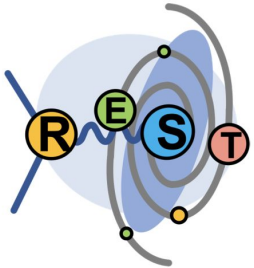


REST-for-Physics Framework

2.2 Geant4 simulations with restG4

25/01/2022 - Luis Obis - lobis@unizar.es

Centro de Astropartículas y Física de Altas Energías
Universidad Zaragoza

- Geant4 is a toolkit for the simulation of radiation transport in matter (more info)
- REST interfaces with Geant4 to configure simulation settings and store data in a compatible format (root files)

- The Geant4Lib library handles all metadata related to simulation: physics list, generator, detectors… It does not link to Geant4! It also provides the data containers for the simulation results: **You do not need to have Geant4 available to analyse simulation results!**
- The restG4 package interfaces with geant4 to run the simulation configured with Geant4Lib

- Geant4 is a framework which requires the user to write their own simulation code linking to Geant4 libraries. It can be difficult for beginners.
- Geant4 also does not provide a standard way to save a full event, only computed observables such as dose/energy
- REST allows configuration of simulation via a single rml file
- The full simulation (all events / tracks / hits) are stored in the REST output file by default, which allows better understanding of the physics

- Type `restG4` in the terminal to see available options

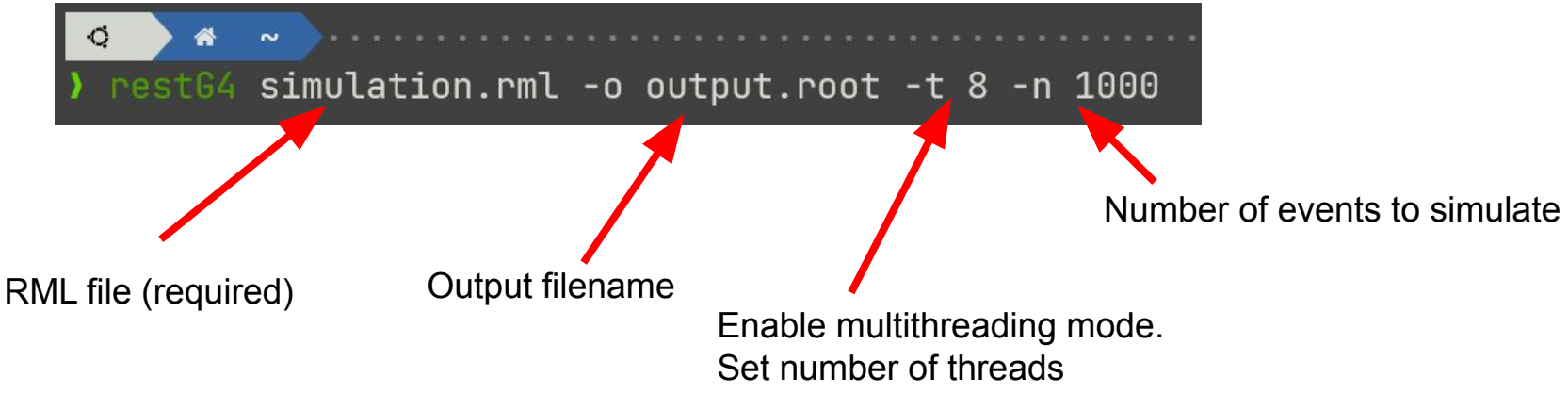
```
❯ restG4

                        888     .d8888b.      d8888
                        888    d88P  Y88b    d8P888
                        888    888    888   d8P 888
888d888 .d88b.  .d8888b  888888 888         d8P  888
888P"  d8P  Y8b 88K      888    888  88888 d88   888
888    88888888 "Y8888b. 888    888    888 8888888888
888    Y8b.          X88 Y88b.  Y88b  d88P       888
888     "Y8888   88888P'  "Y888  "Y8888P88       888

🌏 Source code: https://github.com/rest-for-physics/restG4
🐞 Bug reports: https://github.com/rest-for-physics/restG4/issues

Usage example: restG4 example.rml
there are other convenient optional parameters that override the ones in the rml file:
        --help (-h) | show usage (this text)
        --config (-c) example.rml | specify RML file (same as calling restG4 example.rml)
        --output (-o) output.root | specify output file
        --events (-n) nEvents | specify number of events to be processed (overrides nEvents on rml file)
        --entries (-e) | specify the requested number of entries. The simulation will stop after reaching this number of saved events. Final number may be larger
        --time timeLimit | Sets time limit for the simulation in the format '1h20m30s', '5m20s', '30s', ... If the time limit is reached before simulation ends, i
t will end the simulation and save to disk all events
        --geometry (-g) geometry.gdml | specify geometry file
        --seed (-s) seed | specify random seed (positive integer)
        --interactive (-i) | set interactive mode (disabled by default)
        --threads (-t, -j) | set the number of threads, also enables multithreading which is disabled by default
```

- `restG4` only needs an RML filename as argument
- All other settings can be configured either in the RML or via command line arguments (`restG4 –help` to see all options). Command line arguments take precedence over RML!



- This RML will be used in the examples session afterwards

TRestRun configuration: common to REST rml files
Sets output file and other parameters

```
<TRestRun name="TestRun" title="A basic muon test with 2 active volumes">
    <parameter name="experimentName" value="TestScan"/>
    <parameter name="readOnly" value="false"/>
    <parameter name="runNumber" value="auto"/>
    <parameter name="runDescription" value=""/>
    <parameter name="user" value="${USER}"/>
    <parameter name="verboseLevel" value="1"/>
    <parameter name="overwrite" value="off"/>
    <parameter name="outputFileName" value="Run[fRunNumber]_[fRunType]_[fRunTag]_[fRunUser].root"/>
</TRestRun>
```

Simulation RML files

```
<TRestGeant4Metadata name="restG4 run" title="CosmicMuonTest">

    <parameter name="gdmlFile" value="box.gdml"/>
    <parameter name="subEventTimeDelay" value="100us"/>

    <parameter name="nRequestedEntries" value="1000"/>
    <parameter name="seed" value="137"/>

    <parameter name="saveAllEvents" value="false"/>
    <parameter name="printProgress" value="true"/>

   <generator type="surface" shape="wall" position="(0,10,0)cm" size="(30,30,0)cm" rotationAngle="3.1416/2"
              rotationAxis="(1,0,0)">
        <source particle="mu-" fullChain="on">
            <energy type="TH1D" file="Muons.root" name="cosmicmuon"/>
            <angular type="formula" name="Cos2" direction="(0,-1,0)"/>
        </source>
    </generator>
    <detector>
        <parameter name="activateAllVolumes" value="false"/>

        <volume name="gas" sensitive="true" maxStepSize="1mm"/>
        <volume name="vessel" maxStepSize="1mm"/>
    </detector>

</TRestGeant4Metadata>
```

TRestGeant4Metadata:
Most important section.
Configures most parameters of simulation such as:

- Geometry
- Entries
- Primary generation
- Sensitive volume configuration

TRestGeant4PhysicsLists

- Configuration related to Geant4 physics

```
<TRestGeant4PhysicsLists name="default" title="First physics list implementation." verboseLevel="warning">
    <parameter name="cutForGamma" value="1" units="um"/>
    <parameter name="cutForElectron" value="1" units="um"/>
    <parameter name="cutForPositron" value="1" units="um"/>

    <parameter name="cutForMuon" value="1" units="mm"/>
    <parameter name="cutForNeutron" value="1" units="mm"/>
    <parameter name="minEnergyRangeProductionCuts" value="1" units="keV"/>
    <parameter name="maxEnergyRangeProductionCuts" value="1" units="GeV"/>

    <!-- EM Physics lists -->
    <physicsList name="G4EmLivermorePhysics"/>
    <!-- <physicsList name="G4EmPenelopePhysics"> </physicsList> -->
    <!-- <physicsList name="G4EmStandardPhysics_option3"> </physicsList> -->

    <!-- Decay physics lists -->
    <physicsList name="G4DecayPhysics"/>
    <physicsList name="G4RadioactiveDecayPhysics"/>
    <physicsList name="G4RadioactiveDecay">
        <option name="ICM" value="true"/>
        <option name="ARM" value="true"/>
    </physicsList>

    <!-- Hadron physics lists -->
    <physicsList name="G4HadronElasticPhysicsHP"/>
    <physicsList name="G4IonBinaryCascadePhysics"/>
    <physicsList name="G4HadronPhysicsQGSP_BIC_HP"/>
    <physicsList name="G4EmExtraPhysics"/>

</TRestGeant4PhysicsLists>
```

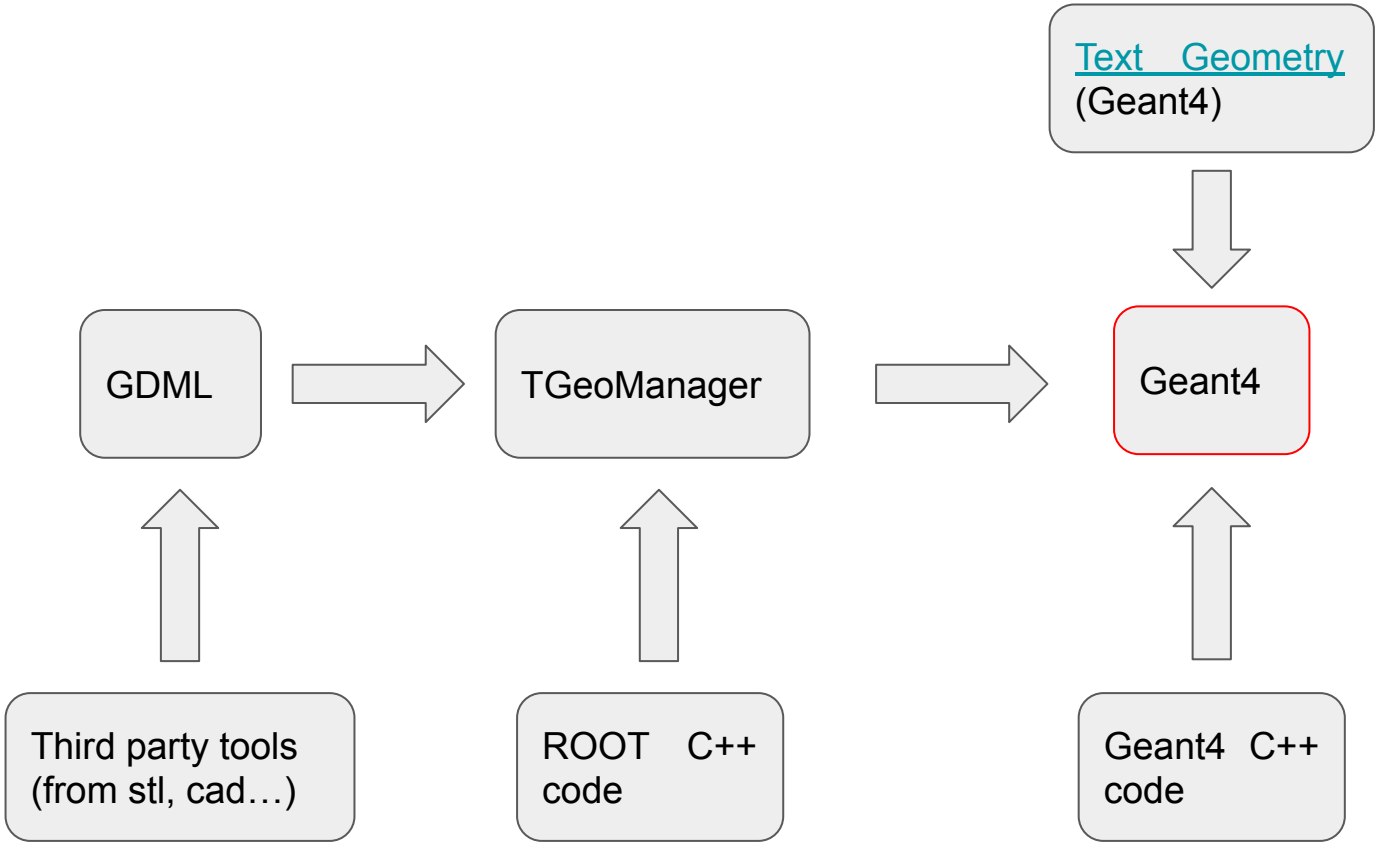
*The Geometry Description Markup Language (GDML) is a **specialized XML-based language** designed as an application-independent persistent format for describing the geometries of detectors (or parts of them) for physics experiments.* [GDML documentation]

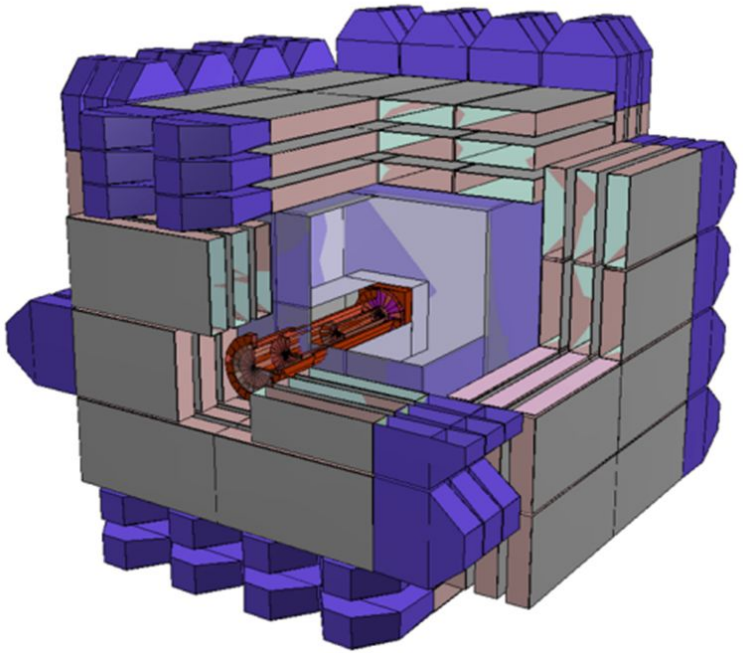
It is integrated in Geant4 and in Root

GDML schema:

- ❑ Definitions
- ❑ Materials
- ❑ Structure (geometry)
- ❑ Setup: top volume of the geometry tree

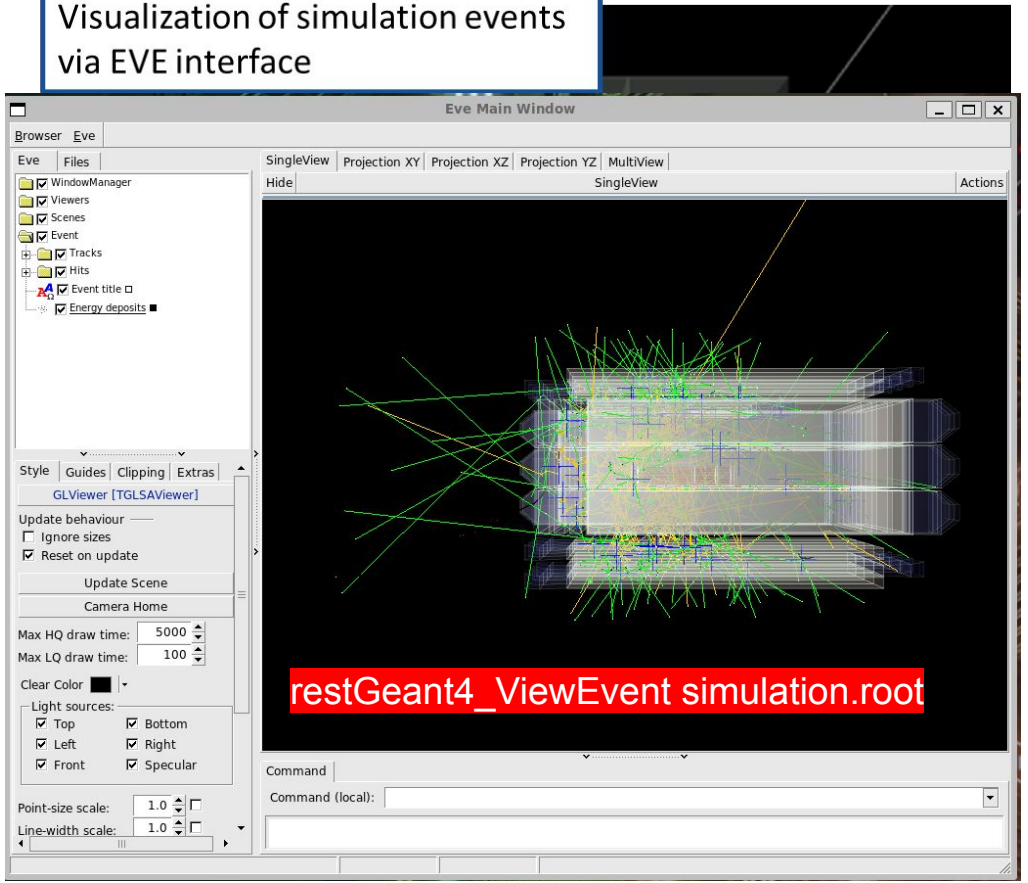
```
<?xml version="1.0" encoding="UTF-8"?>
<gdml xsi:noNamespaceSchemaLocation="schema/gdml.xsd">
        <define>
                …
                <position name="TrackerinWorldpos" unit="mm" x="0" y="0" z="100"/>
        </define>
        <materials>
                …
                <element name="Nitrogen" formula="N"  Z="7.">
                <atom value="14.01"/>
                </element>
                <material formula=" " name="Air" >
                        <D value="1.290" unit="mg/cm3"/>
                        <fraction n="0.7" ref="Nitrogen" />
                        <fraction n="0.3" ref="Oxygen" />
                </material>
        </materials>
        <solids>
                …
                <box lunit="mm" name="Tracker" x="50" y="50" z="50"/>
        </solids>
        <structure>
                …
                <volume name="World" >
                        <materialref ref="Air" />
                        <solidref ref="world" />
                        <physvol>
                        <volumeref ref="Tracker" />
                        <positionref ref="TrackerinWorldpos"/>
                        <rotationref ref="TrackerinWorldrot"/>
                        </physvol>
                </volume>
        </structure>
        <setup name="Default" version="1.0" >
                <world ref="World" />
        </setup>
</gdml>
```


Text Geometry
(Geant4)
GDML
TGeoManager
Geant4
Third party tools
(from stl, cad…)
ROOT C++
code
Geant4 C++
code



TGeoManager geometry stored in simulation file

Visualization of simulation events via EVE interface



- **Many Geant4 examples can be found here. Best way to learn is to play with these!**
- Let's look at example 04 rml generator
- Launch muons from a circular plane. Energy and angular dist defined in histograms present in REST data directory

```
<generator type="surface" shape="circle"
           position="(0,100,0)mm" size="(400,0,0)mm"
           rotationAngle="90deg" rotationAxis="(1,0,0)">
    <source particle="mu-" fullChain="on">
        <energy type="TH1D" file="Muons.root" name="cosmicmuon"/>
        <angular type="formula" name="Cos2" direction="(0,-1,0)"/>
    </source>
</generator>
```

A complete description of all available generators can be found here

- The detector section controls which and how data is stored
- Only events depositing energy in sensitive volumes are recorded
- Active volumes will also be recorded (if the event itself is valid i.e. energy in sensitive volume)
- Different settings such as step sizes can be defined on a per-volume basis

```
<detector>
    <parameter name="activateAllVolumes" value="false"/>

    <volume name="det_dw_01" sensitive="true" maxStepSize="1mm"/>
    <volume name="det_up_01" maxStepSize="1mm"/>
</detector>
```

Some useful restG4 command line options:

- **Multithreading** (-t 8): launches multiple threads, simulation runs much faster!
- **Number of primaries** (-n 1000): launch 1000 primary particles
- **Number of recorded events** (-e 100): stop simulation when 100 events are stored in file
- **Simulation time** (--time 1h20m10s): stop simulation after set time has elapsed
- **Output file** (-o output.root): set the output file

- Let's run a restG4 example from the REST install directory.
- Go to the example directory: `cd $REST_PATH/examples/restG4/04.MuonScan`
- Run the example and save output to some directory with write permissions:

  `restG4 CosmicMuons.rml -o /tmp/simulation.root`

- View events! `restGeant4_ViewEvent /tmp/simulation.root`

- Contents of a restG4 simulation file:
- `restManager filename.root` to get to the root prompt

```
root [0] .ls
TFile**         test.root
 TFile*         test.root
  OBJ: TRestAnalysisTree        AnalysisTree    AnalysisTree : 0 at: 0x555dc894ca30
  OBJ: TTree    EventTree       TRestGeant4EventTree : 0 at: 0x555dc9db1690
  KEY: TGeoManager      Geometry;1      Geometry imported from GDML
  KEY: TRestGeant4Metadata      defaultTRestGeant4Metadata;1    Default TRestGeant4Metadata
  KEY: TRestGeant4PhysicsLists  defaultTRestGeant4PhysicsLists;1        Default TRestGeant4PhysicsLists
  KEY: TRestAnalysisTree        AnalysisTree;1  AnalysisTree
  KEY: TRestRun defaultTRestRun;1       CosmicNeutrons
  KEY: TTree    EventTree;1     TRestGeant4EventTree
root [1]
```

Simulation output is stored in the EventTree (ROOT TTree)

Access the current event:

```
root [1] ev0
(TRestGeant4Event *) 0x555dca58a770
```

```
root [2] ev0->GetID()
(int) 62143
```

Use “tab” key to autocomplete methods and discover new ones! Or check the documentation of each class here

Use TRestRun (run0) methods to get to a new entry

```
(TRestRun *) 0x555dc9385e90
root [5] run0->GetEntry(4)
root [6] ev0->GetID()
(int) 151270
```

Data hierarchy: Run > Event > Track > Hit

Tracks can be accessed from event:

```
root [11] auto track = ev0->GetTracks()[0]
(TRestGeant4Track &) @0x7f2408cbe220
```

Track information can be printed or accessed via many built-in methods

```
root [13] track.PrintTrack()
 * TrackID: 1 - Particle: neutron - ParentID: 0 - Created by 'PrimaryGenerator' with initial KE of 73.03 MeV
   Initial position (63.403, 1763.529, -217.247) mm at time 0 - Time length of 13.42 ns and spatial length of 1.50 m
      - Hit 0 - Energy: 0 - Process: Init - Volume: world_PV - Position: (63.403, 1763.529, -217.247) mm - Time: 0 - KE: 73.03 MeV
      - Hit 1 - Energy: 0 - Process: Transportation - Volume: world_PV - Position: (73.253, 295.000, -225.645) mm - Time: 13.14 ns - KE: 73.03 MeV
      - Hit 2 - Energy: 0 - Process: neutronInelastic - Volume: Shielding_shielding20cm - Position: (73.465, 263.524, -225.825) mm - Time: 13.42 ns - KE: 0
```

Hits can be accessed from track:

```
root [15] auto hits = track.GetHits()
(TRestGeant4Hits &) @0x7f2408cbe3e8
```

```
root [16] hits.GetProcessName(2)
(TString) "neutronInelastic"[16]
```

These methods can be used in ROOT/Python code scripts for fast analysis, or used in jupyter notebooks

The data can be plotted directly with matplotlib

Or we can use REST processes to populate analysis tree (exercises after this session!)