Simulation and Visualisation of e-column in IOTA ring using ParaView

Diletta Milana

under the supervision of

Charles Thangaraj (FAST-IOTA)

and

Chong Shik Park (IOTA), Giulio Stancari (IOTA)

Fermilab

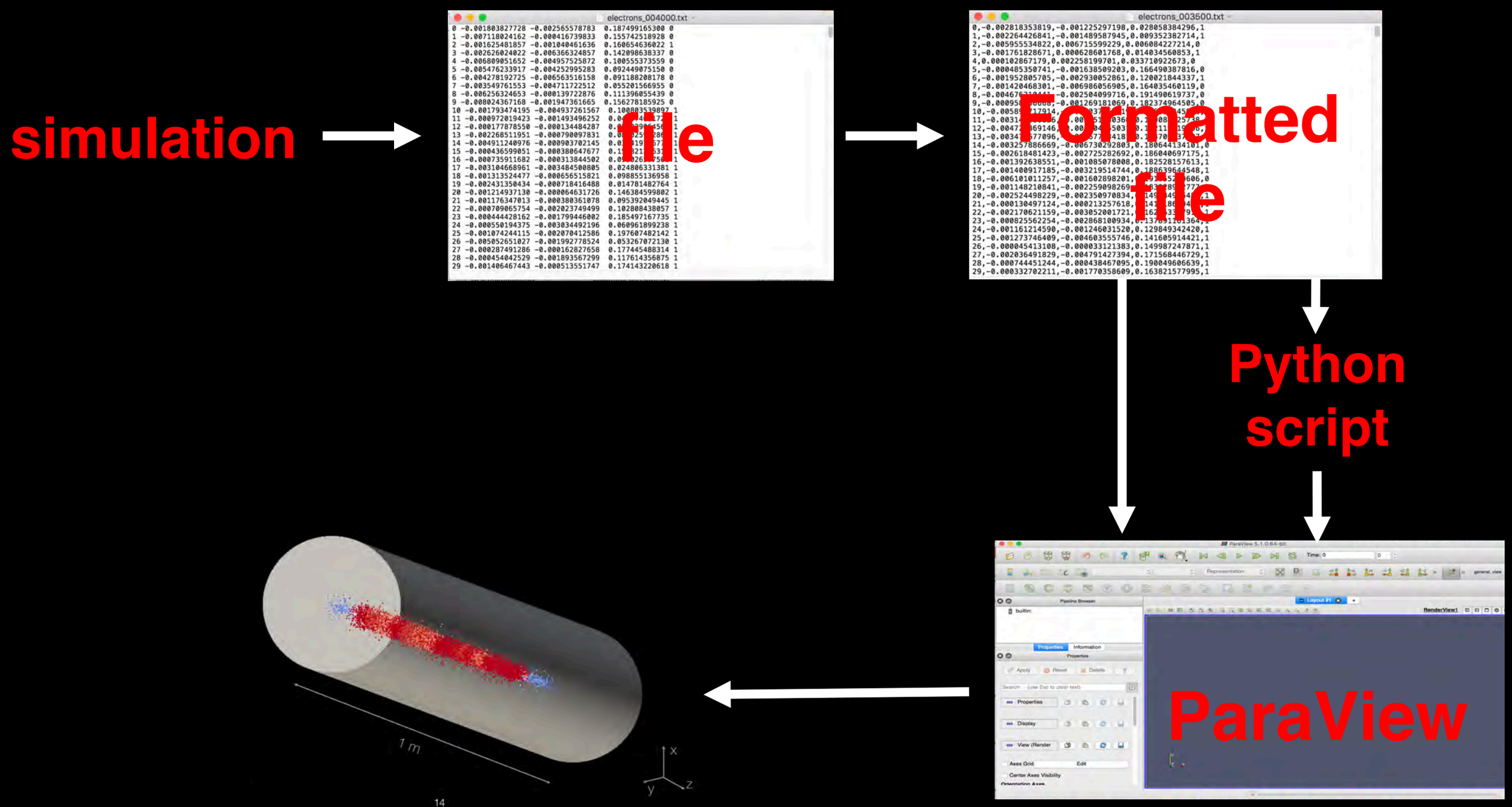
POLITECNICO MILANO 1863

Paraview

An open-source, multi-platform and extremely flexible data analysis tool for extremely large datasets

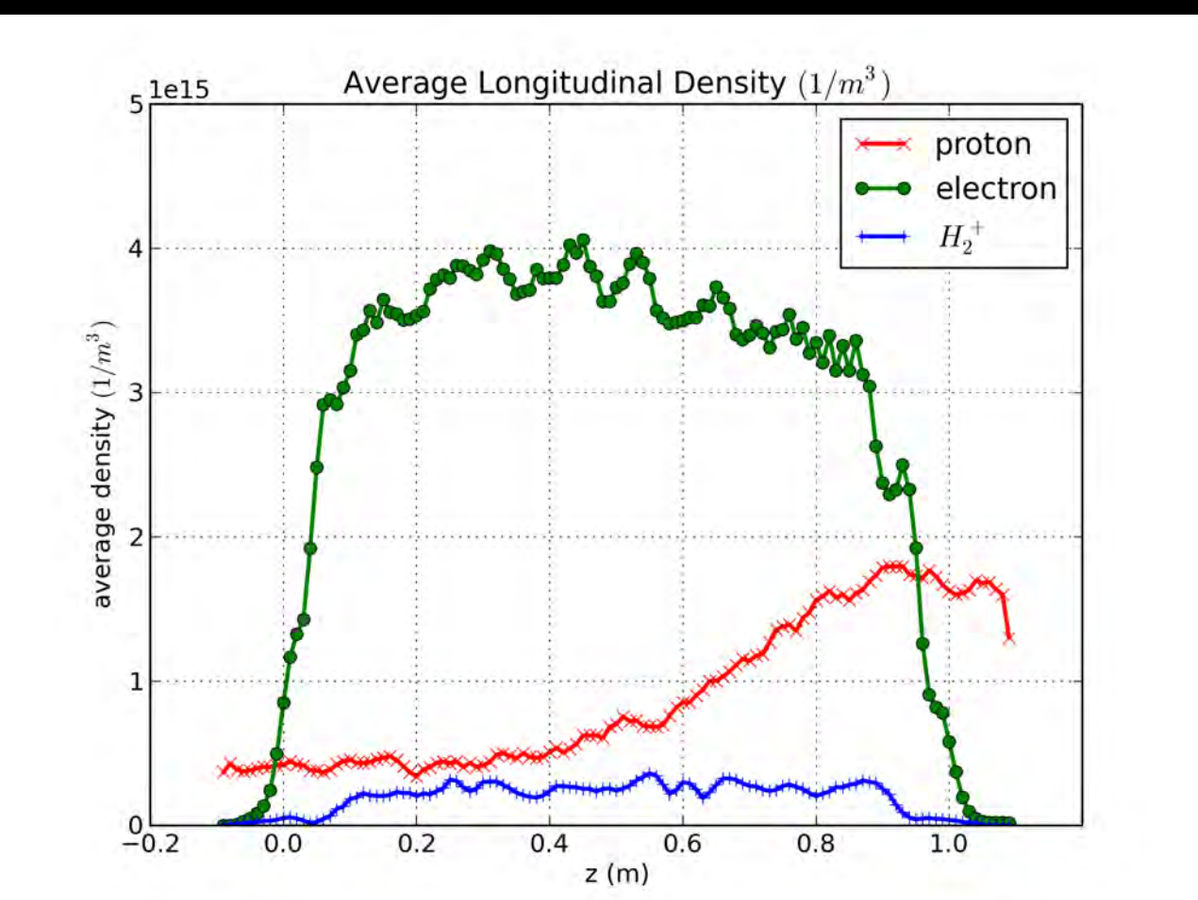
Great, but why do we need this?

The pipeline (1)

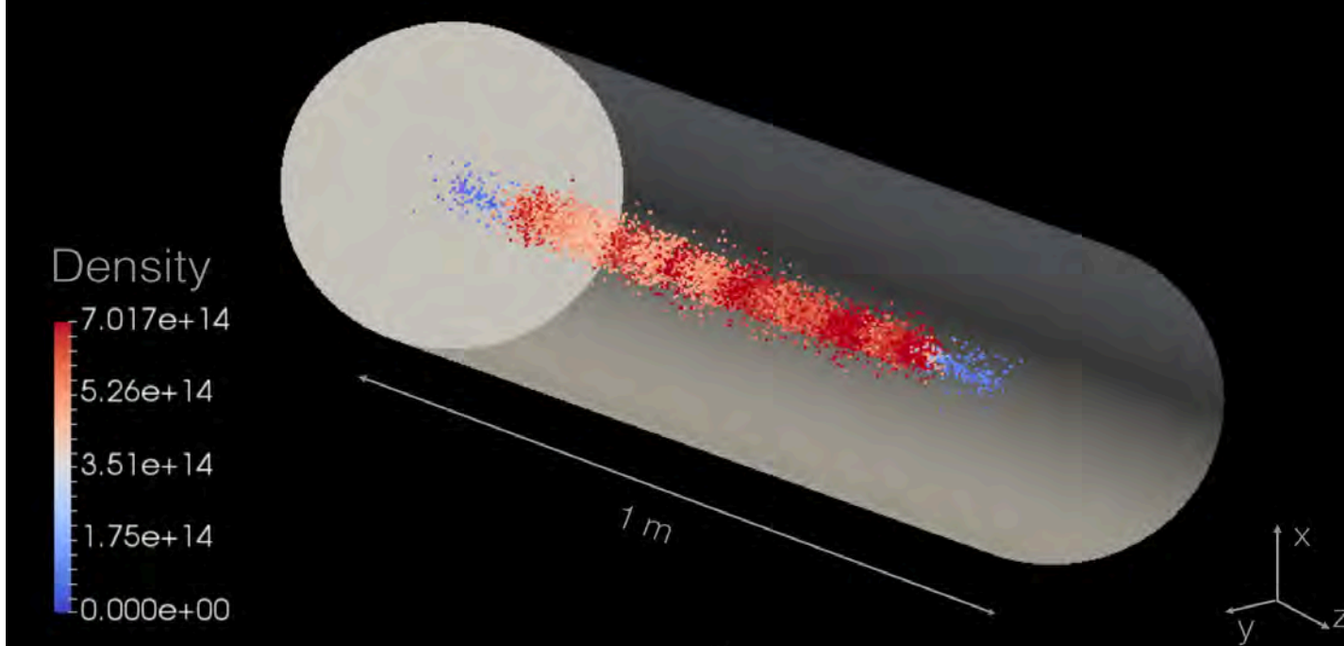


Example: density profile

Traditional 2D view



ParaView's 3D view



A concrete application: the electron column simulation

Why electron columns?

- The performance of high-power accelerators, synchrotrons and storage rings is strongly limited by losses and instabilities, such as the space charge effect.

- Space-charge forces in a beam result from mutual Coulomb repulsion, creating an internal electric field.

- This could lead to beam losses that reduce and limit the current.

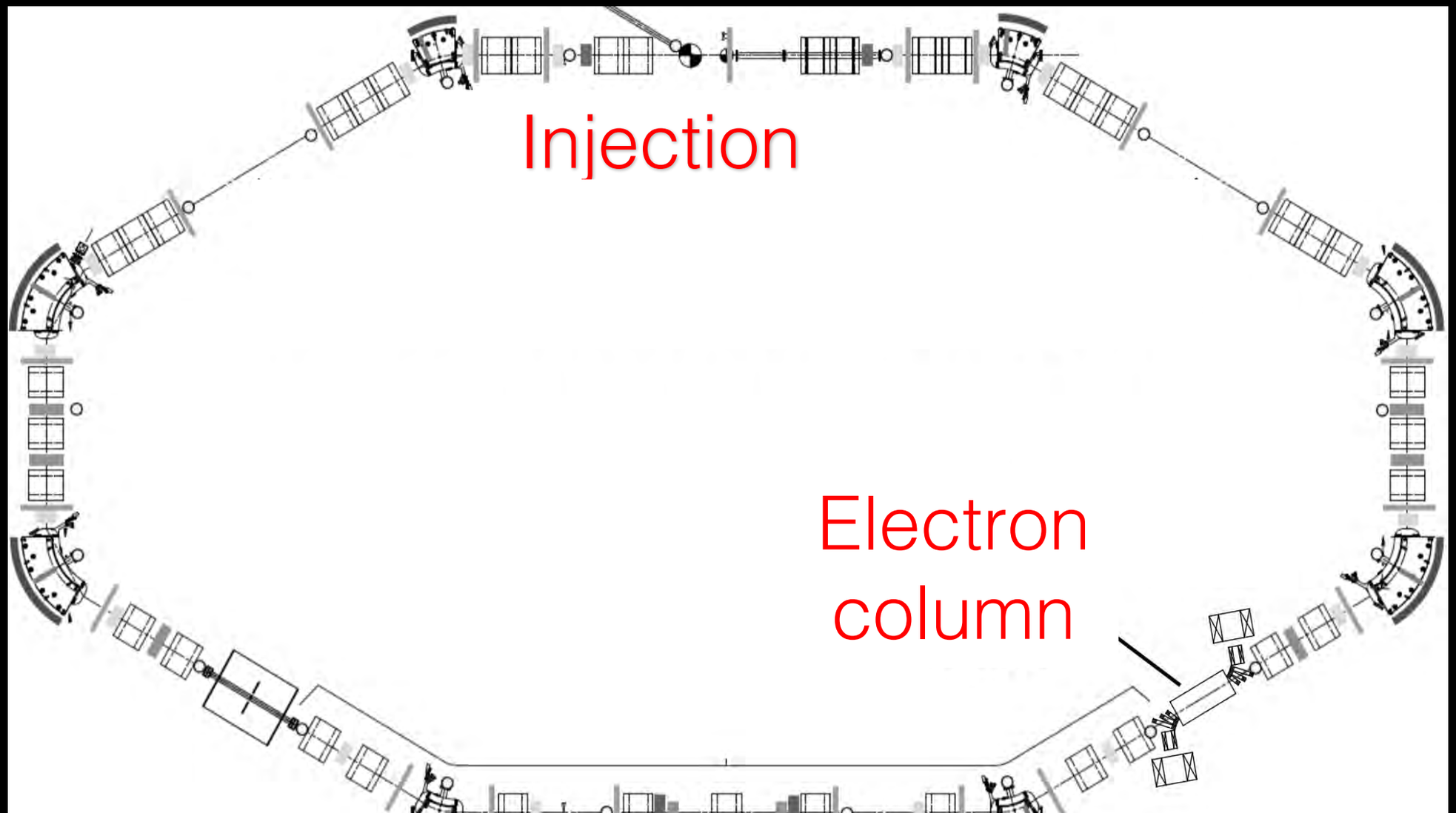
Space Charge Compensation

- To compensate we need opposite charge. Two ways to do it: electron lens or electron column. Here we focus on the e-column method.

- The e-column has potential to improve the performance of circular accelerators.

Schematics of IOTA ring

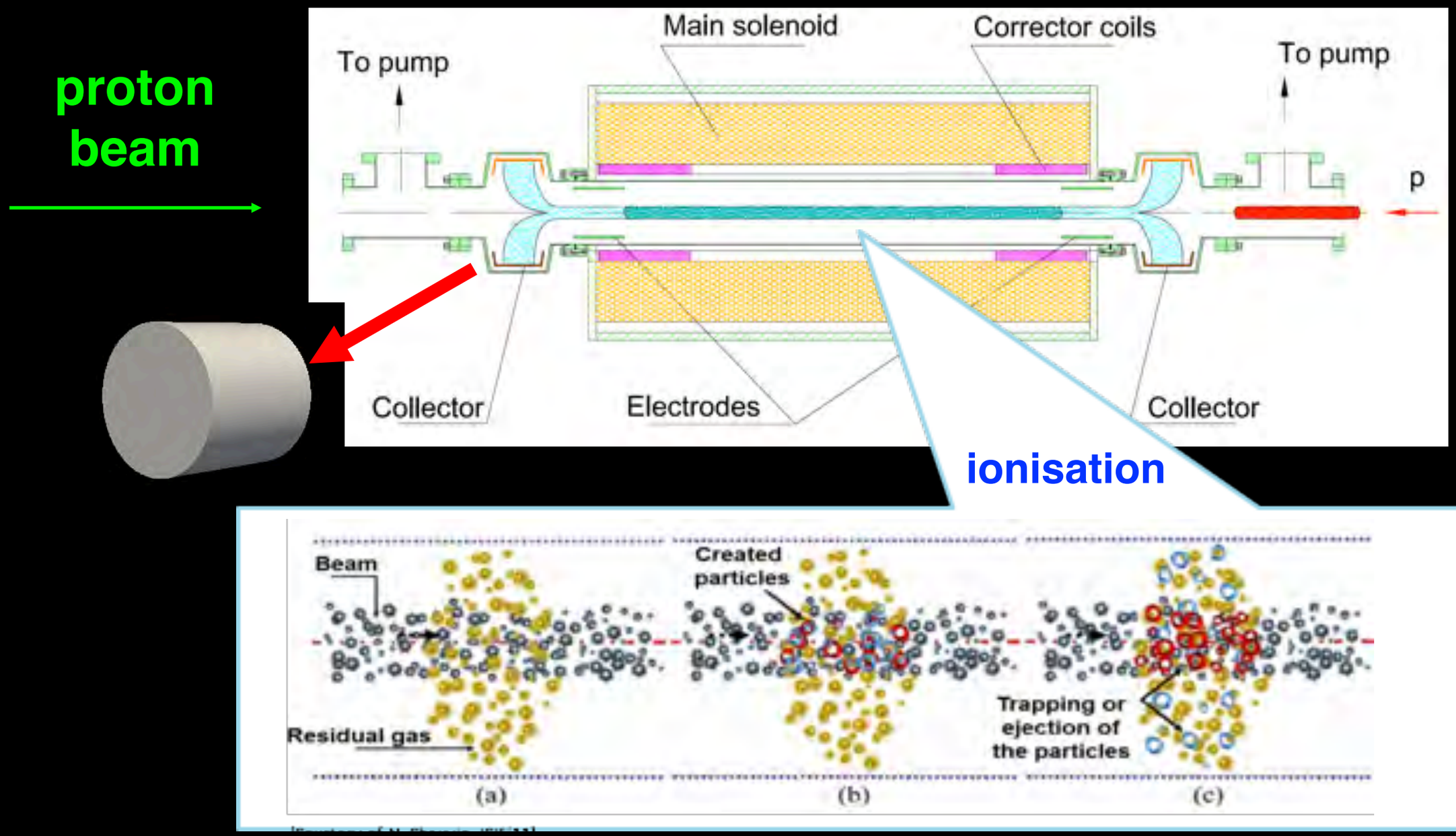
Integrable Optics Test Accelerator



The electron column

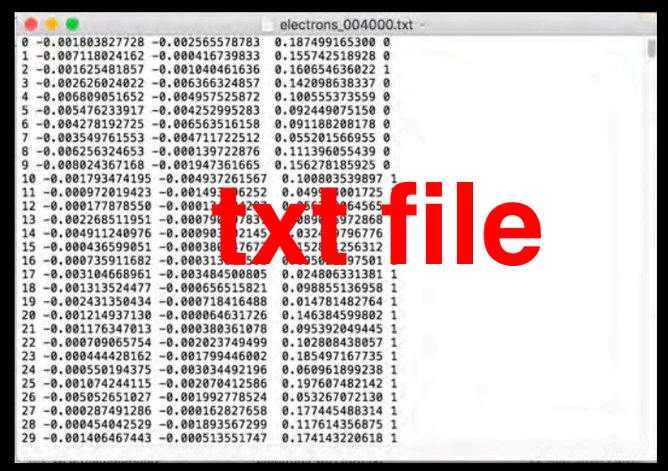
Vladimir Shiltsev's experimental setup

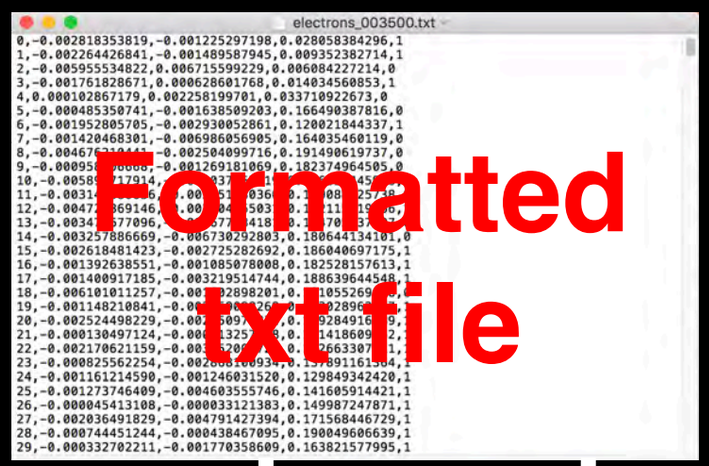
$$p + H_2 \longrightarrow p + H_2^+ + e$$



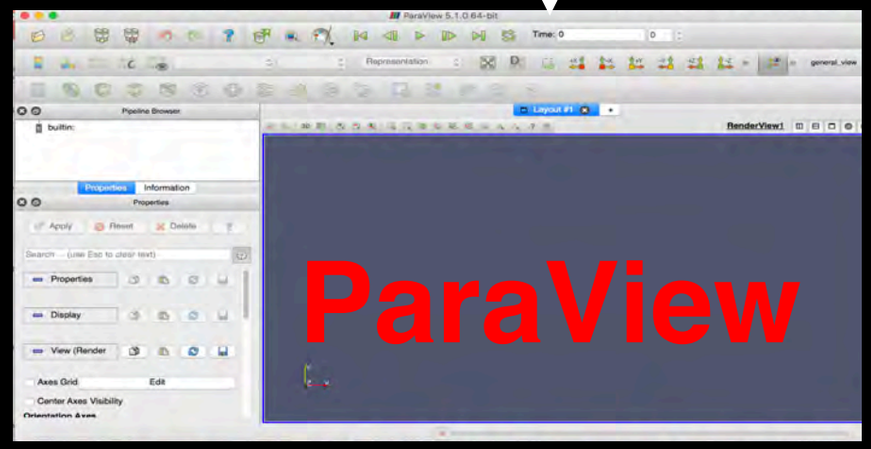
The pipeline (2)

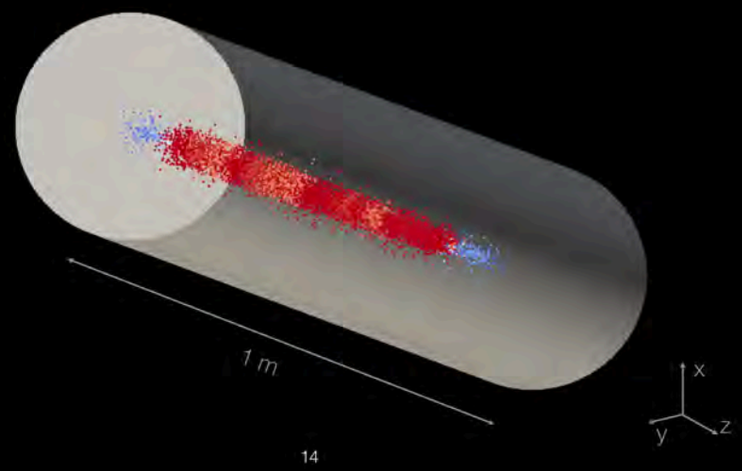
WARP simulation





Python script





Python for Paraview

- Readers, filters, render views.
- Each visualisation can be generated by a python script, that takes as input the results of the simulation.
- Starting from this code, anyone can run the visualisation scripts and setting the appropriate parameters

…let’s get started!

General settings (1)

reset and read

```
from paraview.simple import *
import sys
import os
import glob

#indicates where the output files (of the simulation) can be found
output_path='/Users/diletta/Desktop/format_output/'

#indicates where all imported scripts can be found
PATH_TO_SCRIPTS="/Users/diletta/Desktop/Fermiworking/fermiscripts/paraview/"

sys.path.append(os.path.abspath(PATH_TO_SCRIPTS))

#reset current session
from reset import *
reset_all()

#reading input data
electrons_ = CSVReader(FileName=glob.glob(output_path+"electrons_*"))
```

General settings (2)

text annotations

```
protonsText = Text()
protonsTextDisplay = Show(protonsText, renderView1)
protonsText.Text = 'Protons'
protonsTextDisplay.FontSize = 8
protonsTextDisplay.WindowLocation = 'AnyLocation'
RenameSource('ProtonsText', protonsText)

protonsTextDisplay = GetDisplayProperties(protonsText,
view=renderView1)
protonsTextDisplay.Color = [0.12259098191805905, 1.0,
0.16134889753566797]
protonsTextDisplay.Position= [0.90,0.80]
```

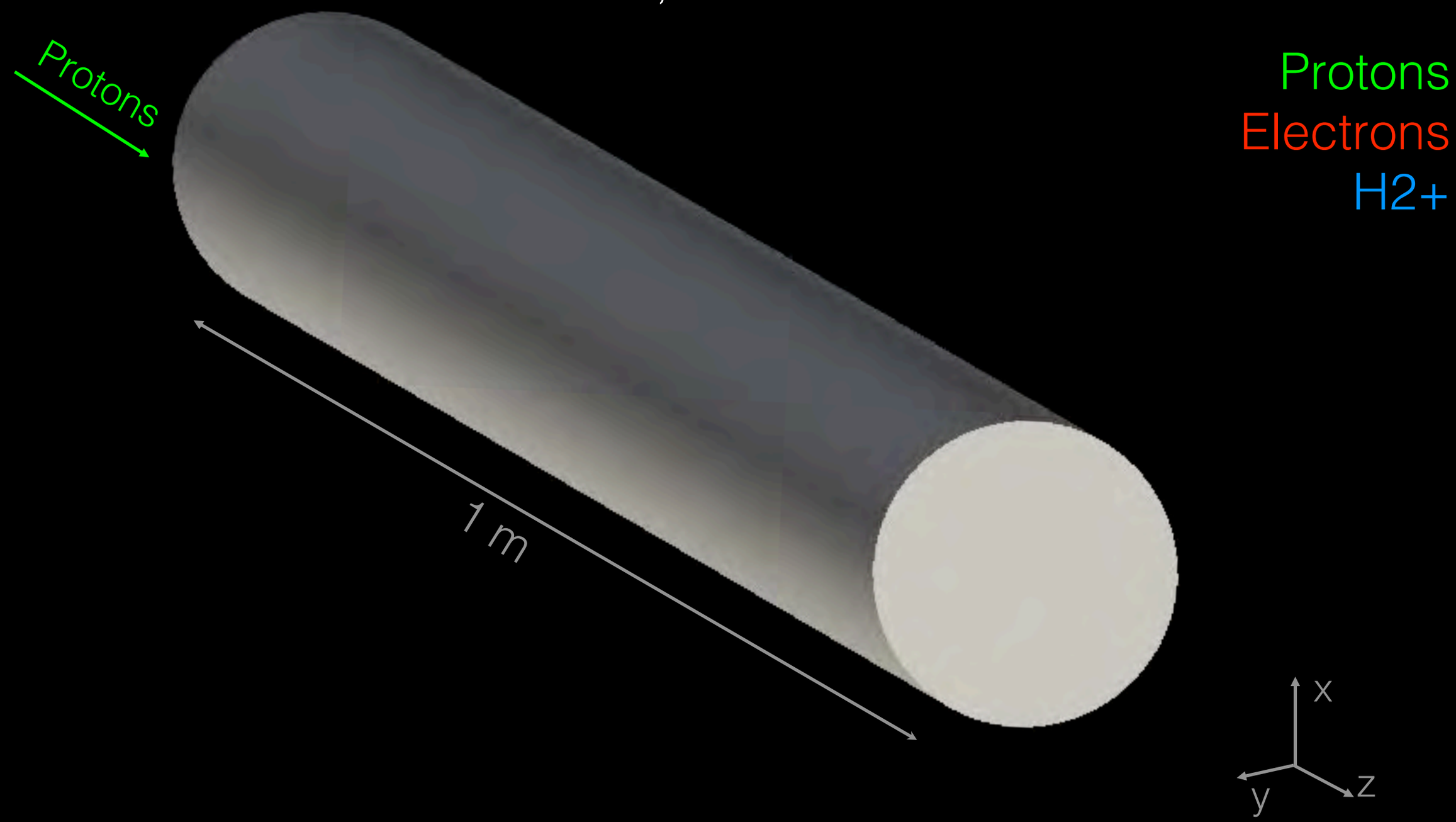
General settings (3)

creating the pipe

```
# create a new cylinder (the pipe)
cylinder1 = Cylinder()
cylinder1.Resolution = 350
cylinder1.Height = 1.1
cylinder1.Radius = 0.03
cylinder1.Center = [0.0, 0.0, 0.0]

# Transforming the pipe
transform1 = Transform(Input=cylinder1)
transform1.Transform = 'Transform'
transform1.Transform.Translate = [0.0, 0.0, 0.55]
transform1.Transform.Rotate = [90.0, 0.0, 90.0]
transform1.Transform.Scale = [1.0, 1.0, 1.0]

# show data in view
transform1Display = Show(transform1, renderView1)
```

Evolution of the electron column

B=0.1 T, V=-10 V



Evolution of the electron column

python script

```
#creating a new reader
electrons_ =
CSVReader(FileName=glob.glob(output_path+"electrons_*"))

# filtering
tableToPoints1 = TableToPoints(Input=electrons_)
tableToPoints1.XColumn = 'Field 1'
tableToPoints1.YColumn = 'Field 2'
tableToPoints1.ZColumn = 'Field 3’

# filtering
clip1 = Clip(Input=tableToPoints1)
clip1.ClipType = 'Scalar'
clip1.Scalars = ['POINTS', 'Field 3']
clip1.Value = 0.0
```

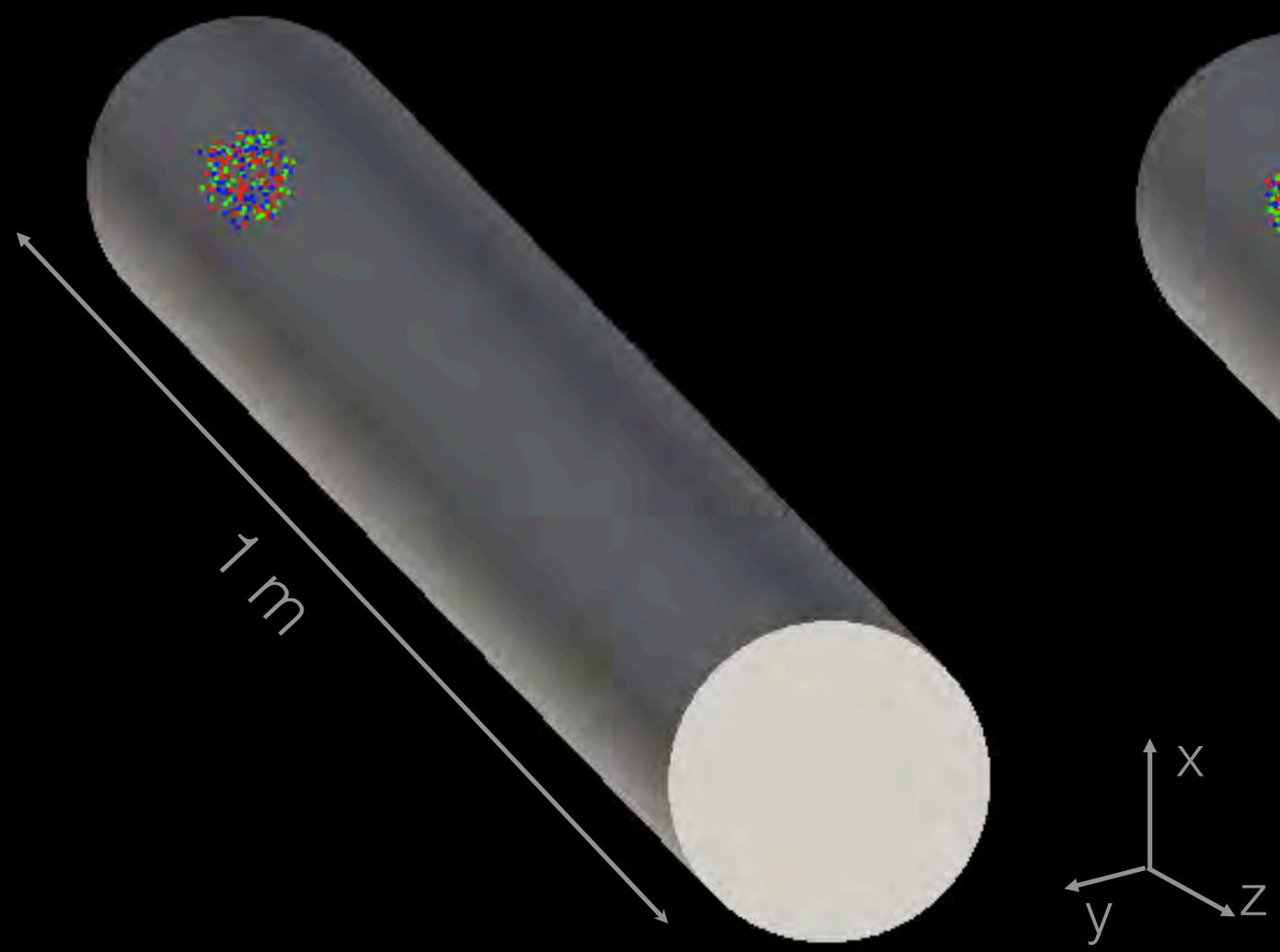
Evolution of the electron column

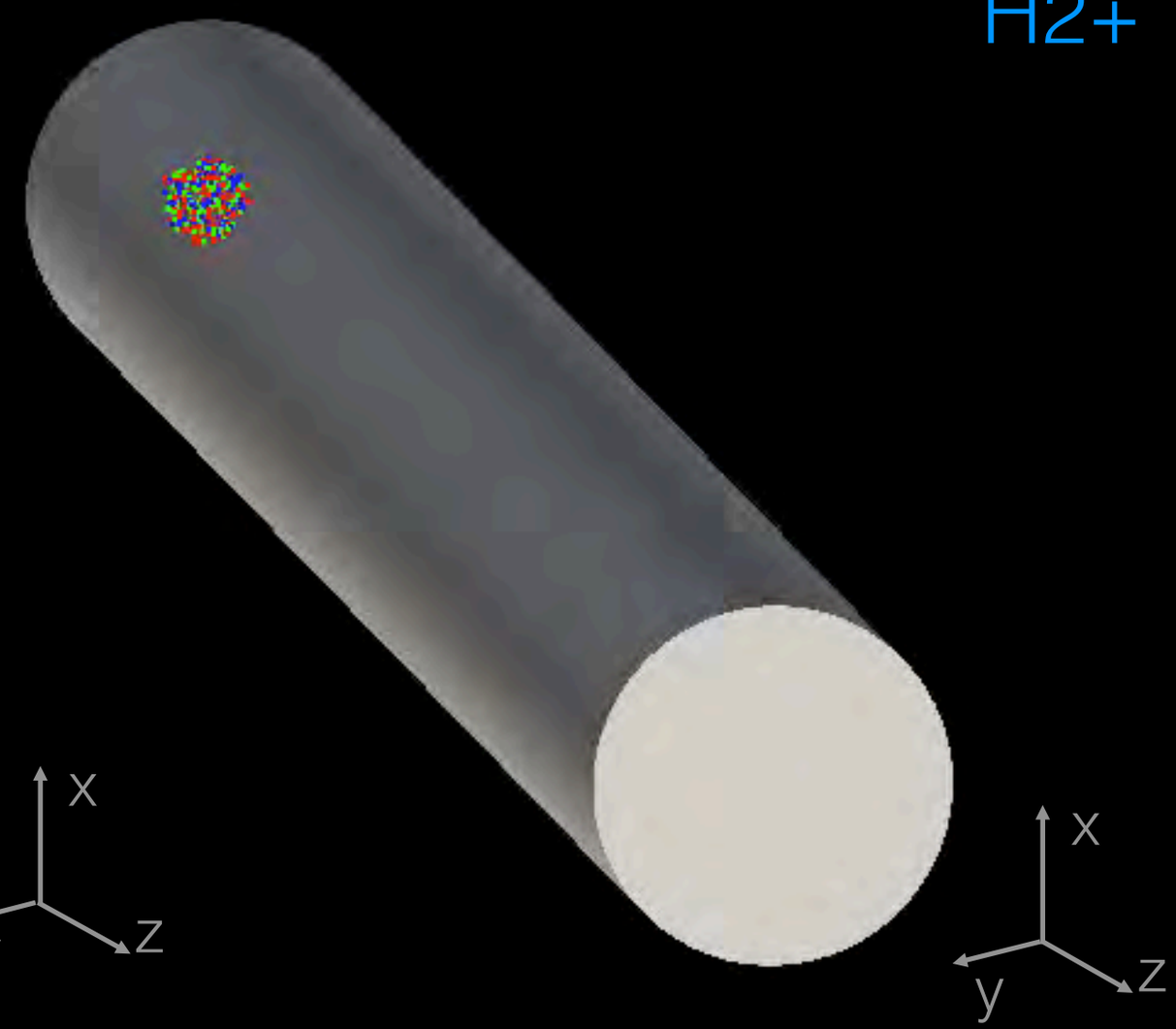
comparing different runs

B=0.1 T , V=-10 V

B=0.1 T , V=-5 V

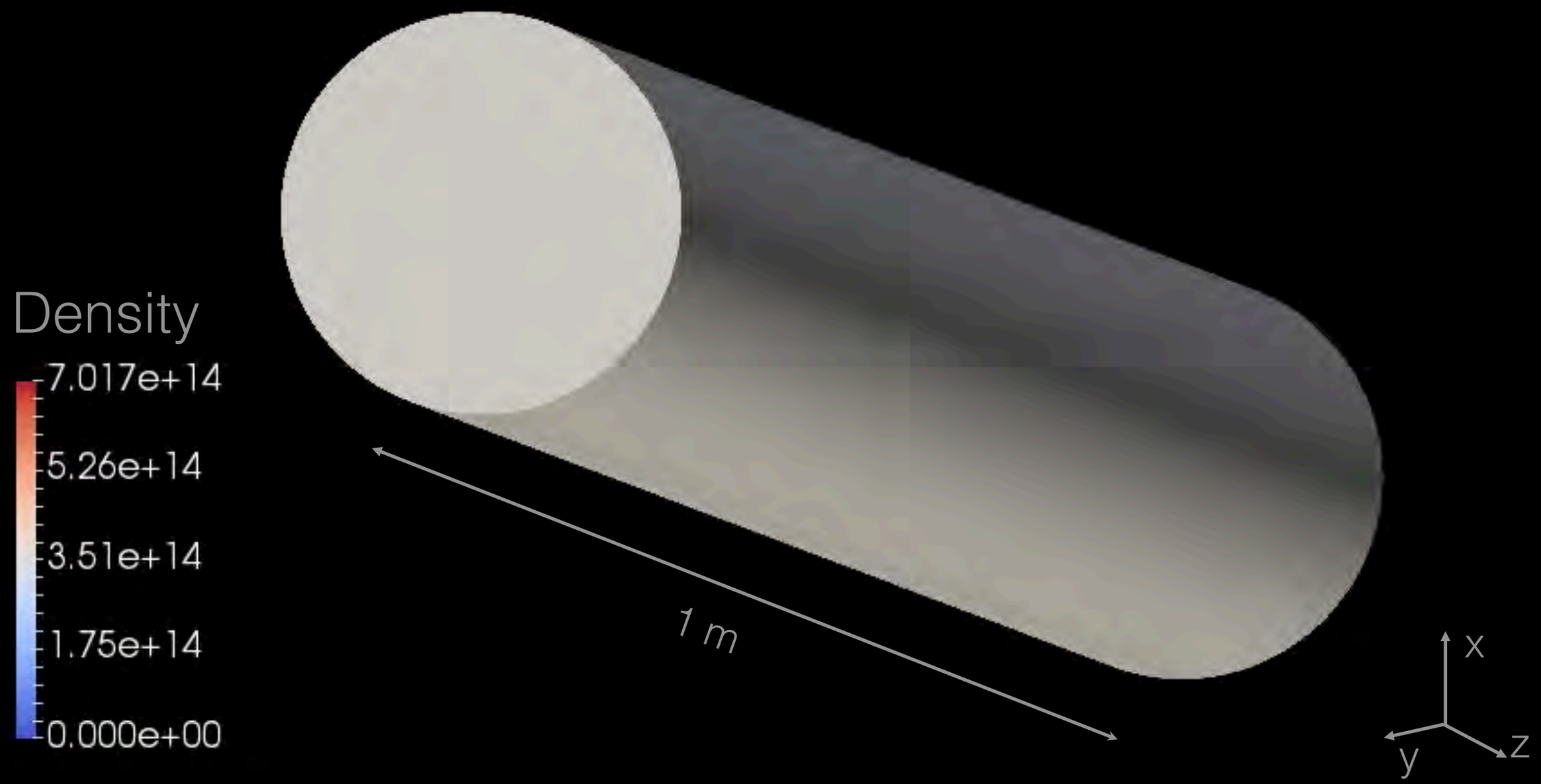
Protons
Electrons
H2+





Mapping the electron density

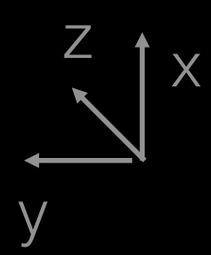
B=0.1 T, V=-10 V

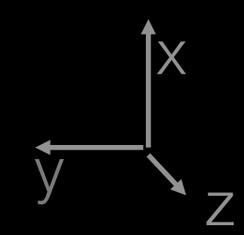


Detail: end views

B=0.1 T, V=-10 V

Protons
Electrons
H2+

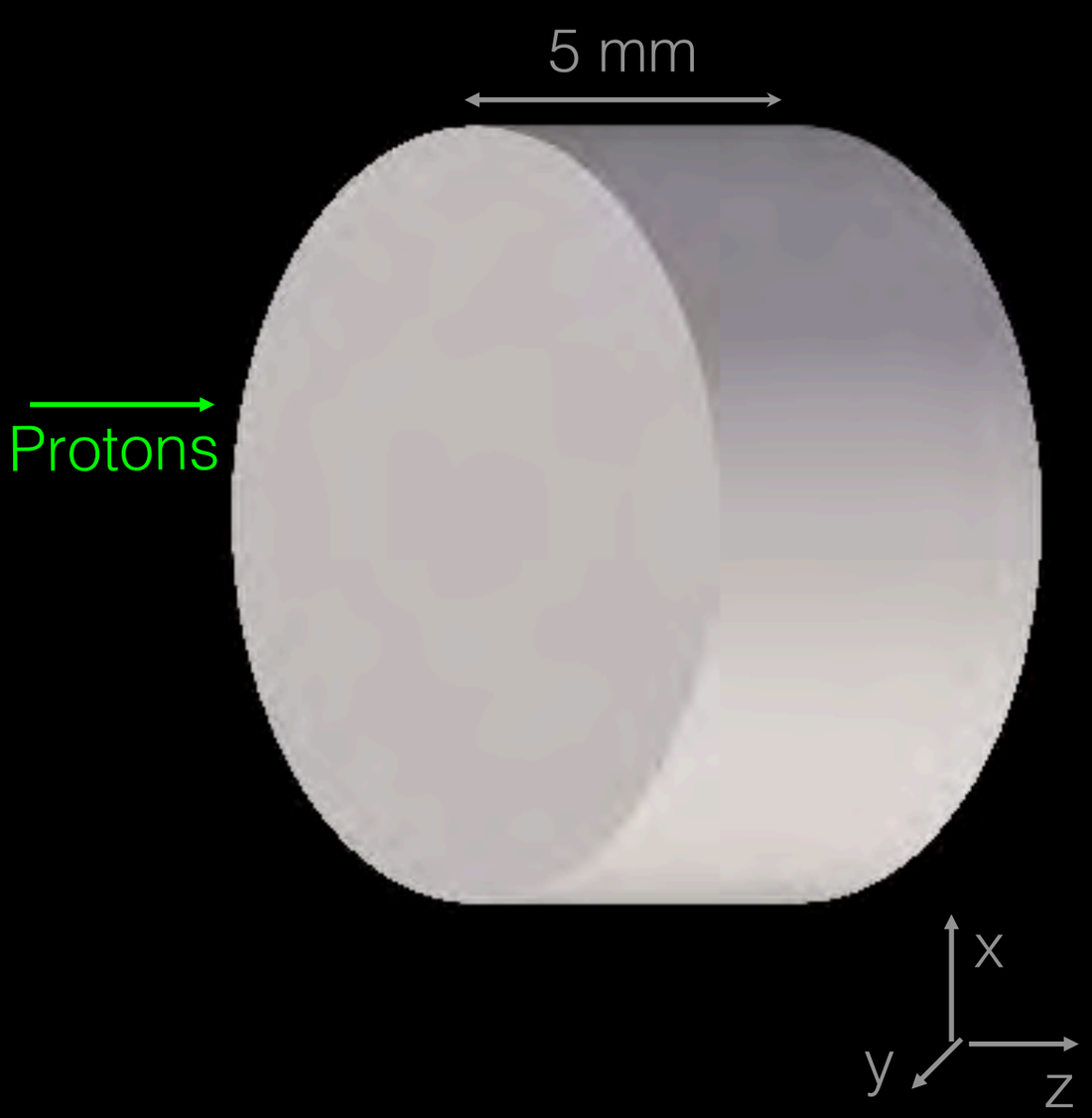


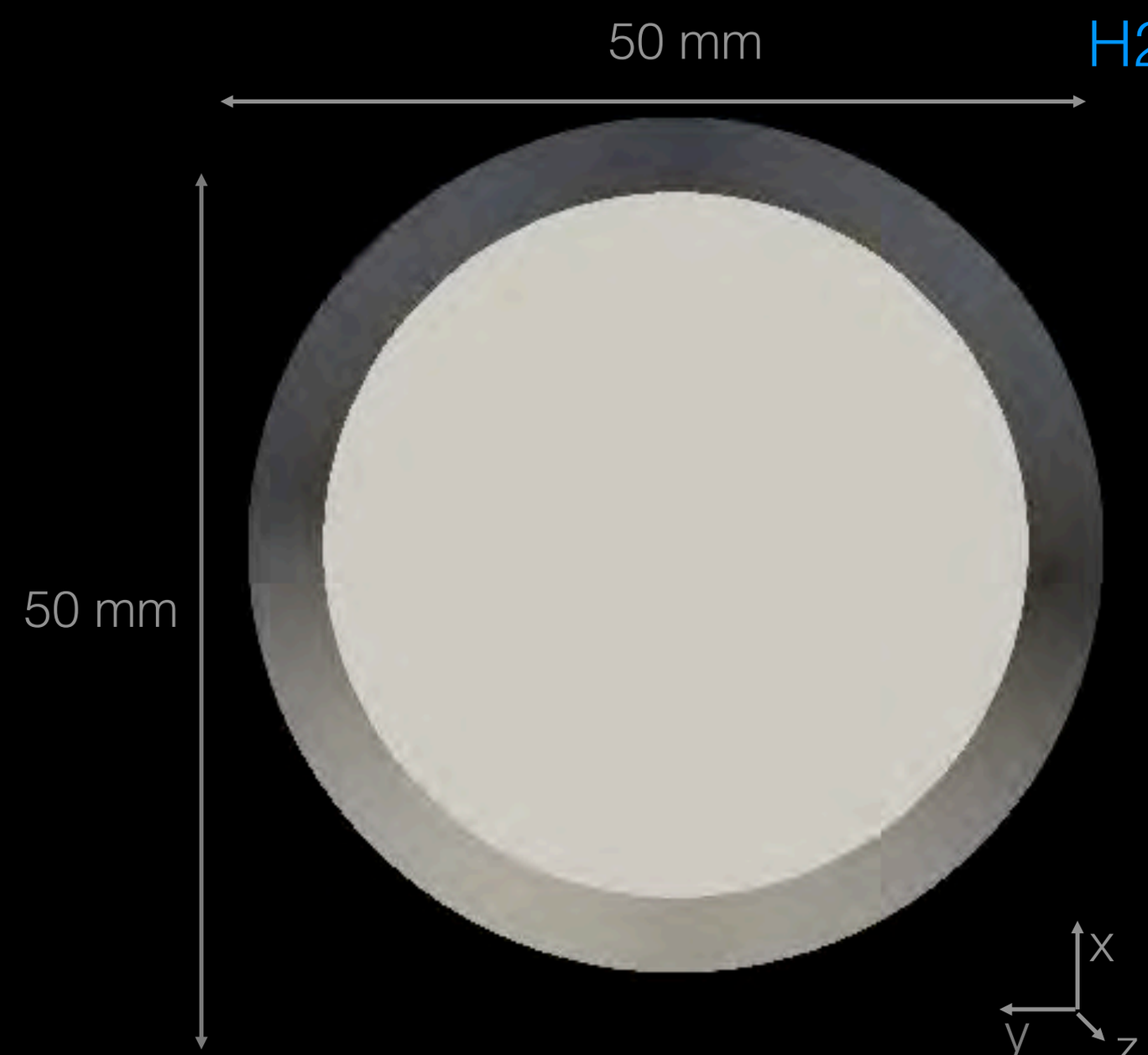


detail: slice views

B=0.1 T, V=-10 V

Protons
Electrons
H2+





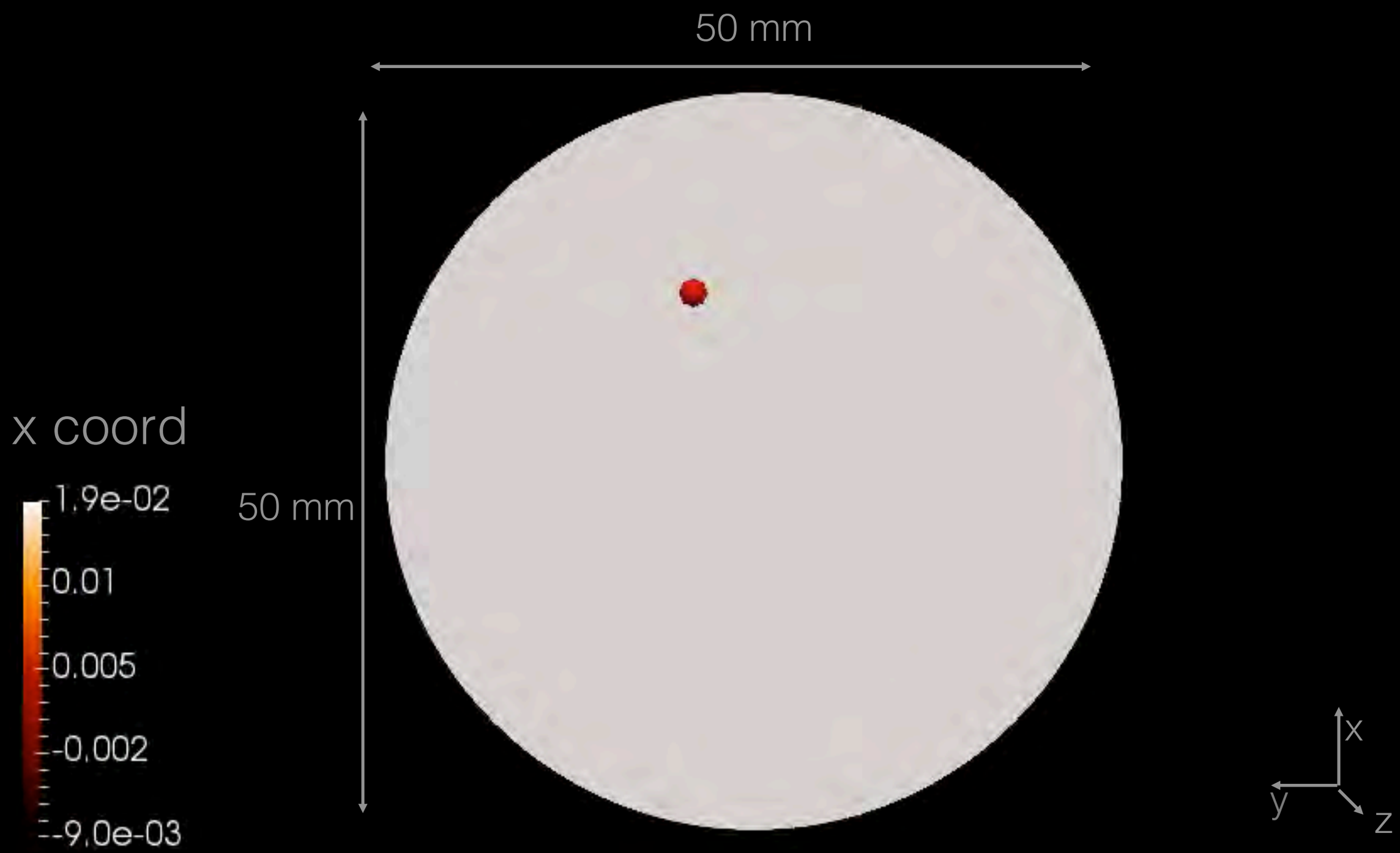
detail: slice views

python script

```
Threshold_min= 0.54 #where the slice begins when z=0
Threshold_max=0.58 #where the slice ends when z=1

# create a new slice
isoVolume1 = IsoVolume(Input=tableToPoints1)
isoVolume1.InputScalars = ['POINTS', 'Field 3']
isoVolume1.ThresholdRange = [Threshold_min,
Threshold_max]

# show data in view
isoVolume1Display = Show(isoVolume1, renderView1)
```

Detail: tracking an electron*

B=0.1 T, V=-10 V



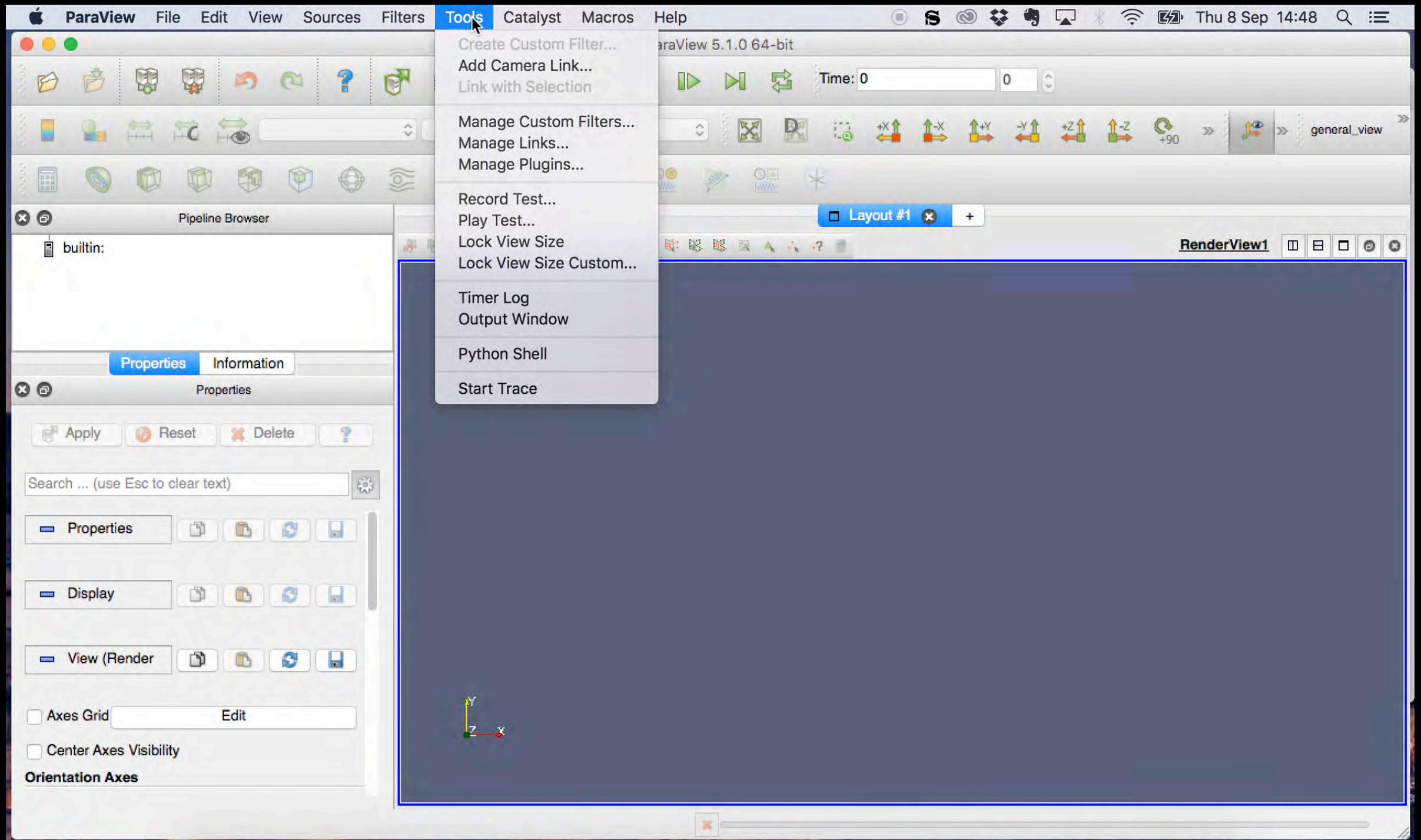
Detail: tracking an electron*

python script

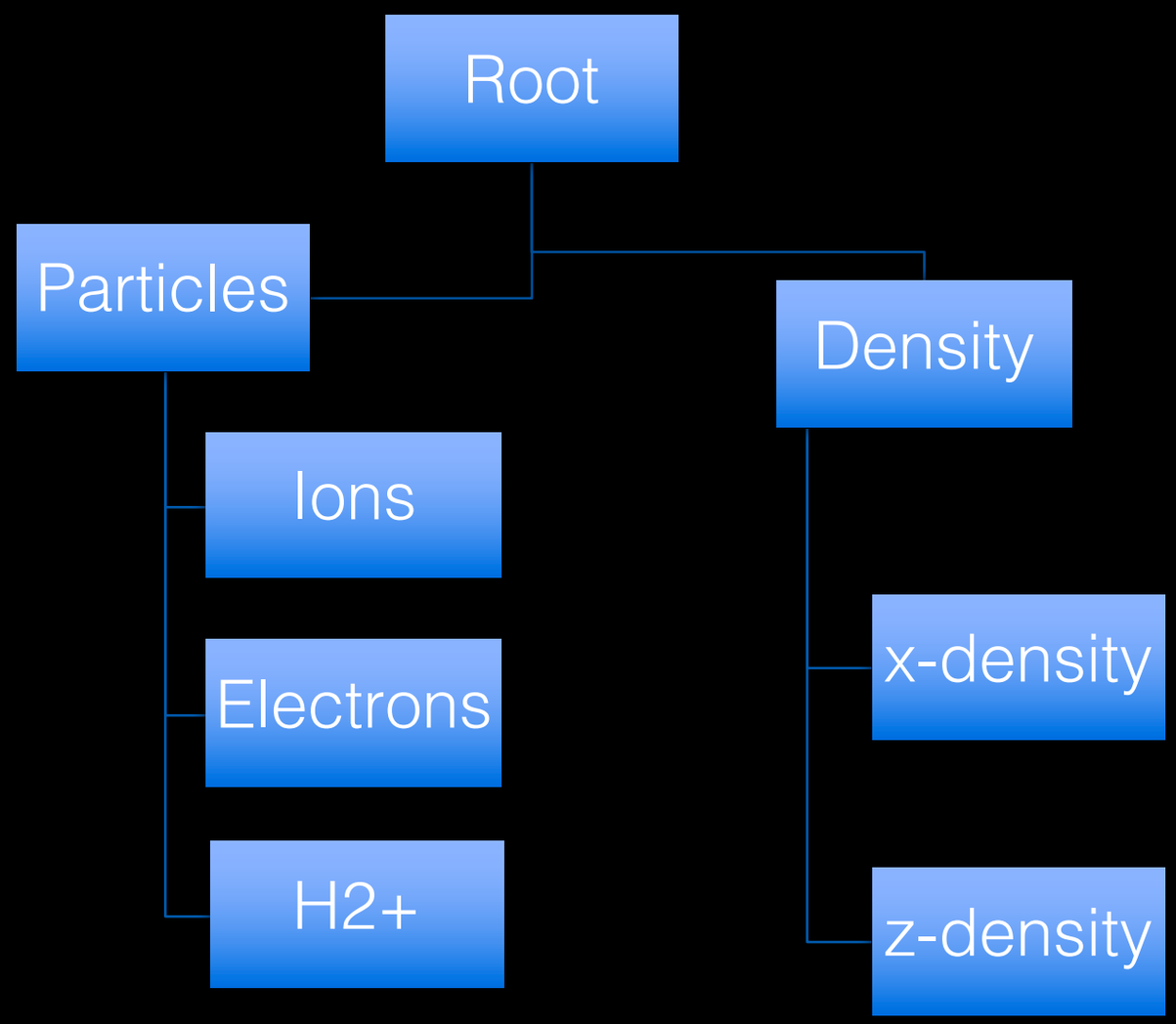
```
MASK_POINTS=10000
PATH_LENGTH= 1000
PARTICLE_RADIUS= 0.002

# create a new 'Temporal Particles To Pathlines'
temporalParticlesToPathlines1 =
TemporalParticlesToPathlines(Input=tableToPoints1,
    Selection=None)
temporalParticlesToPathlines1.MaskPoints = MASK_POINTS
temporalParticlesToPathlines1.MaxTrackLength = PATH_LENGTH
temporalParticlesToPathlines1.MaxStepDistance = [1.0, 1.0, 1.0]
temporalParticlesToPathlines1.IdChannelArray = 'Global or Local IDs’

temporalParticlesToPathlines1Display =
Show(temporalParticlesToPathlines1, renderView1)
```

A tutorial

integrating python and GUI



breaking bottlenecks… HDF5

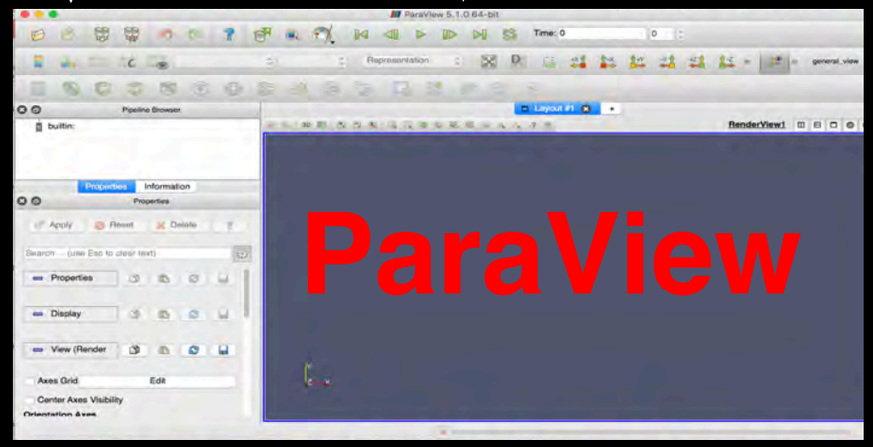


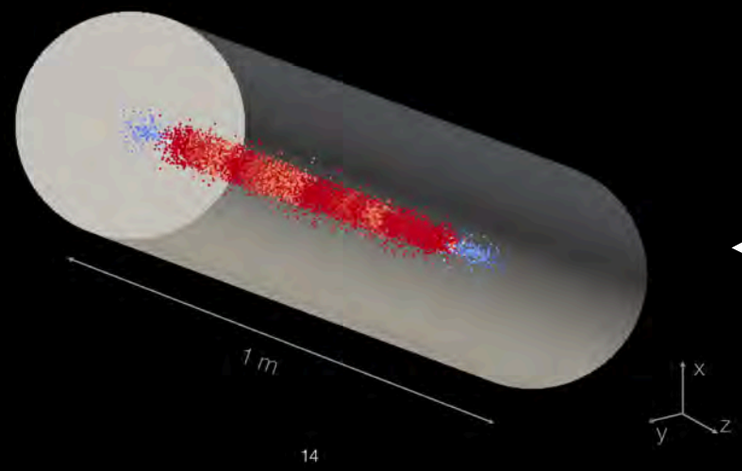
- Open source technology suite for high volume and/or complex data
- Guarantees flexible, efficient storage and I/O
- *Xml*-style hierarchical structure

The pipeline

WARP simulation

Python script





A simple implementation

```
#create an hdf5 file for each timestep
h5file= PWpyt.PW("timestep_%06d.h5" % iter)

#create an array of particles
ions_r = []

#for every ion, add its info to the previous array:
   ions_r.append([i,xions[i],yions[i],zions[i])

#convert the array into a numpy array
ions_np= np.array(ions_r)

#add the array of ions as an attribute of that file
h5file.protons= ions_np

h5file.close()
```

What comes next

- Practical issues such as ring dynamics of the primary proton beam and multiple passes thru the e-columm
- **2016** North American Particle Accelerator Conference (NAPAC16)

What I learnt

- Physics is really complex
- One of the best ways to understand what is going on is to see it
- Understanding is one thing, making it clear to others is a completely different business
- It takes a lot of work to make the complex look simple!

What I hope you'll take home

- Paraview is an extremely flexible and versatile, and it can be used for almost every type of data.
- The power of visualisation goes well beyond fancy graphics
- The strength of a simulation lies not only in the power and efficiency of the code, but also in the usability of its output
- The strength of any script lies in its re-usability

To Charles Thangaraj, Chong Shik Park and Giulio Stancari

To the **Fermilab** Accelerator Science and Technology **(FAST)** – Integrable Optics Test Accelerator **(IOTA)** department

To Giorgio Bellettini, Emanuela Barzi, Simone Donati and the CAIF associates

To my colleagues and roommates – great physics tutors

To Fermilab – a truly incredible opportunity

Thank you!