Status of Detector Description

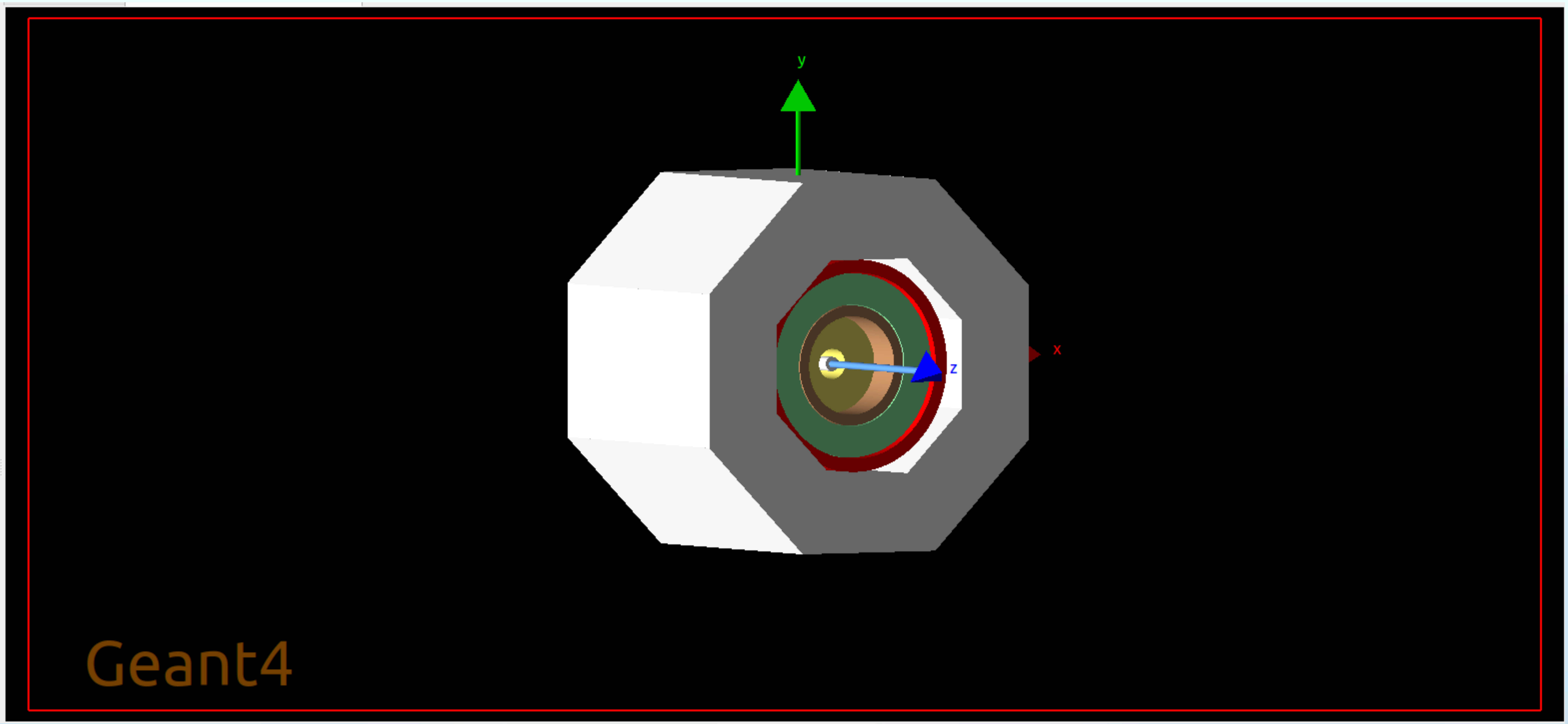
VII SPD collaboration meeting

Aytadzh Allakhverdieva, 23.05.24

Detector description

- Current version of SPD geometry description is based on ROOT's geometry system TGeo.
- GeoModel - a detector description toolkit for HEP (https://geomodel.web.cern.ch/home/).
- A toolkit meets all requirements for SPD geometry description.

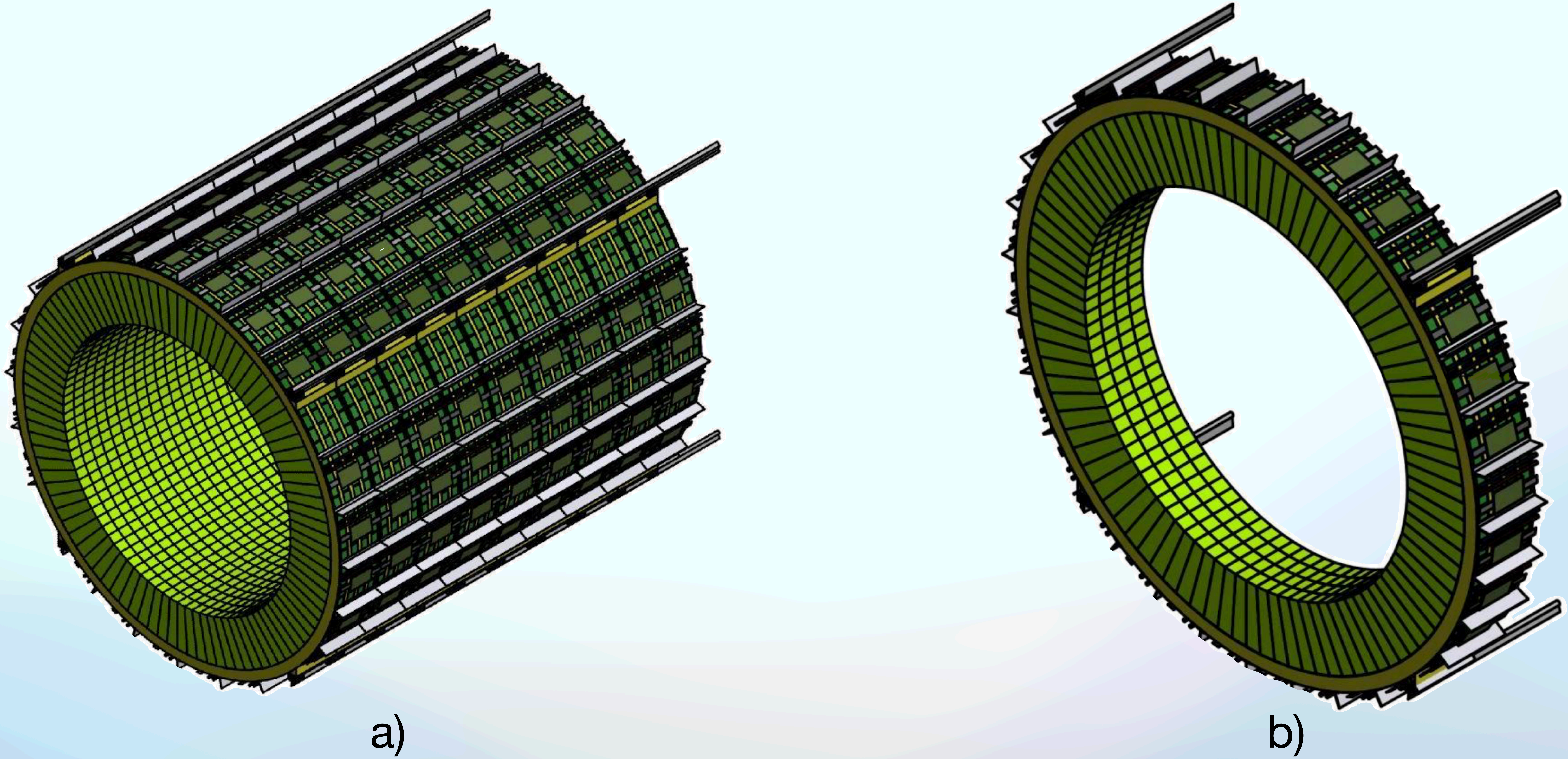
Detector description



SPD geometry description on subsystems' level.

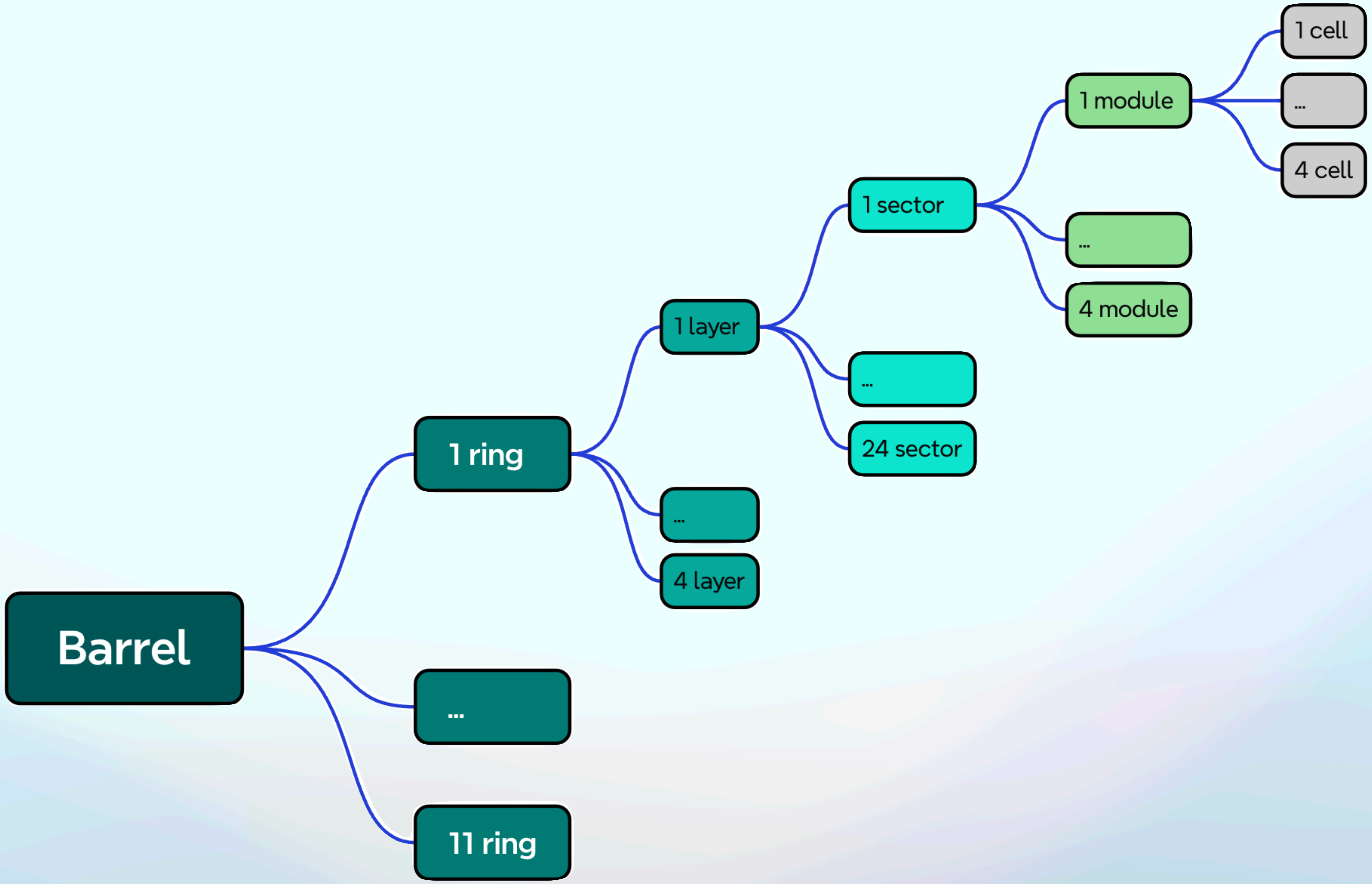
Electromagnetic calorimeter inner structure description

Electromagnetic calorimeter barrel part



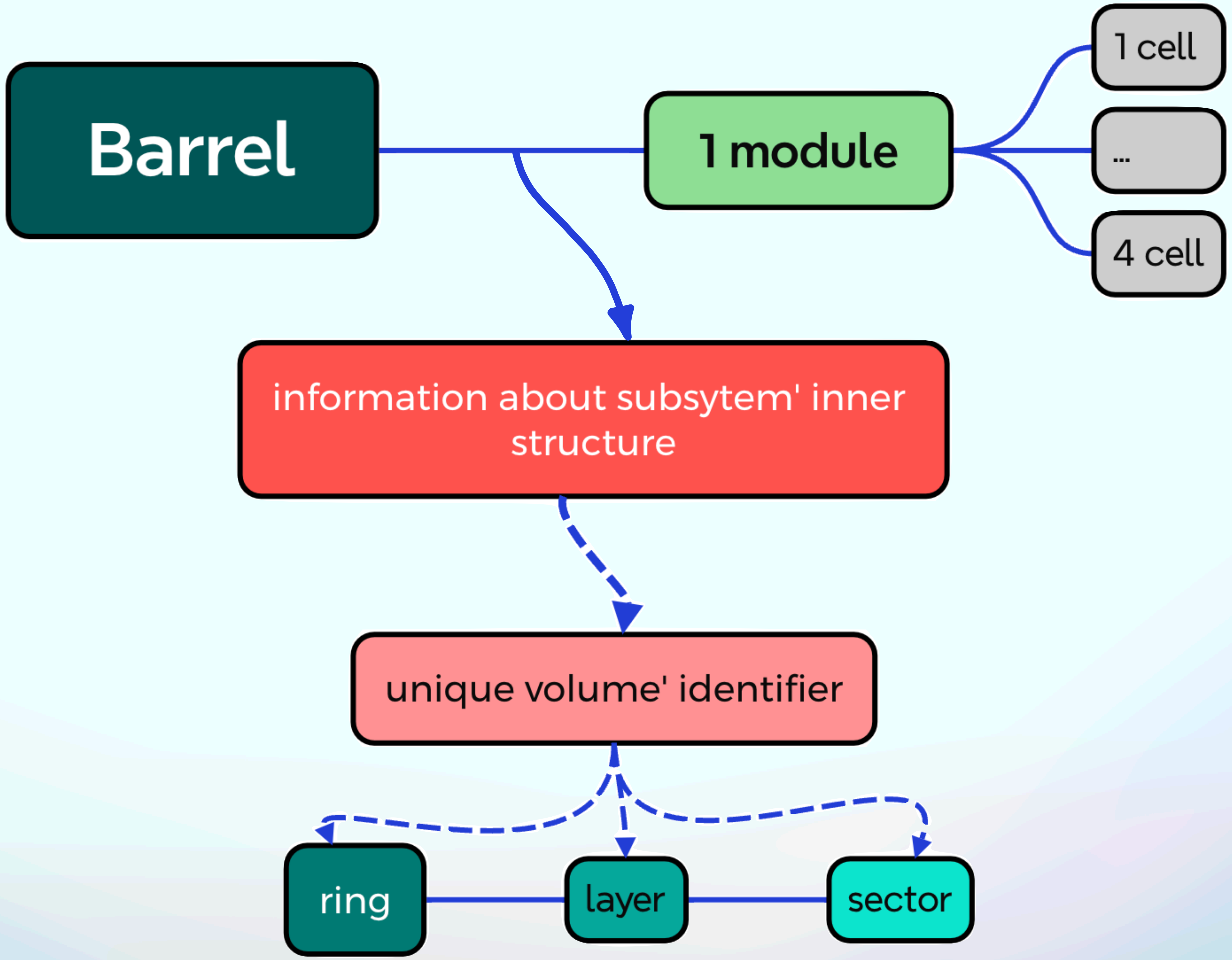
a) The barrel part of the calorimeter. b) A separate ring in more detail.

Electromagnetic calorimeter barrel part



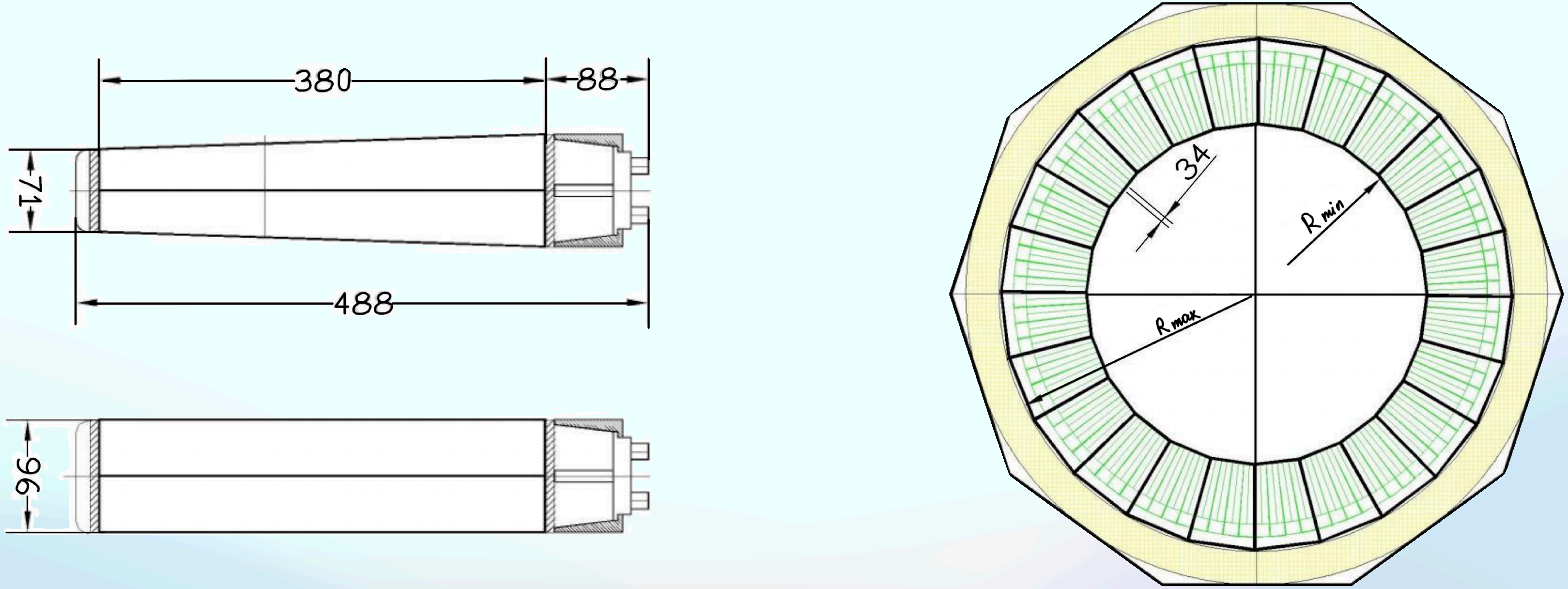
Scheme of the ECal barrel part volumes hierarchy.

Electromagnetic calorimeter barrel part


Barrel
1 module
1 cell
...
4 cell
information about subsytem' inner structure
unique volume' identifier
ring
layer
sector

Scheme of the ECal barrel part volumes hierarchy in detector description. Information about complete structure of ECal barrel is concluded into unique identifier.

Parameterizations for ECal barrel part



a)

b)

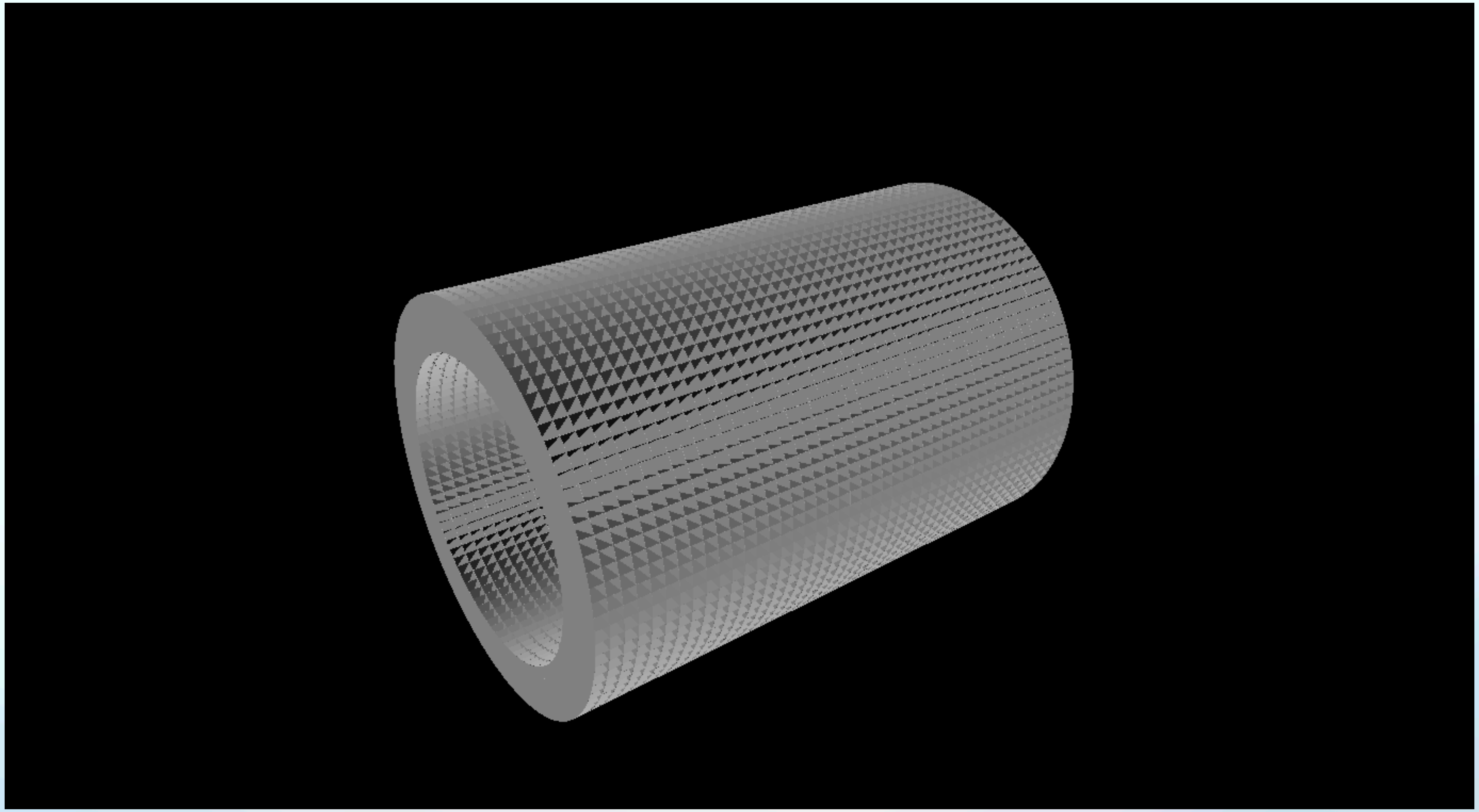
a) ECal module. b) Schematic drawing of a cross-section of the barrel part of the calorimeter.

Parameterizations for ECal barrel part

- translation = Rmin_ecalbar + 0.5 zmodule_height;
- GeoTrf::TranslateX3D = translation*cos(3.75*i*deg);
- GeoTrf::TranslateY3D = translation*sin(3.75*i*deg);
- GeoTrf::RotateY3D((270.0+3.75*i)*deg));

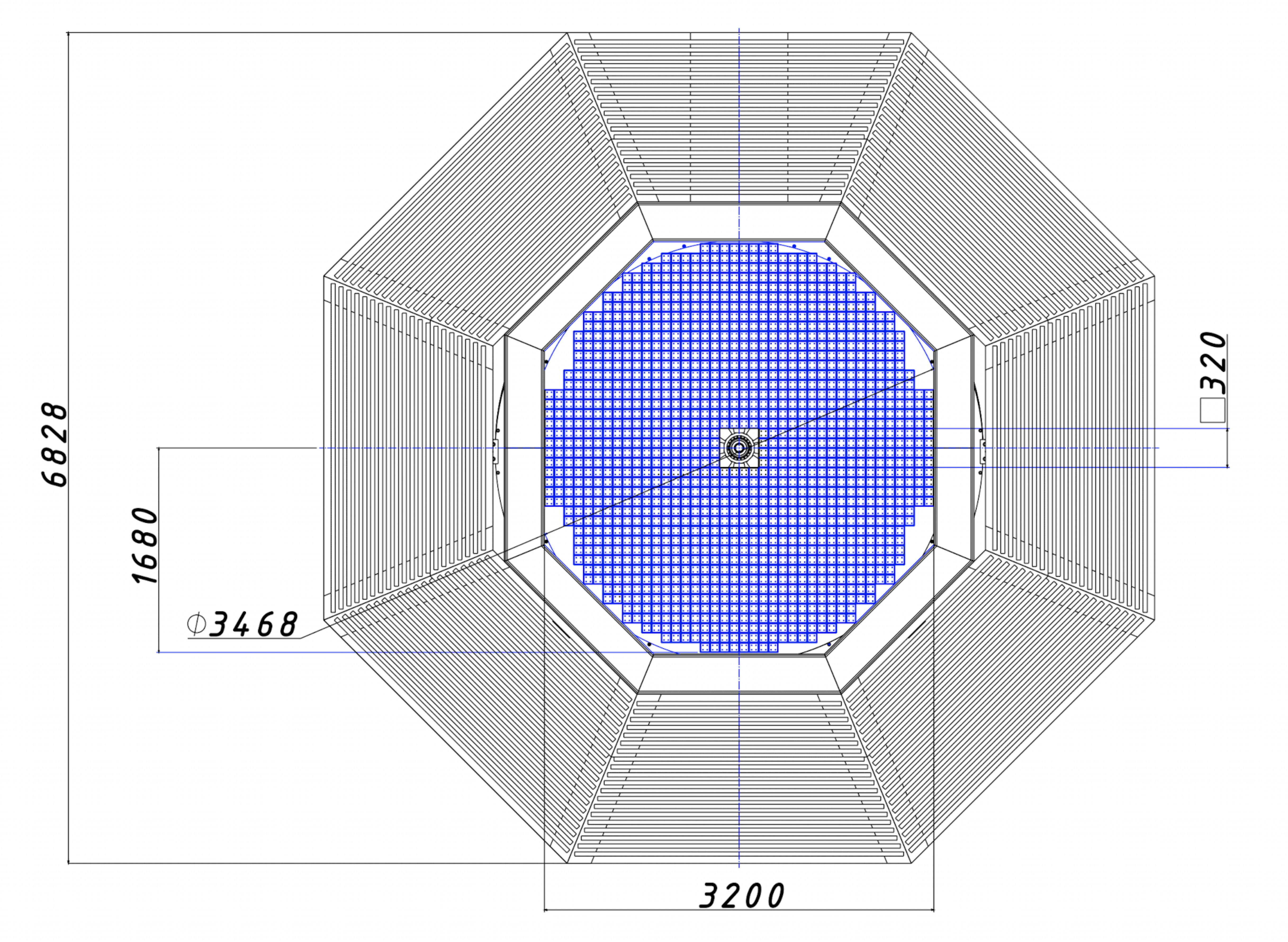
- GeoTrf::TranslateZ3D = 0.5*Length_ecalbar - ymodule_length*j;

Electromagnetic calorimeter barrel part



The barrel part of the ECal.

Electromagnetic calorimeter end-cap part

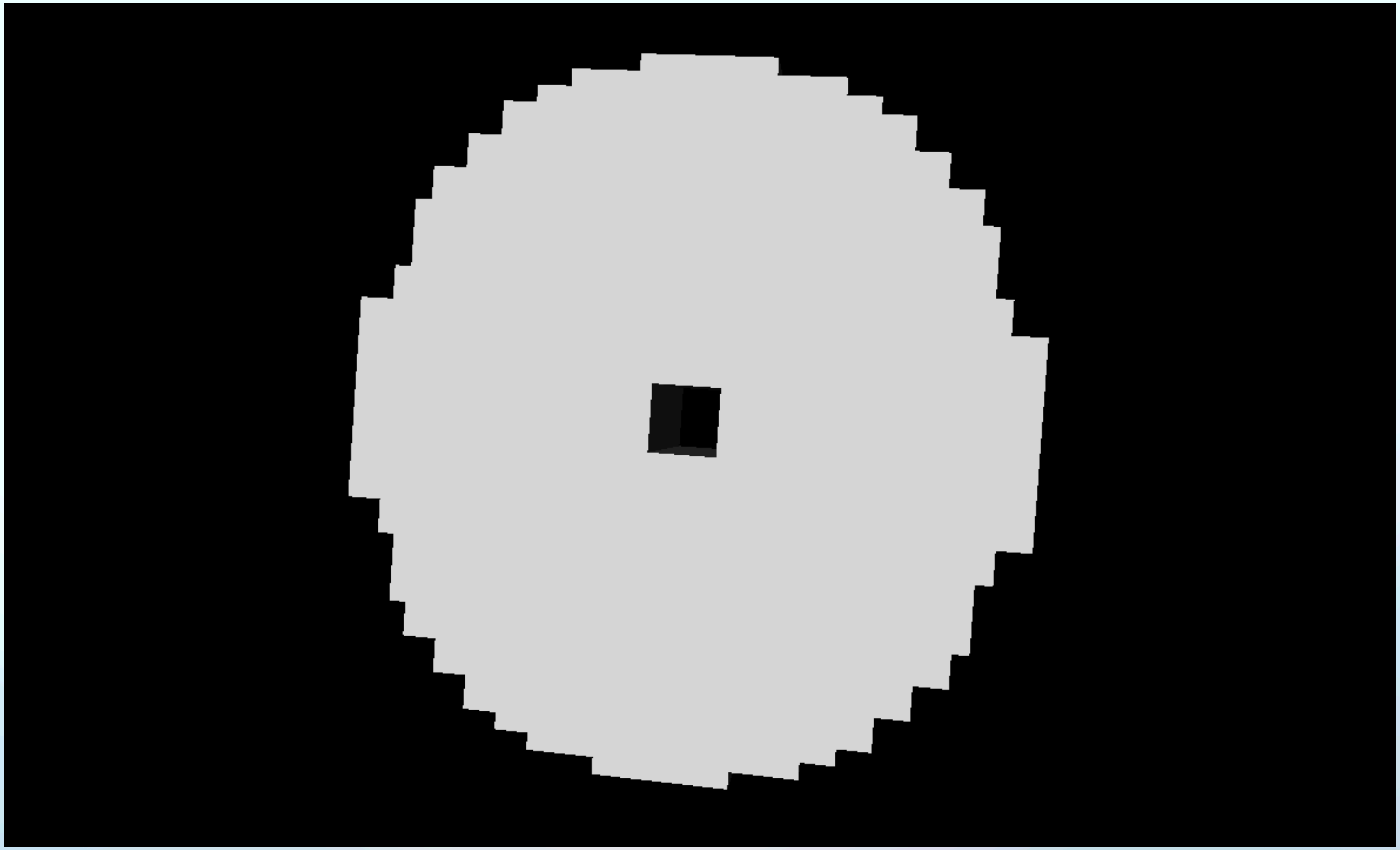


The end-cap part of the calorimeter in the frame, installed in RS. Each end-cap consists of 5136 cells. All dimensions are in millimeters.

Parameterizations for ECal end-cap part

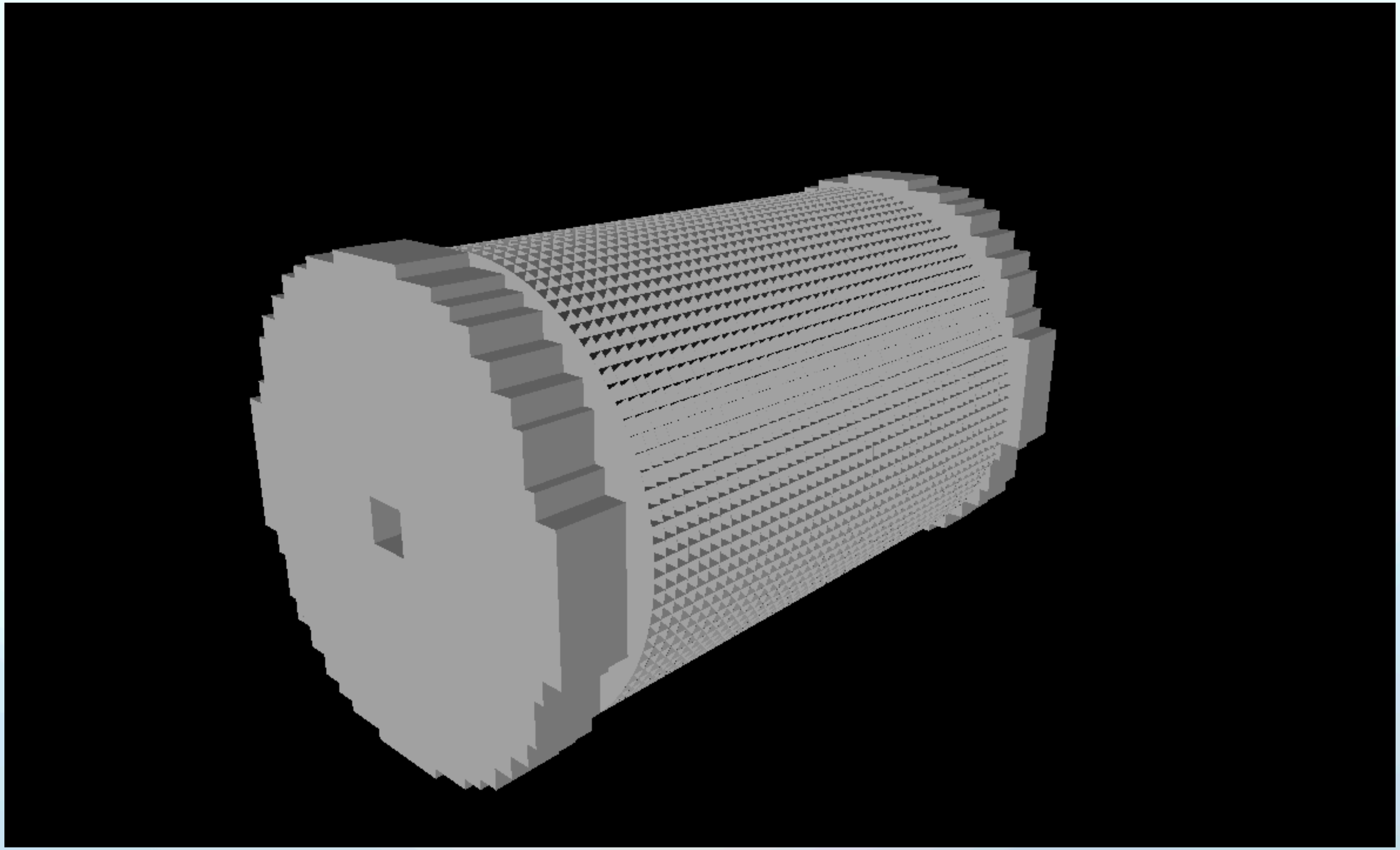
- GeoTrf::TranslateX3D = (2(i-num_of_mod)+1)*0.5*ecal_ec_module_width;
- GeoTrf::TranslateY3D = (41-2*i)*0.5*ecal_ec_module_height;
- GeoTrf::TranslateZ3D = Length_ecalbar + gap + ecal_ec_module_length;

Electromagnetic calorimeter end-cap part



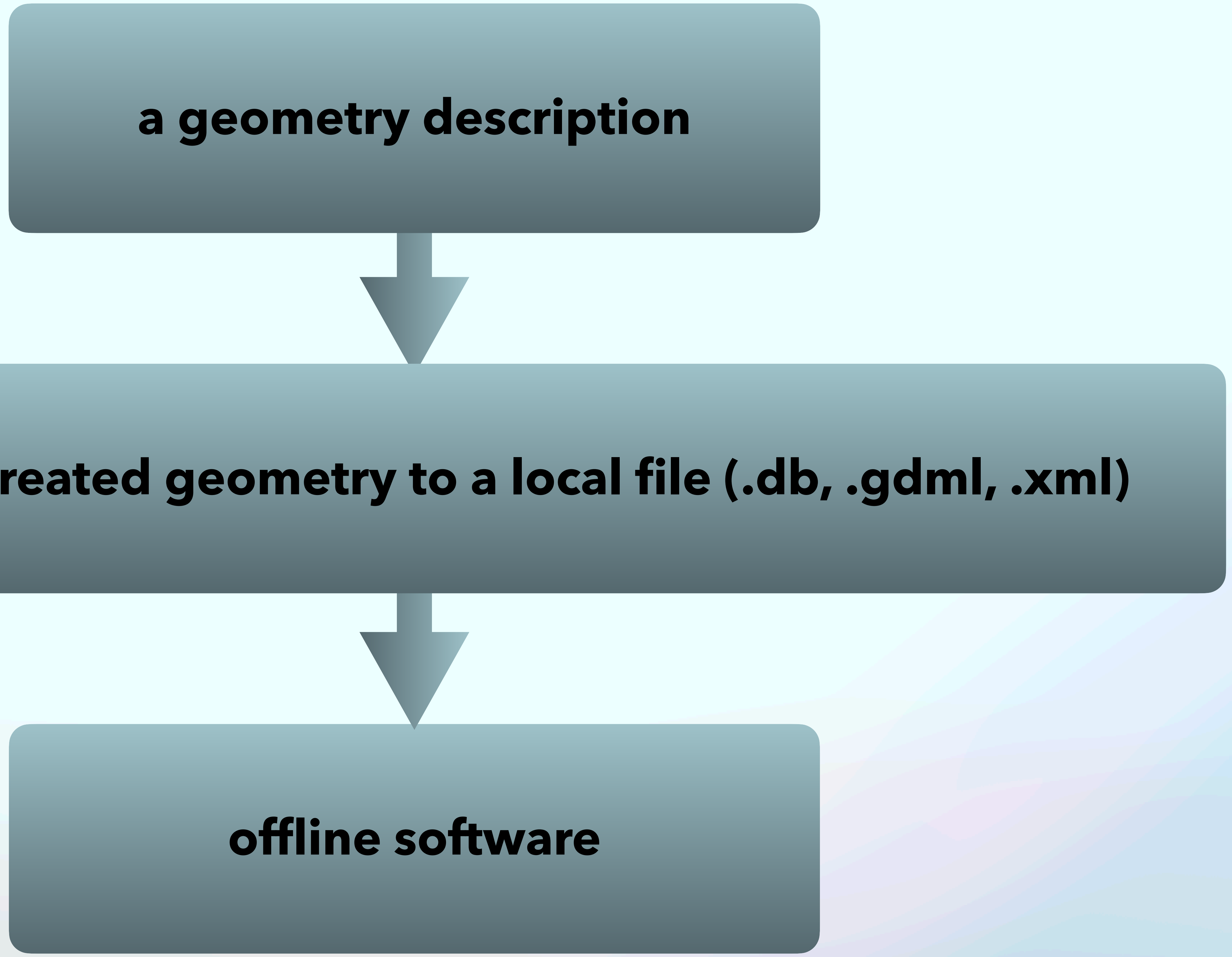
The end-cap part of the ECal.

Electromagnetic calorimeter



Sensitive volumes in Geant4 and GeoModel

The mechanism of interaction with GeoModel



a geometry description

writing the created geometry to a local file (.db, .gdml, .xml)

offline software

Coverting GeoModel data to a Geant4 geometry

```
static const std::string path = "/path to db file/spd.db";

GMDBManager* db = new GMDBManager(path);

GeoModelIO::ReadGeoModel readInGeo = GeoModelIO::ReadGeoModel(db);

GeoVPhysVol* world = readInGeo.buildGeoModel();

ExtParameterisedVolumeBuilder* builder = new ExtParameterisedVolumeBuilder("SPD");

G4LogicalVolume* g4World = builder->Build(world);

G4VPhysicalVolume* physWorld = new G4PVPlacement(0, G4ThreeVector(), g4World, "World", 0, false,
0, true);
```

Sensitive volume in Geant4

- The detector contains a large number of elements. Some of which are necessary to support the facility.

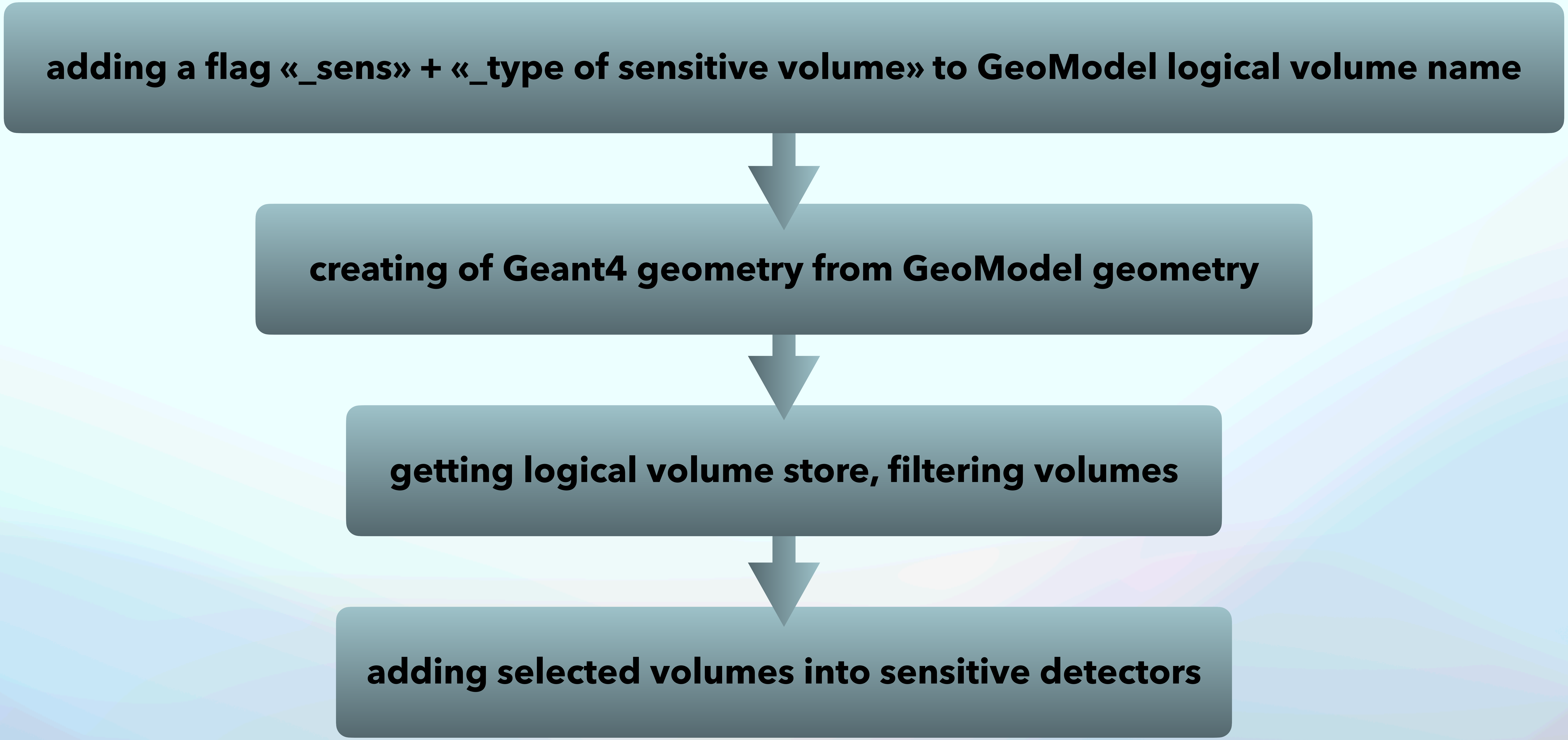
- Sensitive detector is used to distinguish detecting elements from others in Geant4. It is a way to declare a geometric element sensitive to the passage of particles.

The GeoModel doesn't know what sensitive volumes are

↓

Need to be developed

Sensitive detector in Geant4



adding a flag «_sens» + «_type of sensitive volume» to GeoModel logical volume name

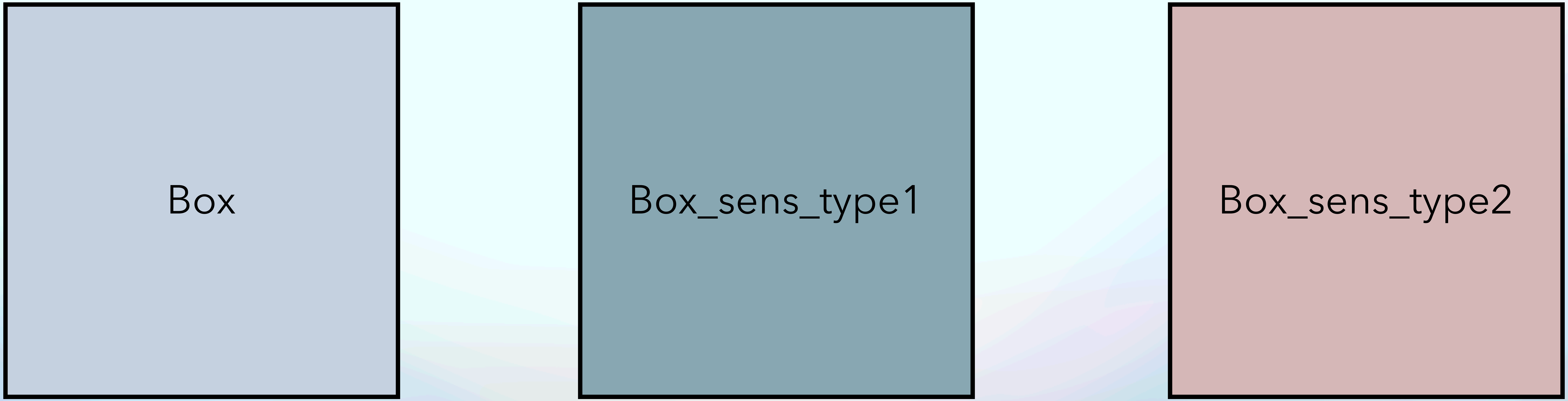
creating of Geant4 geometry from GeoModel geometry

getting logical volume store, filtering volumes

adding selected volumes into sensitive detectors

Sensitive detector in Geant4

Input database (originated in GeoModel):



Box

Box_sens_type1

Box_sens_type2

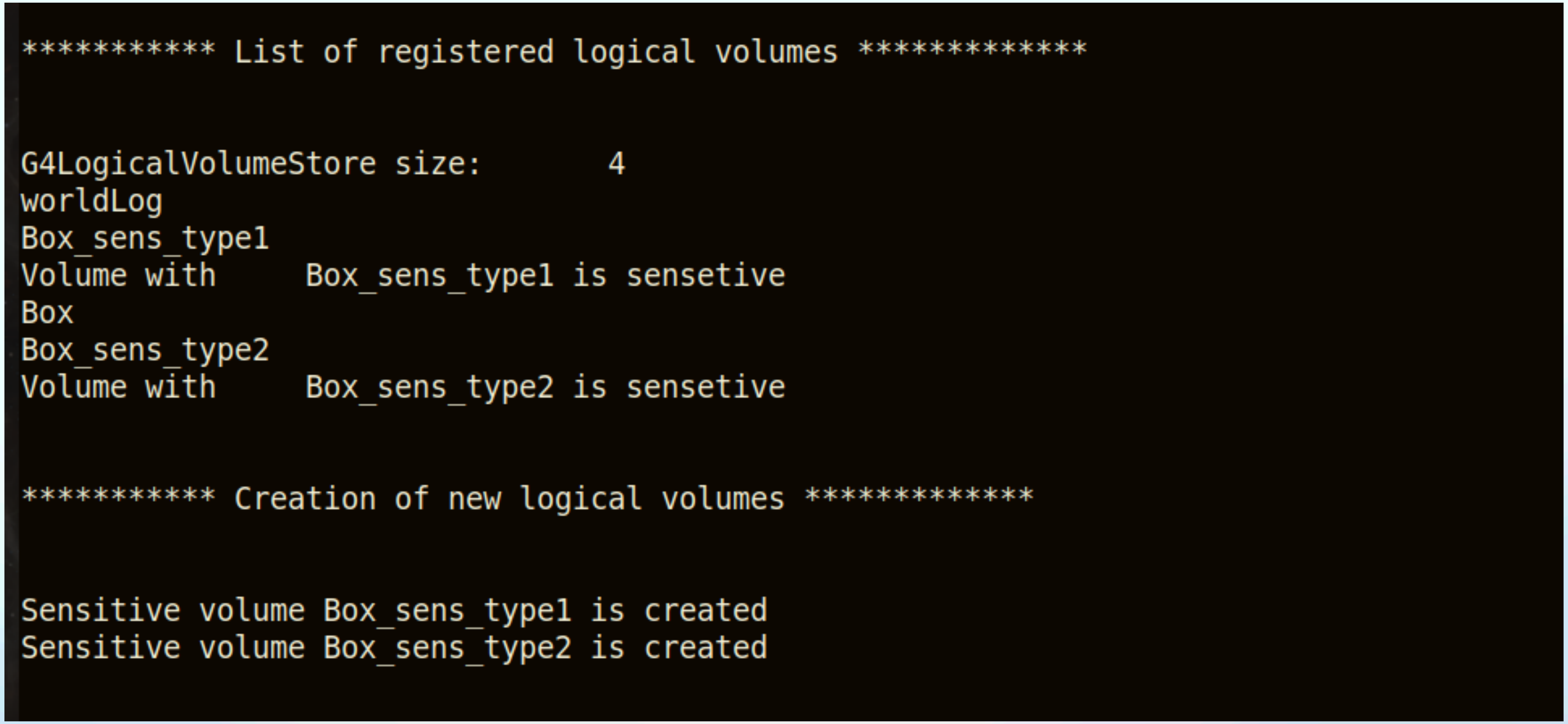
Sensitive detector in Geant4

```
std::cout  << "*********** List of registered logical volumes *************" << std::endl;
std::vector <G4String> SensLVnames;

std::vector<G4LogicalVolume*>* lvStore = G4LogicalVolumeStore::GetInstance();
std::size_t nlv=lvStore->size();
std::cout<<"G4LogicalVolumeStore size:\t "<<nlv<<std::endl; //info for debugging

for (std::size_t i=0;  i<nlv; i++)
{
    G4LogicalVolume* lv = (*lvStore)[i];
    G4String name = lv->GetName();
    std::cout << name << std::endl; //info for debugging

    if (name.contains(«_sens"))
    {
        SensLVnames.push_back(name);
         std::cout<< "Volume with \t"<< name << " is sensetive" << std::endl; //info for debugging
    }
}
```

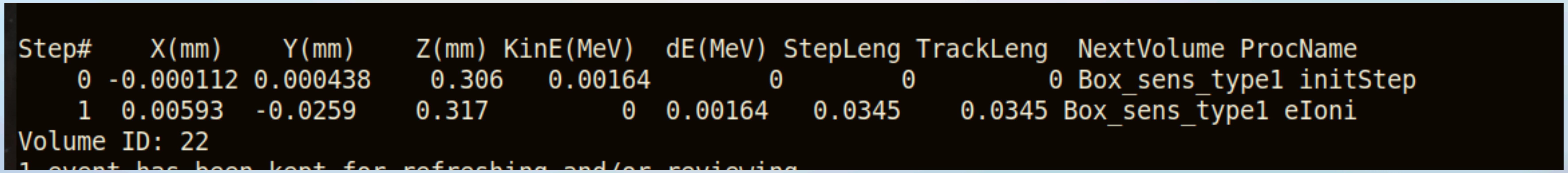
Sensitive detector in Geant4

```
for(int i=0; i<SensLVnames.size(); i++)
{
    G4String names = SensLVnames[i];

    if(names.contains("_type1"))
    {
        box_type1 = G4LogicalVolumeStore::GetInstance()->GetVolume(names);
        std::cout << «Sensitive volume\t» << names << «is created» << std::endl; //info for debugging
    }
    if(names.contains(«_type2"))
    {
        box_type2 = G4LogicalVolumeStore::GetInstance()->GetVolume(names);
        std::cout << «Sensitive volume\t» << names << «is created» << std::endl; //info for debugging
    }
}
```

Sensitive detector in Geant4



```
*********** List of registered logical volumes *************


G4LogicalVolumeStore size:      4
worldLog
Box_sens_type1
Volume with     Box_sens_type1 is sensetive
Box
Box_sens_type2
Volume with     Box_sens_type2 is sensetive


*********** Creation of new logical volumes *************


Sensitive volume Box_sens_type1 is created
Sensitive volume Box_sens_type2 is created
```



```
Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0 -0.000112 0.000438    0.306   0.00164        0        0         0 Box_sens_type1 initStep
    1  0.00593  -0.0259    0.317         0  0.00164   0.0345    0.0345 Box_sens_type1 eIoni
Volume ID: 22
1 event has been kept for refreshing and/or reviewing
```

Next steps

- Update geometry description in accordance with last version of TDR;
- Range system inner structure description;
- Development of methods that produce the necessary geometric parameters (for reconstruction).

Thank you for your attention!