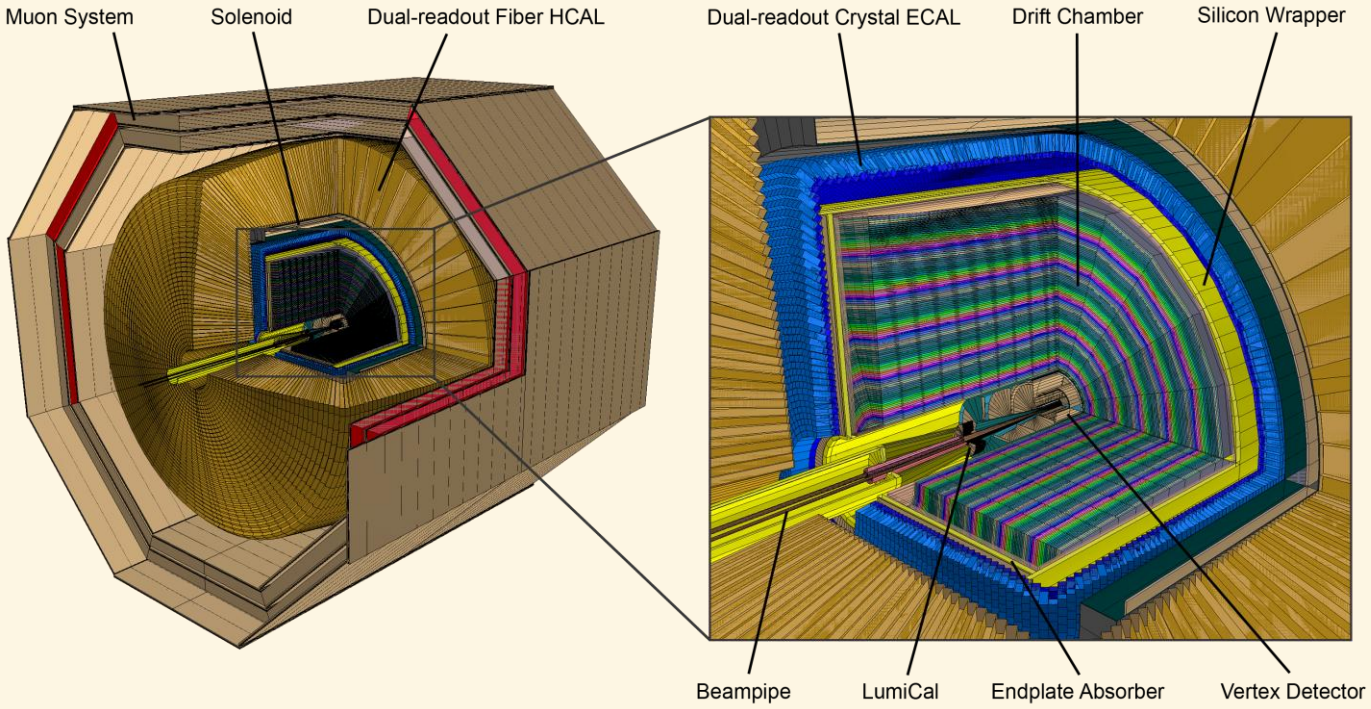
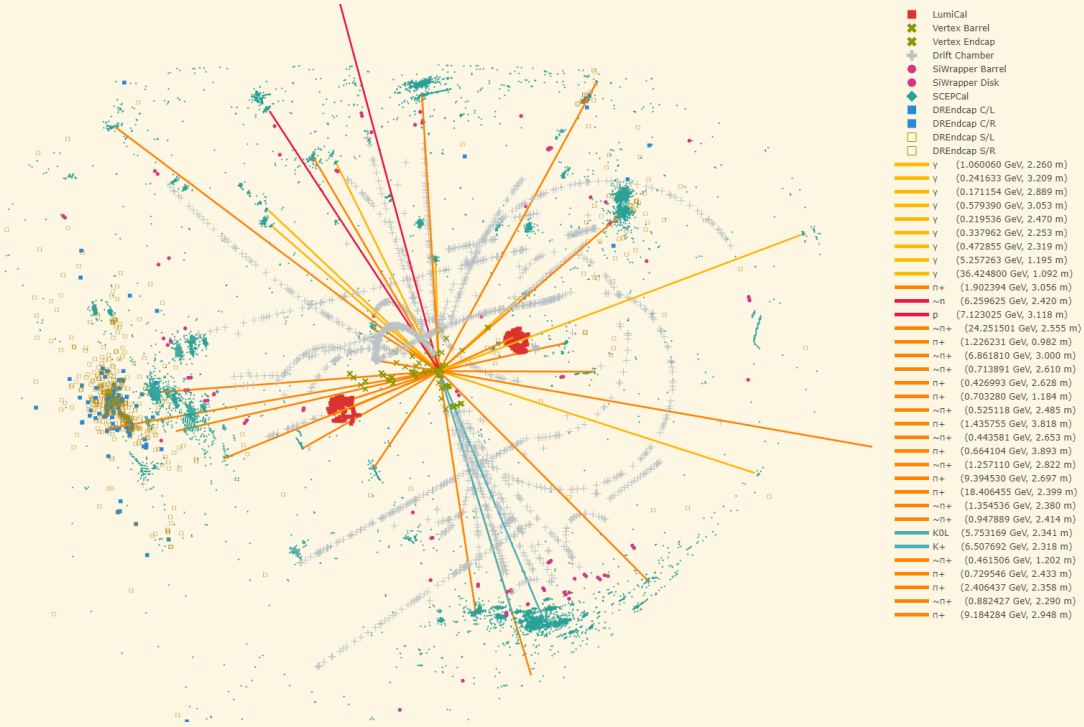
Full Detector Simulation in DD4hep

Concepts and Examples



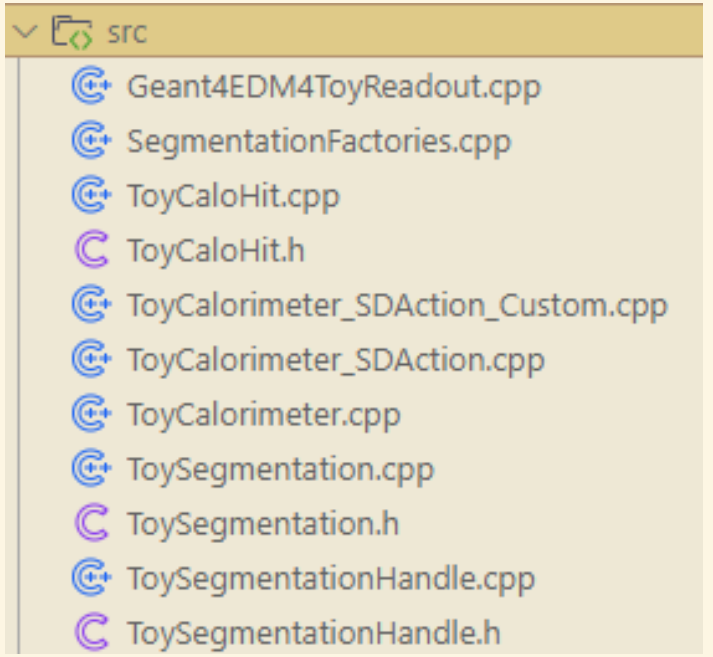


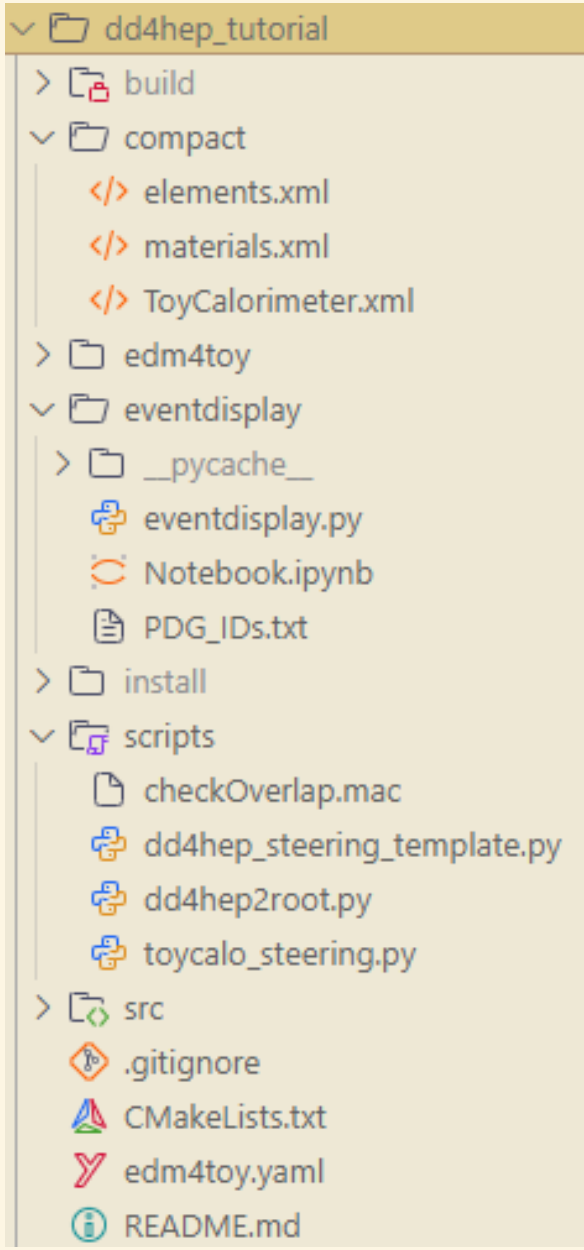
Wonyong Chung
Princeton University

April 2025
Fermilab

Introduction and Goals

- Why full simulation?
  - Some things you can't parameterize
  - ML surrogates – interesting approach but in progress
- DD4hep vs pure Geant4
  - Essentially a modular wrapping of Geant4 components
- What you need for a new detector simulation
  - Compact XML + Steering file
  - Detector constructor
  - Segmentation (detector coordinates to global 3D coordinates)
  - Sensitive action (optional, custom detector response)
  - Readout class (optional, for custom data fields)
- Will go through each of these
  - Exercises: complete implementation of toy detector






```
git clone https://github.com/wonyongc/dd4hep_tutorial

source /cvmfs/sw.hsf.org/key4hep/setup.sh
```

Compact XML

- XML
- Specifies the detector configuration
  - Material properties
  - Visualization
  - Subdetectors to use
    - Dimensions/constants to pass in
  - Readout/segmentation

```
<properties>
  <matrix name="RI_PbWO" coldim="2" values="···
  "/>
  <matrix name="AbsLen_PbWO" coldim="2" values="···
  "/>
  <matrix name="scintFast_PbWO" coldim="2" values="···
  "/>
</properties>

<materials>
  <material name="PbWO4">···
  </material>
</materials>

<readouts>
  <readout name="ToyCalorimeterReadout">
    <segmentation type="ToySegmentation"/>
    <id>system:5,phi:9,theta:9,depth:9</id>
  </readout>
</readouts>

<display>
  <vis name="ToyCalorimeterGlobalVis" alpha="0.1" r="1." g="0
  <vis name="boxVis"                  alpha="1"   r="1.0" g="0
</display>

<define>
  <constant name="world_size" value="20 *m"/>
```

Steering File

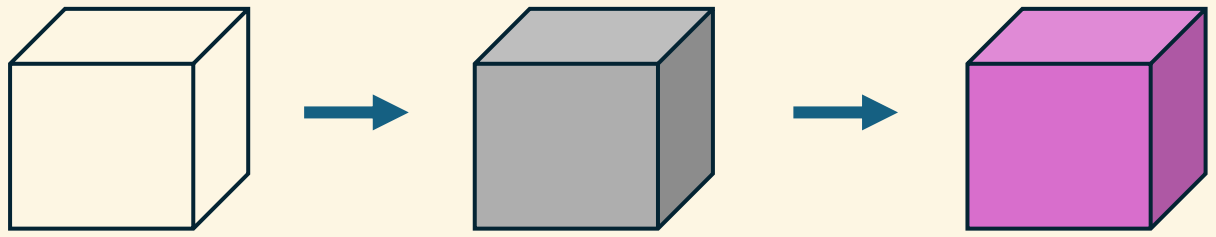
- Python
- Settings for the simulation using the detector
  - Event specification (input/output)
  - Physics list
  - Sensitive actions and filters (thresholds)
  - Magnetic field
  - Randomness
  - MC truth

```
SIM.gun.particle        = settings['particle']
SIM.gun.position        = (0, 0, 0)
SIM.gun.momentumMin     = settings['momentum']-settings['plusminus']
SIM.gun.momentumMax     = settings['momentum']+settings['plusminus']
SIM.gun.phiMin          = settings['phi'][0]
SIM.gun.phiMax          = settings['phi'][1]
SIM.gun.thetaMin        = settings['theta'][0]
SIM.gun.thetaMax        = settings['theta'][1]
#~~~~~~~~~~~~~~~~~ Vertex ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SIM.crossingAngleBoost = 0.0
SIM.vertexOffset = [0.0, 0.0, 0.0, 0.0]
SIM.vertexSigma  = [0.0, 0.0, 0.0, 0.0]
#~~~~~~~~~~~~~~~~~ Filters ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SIM.filter.filters = {
    'geantino':{'name':      'GeantinoRejectFilter/GeantinoRejector
    |      |      | 'parameter':{ }},
    'edep':     {'name':      'EnergyDepositMinimumCut/edep',
    |      |      | 'parameter':{'Cut':settings['edep']}},
}
#~~~~~~~~~~~~~~~~~ Calorimeter ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SIM.action.calorimeterSDTypes = ['calorimeter']
SIM.action.mapActions["MyToyCalorimeter"]      = "ToyCalorimeter_SD
SIM.filter.mapDetFilter['MyToyCalorimeter']    = 'edep'
```

Detector Constructor

- Pull in parameters from XML file
  - Dimensions
  - Materials
  - Visualization
- Initialize the main detector element and segmentation
- **Construct the detector geometry**
  - Make a Shape
  - Make a Volume from a Shape
  - Make a PlacedVolume from a Volume
  - Set physical volume ID for PlacedVolumes
  - Nest geometry as desired
  - NB: Volumes are placed into the local coordinate system of the parent volume! Account for rotations and displacements accordingly
- For complex geometries, start to need intermediate envelope volumes (around ~1000 sub-volumes per volume)
- Check overlaps!

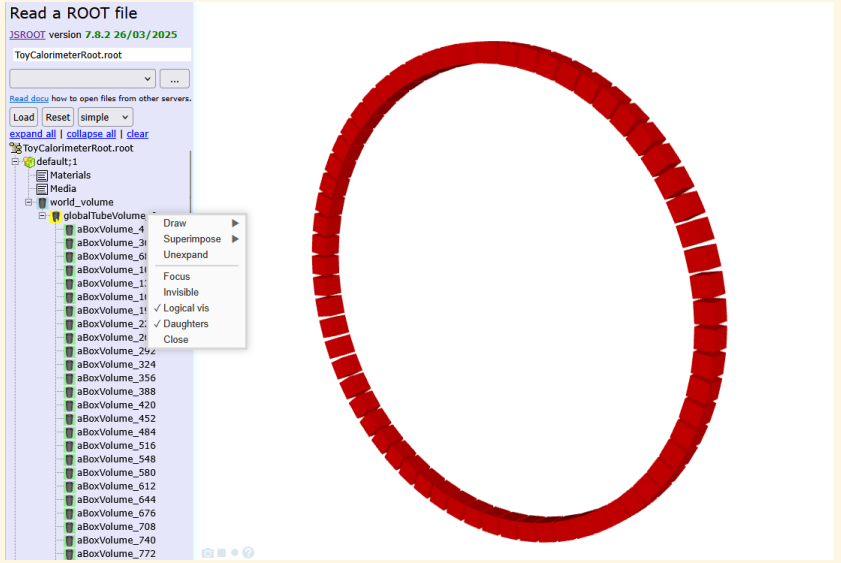
```
static Ref_t create_toy_calorimeter(Detector& theDetector, xml_h xmlElement, SensitiveD

  // ------------------------------------------------------------------
  // 1) Load the XML elements
  // ------------------------------------------------------------------

  // Handle to the XML <detector> node
  xml_det_t  detectorXML                =xmlElement;

  // Handle to custom XML elements like <dim>, <barrel>, etc.
  xml_comp_t dimXML                     =detectorXML.child(_Unicode(dim));
  xml_comp_t boxXML                     =detectorXML.child(_Unicode(box));

  std::string detName                   =detectorXML.nameStr();
  int detId                             =detectorXML.id();

  const double  PHI_SEGMENTS            =dimXML.attr<double>(_Unicode(phiSegments));
  const double  BARREL_HALF_Z           =dimXML.attr<double>(_Unicode(barrelHalfZ));
  const double  BARREL_INNER_R          =dimXML.attr<double>(_Unicode(barrelInnerR));
  const double  BARREL_OUTER_R          =dimXML.attr<double>(_Unicode(barrelOuterR));

  const double  BOX_HALF_X              =boxXML.attr<double>(_Unicode(boxHalfX));
  const double  BOX_HALF_Y              =boxXML.attr<double>(_Unicode(boxHalfY));
  const double  BOX_HALF_Z              =boxXML.attr<double>(_Unicode(boxHalfZ));

  // Initialize the top-level detector element
  dd4hep::DetElement calorimeterDet(detName, detId );

  // Get the world volume and call it experimentalHall
  dd4hep::Volume experimentalHall=theDetector.pickMotherVolume(calorimeterDet);
```



Shape → Volume → PlacedVolume (w/ PhysVolID)

Checkpoint – View the geometry

- Clone the repo and source key4hep
  - `git clone https://github.com/wonyongc/dd4hep_tutorial.git`
  - `source /cvmfs/sw.hsf.org/key4hep/setup.sh`
  - `cd dd4hep_tutorial; mkdir build install`



- Implement your geometry in `src/ToyCalorimeter.cpp` , then
  - `cd build`
  - `cmake -Wno-dev -Wno-cpp -DCMAKE_INSTALL_PREFIX=../install ..`
  - `make install -j8`
  - `export LD_LIBRARY_PATH=../install/lib:$LD_LIBRARY_PATH`
    - `once per shell instance`
- Check overlaps (running from inside build folder)
  - `ddsim --compactFile ../compact/ToyCalorimeter.xml --runType run --macroFile ../scripts/checkOverlap.mac`
- Visualization - export geometry to a ROOT file, scp it to your laptop, and use online JSROOT viewer
  - `python3 ../scripts/dd4hep2root.py -c ../compact/ToyCalorimeter.xml -o ToyCalorimeterRoot.root`
  - https://root.cern/js/latest/

Geometry <-> Simulation: Readout/Segmentation

- Effectively a utility class to bridge information between detector geometry and sensitive action during simulation

- Two essential functions of detector segmentation
  - **Define readout bit fields** (detector coordinates) used for physical volume ID
    - Number of bits per field assignable at runtime in XML
  - **Store/convert detector coordinates** to global 3D coordinates for hit positions

- Other user-defined functions can be defined as desired
- Relevant files (simple, but lots of repetition):
  - `src/ToySegmentation.h`
  - `src/ToySegmentation.cpp`
  - `src/ToySegmentationHandle.h`
  - `src/ToySegmentationHandle.cpp`
  - `src/SegmentationFactories.cpp`

```
<readouts>
  <readout name="ToyCalorimeterReadout">
    <segmentation type="ToySegmentation"/>
    <id>system:5,phi:9,theta:9,depth:9</id>
  </readout>
</readouts>
```

```
                  32 bits
 00000 000000000 000000000 000000000
system    phi       theta     depth
```

Sensitive Action

- Models the detector response
  - Sets up detector hit type, ROOT collections (trees)
  - Custom hit types and collections possible (advanced)
- process() function runs on every Geant4 step
- Implements physics/detector logic
  - Baseline: Create/accumulate detector hits, save energy deposit
  - Terminate certain tracks if desired (e.g. optical photons)
  - Etc.
- Relevant files:
  - **`src/ToyCalorimeter_SDAction.cpp`**
- If using custom hit info:
  - **`edm4toy.yaml`**
  - **`src/ToyCalorimeter_SDAction_Custom.cpp`**
  - **`src/ToyCaloHit.h`**
  - **`src/ToyCaloHit.cpp`**
  - **`src/Geant4EDM4ToyReadout.cpp`**
    - Save your custom hit in `void Geant4EDM4ToyReadout::saveCollection`

```
namespace ToyCalorimeter {
  class ToyCalorimeter_SDAction {
    public:
      // typedef dd4hep::sim::Geant4Tracker::Hit Hit;
      typedef dd4hep::sim::Geant4Calorimeter::Hit Hit;
  };
}
namespace dd4hep {
  namespace sim {

    using namespace ToyCalorimeter;

    template <> void Geant4SensitiveAction<ToyCalorimeter_SDAction>::defineCollecti
      m_collectionID = defineCollection<ToyCalorimeter_SDAction::Hit>("ToyCalorimet
    }

    template <> bool
    Geant4SensitiveAction<ToyCalorimeter_SDAction>::process(const G4Step* step,G4To
      G4StepPoint        *thePrePoint =step->GetPreStepPoint();
      G4TouchableHandle  thePreStepTouchable =thePrePoint->GetTouchableHandle();

      // Uncomment if you want to save MC step contributions
      // Geant4StepHandler h(step);
      // Geant4HitData::MonteCarloContrib contrib = Geant4HitData::extractContribut

      auto cellID =thePreStepTouchable->GetCopyNumber(0);
      G4Track *track =step->GetTrack();

      // Get the segmentation for the sensitive detector
      dd4hep::Segmentation *_geoSeg=&m_segmentation;
      auto segmentation=dynamic_cast<dd4hep::DDSegmentation::ToySegmentation *>(_ge

      // Get the position of the detector cell from the segmentation
      DDSegmentation::Vector3D pos =segmentation->position(cellID);
      Position global(pos.x(),pos.y(),pos.z());

      // Get the energy deposited in the step
      G4double edep =step->GetTotalEnergyDeposit();
```

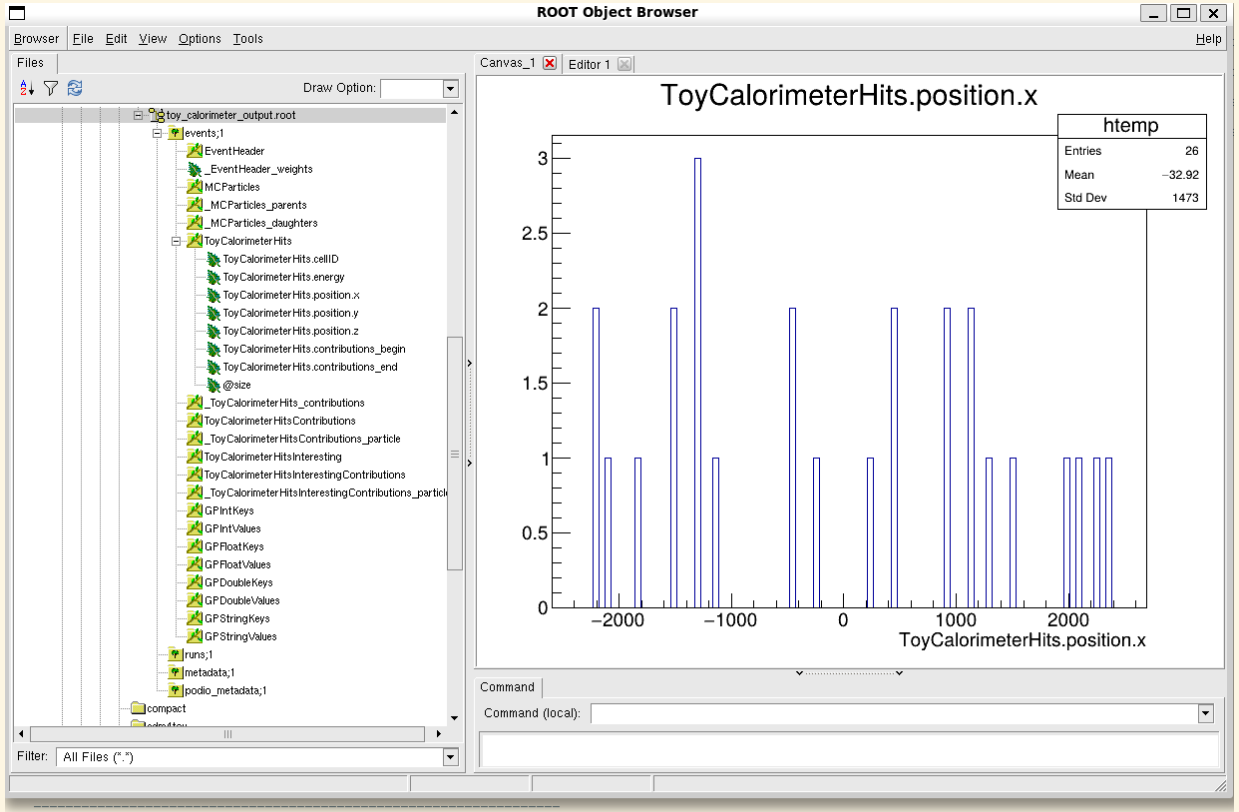
Checkpoint – Run the simulation

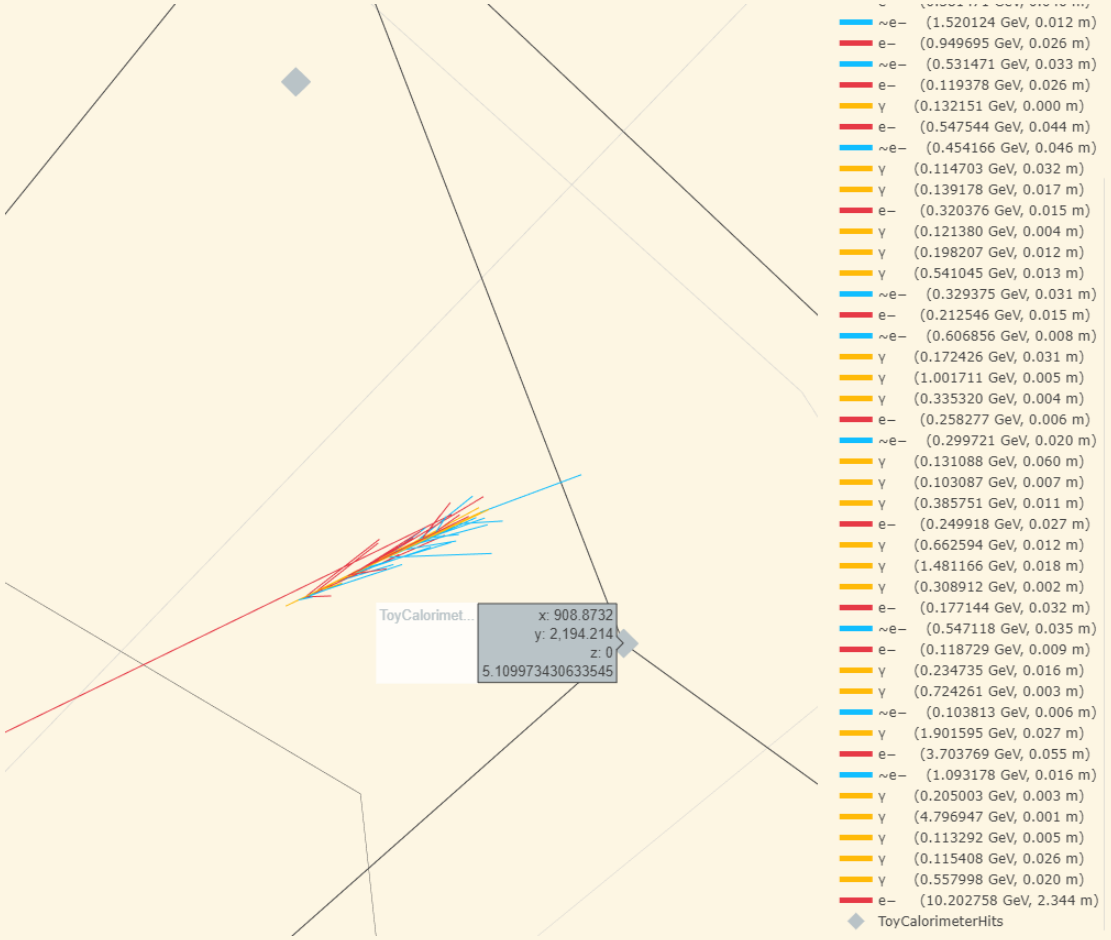
- Implement any changes desired in the relevant segmentation and sensitive action files, then
  - **`cd build`**
  - **`cmake -Wno-dev -Wno-cpp -DCMAKE_INSTALL_PREFIX=../install ..`**
  - **`make install -j8`**
  - **`export LD_LIBRARY_PATH=../install/lib:$LD_LIBRARY_PATH`**
    - **once per shell instance**
- Run the simulation (from the build folder)
  - **`ddsim --steeringFile  ../scripts/toycalo_steering.py`**

```
[18:28:02]*[master][argama:~/src/hep/dd4hep_tutorial/build] ddsim --steeringFile ../toycalo_steering.py
PersistencyIO    INFO  +++ Set Streamer to dd4hep::OpaqueDataBlock
Info in <TGeoManager::TGeoManager>: Geometry default, Detector Geometry created
Info in <TGeoNavigator::BuildCache>: --- Maximum geometry depth set to 100
XMLLoader        INFO  +++ Processing XML file: file:/home/wonyongc/src/hep/dd4hep_tutorial/compact/ToyCalorimete
DocumentHandler  INFO  +++ Loading document URI: file:/home/wonyongc/src/hep/dd4hep_tutorial/compact/ToyCalorimet
gc/src/hep/dd4hep_tutorial/compact/ToyCalorimeter.xml']
DocumentHandler  INFO  +++ Document file:/home/wonyongc/src/hep/dd4hep_tutorial/compact/ToyCalorimeter.xml succes
.
DocumentHandler  INFO  +++ Loading document URI: /home/wonyongc/src/hep/SCEPCal-private/spack/local/.spack-env/v
s.xml [Resolved:'/home/wonyongc/src/hep/SCEPCal-private/spack/local/.spack-env/view/DDDetectors/compact/elements
DocumentHandler  INFO  +++ Document /home/wonyongc/src/hep/SCEPCal-private/spack/local/.spack-env/view/DDDetecto
ully parsed with TinyXML .....
DD4hep           WARN  ++ STD conditions NOT defined by client.  NTP defaults taken
```

Check outputs

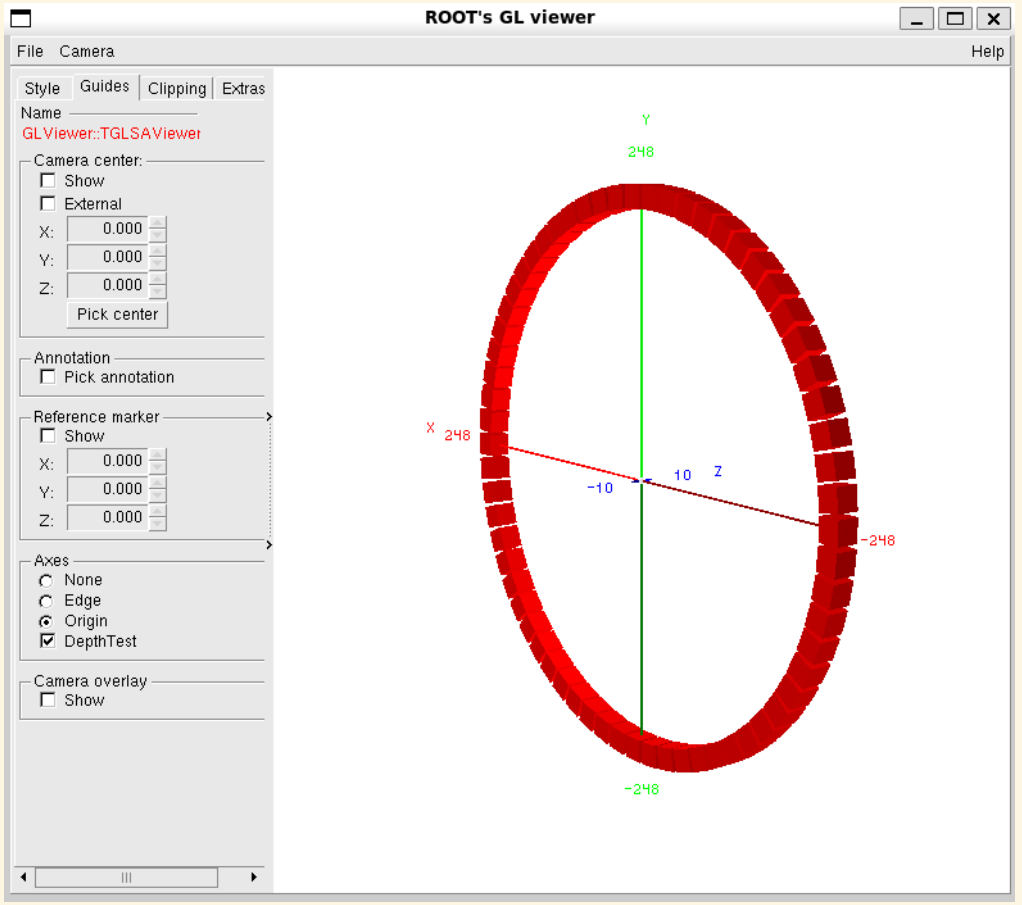
- Check histograms with TBrowser or JSROOT
- Event display in python notebook provided





Exercise – Implement a new toy calorimeter

- **Make decisions:**
  - Homogeneous, sampling
  - Dimensions, material, geometry
  - Stubs for different shapes/materials provided
- **Identify main geometry constructor loop**
  - Currently makes one ring in phi of cube crystals as shown here
- **Complete the barrel**
  - Do simple calculations to radius and crystal dimensions to make them touch each other exactly along edges (instead of having gaps)
  - Add a loop over theta to extend the barrel
  - Try other shapes, make more layers, etc.
- **Validate geometry with visualization and overlap check**
- **Run particle gun simulation**



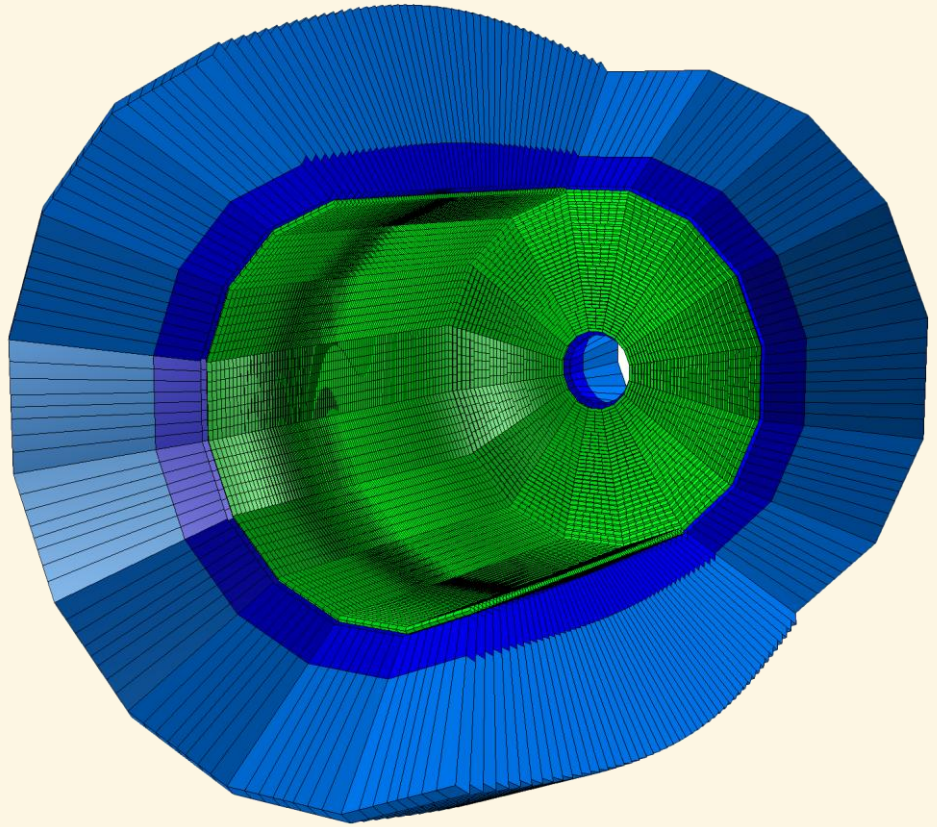
Exercises (Advanced)

- **Customize the detector response and readout beyond energy deposit**

- **Pick an interesting physics quantity to read out**
  - Incident angle of particle momentum, timing information, particle status/process, etc.

- **Implement the new quantity in the relevant files**
  - `edm4toy.yaml`
  - `src/ToyCalorimeter_SDAction_Custom.cpp`
  - `src/ToyCaloHit.h`
  - `src/ToyCaloHit.cpp`
  - `src/Geant4EDM4ToyReadout.cpp`

- **Change SD action and output config in steering file**



Event display

- scp the simulation output ROOT file to your laptop
  - **scp wochung@lxplus.cern.ch:/src/dd4hep_tutorial/build/toy_calorimeter_output.root /local/dd4hep_tutorial/eventdisplay/**
- Recreate the edm4hep classes on your laptop through ROOT
  - **cd /local/dd4hep_tutorial/eventdisplay/**
  - **root -l '../scripts/createProject.c("toy_calorimeter_output.root")'**
  - **.q**
  - **cd ToyCalorimeter**
  - **./MAKEP**
- Then run the python notebook in the eventdisplay folder

```
[14:06:38][master][argama:~/src/hep/dd4hep_tutorial/build] root -l '../scripts/createProject.c("toy_calorimeter_output.root")'
root [0]
Processing ../scripts/createProject.c("toy_calorimeter_output.root")...
MakeProject has generated 0 classes in ToyCalorimeter
ToyCalorimeter/MAKEP file has been generated
Shared lib ToyCalorimeter/ToyCalorimeter.so has been generated
Shared lib ToyCalorimeter/ToyCalorimeter.so has been dynamically linked
Project created in directory: ToyCalorimeter
root [1] .q
[14:06:44][master][argama:~/src/hep/dd4hep_tutorial/build] cd ToyCalorimeter
[14:06:48][master][argama:~/src/hep/dd4hep_tutorial/build/ToyCalorimeter] ./MAKEP
[14:06:52][master][argama:~/src/hep/dd4hep_tutorial/build/ToyCalorimeter]
```