GEMC

a database-driven Monte Carlo simulation program

Homepage src libs

Maurizio Ungaro

Thomas Jefferson National Accelerator Facility

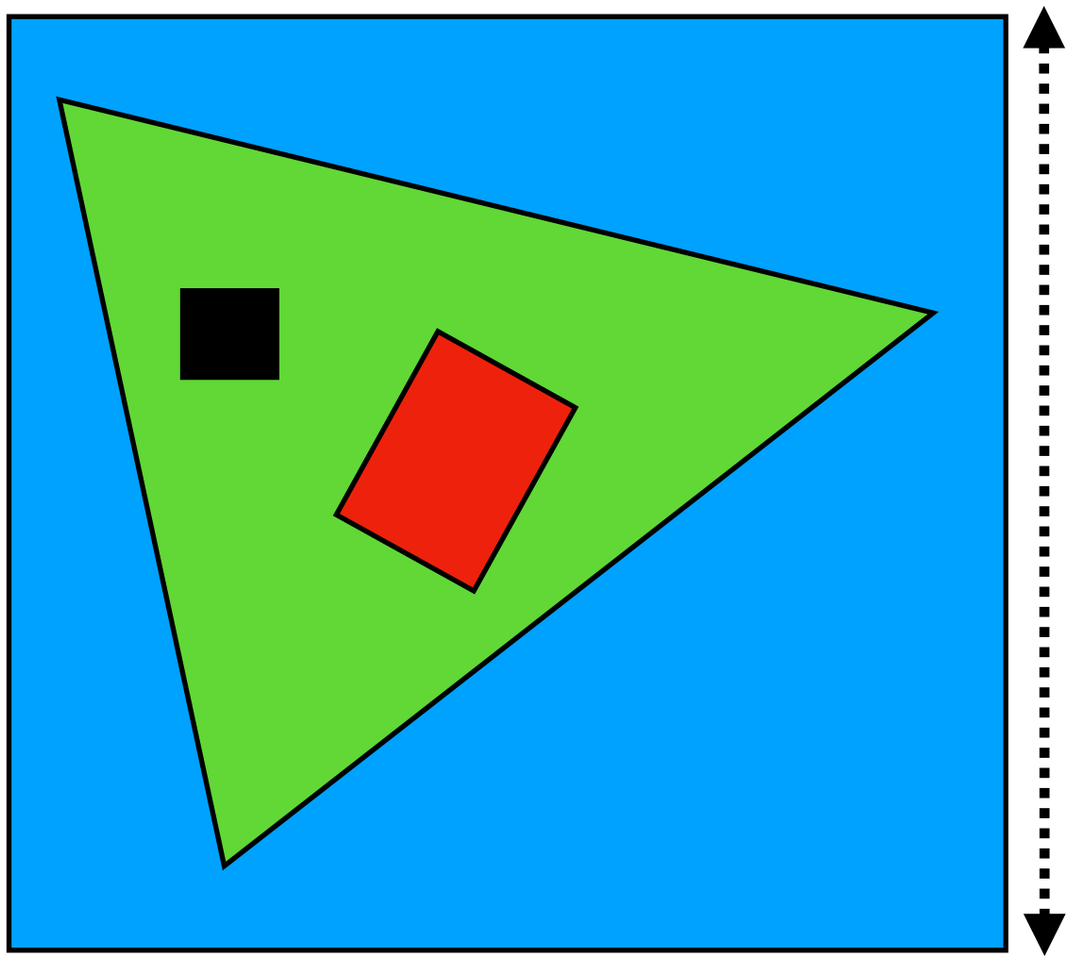
Jefferson Lab

U.S. DEPARTMENT OF ENERGY | Office of Science

JSA

Q1: No hardcoded numbers?

Can we build Geant4 application that reads parameters from a database?



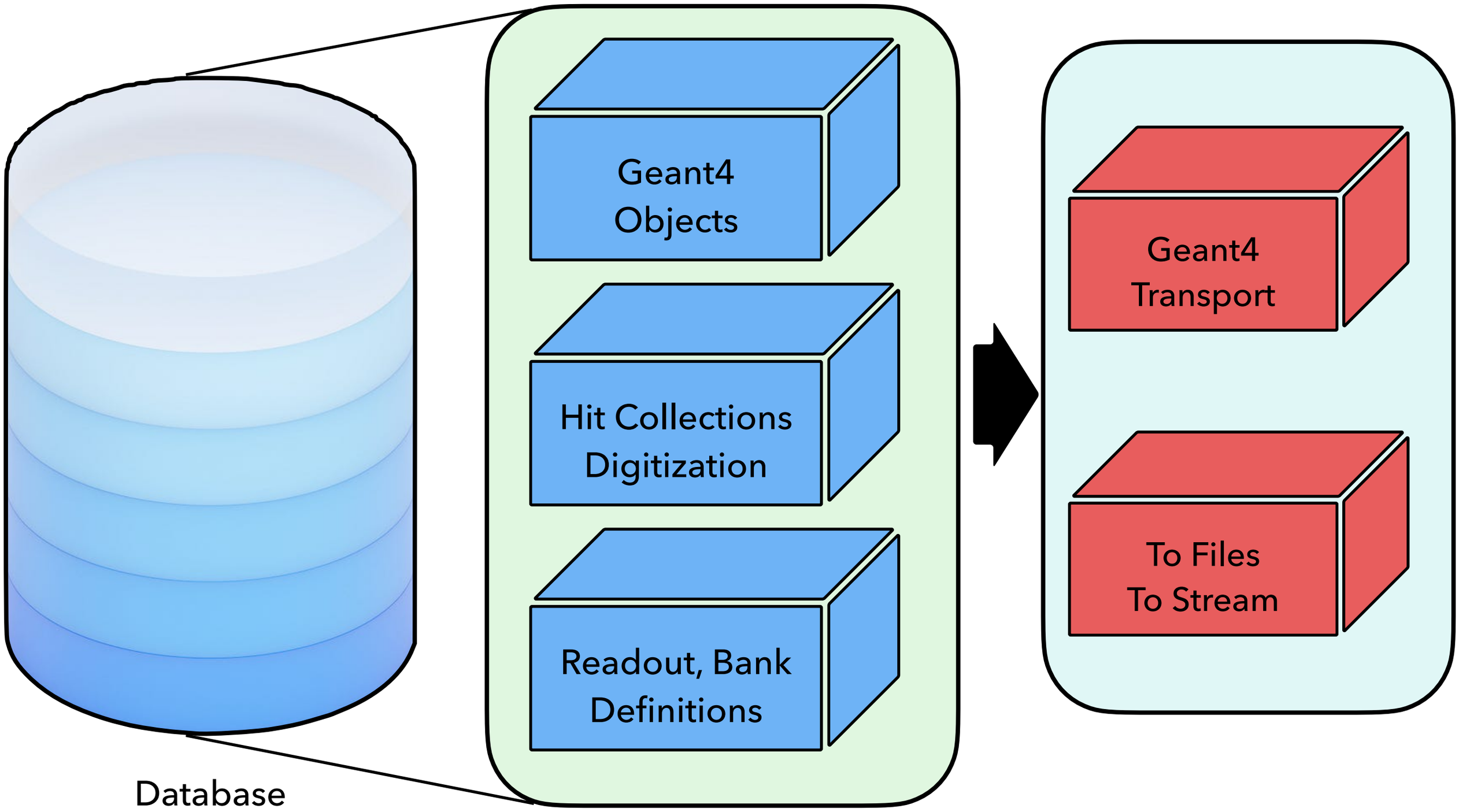

```
G4Box(‘box’, 20, 30, 40)
```

```
G4Box(name, a, b, c)
```

```
name, a, b, c = “select name, dimensions from table  geometry where….”
```

Q2: Database, not code

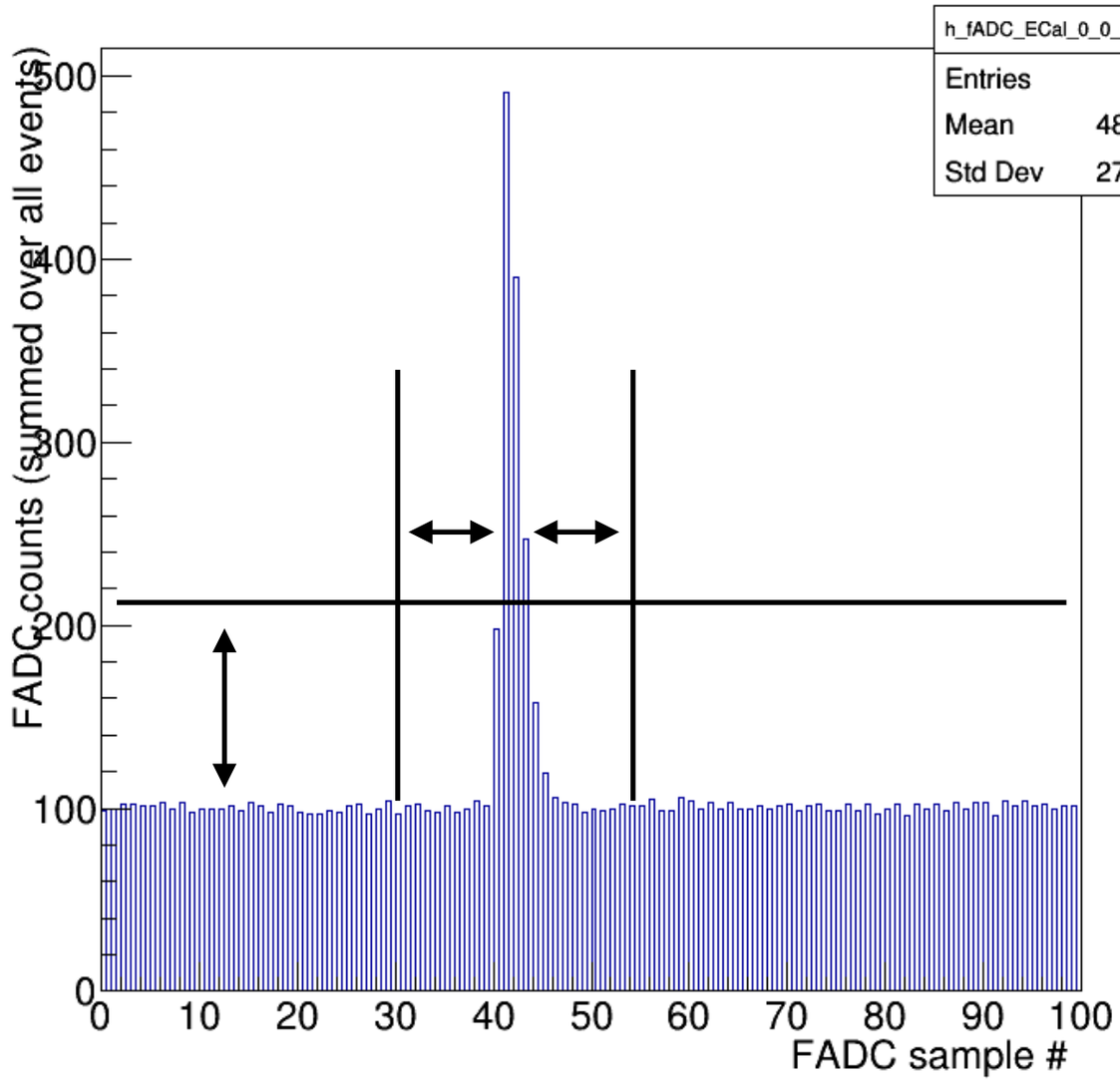
Can we define a geant4 simulation in its entirety from a database?



Q3: Let's be real

Can we make such DB simulation 'realistic' ?

1. Calibration / Digitization constants
2. Geometry <u>variations</u>
3. Emulation of Electronic Readout
4. Hardware status
5. Custom user digitization
6. Energy sharing / hits duplication



Ultimate goal: MC indistinguishable from data

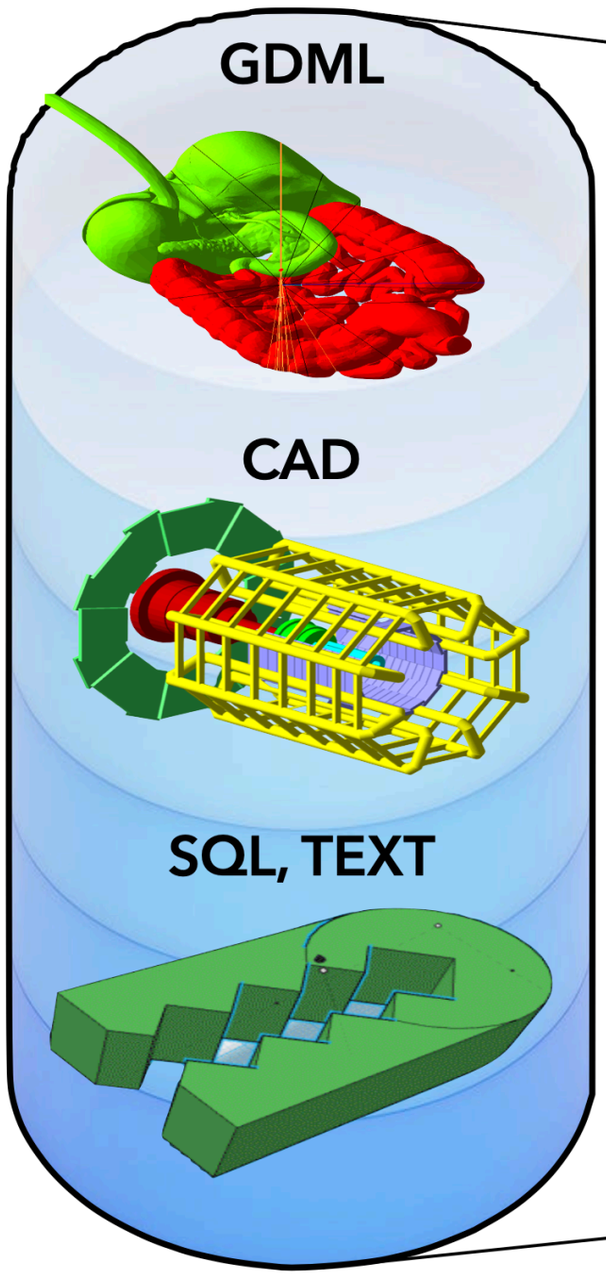
Q4: Turnkey MC simulations

Can we create and run complex setups w/o programming knowledge ?

1. Create DB entries, no c++ / geant4 coding
2. Intuitive, easy to use API
3. Allows user to focus on design and detector response
4. Turnkey executable provides out of the box:
   - MT handling
   - Variations
   - Pre-defined digitizations such as flux and dosimeter
   - Built-in text and ROOT output

GEMC: turnkey database-driven MC simulations program

Run options: tilts, displacements, calibration, inefficiencies



Geant4 Objects

Digitization Hits Collection

Readout, Bank Definitions

Geant4 Transport

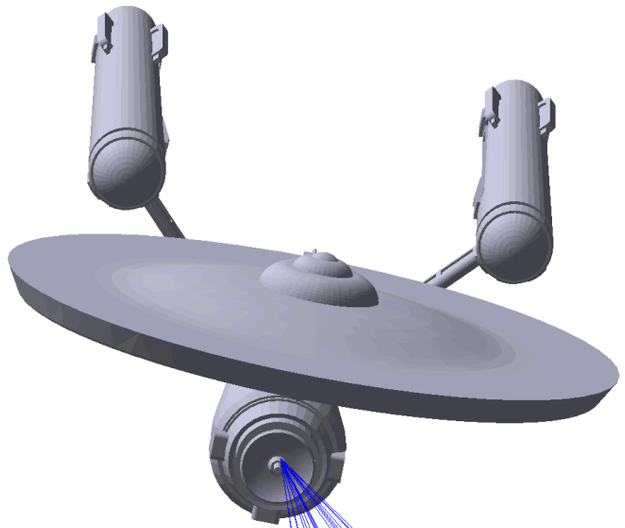
Files on Disk Data Streaming

Physics Cross Sections

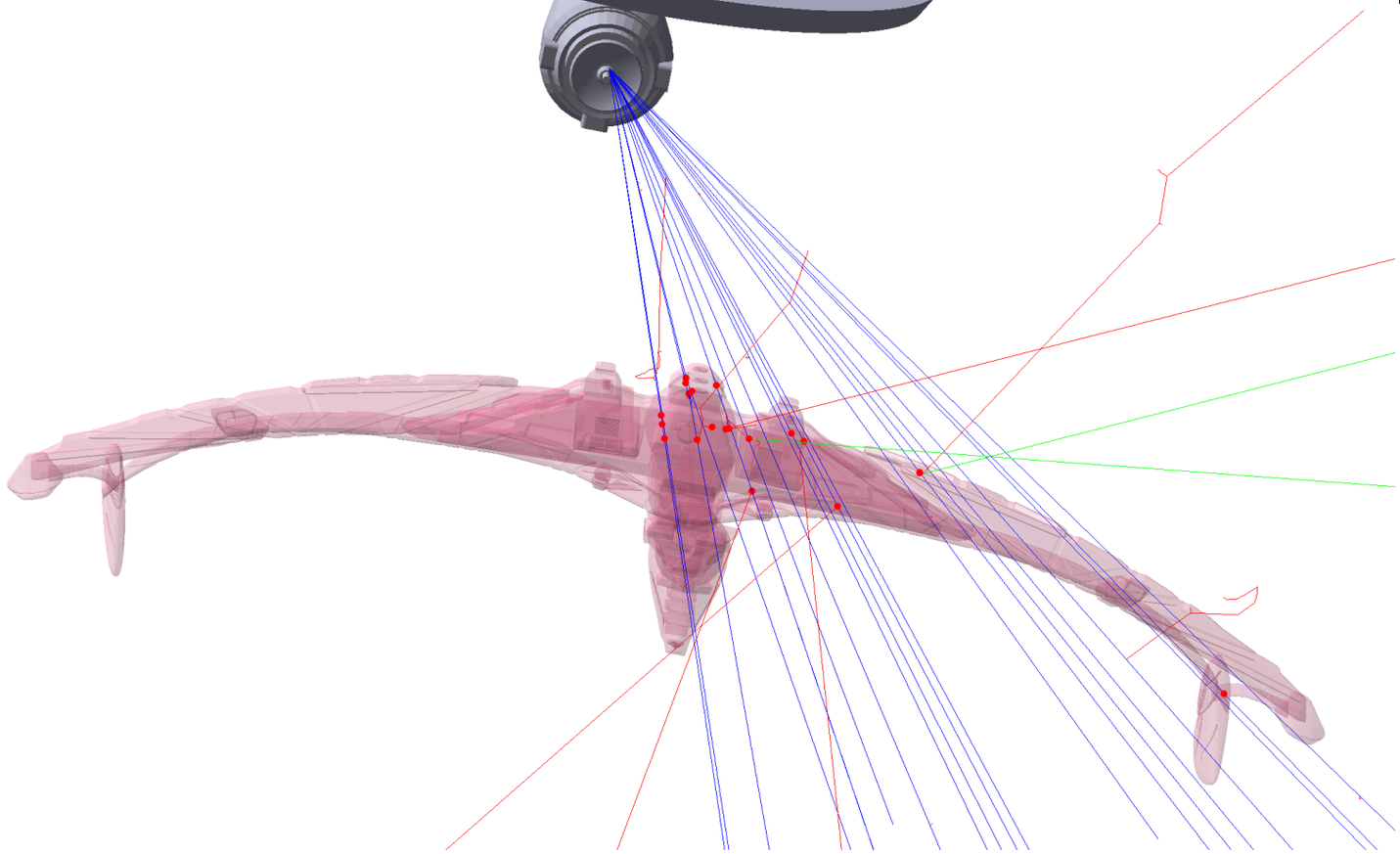
EM Fields

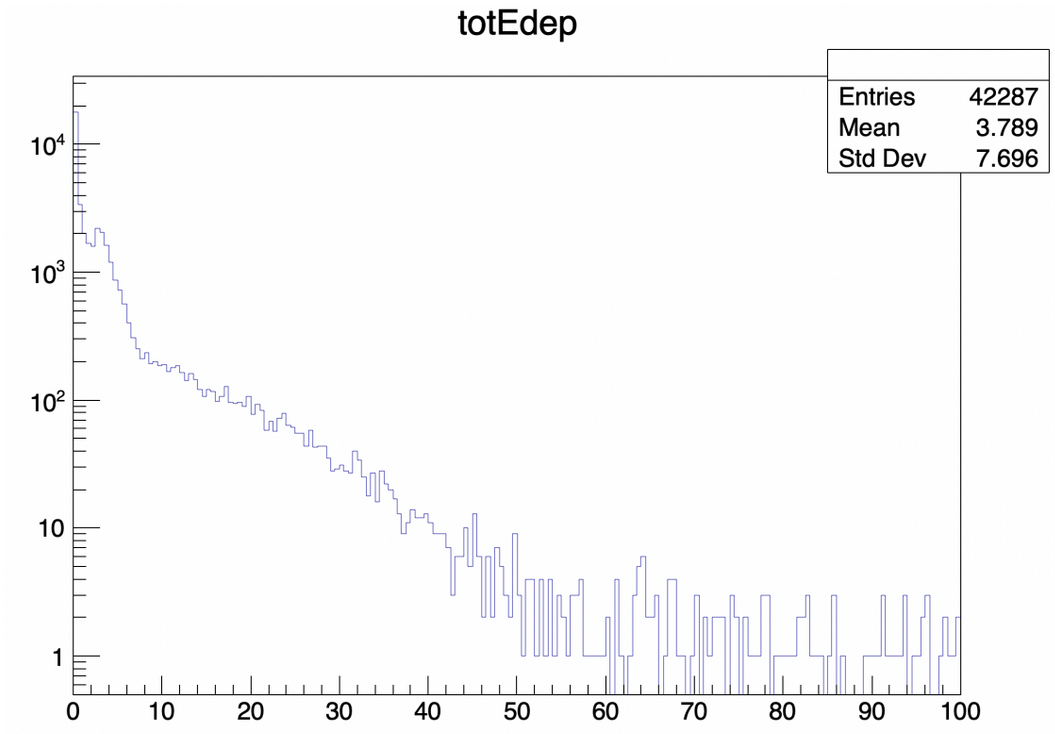
Digitization Plugins

CAD Example: edit 2 lines (JSON)



1. Grab STL files
2. Assign properties: edit json file
3. Run
4. ROOT, TEXT files with true information

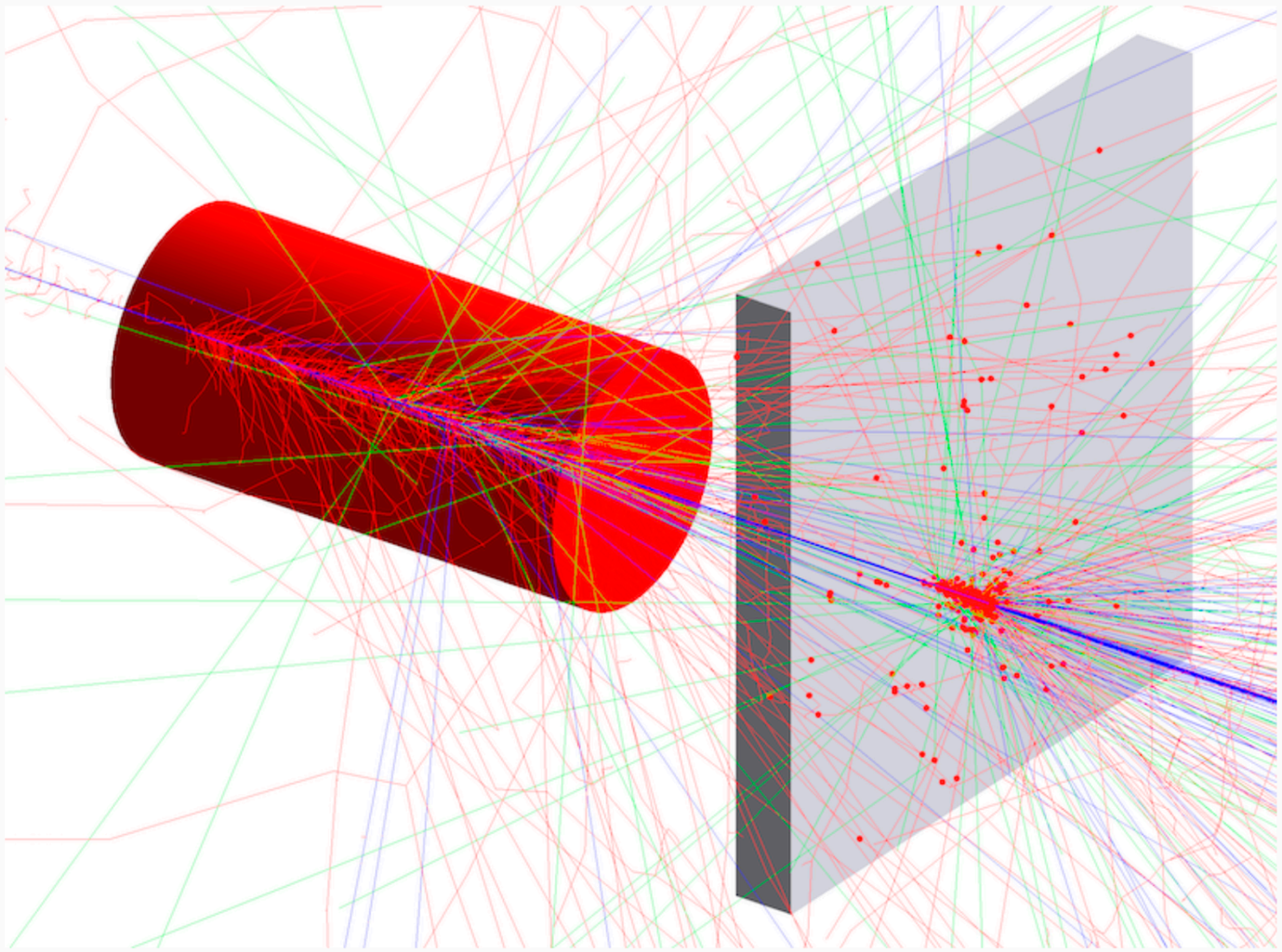





```
"romulans":    {"color": "ff99bb4", "digitization":  "flux"}
```

Python API

Python API



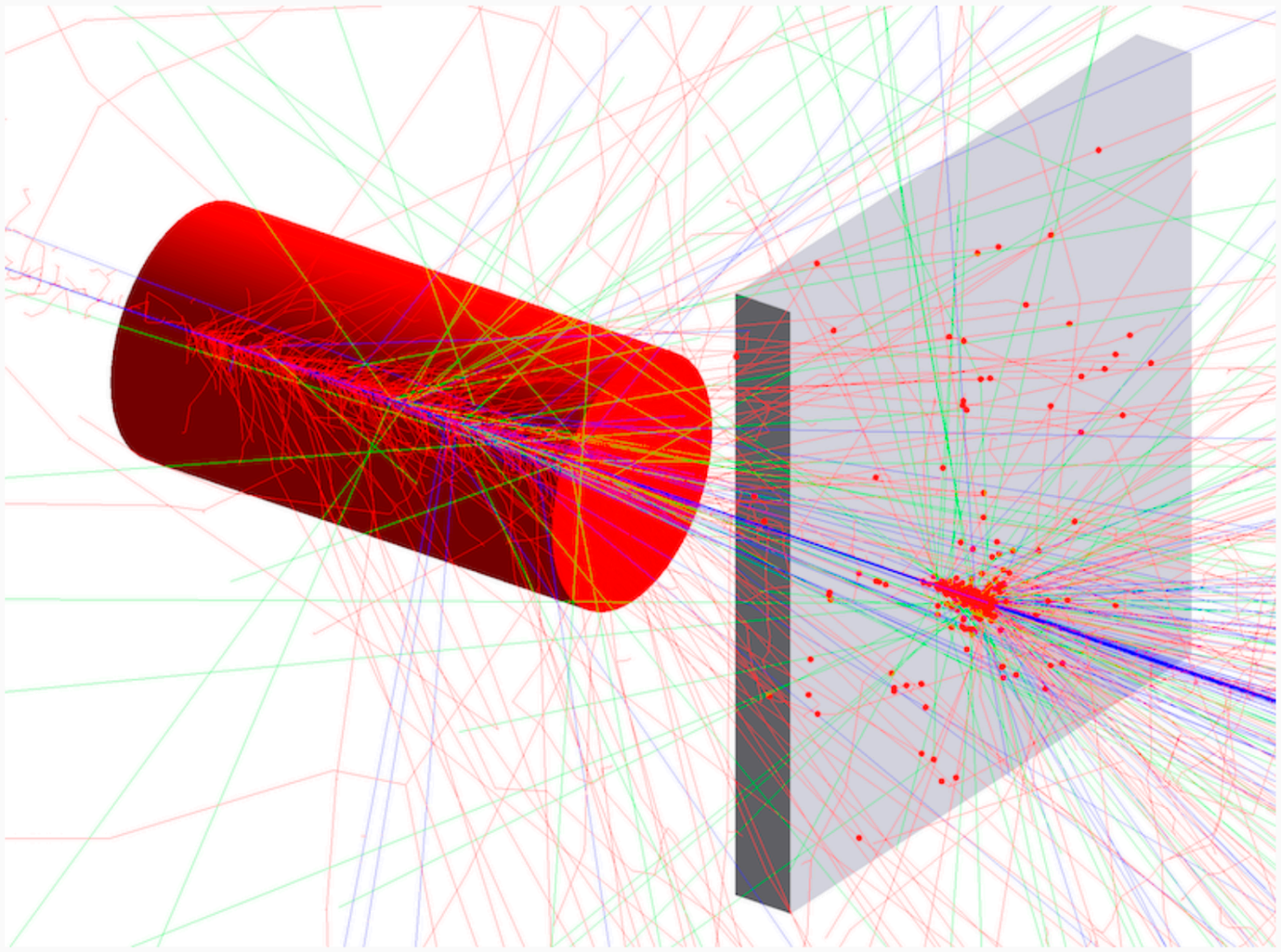
```
gvolume = GVolume('target')
gvolume.description = 'Liquid Hydrogen Target'
gvolume.make_tube(0, 20, 40, 0, 360)
gvolume.material    = 'G4_lH2'
gvolume.color       = 'ff0000'
gvolume.publish(configuration)

gvolume = GVolume('paddle')
gvolume.description = 'Scintillator paddle'
gvolume.make_box(5, 0.5, 5, 'cm')
gvolume.material = 'G4_PLASTIC_SC_VINYLTOLUENE'
gvolume.set_rotation(90, 0, 0)
gvolume.set_position(0, 2, 10, 'cm')
gvolume.color       = 'f4f4ff'
gvolume.digitization = 'flux'
gvolume.set_identifier('paddleid', 5)
gvolume.publish(configuration)
```

Geant4 volumes are built using the sci-g python API. An example geometry: a flux scintillator paddle collects hits from protons impinging on a liquid hydrogen target

The above snippet is the only code needed to build the geometry and record all tracks hitting the paddle.

Python API

Python API



```
gvolume = GVolume('target')
gvolume.description = 'Liquid Hydrogen Target'
gvolume.make_tube(0, 20, 40, 0, 360)
gvolume.material    = 'G4_lH2'
gvolume.color       = 'ff0000'
gvolume.publish(configuration)
```

```
gvolume = GVolume('paddle')
gvolume.description = 'Scintillator paddle'
gvolume.make_box(5, 0.5, 5, 'cm')
gvolume.material = 'G4_PLASTIC_SC_VINYLTOLUENE'
gvolume.set_rotation(90, 0, 0)
gvolume.set_position(0, 2, 10, 'cm')
gvolume.color       = 'f4f4ff'
gvolume.digitization = 'flux'
gvolume.set_identifier('paddleid', 5)
gvolume.publish(configuration)
```

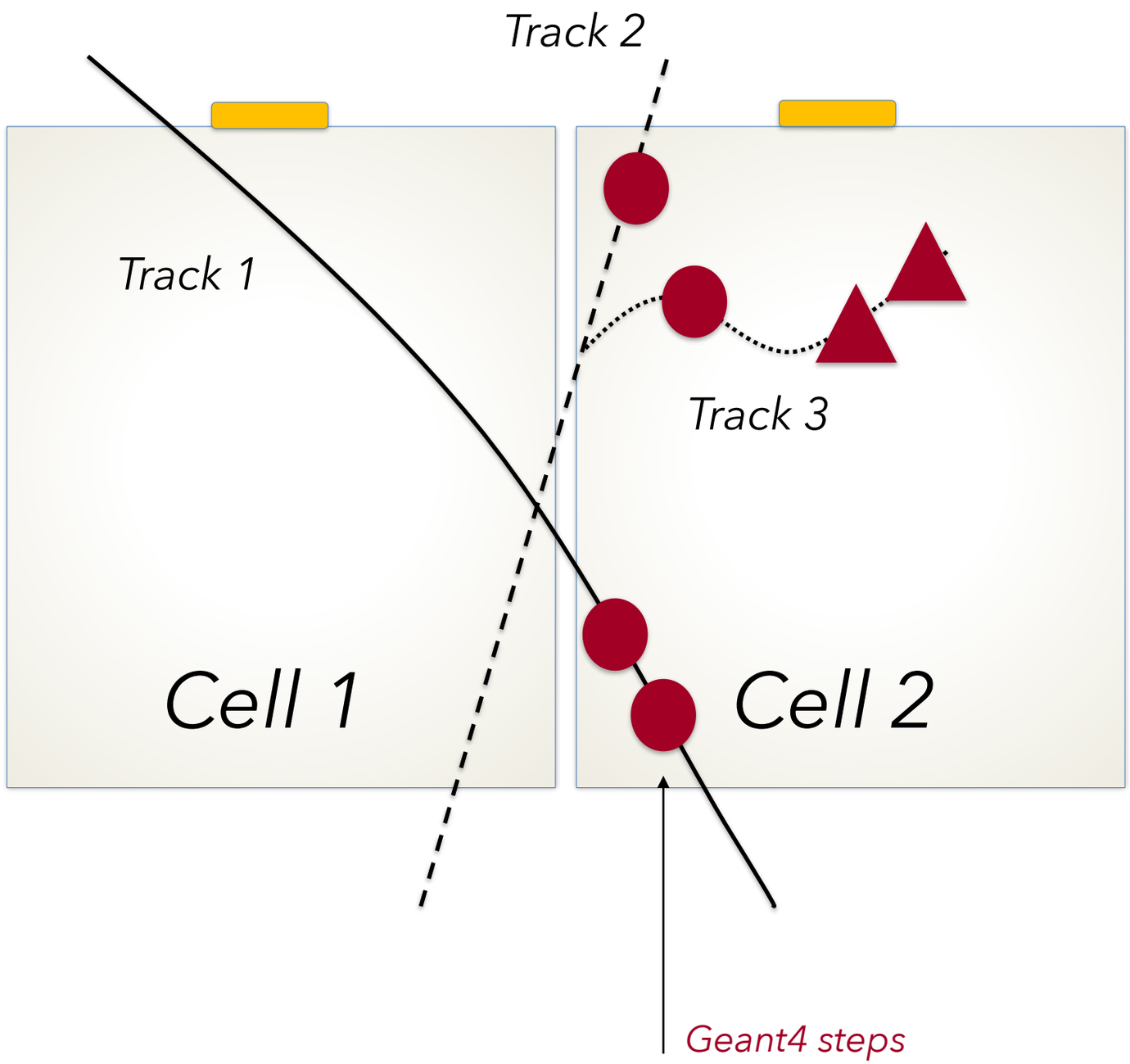
Geant4 volumes are built using the sci-g python API. An example geometry: a flux scintillator paddle collects hits from protons impinging on a liquid hydrogen target

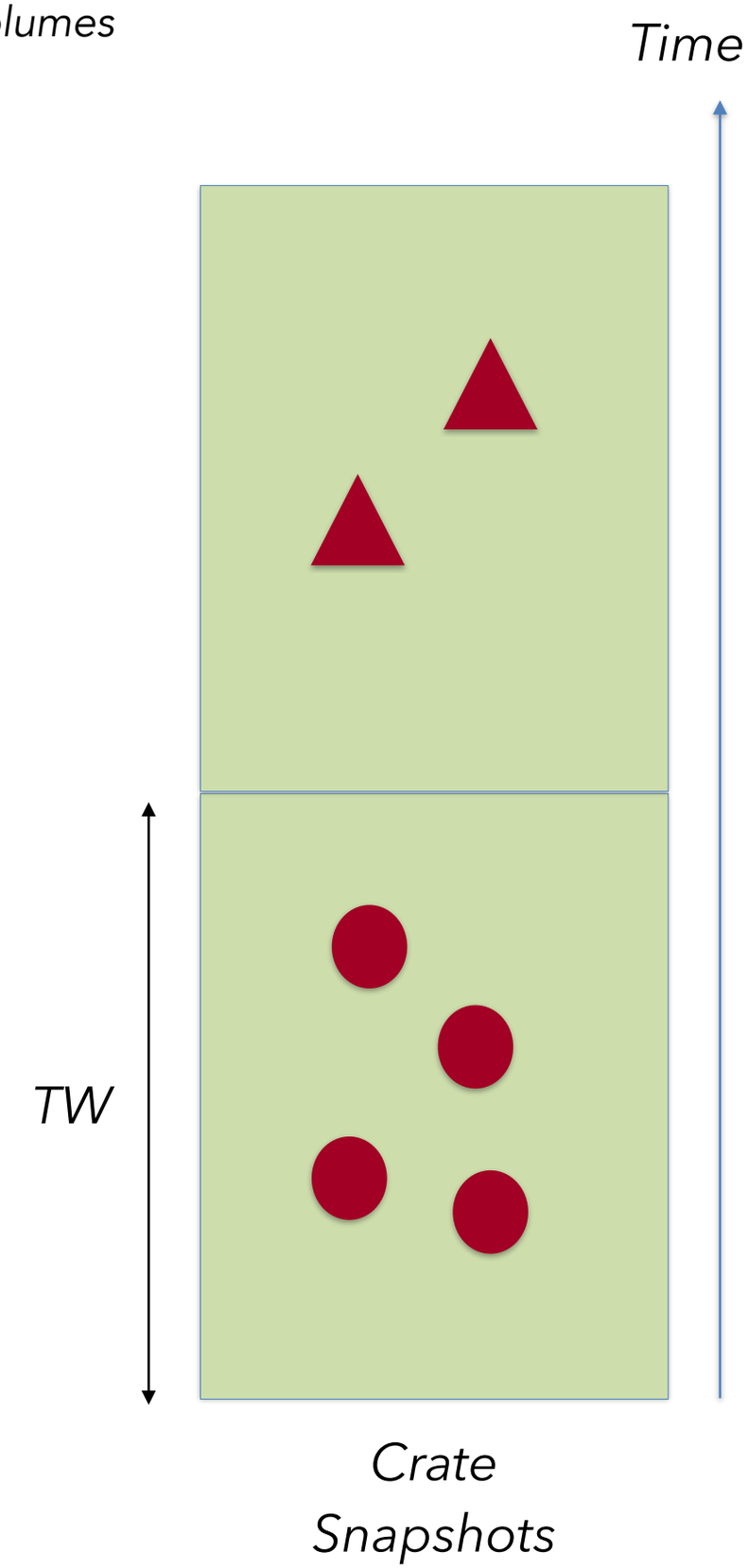
The above snippet is the only code needed to build the geometry and record all tracks hitting the paddle.

```
scigTemplate.py -gv G4Box
```

Electronic Time Window

Mechanism provided by GEMC to all sensitive volumes

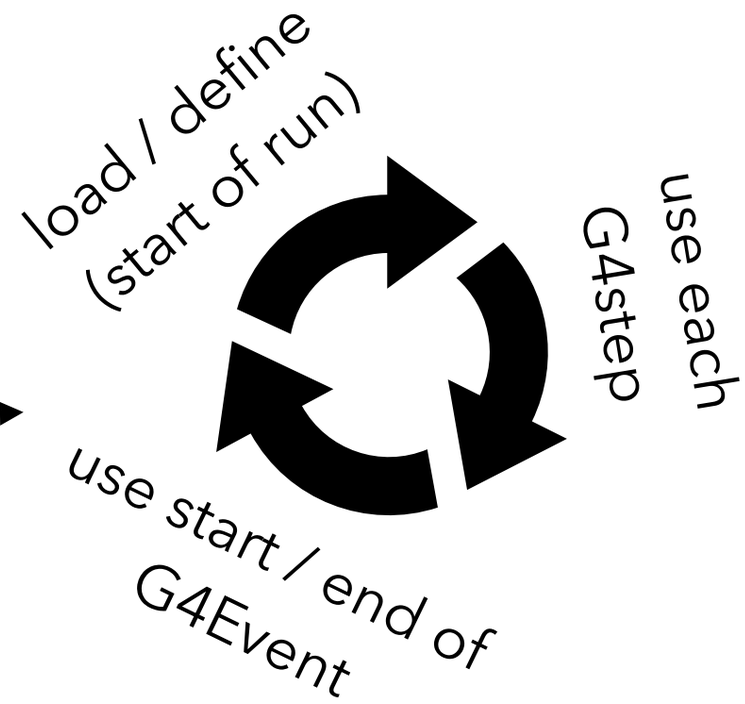




Digitization

c++ code

- external plugin - loaded on demand
- formalized access to g4step information
- formalized workflow

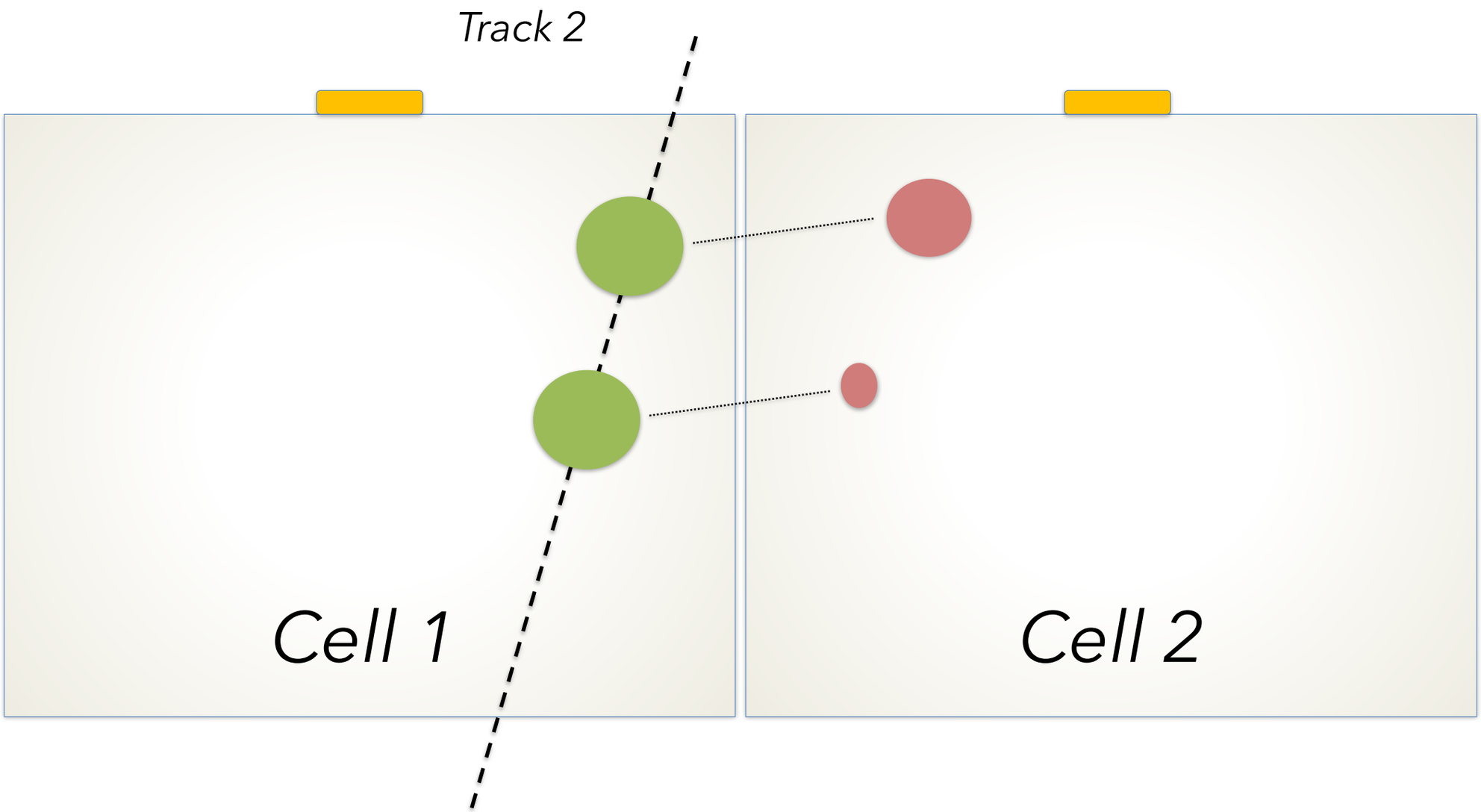


<u>Hooks to:</u>

- Define readout electronics (time window)
- Define Energy Sharing / Hit Proliferation mechanism
- Calibration / Digitization constants
- Load Translation Table
- Digitized Hit
- Define Streaming Readout
- Define output bank (ADC, TDC, FADC, SRO payload

Energy Sharing

Digitization Hook



True geant4 step

Generated step

GEMC Data Streamers

Concurrent Files or Streaming Readout

Event Data Collection

true / digitized data

indexed by event

Formats:

- TEXT
- ROOT
- User defined (plugin)

Frame Data Collection

collections of event data, can refer to multiple events

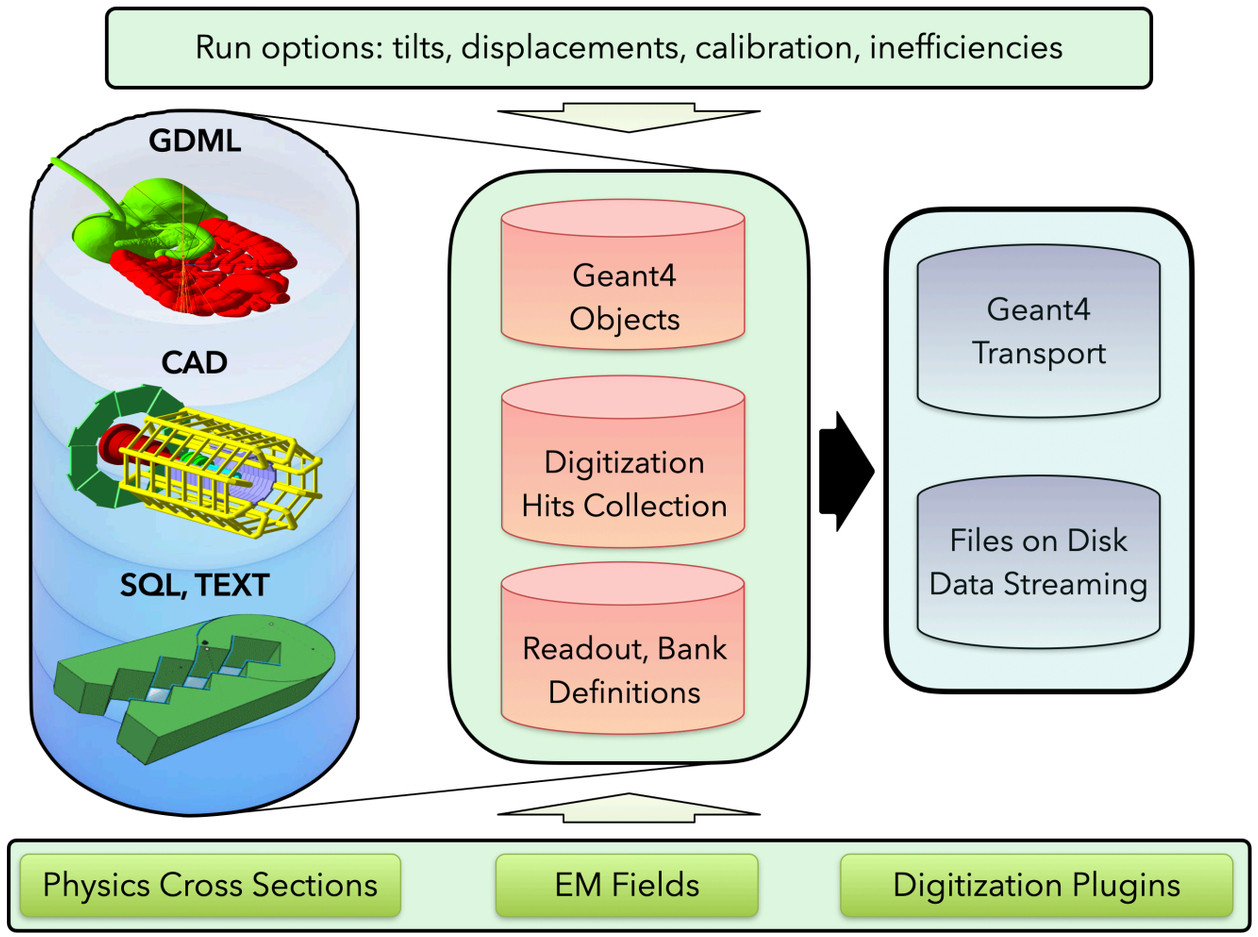
Frame Header

Payload

Formats:

- VTP Binary
- User defined (plugin)

Summary



- GEMC: turnkey database-driven MC simulations program
- Full geant4 capabilities
- Realistic output
- Easy Intuitive interface

CHEP2023: consensus to support R&D for software across project or discipline boundaries

Please keep in mind this project: database driven - experiment independent

Homepage src libs