Creating new detector in lcgeo/DD4hep

- compact.xml
- geo_driver.cpp

Shaojun Lu
shaojun.lu@desy.de

22.02.2016

compact.xml

```
<lccdd>
<define>
  <constant name="env_safety" value="0.001*mm"/>

  <constant name="Calo_dim_x" value="375*mm"/>
  <constant name="Calo_dim_y" value="375*mm"/>
  <constant name="Calo_dim_z" value="474*mm"/>

  <constant name="Calo_Layer_ncell_x" value="12"/>
  <constant name="Calo_Layer_ncell_y" value="12"/>
</define>

<readouts>
  <readout name="HBUCollection">
    <segmentation type="TiledLayerGridXY" grid_size_x="3" grid_size_y="3" offset_x="-Calo_dim_x/2.0"
offset_y="-Calo_dim_y/2.0" identifier_x="I" identifier_y="J" identifier_layer="K"/>
    <id>system:8,K:8,I:8,J:8</id>
  </readout>
</readouts>

<display>
  <vis name="HBUVis"         alpha="0.5"  r="0.7" g="0.7"  b="0.0" showDaughters="true"  visible="false"/>
  <vis name="TungstenVis"   alpha="1.0"  r="1.0" g="0.0"  b="0.0" showDaughters="true"  visible="true"/>
  <vis name="AirVis"           alpha="1.0"  r="0.0" g="1.0"  b="0.0" showDaughters="true"  visible="false"/>
  <vis name="SciVis"           alpha="1.0"  r="0.0" g="0.0"  b="1.0" showDaughters="true"  visible="true"/>
  <vis name="PCBVis"         alpha="1.0"  r="0.0" g="1.0"  b="0.0" showDaughters="true"  visible="true"/>
</display>
```

compact.xml

```
<detectors>

   <detector name="HBUTestBeam"  type="CaloPrototype_v01" vis="HBUVis" id="3"
readout="HBUCollection" insideTrackingVolume="false" >
    <type_flags type="1" />

    <envelope vis="BlueVis">
      <shape type="Box" dx="Calo_dim_x/2.0 + env_safety" dy="Calo_dim_y/2.0 + env_safety"
dz="Calo_dim_z/2.0 + 2.0*env_safety" material="Air" />
      <rotation x="0" y="0" z="0"/>
      <!--position x="0" y="0" z="0*mm"-->
    </envelope>

    <layer repeat="30" vis="HBUVis">
     <slice material = "TungstenDens24"    thickness = "10.0*mm"  vis="TungstenVis"                    />
     <slice material = "Air"               thickness = "1.0*mm"   vis="AirVis"                         />
     <slice material = "Cu"                thickness = "0.1*mm"                                        />
     <slice material = "PCB"               thickness = "0.7*mm"   vis="PCBVis"                         />
     <slice material = "G4_POLYSTYRENE"    thickness = "3.0*mm"   vis="SciVis"  sensitive = "yes"      />
     <slice material = "Air"               thickness = "1.0*mm"   vis="AirVis"                         />
    </layer>

   </detector>

 </detectors>
</lccdd>
```

geo_driver.cpp

- create and name a new geometry driver
  - and define the envelope
- access the parameters in compact.xml
- build the detail layers geometry

```
//====================================================================
//  DD4hep Geometry driver for Sampling Calo BOX prototype
//--------------------------------------------------------------------
//  S.Lu, DESY
//  $Id:  $
//====================================================================
#include "DD4hep/Printout.h"
#include "DD4hep/DetFactoryHelper.h"
#include "XML/Layering.h"
#include "XML/Utilities.h"
#include "DDRec/DetectorData.h"
#include "DDSegmentation/TiledLayerGridXY.h"
#include "LcgeoExceptions.h"

#include <iostream>
#include <vector>

using namespace std;
using namespace DD4hep;
using namespace DD4hep::Geometry;
using namespace lcgeo ;

// workaround for DD4hep v00-14 (and older)
#ifndef DD4HEP_VERSION_GE
#define DD4HEP_VERSION_GE(a,b) 0
#endif


static Ref_t create_detector(LCDD& lcdd, xml_h element, SensitiveDetector sens)  {

  xml_det_t   x_det       = element;
  string      det_name    = x_det.nameStr();
```

geo_driver.cpp

```
static Ref_t create_detector(LCDD& lcdd, xml_h element, SensitiveDetector sens)  {

  xml_det_t   x_det        = element;
  string       det_name     = x_det.nameStr();
  DetElement  sdet( det_name,x_det.id() );

  Layering    layering(x_det);

  // --- create an envelope volume and position it into the world --------

  Volume envelope = XML::createPlacedEnvelope( lcdd,  element , sdet ) ;

  XML::setDetectorTypeFlag( element, sdet ) ;

  if( lcdd.buildType() == BUILD_ENVELOPE ) return sdet ;

  //————————————————————————————————————————————————————————————

                 Here this block will be shown in next two slides:
                      Access the parameters in compact.xml
                        Build the detail layers geometry

  //————————————————————————————————————————————————————————————

    return sdet;
}

DECLARE_DETELEMENT(CaloPrototype_v01, create_detector)
```

geo_driver.cpp

```
//================================================================
//
// Read all the constant from compact.xml, user can update the value.
// Use them to build a calo box prototye.
//
//================================================================
```

```
double      Calo_half_x          = lcdd.constant<double>("Calo_dim_x")/2.0;
double      Calo_half_y          = lcdd.constant<double>("Calo_dim_y")/2.0;
double      Calo_half_z          = lcdd.constant<double>("Calo_dim_z")/2.0;
double      Calo_Layer_ncell_x   = lcdd.constant<int>("Calo_Layer_ncell_x");
double      Calo_Layer_ncell_y   = lcdd.constant<int>("Calo_Layer_ncell_y");
```

```
printout( DD4hep::DEBUG,  "building SamplingCaloBoxPrototype_v01",
         "Calo_half_x : %e    Calo_half_y: %e    Calo_half_z: %e ",
         Calo_half_x, Calo_half_y, Calo_half_z) ;
```

```
Readout  readout = sens.readout();
Segmentation seg = readout.segmentation();

std::vector<double> cellSizeVector = seg.segmentation()->cellDimensions(0);
double cell_sizeX      = cellSizeVector[0];
double cell_sizeY      = cellSizeVector[1];
```

geo_driver.cpp

```
//==================================================
// Chambers in the CaloBox
==================================================
    int layer_num = 0;
    int layerType   = 0;

    double layer_pos_z = - Calo_half_z;

    for (xml_coll_t c(x_det, _U(layer)); c; ++c) {
        xml_comp_t x_layer = c;
        int repeat = x_layer.repeat();              // Get number of times to repeat
        const Layer* lay = layering.layer(layer_num); // Get the layer from the layer
        double layer_thickness = lay->thickness();
        string layer_type_name   = _toString(layerType,"layerType%d");

        // Loop over repeats for this layer.
        for (int j = 0; j < repeat; j++) {
            string layer_name = _toString(layer_num, "layer%d");
            DetElement layer(layer_name, layer_num);

            // Layer box & volume
            Volume layer_vol(layer_type_name, Box(cal_hx, cal_hy, layer_thickness

            // Create the slices (sublayers) within the layer.
            double slice_pos_z = -(layer_thickness / 2);
            int slice_number = 0;

            for (xml_coll_t k(x_layer, _U(slice)); k; ++k) {
                xml_comp_t x_slice = k;
                string slice_name = _toString(slice_number, "slice%d");
                double slice_thickness = x_slice.thickness();
                Material slice_material = lcdd.material(x_slice.materialStr());

                slice_pos_z += slice_thickness / 2;
                // Slice volume & box
                Volume slice_vol(slice_name, Box(cal_hx, cal_hy, slice_thickness / 2),
slice_material);
```

```
                if (x_slice.isSensitive()) {
                    sens.setType("calorimeter");
                    slice_vol.setSensitiveDetector(sens);
                }


                // Set region, limitset, and vis.
                slice_vol.setAttributes(lcdd, x_slice.regionStr(), x_slice.limitsStr(), x_slic
                // slice PlacedVolume
                layer_vol.placeVolume(slice_vol, Position(0, 0, slice_pos_z));

                // Increment Z position for next slice.
                slice_pos_z += slice_thickness / 2;
                // Increment slice number.
                ++slice_number;
            }


            // Set region, limitset, and vis.
            layer_vol.setAttributes(lcdd, x_layer.regionStr(), x_layer.limitsStr(), x_layer

            // Layer position in Z within the stave.
            layer_pos_z += layer_thickness / 2;
            // Layer physical volume.
            PlacedVolume layer_phv = envelope.placeVolume(layer_vol, Position(0, 0
layer_pos_z));
            //layer_phv.addPhysVolID("layer", layer_num);
            layer_phv.addPhysVolID("K", layer_num);
            layer.setPlacement(layer_phv);

            // Increment the layer Z position.
            layer_pos_z += layer_thickness / 2;
            // Increment the layer number.
            ++layer_num;
        }

        ++layerType;
    }
```

Summary

- geometry driver, envelope (global cooperation interface)
  - details, realistic and complex built within envelope (sub-det. space holder)
- active layer segmentation (generic sensitive detector digitisation for LC layer-wise detectors)
  - if proved, the gaps between readout channels are not necessary in simulation within a layer, virtual channels could be generated with segmentation from mega layer for improving full simulation performance
- compact file (user interface, and sharing of the common geometry driver)
  - user detector size, number of layer, material, position, rotation

Summary

- two chances to create your model
  - pickup one existed geometry driver that fit your request, setup your values and material in the compact file
  - or, create a geometry driver by user itself to fit the user requirement
- demonstrated how to create a geometry driver and compact XML
  - this example is a simple box with multi-layers
    - envelope, segmentation, create layers with envelope, and setup compact file
    - run a test with particle gun
    - check your result