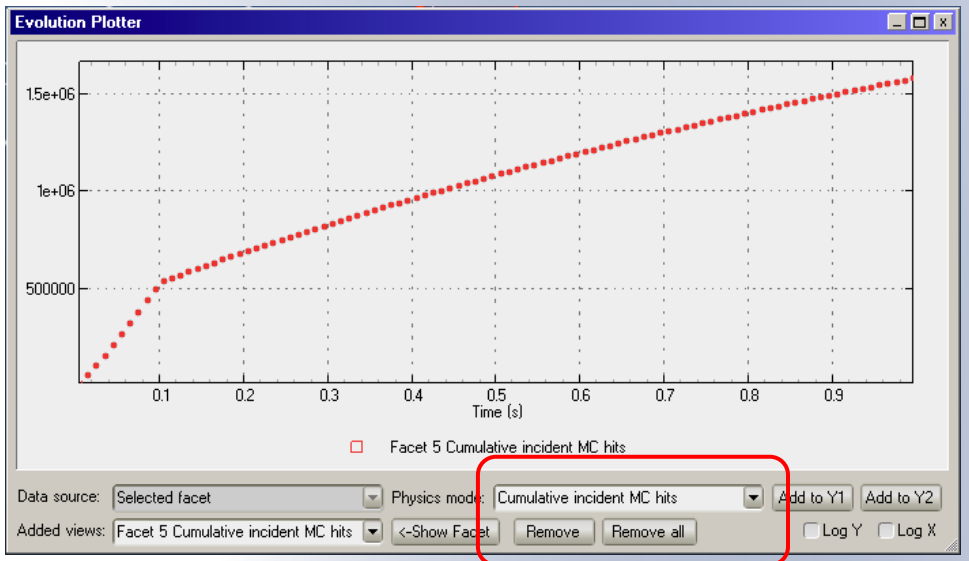
# New features

Add temperature-dependent residence time





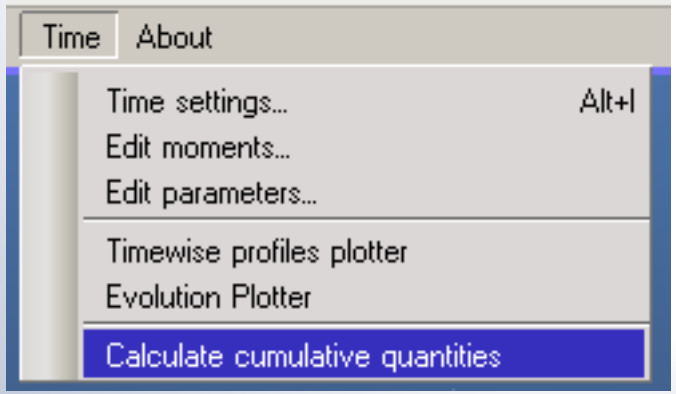
New plotter window

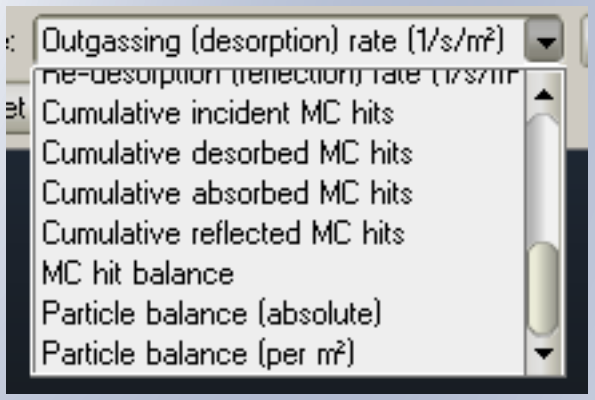



Add reflection counters

| # | Hits | Des | Abs | Refl |
|---|---|---|---|---|
| 1 | 1705444 | 237! | 1.70! | 0 |
| 2 | 670154 | 0 | 670 | 0 |
| 3 | 2936831 | 0 | 0 | 2.93683e+06 |
| 4 | 2935478 | 0 | 0 | 2.93548e+06 |
| 5 | 2936135 | 0 | 0 | 2.93614e+06 |
| 6 | 2935874 | 0 | 0 | 2.93587e+06 |
| 7 | 2935099 | 0 | 0 | 2.9351e+06 |

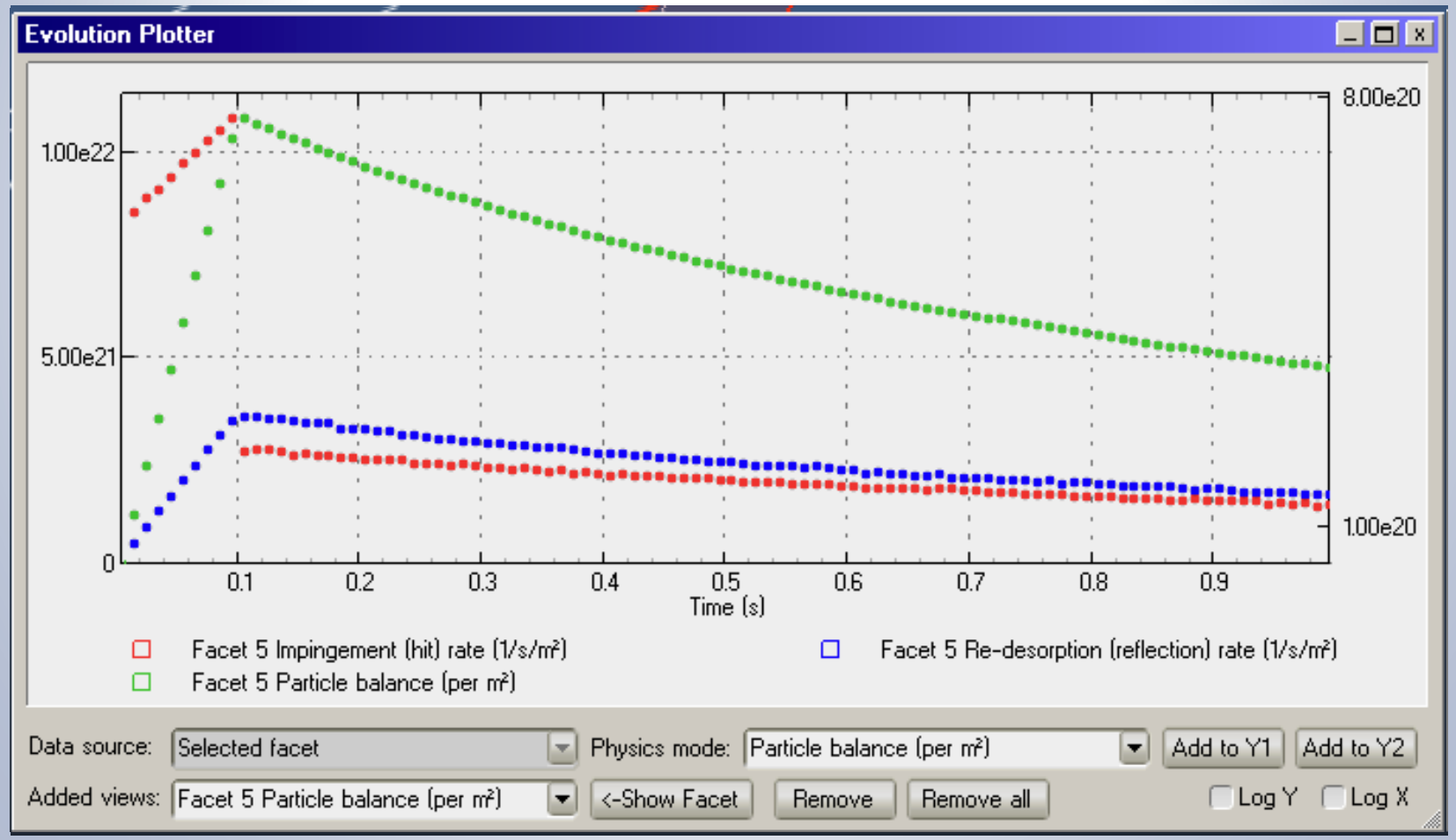
Integrate over time



# Particle balance example

- Mean residence time: 0.2s
- Outgassing: from 0 to 0.1s



# Automation with MolFlow

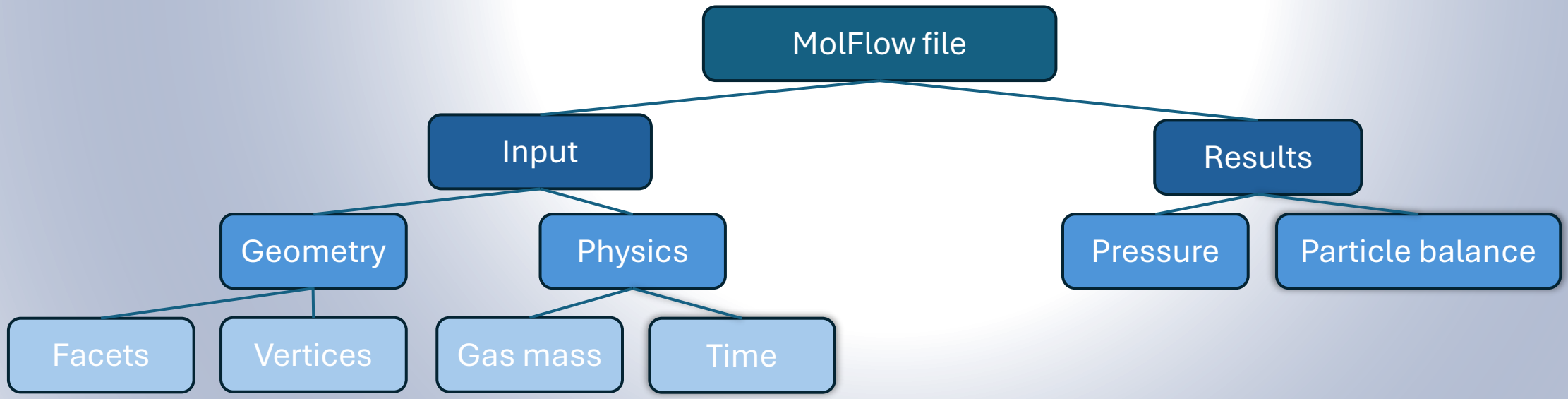

- Parameter sweep
- Iterative simulations
- Result-based optimization

molflowCLI

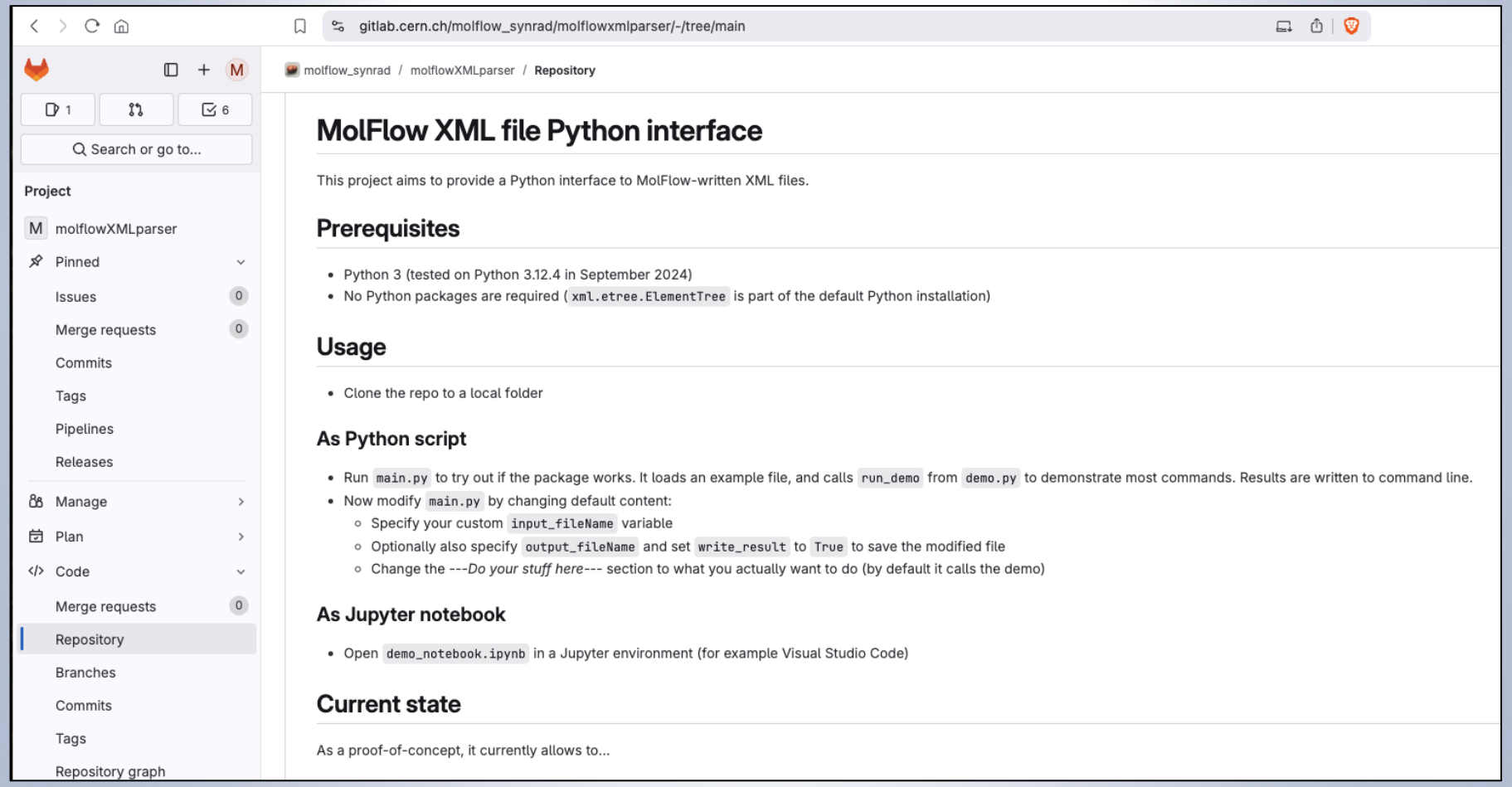
```
molflowCLI.exe --file inputFile.xml -t 3600 -o ./outputFile.xml
```

# Python XML interface

- Python is an easy to learn, popular programming language
- XML is a file format – both machine and human-readable
  - Looks like a tree
  - Non-linear, can be searched



# Published as Python module or Jupyter notebook



# Examples

Set what to work on:

## Specify input and output paths

In this example we don't write the modified file, otherwise set write_result to `True`

```
input_fileName = "examples/quickpipe.xml" # .xml or .zip
output_fileName = "out/output.zip" # .xml or .zip
write_result = False # save loaded file at script end
```

Read geometry:

## Facet count

```
facet_count = get_facet_count(root)
print(f"The geometry has {facet_count} facets.")
```

```
The geometry has 7 facets.
```

```
y_coord_2 = get_vertex(root, 2).y
print(f"The Y coordinate of vertex 2 is {y_coord_2}")
```

```
The Y coordinate of vertex 2 is 0.9510565162951535
```

Get results:

## Facets

Print pressure on facet at moment 0 (const. flow)

```
print(f"Pressure on facet id {facet_id} is {get_facet_pressure(root, facet_id, moment_id)} mbar")
```

```
Pressure on facet id 3 is 0.17865621366500498 mbar
```

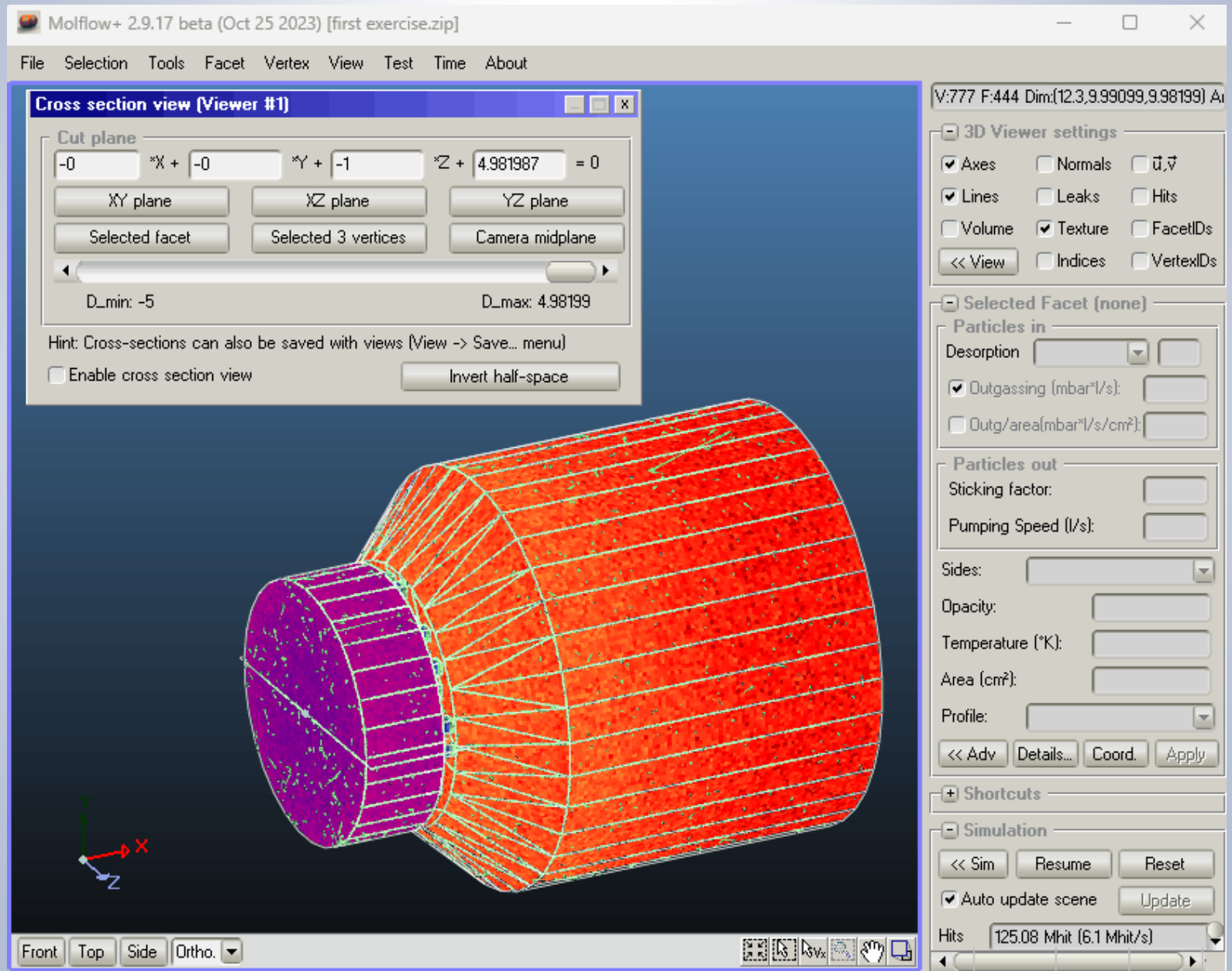
Change physics:

Update the temperature on facet to 400K

```
print(f"Temp on facet id {facet_id} was {get_temperature(facet_node)} K")
set_temperature(facet_node, 400.0)
print(f"Temp on facet id {facet_id} is now {get_temperature(facet_node)} K")
```

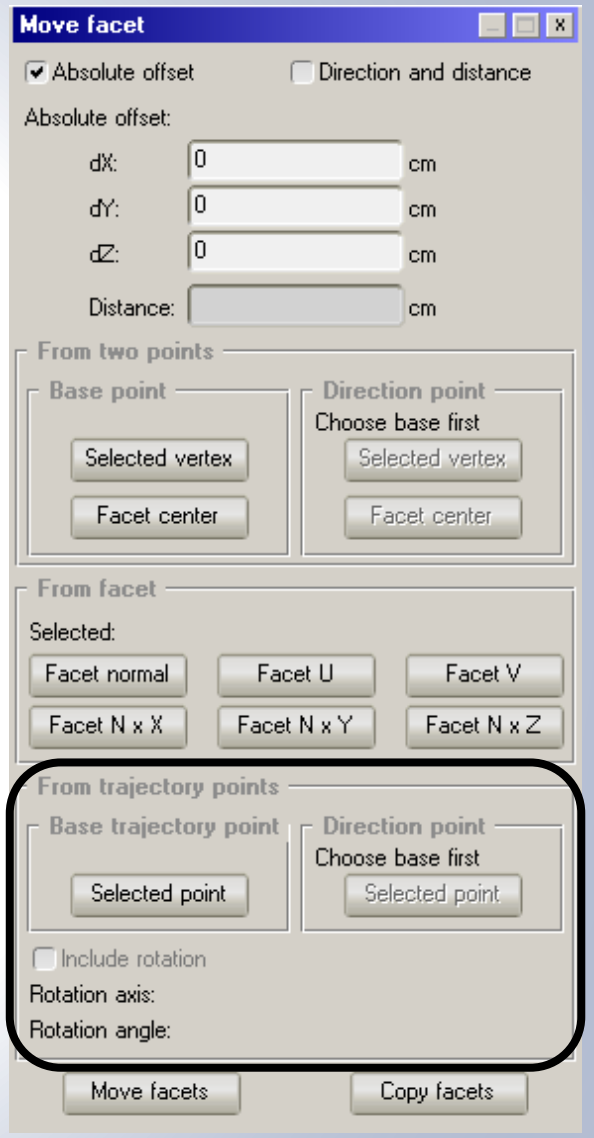
```
Temp on facet id 3 was 293.15 K
Temp on facet id 3 is now 400.0 K
```

# Cross-section viewer



# Rich facet and vertex movement

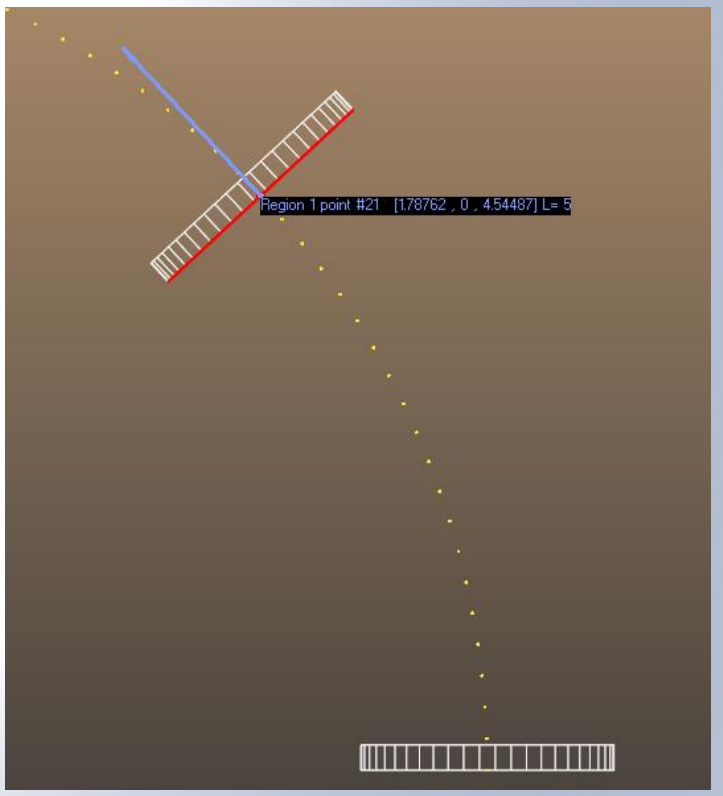
- SynRad movement using beam trajectory points
  - Two points of a trajectory define the movement parameters:
    - Direction,
    - Distance,
    - Rotation.



# Rich facet and vertex movement

- SynRad movement using beam trajectory points
  - Two points of a trajectory define the movement parameters:
    - Direction,
    - Distance,
    - Rotation.

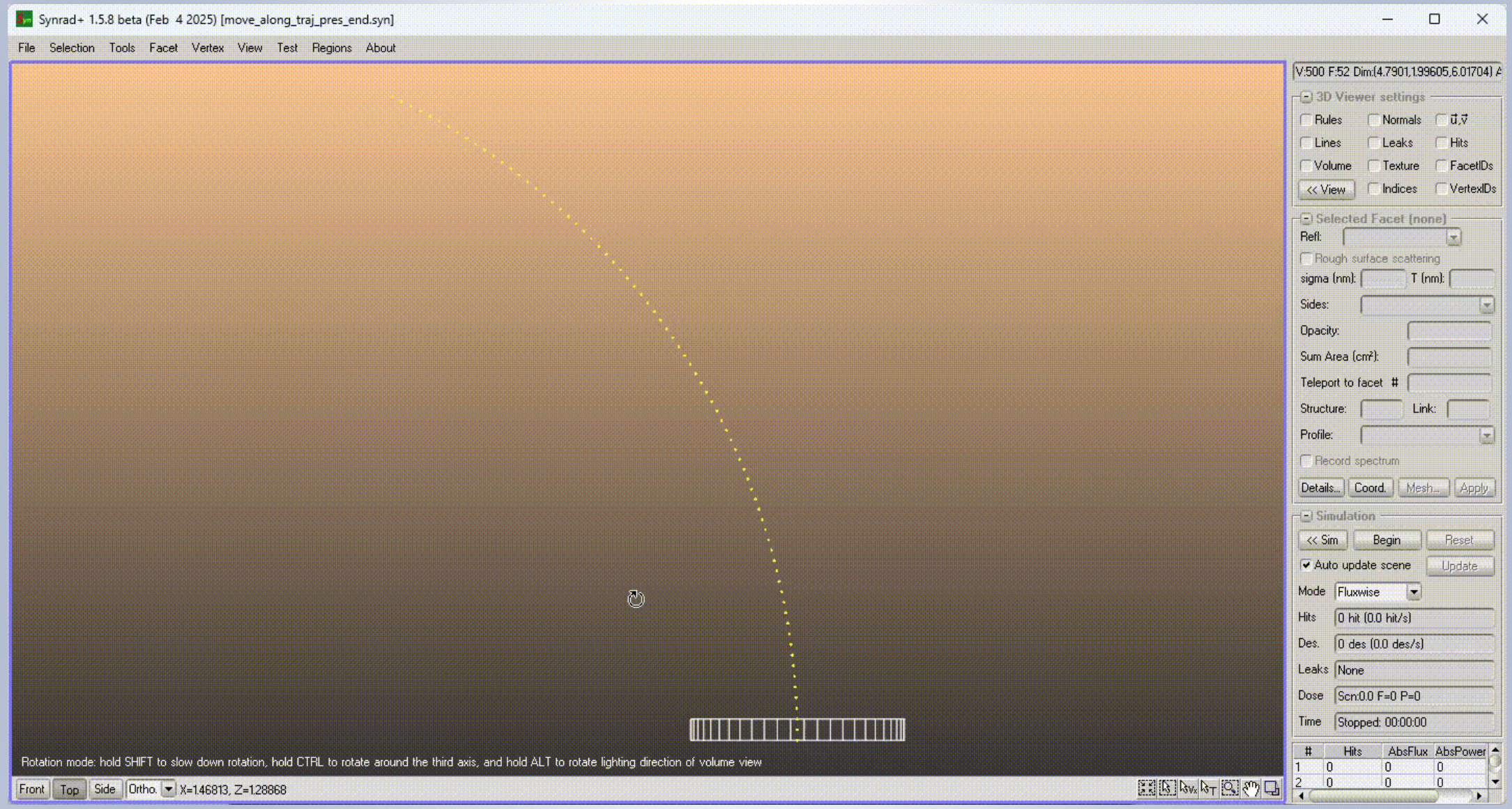
**Example (top view):**



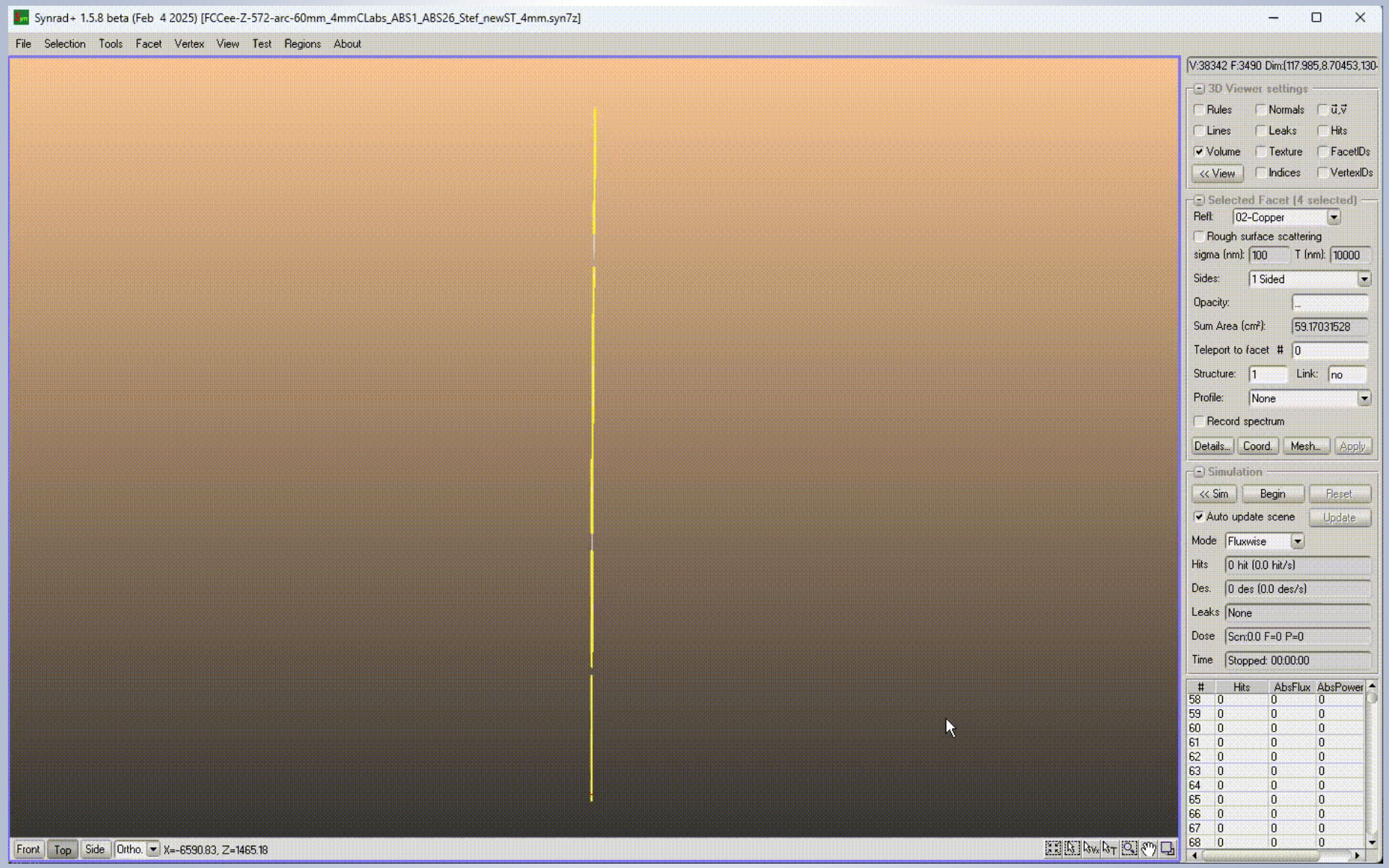
# Rich facet and vertex movement

- SynRad movement using beam trajectory points:
  - Why add this feature?
- Problem:
  - Moving or duplicating smaller pieces of geometry along longer pipes.
  - Long pipes usually contain beams whose trajectory is aligned with the pipe.
- The old “solution”:
  - Using the *Move* and *Rotate* options separately, manually calculating the rotation.
- The new solution:
  - Selecting the appropriate trajectory points while SynRad does the rest.

# Rich facet and vertex movement – a simple demo

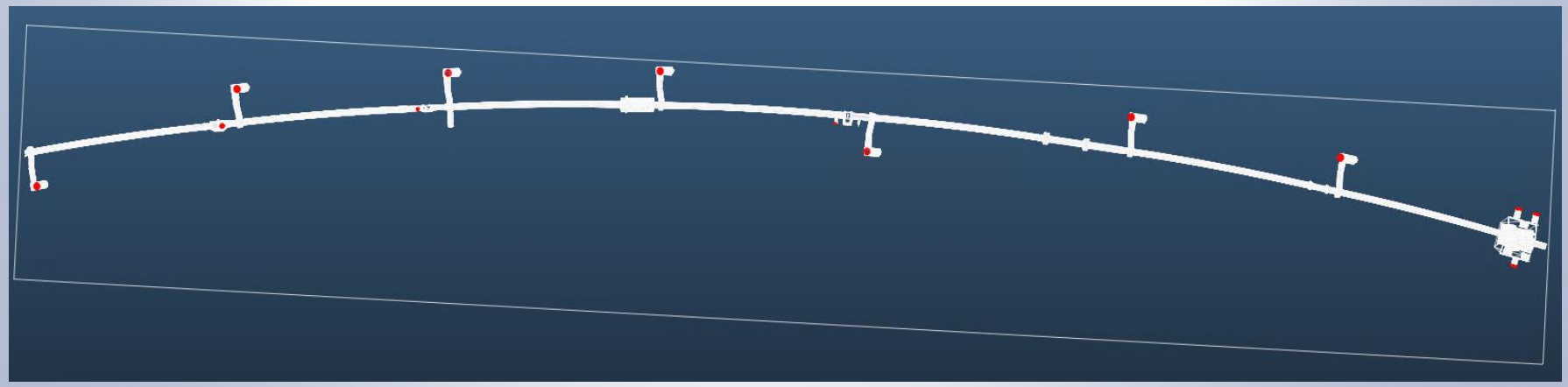


# Rich facet and vertex movement – FCCee demo



# New feature – extracting data along a path

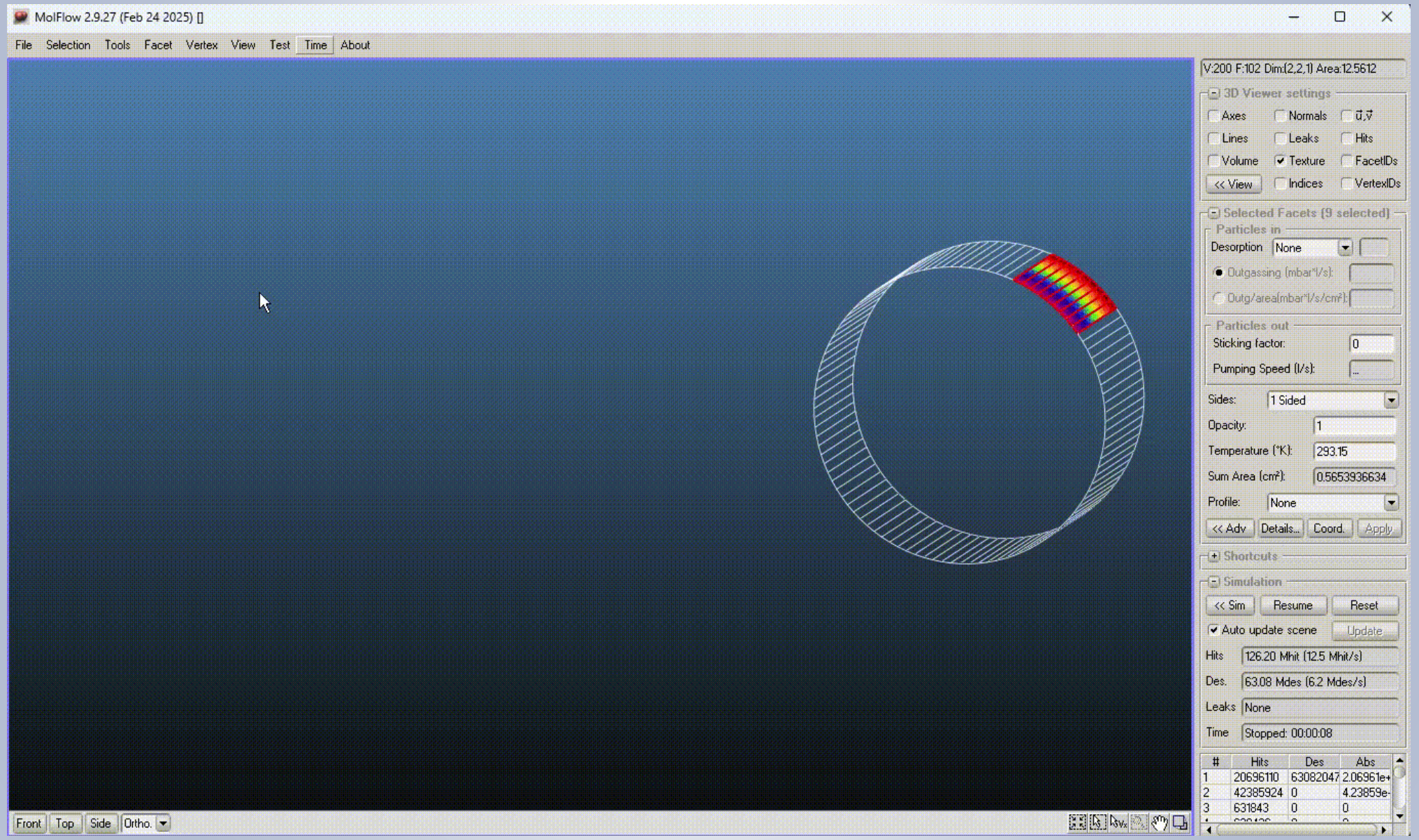
- Problem:
  - It is often required to extract simulation results (e.g. pressure, impingement rate…) along a certain path in the geometry.
- The old solution:
  - For straight paths, it could be sufficient to add a transparent facet that stretches along the desired path and “records” results.
  - For curved paths, this can be hard:



# New feature – extracting data along a path

- Problem:
  - It is often required to extract simulation results (e.g. pressure, impingement rate…) along a certain path in the geometry.
- The old solution:
  - For straight paths, it could be sufficient to add a transparent facet that stretches along the desired path and “records” results.
  - For curved paths, this can be hard.
- The new solution:
  - Allow the user to define a data extraction path by specifying the points in space it consists of.
  - Read simulation results from the surrounding facets.

# Extracting data along a path – a preview



# Extracting data along a path – an important use case

- FCC-ee:
  - Using curved geometries at CERN is a very common practice.
  - The new feature will be useful for simulations for the FCC-ee.
  - The SynRad version will include data extraction along trajectory points