Changing Geometry in DD4hep

André Sailer

CERN-EP-SFT

FCC Software Hands-on Tutorial
October 19–21, 2022

Table Of Contents

Scope and Goals

Changing DD4hep Geometries

In this set of tutorials you will learn how to

- simulate an existing detector model
- modify its XML files
- check for overlaps with Geant4
- get a better understand of what DD4hep does under the hood
- Write your own c++ detector constructor and run with it

Setup and Requirements

Setup

Assuming you are on some CentOS7 machine, lxplus, VM, Docker with CVMFS

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh
mkdir mydd4heptutorial
cd mydd4heptutorial
cp -r $LCGEO/FCCee/compact/FCCee_o1_v05 .
pip install --user uproot3
```

You should now see something like this when listing the current working directory

```
ls -ltra
total 8
drwxr-xr-x. 28 sailer zf 4096 Oct 14 11:29 ..
drwxr-xr-x.  3 sailer zf   26 Oct 14 11:30 .
drwxr-xr-x.  2 sailer zf 4096 Oct 14 11:30 FCCee_o1_v05
```

Show Some Distribution Script

Save this a `showPlots.py`, with `uproot3` this can draw histograms in the terminal

```
#!/bin/env python
import sys, ROOT, uproot3 as uproot
from ROOT import TFile, TTree
ROOT.gROOT.SetBatch(True)
if len(sys.argv) < 2:
  print("Please specify input file"); sys.exit(1)
inputFile = sys.argv[1]; print("Reading:", inputFile)
tfile = ROOT.TFile.Open(inputFile)
myTree = tfile.Get("events")
myTree.Draw("ECalEndcapCollection.position.z>>zHist(100, 2300, 2510)",
            "ECalEndcapCollection.position.z > 0")
myTree.Draw("ECalEndcapCollection.energy>>eHist(30, 0, 0.002)")
zHist = tfile.Get("zHist")
eHist = tfile.Get("eHist")
oFile = ROOT.TFile.Open("temp.root", "RECREATE")
eHist.Write(); zHist.Write(); oFile.Close()
uFile = uproot.open("temp.root")
uFile["zHist"].show()
uFile["eHist"].show()
```


Simulate some Muons in CLD

Simulation of 100 Muons

Let's use this detector model to simulate a couple of Muons

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --enableGun \
      --gun.distribution uniform \
      --gun.energy "10*GeV" \
      --gun.particle mu- \
      --numberOfEvents 100 \
      --outputFile Step1_edm4hep.root
...
DDSim INFO DDSim INFO  Total Time:   152.55 s (User), 0.98 s (System)
DDSim INFO DDSim INFO  StartUp Time: 136.30 s, Event Processing: 16.25 s (0.16 s/
```

And plot the distribution with `showPlots.py`

```
python showPlots.py Step1_edm4hep.root
```

Removing an Included XML file

Removing an Include

The Silicon Tracker has a lot of volumes, for the next important bit (overlap checking) we will remove those sub-detectors, because we are on a schedule.

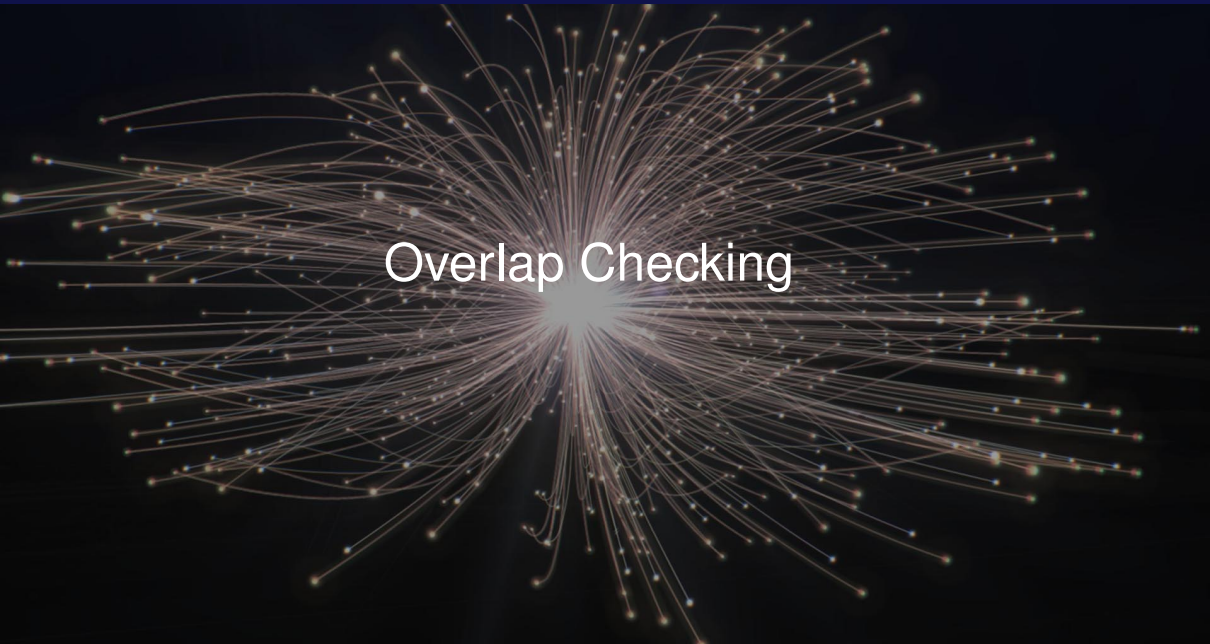
1. Open `FCCee_o1_v05/FCCee_o1_v05.xml`
2. Find
```
<include ref="InnerTracker_o2_v06_02.xml"/>
<include ref="OuterTracker_o2_v06_02.xml"/>
```
3. And replace, or delete
```
<!-- <include ref="InnerTracker_o2_v06_02.xml"/> -->
<!-- <include ref="OuterTracker_o2_v06_02.xml"/> -->
```
4. Comment out the `<plugins> ... </plugins>` in the same file
   - They use some constants defined in the commented files
   - We don't need them for the simulations today

Simulate the model without the Tracker

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --enableGun \
      --gun.distribution uniform \
      --gun.energy "10*GeV" \
      --gun.particle mu- \
      --numberOfEvents 100 \
      --outputFile Step2_edm4hep.root
...
DDSim INFO DDSim INFO  Total Time:   7.79 s (User), 0.46 s (System)
DDSim INFO DDSim INFO  StartUp Time: 2.71 s, Event Processing: 5.08 s (0.05 s/Eve
```

Fast enough for repeated testing

The distributions should not have noticeably changed



Overlap Checking

Overlap Checking

Whenever you change the geometry in a non-trivial way there are the possibilities of overlaps and the following things should be kept in mind

1. There are no trivial changes

Overlap Checking

Whenever you change the geometry in a non-trivial way there are the possibilities of overlaps and the following things should be kept in mind

1. There are no trivial changes
2. See point 1

Overlap Checking

Whenever you change the geometry in a non-trivial way there are the possibilities of overlaps and the following things should be kept in mind

1. There are no trivial changes
2. See point 1
3. Run the overlap check

Running the Geant4 Overlap Check

Create the following file as `overlap.mac`

```
/geometry/test/run
exit
```

And then we run `ddsim` with this macro file, and dump the output to a text file for easy browsing

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

- With the full detector model including the tracker this would take about 30 minutes

Running the Geant4 Overlap Check

Create the following file as `overlap.mac`

```
/geometry/test/run
exit
```

And then we run `ddsim` with this macro file, and dump the output to a text file for easy browsing

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

- With the full detector model including the tracker this would take about 30 minutes
- There are some exceptions. . . someone didn't follow step 3?

Running the Geant4 Overlap Check

Create the following file as `overlap.mac`

```
/geometry/test/run
exit
```

And then we run `ddsim` with this macro file, and dump the output to a text file for easy browsing

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

- With the full detector model including the tracker this would take about 30 minutes
- There are some exceptions. . . someone didn't follow step 3?
  - The exceptions about the VertexEndcapModule are due to a too small envelope, and could be easily fixed

Running the Geant4 Overlap Check

Create the following file as `overlap.mac`

```
/geometry/test/run
exit
```

And then we run `ddsim` with this macro file, and dump the output to a text file for easy browsing

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

- With the full detector model including the tracker this would take about 30 minutes
- There are some exceptions. . . someone didn't follow step 3?
  - The exceptions about the VertexEndcapModule are due to a too small envelope, and could be easily fixed
  - The exceptions about the Boolean Volume are more vexing. . . , let's ignore those for now. Maybe drop the HOMAbsorber include as well

Modifying the ECal

Modifying the ECal

- Now let's do some "real" modifications to the detector model
- Let's change the number of layers and silicon thicknesses of the ECal Endcap
- Open the file `FCCee_o1_v05/ECalEndcap_o2_v01_03.xml`
- Find this block:

```
<layer repeat="40" vis="ECalLayerVis">
  <slice material = "TungstenDens24" thickness = "1.90*mm" vis="ECalAbsorberVis" radiator="yes"/
  <slice material = "G10"            thickness = "0.15*mm" vis="InvisibleNoDaughters"/>
  <slice material = "GroundOrHVMix"  thickness = "0.10*mm" vis="ECalAbsorberVis"/>
  <slice material = "Silicon"        thickness = "0.50*mm" sensitive="yes" limits="cal_limits"
        vis="ECalSensitiveVis"/>
  <slice material = "Air"            thickness = "0.10*mm" vis="InvisibleNoDaughters"/>
  <slice material = "siPCBMix"       thickness = "1.30*mm" vis="ECalAbsorberVis"/>
  <slice material = "Air"            thickness = "0.25*mm" vis="InvisibleNoDaughters"/>
  <slice material = "G10"            thickness = "0.75*mm" vis="InvisibleNoDaughters"/>
</layer>
```

Modifying the ECal

- Now let's do some "real" modifications to the detector model
- Let's change the number of layers and silicon thicknesses of the ECal Endcap
- Open the file `FCCee_o1_v05/ECalEndcap_o2_v01_03.xml`
- and change the number of layers and the silicon thickness:

```
<layer repeat="20" vis="ECalLayerVis">
  <slice material = "TungstenDens24" thickness = "1.90*mm" vis="ECalAbsorberVis" radiator="yes"/
  <slice material = "G10"            thickness = "0.15*mm" vis="InvisibleNoDaughters"/>
  <slice material = "GroundOrHVMix"  thickness = "0.10*mm" vis="ECalAbsorberVis"/>
  <slice material = "Silicon"        thickness = "1.00*mm" sensitive="yes" limits="cal_limits"
        vis="ECalSensitiveVis"/>
  <slice material = "Air"            thickness = "0.10*mm" vis="InvisibleNoDaughters"/>
  <slice material = "siPCBMix"       thickness = "1.30*mm" vis="ECalAbsorberVis"/>
  <slice material = "Air"            thickness = "0.25*mm" vis="InvisibleNoDaughters"/>
  <slice material = "G10"            thickness = "0.75*mm" vis="InvisibleNoDaughters"/>
</layer>
```

Simulate with the new ECal

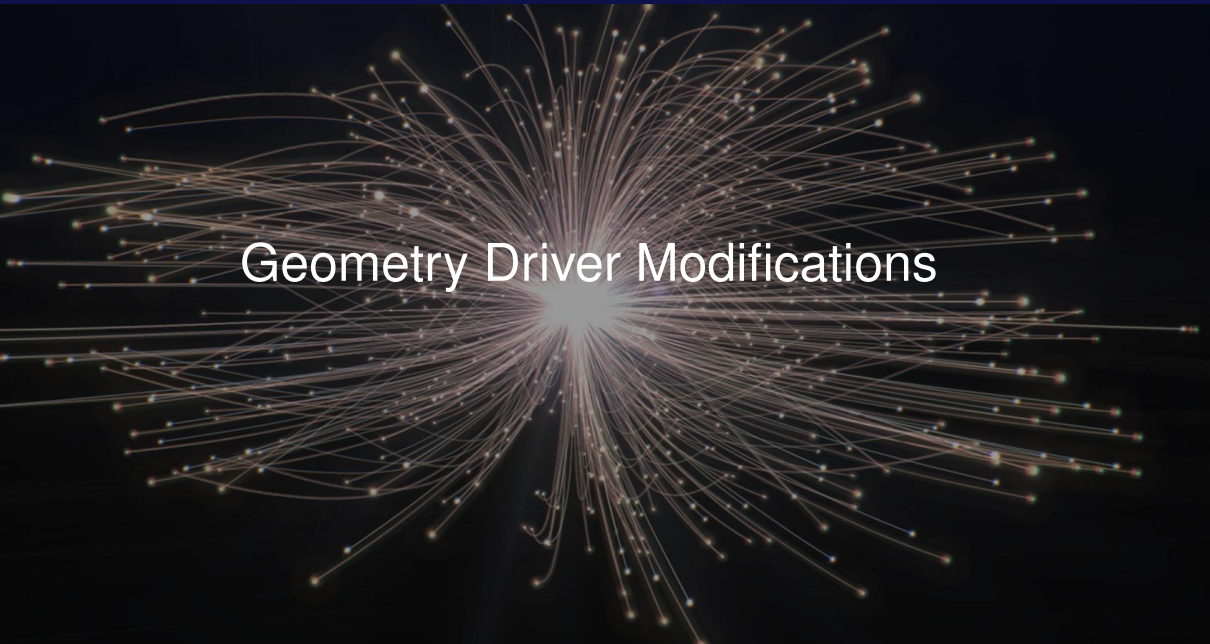
```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --enableGun \
      --gun.distribution uniform \
      --gun.energy "10*GeV" \
      --gun.particle mu- \
      --numberOfEvents 100 \
      --outputFile Step3_edm4hep.root
```

Compare the distributions

- Now compare the output of showPlots for Step2 and Step3
- Note how the distributions changed?

Congratulations

You are now an expert in modifying detector geometries!



Geometry Driver Modifications

DD4hep Plugin Basics

- You just saw how changing the XML file changes the detector geometry, but how does this work under the hood?
- The `type` attribute of the detector tag tells DD4hep which detector constructor to load

```
<detector name="ECalEndcap"
          type="GenericCalEndcap_o1_v01"
          id="DetID_ECal_Endcap"
          readout="ECalEndcapCollection"
          vis="ECALVis" >
  <!-- more XML -->
</detector>
```

- DD4hep's plugin service will look in the `components` files it finds via the LD_LIBRARY_PATH (or DD4HEP_LIBRARY_PATH on macOS because of SIP) environment variables, and load the library on-demand, and then instantiate the function
- How can we know which library contains the `GenericCalEndcap` plugin?

```
for DIR in $(echo $LD_LIBRARY_PATH | tr ":" " ") ; do
  grep GenericEndcap $DIR/*.components 2> /dev/null;
done
```

Boilerplate Project

Usually we would add our detector to an existing project, but because overwriting existing libraries in the environment is a bit of a pain, we start with a new project

```
mkdir MyFirstDetector
cd MyFirstDetector
git init
touch CMakeLists.txt
mkdir src
touch src/MyFirstDetector.cpp
```

CMakeLists.txt

In `MyFirstDetector` create a `CMakeLists.txt` file with this content

```
cmake_minimum_required(VERSION 3.12 FATAL_ERROR)

set(PackageName MyFirstDetector)
project(${PackageName})

find_package(DD4hep REQUIRED COMPONENTS DDG4)

# our sources
set(sources ./src/MyFirstDetector.cpp)

# create our library and make the components file
add_dd4hep_plugin(${PackageName} SHARED ${sources})

# link it with DDCore, or whatever you need
target_link_libraries(${PackageName} DD4hep::DDCore)

# Create this_package.sh file, and install
dd4hep_instantiate_package(${PackageName})
```

MyFirstDetector.cpp

```cpp
#include "DD4hep/DetFactoryHelper.h"
#include "XML/Layering.h"
#include "XML/Utilities.h"
#include "DDRec/DetectorData.h"
static dd4hep::Ref_t create_detector(
 dd4hep::Detector& theDetector,
 xml_h entities,
 dd4hep::SensitiveDetector sens)  {
  // XML Detector Element (confusingly also XML::DetElement)
  xml_det_t x_det = entities;
  // DetElement of our detector instance, attach additional information, sub-elements...
  // uses name of detector and ID number as defined in the XML detector tag
  std::string detName  = x_det.nameStr();
  sens.setType("calorimeter");
  dd4hep::DetElement sdet (detName, x_det.id());
  return sdet;
}
DECLARE_DETELEMENT(MyFirstDetector,create_detector)
```

Overall Dimensions, and First Cylinder

Add this just before the `return`:

```
//get the dimensions tag
xml_dim_t dim = x_det.dimensions();
//read its attributes
double rmin = dim.rmin();
double rmax = dim.rmax();
double zmax = dim.zmax();

//Make a Cylinder
dd4hep::Tube envelope(rmin, rmax, zmax);
dd4hep::Material air = theDetector.air();
dd4hep::Volume envelopeVol(detName+"_envelope",
                           envelope,
                           air);
dd4hep::PlacedVolume physvol =
  theDetector.pickMotherVolume(sdet).placeVolume(envelopeVol);
  // add system ID and identify as barrel (as opposed to endcap +/-1)
physvol.addPhysVolID("system", sdet.id()).addPhysVolID(_U(side),0);
sdet.setPlacement(physvol);
```

Compile

We are still in the `MyFirstDetector` directory

```
mkdir build install
cd build
cmake -D CMAKE_INSTALL_PREFIX=$PWD/../install ..
make install
source ../install/bin/thisMyFirstDetector.sh
```

Look at `$PWD/../install` and note the content of the `bin`, `lib` directories, have a look at the `components` file

Use It

1. In the `FCCee_o1_v05` directory, create a `mydetector.xml` file
2. Modify the `FCCee_o1_v05/FCCee_o1_v05.xml` and replace the include for the ECalBarrel with `mydetector.xml`

MyDetector.xml

Fill `mydetector.xml` with

```
<lccdd>
  <!-- Constants, Readout, VIS goes here -->
  <detectors>
    <detector name="MyDetectorName"
              type="MyFirstDetector"
              id="1234"
              readout="MyReadout"
              vis="MyVis" >
      <dimensions
          zmax="ECalBarrel_half_length"
          rmin="ECalBarrel_inner_radius"
          rmax="ECalBarrel_outer_radius"/>

    </detector>
  </detectors>
</lccdd>
```

and also see next page

XML: Constants, Readout, Visualisation

Add to the `mydetector.xml` file:

```
<define>
  <constant name="ECal_cell_size" value="5.1*mm"/>
</define>

<readouts>
  <readout name="MyReadout">
    <segmentation type="GridRPhiEta"
                  grid_size_eta="ECal_cell_size"     phi_bins="360"
                  offset_r="ECalBarrel_inner_radius"
                  grid_size_r="1*cm" />
    <id>system:5,side:2,module:8,stave:4,layer:9,submodule:4,r:32:10,eta:-11,phi:-11</id>
  </readout>
</readouts>

<display>
  <vis name="MyVis" alpha="0.1"
       r="0.1" g=".5" b=".5"
       showDaughters="true"
       visible="false"/>
</display>
```

Check for Overlaps

Go back to the base folder, or change the path to the XML file.

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

Layers in the XML

```
<!-- In the <detector></detector> -->

<layer repeat="10" vis="MyVis">
  <slice material="Iron"     thickness="1*cm" />
  <slice material="G10"      thickness="1*mm" />
  <slice material="Silicon"  thickness="1*mm" sensitive="true" />
  <slice material="G10"      thickness="1*mm" />
</layer>
```

Add interpretation of the layers to `MyFirstDetector.cpp`

```
double currentInnerRadius = rmin; // running inner radius
dd4hep::Layering layering(x_det); // convenience class
int layerNum = 0;
for(xml_coll_t c(x_det,_U(layer)); c; ++c, ++layerNum) {
  xml_comp_t x_layer = c;
  const dd4hep::Layer* lay = layering.layer(layerNum); // Get the layer from the layering engine.
  const double layerThickness = lay->thickness();
  //loop over the number of repetitions
  for(int i=0, repeat=x_layer.repeat(); i<repeat; ++i, ++layerNum)  {

    std::string layerName = detName + dd4hep::_toString(layerNum,"_layer%d");
    //make a volume for the layer
    dd4hep::Tube layerTube(currentInnerRadius,
                           currentInnerRadius + layerThickness, zmax);
    dd4hep::Volume layerVol(layerName, layerTube, air);
    dd4hep::DetElement layerElement(sdet, layerName, layerNum);
    dd4hep::PlacedVolume layerVolPlaced = envelopeVol.placeVolume(layerVol);
    layerVolPlaced.addPhysVolID("layer",layerNum);
    //loop over slices
  } //repetitions
 }//layers
```

Add interpretation of the slices to `MyFirstDetector.cpp`

```
int sliceNum = 0;
for(xml_coll_t slice(x_layer,_U(slice)); slice; ++slice, ++sliceNum)  {
  xml_comp_t x_slice = slice;
  double sliceThickness = x_slice.thickness();
  dd4hep::Material sliceMat  = theDetector.material(x_slice.materialStr());
  std::string sliceName = layerName + dd4hep::_toString(sliceNum,"slice%d");
  dd4hep::Tube sliceTube(currentInnerRadius,
                         currentInnerRadius + sliceThickness, zmax);
  dd4hep::Volume sliceVol(sliceName, sliceTube, sliceMat);
  if ( x_slice.isSensitive() ) {
    sliceVol.setSensitiveDetector(sens);
  }
  //place the slice in the layer
  layerVol.placeVolume(sliceVol);

  currentInnerRadius += sliceThickness;
 } // slices
```

Check for Overlaps

- Compile your detector again
- Go back to the base folder, or change the path to the XML file.

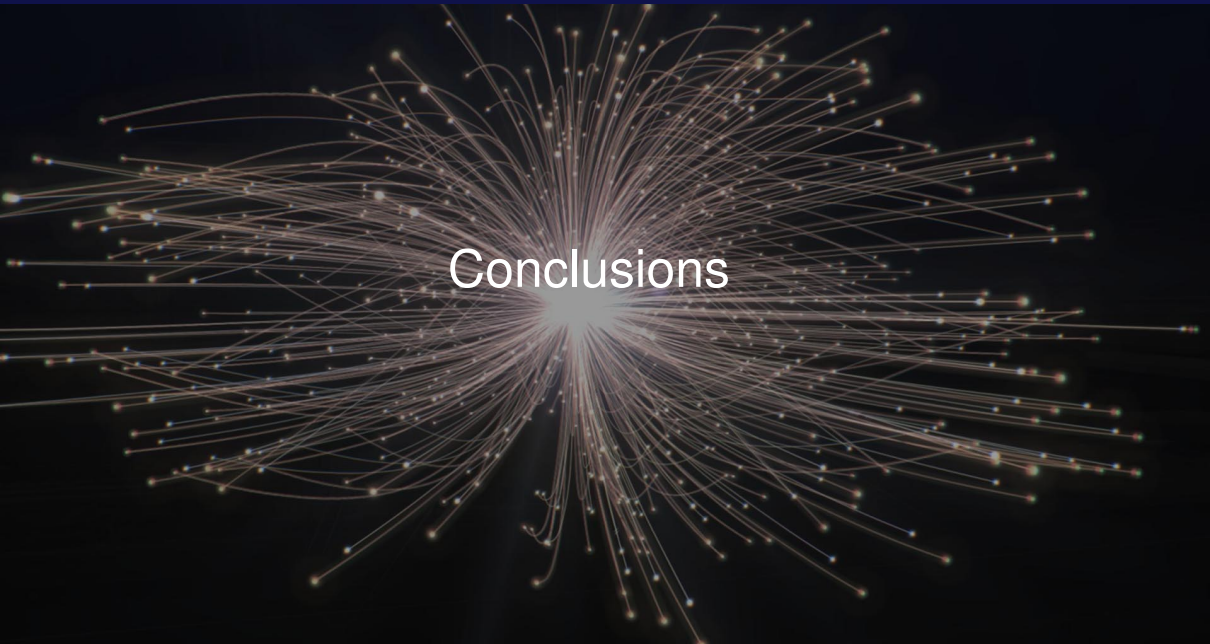
```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --runType run \
      --macroFile overlap.mac > overlapDump &
```

Simulate

Go back to the base folder, or change the path to the XML file.

```
ddsim --compactFile FCCee_o1_v05/FCCee_o1_v05.xml \
      --enableGun \
      --gun.distribution uniform \
      --gun.energy "10*GeV" \
      --gun.particle mu- \
      --numberOfEvents 100 \
      --outputFile Step4_edm4hep.root
```

Modify `showPlots.py` to display properties from this collection (MyReadout)



Conclusions

Conclusions

- Changing geometries with DD4hep can range from trivial to sophisticated
- use the overlap checker
- When you have questions:
  1. Browse the **DD4hep documentation**
  2. Ask us at **https://github.com/aidasoft/dd4hep**