GEMC

a database-driven Monte Carlo simulation program

Maurizio Ungaro

Thomas Jefferson National Accelerator Facility
Can we build Geant4 application that reads parameters from a database?

name, a, b, c = "select name, dimensions from table geometry where...."

G4Box('box', 20, 30, 40)

G4Box(name, a, b, c)
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 variations
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

GEMC Components:
- GDML
- CAD
- SQL, TEXT

Processing Flow:
1. GDML
2. CAD
3. SQL, TEXT
4. Geant4 Objects
5. Digitization Hits Collection
6. Readout, Bank Definitions
7. Geant4 Transport
8. 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"}
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.
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
```
Mechanism provided by GEML to all sensitive volumes
Digitization

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

Hooks to:

- 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

Cell 1

True geant4 step

Generated step

Cell 2

Track 2
GEMC Data Streamers

Event Data Collection
true / digitized data
indexed by event

Frame Data Collection
collections of event data, can refer to multiple events
Frame Header
Payload

Formats:
• TEXT
• ROOT
• User defined (plugin)

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