```
// ==================== DetectorConstruction.cc Class ========================
//
// Developed by Bryan V. Egner, Darren E. Holland, and Julie V. Logan
// Modified by Darren Holland 2020-11-02
// ===========================================================================
//
// This file creates creates the model geometry, materials, and also the applies
// the functional detector capabilities to utilize built-in scorers
// ===========================================================================
//
#include "B4DetectorConstruction.hh"
// Geant4 Material Classes:
#include "G4Material.hh"
#include "G4NistManager.hh"
// Geant4 Geometric Shapes:
#include "G4Box.hh"
#include "G4UnionSolid.hh"
#include "G4Sphere.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4PVReplica.hh"
#include "G4Tubs.hh"
#include "G4Orb.hh"
#include "G4SubtractionSolid.hh"
#include "G4GeometryManager.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4SolidStore.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"
#include "G4UnitsTable.hh"
#include <iterator>
#include <vector>
// CADMESH //
#include "G4String.hh"
#include "G4TessellatedSolid.hh"
#include "G4TriangularFacet.hh"
#include "G4QuadrangularFacet.hh"
#include "G4Tet.hh"
#include "G4UIcommand.hh"
// GEANT4 //
#include "globals.hh"
#include "G4ThreeVector.hh"
#include "G4Transform3D.hh"
#include "G4AssemblyVolume.hh"
#include "Settings.hh"
#include "G4SDManager.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4VPrimitiveScorer.hh"
#include "G4PSEnergyDeposit.hh"

#include <iostream>
#include <fstream>

using namespace std;
// Create instance and variable for checking geometry overlaps
B4DetectorConstruction::B4DetectorConstruction()
 : G4VUserDetectorConstruction(),
   fCheckOverlaps(true)
{}

// Destroy instance
```

```
B4DetectorConstruction::~B4DetectorConstruction()
{}

// Create volumes
G4VPhysicalVolume* B4DetectorConstruction::Construct()
{
  return DefineVolumes();
}

// ============================================================================
//
// Define volumes
// ============================================================================
//
G4VPhysicalVolume* B4DetectorConstruction::DefineVolumes()
{
  // Initializes Variables/Constant Values (type):
  G4double z, a, fractionmass, density;
  G4String name, symbol;
  G4int ncomponents, natoms;
  // Initiate Nist Manager for Materials:
  G4NistManager* nistManager = G4NistManager::Instance();

  //
// ============================================================================ //
  // Get materials
  //
// ============================================================================ //
  // Environmental Materials:
  // Air (0.00120479 g/cc):
  G4Material* air_mat = nistManager->FindOrBuildMaterial("G4_AIR");

  // Mask Materials:
  // Acrylic (1.19):
  G4Material* plexi_mat = nistManager->FindOrBuildMaterial("G4_PLEXIGLASS");

  // Create materials for organic Elements: Carbon, Hydrogen, Oxygen:
  // Element properties
  a = 1.00794*g/mole;
  G4Element* ele_H = new G4Element("Hydrogen",symbol="H",z=1.,a);
  G4Material* G4_H = nistManager->FindOrBuildMaterial("G4_H");

  a = 15.999*g/mole;
  G4Element* ele_O = new G4Element("Oxygen",symbol="O",z=8.,a);
  G4Material* G4_O = nistManager->FindOrBuildMaterial("G4_O");

  a = 12.0107*g/mole;
  G4Element* ele_C = new G4Element("Carbon",symbol="C",z=6.,a);
  G4Material* G4_C = nistManager->FindOrBuildMaterial("G4_C");

  // Create materials for elements
  G4Material* G4_Mg = nistManager->FindOrBuildMaterial("G4_Mg");
  G4Material* G4_Al = nistManager->FindOrBuildMaterial("G4_Al");
  G4Material* G4_Si = nistManager->FindOrBuildMaterial("G4_Si");
  G4Material* G4_Ti = nistManager->FindOrBuildMaterial("G4_Ti");
  G4Material* G4_Cr = nistManager->FindOrBuildMaterial("G4_Cr");
  G4Material* G4_Mn = nistManager->FindOrBuildMaterial("G4_Mn");
  G4Material* G4_Fe = nistManager->FindOrBuildMaterial("G4_Fe");
  G4Material* G4_Cu = nistManager->FindOrBuildMaterial("G4_Cu");
  G4Material* G4_Zn = nistManager->FindOrBuildMaterial("G4_Zn");

  // Detector Materials:
  // NaI Scintillator (g/cc):
  G4Material* NaI_mat = nistManager->FindOrBuildMaterial("G4_SODIUM_IODIDE");
```

```
  // Aluminum Alloy Casing 6061 (2.7 g/cc): (PNNL-15870 Compendium)
  G4Material* AlAlloy6061 = new G4Material(name="AlAlloy6061", density=
2.7*g/cm3, ncomponents=9);
        AlAlloy6061->AddMaterial(G4_Mg, fractionmass=0.01);
        AlAlloy6061->AddMaterial(G4_Al, fractionmass=0.97207);
        AlAlloy6061->AddMaterial(G4_Si, fractionmass=0.006);
        AlAlloy6061->AddMaterial(G4_Ti, fractionmass=0.00088);
        AlAlloy6061->AddMaterial(G4_Cr, fractionmass=0.001950);
        AlAlloy6061->AddMaterial(G4_Mn, fractionmass=0.000880);
        AlAlloy6061->AddMaterial(G4_Fe, fractionmass=0.004090);
        AlAlloy6061->AddMaterial(G4_Cu, fractionmass=0.002750);
        AlAlloy6061->AddMaterial(G4_Zn, fractionmass=0.00146);

  // PMT Materials:
  // Pseudo PMT Material (0.39 g/cc): (assumed to be Aluminum with a lighter
density)
  density = 0.39*g/cm3;
  G4Material* PMT_Mat = new G4Material("PMT_Mat",density,ncomponents=1);
        PMT_Mat->AddMaterial(G4_Al, fractionmass=1.0);
  // Borosilicate Glass (2.23 g/cc):
  G4Material* BoroSi_Glass = nistManager->FindOrBuildMaterial("G4_Pyrex_Glass");

  //
========================================================================== //
  // Define Geometry/Fill with Materials
  //
========================================================================== //
  // World:
  // The world is a sphere with radius that is 1 cm larger than the source
distance
  G4double worldRadius = (Settings::SourceDist+1)*cm; // Sphere Radius
  G4VSolid* worldS
    = new G4Orb("World", worldRadius); // Sphere

  G4LogicalVolume* worldLV
    = new G4LogicalVolume(
                worldS,           // its solid
                air_mat,          // its material
                "World");         // its name

  G4VPhysicalVolume* worldPV
    = new G4PVPlacement(
                0,                // no rotation
                G4ThreeVector(),  // at (0,0,0)
                worldLV,          // its logical volume

                "World",          // its name
                0,                // its mother volume
                false,            // no boolean operation
                0,                // copy number
                fCheckOverlaps);  // checking overlaps
  worldLV->SetVisAttributes(G4VisAttributes::Invisible);  // Makes world
invisible in visualization

  // The mask geometry
  //MASK: READ IN THE NODES TO 3 VECTORS (x,y,z)
  ifstream infile;
  infile.open(Settings::fname_nodes);
  std::vector<double> xVals;
  std::vector<double> yVals;
  std::vector<double> zVals;
  string line;
  while(getline(infile,line)) // To get you all the lines.
        {
```

```
        istringstream buf(line);
        istream_iterator<std::string> beg(buf), end;
        vector<std::string> tokens(beg, end);
        // remove the comma
        tokens[0].pop_back();
        tokens[1].pop_back();
        tokens[2].pop_back();

        xVals.push_back(stod (tokens[1].c_str()));
        yVals.push_back(stod (tokens[2].c_str()));
        zVals.push_back(stod (tokens[3].c_str()));
        }
  infile.close();

  // Total mass from all tetrahedrons
  double totMass = 0.;

  // Create volume
  G4AssemblyVolume * assembly = new G4AssemblyVolume();
  G4Transform3D assembly_transform = G4Translate3D();

  //READ IN THE ELEMENTS TO 8 VECTORS (1,2,3,4,5,6,7,8)
  ifstream infile2;
  infile2.open(Settings::fname_ele);
  std::vector<int> onePoint;
  std::vector<int> twoPoint;
  std::vector<int> threePoint;
  std::vector<int> fourPoint;
  std::vector<int> fivePoint;
  std::vector<int> sixPoint;
  std::vector<int> sevenPoint;
  std::vector<int> eightPoint;
  while(getline(infile2,line)) // To get you all the lines.
        {
        istringstream buf(line);
        istream_iterator<std::string> beg(buf), end;
        vector<std::string> tokens(beg, end);
        //remove the comma
        tokens[0].pop_back();
        tokens[1].pop_back();
        tokens[2].pop_back();
        tokens[3].pop_back();
        tokens[4].pop_back();
        tokens[5].pop_back();
        tokens[6].pop_back();
        tokens[7].pop_back();

        onePoint.push_back(stoi (tokens[1].c_str()));
        twoPoint.push_back(stoi (tokens[2].c_str()));
        threePoint.push_back(stoi (tokens[3].c_str()));
        fourPoint.push_back(stoi (tokens[4].c_str()));
        fivePoint.push_back(stoi (tokens[5].c_str()));
        sixPoint.push_back(stoi (tokens[6].c_str()));
        sevenPoint.push_back(stoi (tokens[7].c_str()));
        eightPoint.push_back(stoi (tokens[8].c_str()));
        }
  infile2.close();
  // Set visualization options
  G4VisAttributes* visAttributes2 = new G4VisAttributes(G4Colour(0.0, 1.0,
0.0)); //green
  G4VisAttributes* visAttributes3 = new G4VisAttributes(G4Colour(0.0, 1.0,
0.0)); //green
  G4VisAttributes* visAttributes4 = new G4VisAttributes(G4Colour(0.0, 1.0,
0.0)); //green
```

```
  G4VisAttributes* visAttributes5 = new G4VisAttributes(G4Colour(0.0, 1.0,
0.0)); //green
  G4VisAttributes* visAttributes6 = new G4VisAttributes(G4Colour(0.0, 1.0,
0.0)); //green

  G4RotationMatrix * element_rotation = new G4RotationMatrix();
  G4ThreeVector element_position = G4ThreeVector();

  // Create tesselated solid according to the read in lists and add to assembly
  for (int i=0; i<onePoint.size(); i++) {
        // Get points on tesselated solid
        G4ThreeVector p1 = G4ThreeVector(xVals[onePoint[i]-1]*cm,
yVals[onePoint[i]-1]*cm, zVals[onePoint[i]-1]*cm);
        G4ThreeVector p2 = G4ThreeVector(xVals[twoPoint[i]-1]*cm,
yVals[twoPoint[i]-1]*cm, zVals[twoPoint[i]-1]*cm);
        G4ThreeVector p3 = G4ThreeVector(xVals[threePoint[i]-1]*cm,
yVals[threePoint[i]-1]*cm, zVals[threePoint[i]-1]*cm);
        G4ThreeVector p4 = G4ThreeVector(xVals[fourPoint[i]-1]*cm,
yVals[fourPoint[i]-1]*cm, zVals[fourPoint[i]-1]*cm);
        G4ThreeVector p5 = G4ThreeVector(xVals[fivePoint[i]-1]*cm,
yVals[fivePoint[i]-1]*cm, zVals[fivePoint[i]-1]*cm);
        G4ThreeVector p6 = G4ThreeVector(xVals[sixPoint[i]-1]*cm,
yVals[sixPoint[i]-1]*cm, zVals[sixPoint[i]-1]*cm);
        G4ThreeVector p7 = G4ThreeVector(xVals[sevenPoint[i]-1]*cm,
yVals[sevenPoint[i]-1]*cm, zVals[sevenPoint[i]-1]*cm);
        G4ThreeVector p8 = G4ThreeVector(xVals[eightPoint[i]-1]*cm,
yVals[eightPoint[i]-1]*cm, zVals[eightPoint[i]-1]*cm);

        // Create unique name
        G4String tess_name = G4String("Mask_tess_") +
G4UIcommand::ConvertToString(i);

        // First declare a tessellated solid
        auto solidTarget = new G4TessellatedSolid(tess_name +
G4String("_solid"));
        // Define the facets which form the solid
        G4QuadrangularFacet *facet1 = new G4QuadrangularFacet
(p4,p3,p2,p1,ABSOLUTE);
        G4QuadrangularFacet *facet2 = new G4QuadrangularFacet
(p5,p6,p7,p8,ABSOLUTE);
        G4QuadrangularFacet *facet3 = new G4QuadrangularFacet
(p8,p7,p3,p4,ABSOLUTE);
        G4QuadrangularFacet *facet4 = new G4QuadrangularFacet
(p1,p5,p8,p4,ABSOLUTE);
        G4QuadrangularFacet *facet5 = new G4QuadrangularFacet
(p1,p2,p6,p5,ABSOLUTE);
        G4QuadrangularFacet *facet6 = new G4QuadrangularFacet
(p6,p2,p3,p7,ABSOLUTE);
        // Now add the facets to the solid
        solidTarget->AddFacet((G4VFacet*) facet1);
        solidTarget->AddFacet((G4VFacet*) facet2);
        solidTarget->AddFacet((G4VFacet*) facet3);
        solidTarget->AddFacet((G4VFacet*) facet4);
        solidTarget->AddFacet((G4VFacet*) facet5);
        solidTarget->AddFacet((G4VFacet*) facet6);

        // Finally declare the solid is complete and create logical volume
        solidTarget->SetSolidClosed(true);
        G4LogicalVolume * tess_logical = new G4LogicalVolume(solidTarget,
        // Set mask material
          plexi_mat, tess_name + G4String("_logical"), 0, 0, 0);
        // Add mass and place in volume
        totMass = totMass + tess_logical->GetMass();
        assembly->AddPlacedVolume(tess_logical, assembly_transform);
```

```
        // Set visulization options
        visAttributes2->SetVisibility(true);
        visAttributes3->SetVisibility(true);
        visAttributes4->SetVisibility(true);
        visAttributes5->SetVisibility(true);
        visAttributes6->SetVisibility(true);
        // Color mask
        if (i % 2 == 0) {
             tess_logical->SetVisAttributes(visAttributes2); }
        if (i % 2 == 1) {
             tess_logical->SetVisAttributes(visAttributes3); }
        }
  // Add assembly to world
  assembly->MakeImprint(worldLV, assembly_transform, 0, 0);

  // Print out and save the total mask mass
  G4cout << "***** TO CHECK ******* the mass of the mask was: " <<
G4BestUnit(totMass, "Mass") << G4endl;
  string fname_mass=Settings::fname_out + "mass.txt";
  ofstream ofile;
  ofile.open (fname_mass, ios::out);     // ascii file
  ofile << "Mass " << G4BestUnit(totMass, "Mass");
  ofile.close();
  // END OF GENERATING MASK GEOMETRY

  //
============================================================================ //
  // Add other system components
  //
============================================================================ //
  // For this model there are currently 13 different components used, more may
be added:
  //            1.) NaI detector
  //            2.) Cell Glass Window
  //            3.) End-Cap Top
  //            4.) End Cap Sides
  //            5.) Upper End-Cap Flange
  //            6.) Lower End-Cap Flange
  //            7.) PMT
  //            8.) Upper PMT Holder
  //            9.) Lower PMT Holder
  //            10.) PMT Holder Back
  //            11.) Detector Lid
  //            12.) Detector Tube
  //            13.) Mask Holder

  // Detector Geometry
  // 1.) Cylindrical Scintillator:
  G4Tubs* detectorNaI_Solid
    = new G4Tubs("Detect",             // its name
                 0.*cm,                // inner radius (cylinder)
                 Settings::DetRad*cm,  // outer radius
                 Settings::DetHeight*cm,    // half height
                 0.*deg,               // start angle
                 360.*deg);            // end angle
  G4LogicalVolume* detectorNaI_Logical
    = new G4LogicalVolume(
                 detectorNaI_Solid,    // its solid
                 NaI_mat,              // its material
                 "DetectLV");          // its name
  G4VPhysicalVolume* detectorNaI_Physical
    = new G4PVPlacement(
                 0,                     // no rotation
                 G4ThreeVector(0.*cm,0.*cm,0.*cm),  // center
```

```
                detectorNaI_Logical,   // its logical volume

                "Detect",              // its name
                worldLV,               // its mother volume
                false,                 // no boolean operation
                0,                     // copy number
                fCheckOverlaps);       // checking overlaps
  // Detector visualization
  G4VisAttributes* visAttributes9 = new G4VisAttributes(G4Colour(0.0, 0.0,
1.0)); // Blue
  detectorNaI_Logical->SetVisAttributes(visAttributes9);

  // 2.) Glass Cell Cap: Borofloat Window --->Borosilicate glass
  G4Tubs* CellWindow_Solid
    = new G4Tubs("CellWindow",                      // its name
                0.*cm,                              // inner radius (cylinder)
                1.6002*cm,                          // outer radius
                0.2413*cm,                          // half height
                0.*deg,                             // start angle
                360.*deg);                          // end angle
  G4LogicalVolume* CellWindow_Logical
    = new G4LogicalVolume(
                CellWindow_Solid,                   // its solid
                BoroSi_Glass,                       // its material
                "CellWindow");                      // its name
  G4VPhysicalVolume* CellWindow_Physical
    = new G4PVPlacement(
                0,                                  // no rotation
                G4ThreeVector(0.*cm,0.*cm,-1.5113*cm),  // center
                CellWindow_Logical,                 // its logical volume
                "CellWindow",                       // its name
                worldLV,                            // its mother volume
                false,                              // no boolean operation
                0,                                  // copy number
                fCheckOverlaps);                    // checking overlaps
  // Glass cell cap visualization
  G4VisAttributes* visAttributes10 = new G4VisAttributes(G4Colour(1.0, 0.0,
1.0)); // Magenta
  CellWindow_Logical->SetVisAttributes(visAttributes10);

  // 3.) Al End-Cap Face: Aluminum
  G4Tubs* CellAlEndCapFace_Solid
    = new G4Tubs("CellAlEndCapFace",                // its name
                0.*cm,                              // inner radius (cylinder)
                1.4224*cm,                          // outer radius
                0.0762*cm,                          // half height
                0.*deg,                             // start angle
                360.*deg);                          // end angle
  G4LogicalVolume* CellAlEndCapFace_Logical
    = new G4LogicalVolume(
                CellAlEndCapFace_Solid,             // its solid
                AlAlloy6061,                        // its material
                "CellAlEndCapFace");                // its name
  G4VPhysicalVolume* CellAlEndCapFace_Physical
    = new G4PVPlacement(
                0,                                  // no rotation
                G4ThreeVector(0.*cm,0.*cm,1.3462*cm),// center
                CellAlEndCapFace_Logical,           // its logical volume
                "CellAlEndCapFace",                 // its name
                worldLV,                            // its mother volume
                false,                              // no boolean operation
                0,                                  // copy number
                fCheckOverlaps);                    // checking overlaps
  // Al end-cap face visualization
```

```
  G4VisAttributes* visAttributes11 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  CellAlEndCapFace_Logical->SetVisAttributes(visAttributes11);

  // 4.) Al End-Cap Sides: Aluminum
  G4Tubs* CellAlEndCapSides_Solid
    = new G4Tubs("CellAlEndCapSides",               // its name
                 1.27*cm,                           // inner radius (cylinder)
                 1.4224*cm,                         // outer radius
                 1.27*cm,                           // half height
                 0.*deg,                            // start angle
                 360.*deg);                         // end angle
  G4LogicalVolume* CellAlEndCapSides_Logical
    = new G4LogicalVolume(
                 CellAlEndCapSides_Solid,           // its solid
                 AlAlloy6061,                       // its material
                 "CellAlEndCapSides");              // its name
  G4VPhysicalVolume* CellAlEndCapSides_Physical
    = new G4PVPlacement(
                 0,                                 // no rotation
                 G4ThreeVector(0.*cm,0.*cm,0.*cm),  // center
                 CellAlEndCapSides_Logical,         // its logical volume
                 "CellAlEndCapSides",               // its name
                 worldLV,                           // its mother volume
                 false,                             // no boolean operation
                 0,                                 // copy number
                 fCheckOverlaps);                   // checking overlaps
  // Al end-cap sides visualization
  G4VisAttributes* visAttributes12 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // white
  CellAlEndCapSides_Logical->SetVisAttributes(visAttributes12);

  // 5.) Al End-Cap Flange Top: Aluminum
  G4Tubs* CellAlEndCapFlangeTop_Solid
    = new G4Tubs("CellAlEndCapFlangeTop",           // its name
                 1.4224*cm,                         // inner radius (cylinder)
                 1.9050*cm,                         // outer radius
                 0.4953*cm,                         // half height
                 0.*deg,                            // start angle
                 360.*deg);                         // end angle
  G4LogicalVolume* CellAlEndCapFlangeTop_Logical
    = new G4LogicalVolume(
                 CellAlEndCapFlangeTop_Solid,       // its solid
                 AlAlloy6061,                       // its material
                 "CellAlEndCapFlangeTop");          // its name
  G4VPhysicalVolume* CellAlEndCapFlangeTop_Physical
    = new G4PVPlacement(
                 0,                                 // no rotation
                 G4ThreeVector(0.*cm,0.*cm,-0.7747*cm),// center
                 CellAlEndCapFlangeTop_Logical,     // its logical volume
                 "CellAlEndCapFlangeTop",           // its name
                 worldLV,                           // its mother volume
                 false,                             // no boolean operation
                 0,                                 // copy number
                 fCheckOverlaps);                   // checking overlaps
  // Al end-cap flange top visualization
  G4VisAttributes* visAttributes13 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // white
  CellAlEndCapFlangeTop_Logical->SetVisAttributes(visAttributes13);

  // 6.) Al End-Cap Flange Bottom: Aluminum
  G4Tubs* CellAlEndCapFlangeBottom_Solid
    = new G4Tubs("CellAlEndCapFlangeBottom",        // its name
                 1.6002*cm,                         // inner radius (cylinder)
```

```
                1.9050*cm,                          // outer radius
                0.2413*cm,                          // half height
                0.*deg,                             // start angle
                360.*deg);                          // end angle
  G4LogicalVolume* CellAlEndCapFlangeBottom_Logical
    = new G4LogicalVolume(
                CellAlEndCapFlangeBottom_Solid,    // its solid
                AlAlloy6061,                        // its material
                "CellAlEndCapFlangeBottom");       // its name
  G4VPhysicalVolume* CellAlEndCapFlangeBottom_Physical
    = new G4PVPlacement(
                0,                                  // no rotation
                G4ThreeVector(0.*cm,0.*cm,-1.5113*cm),// center
                CellAlEndCapFlangeBottom_Logical,  // its logical volume
                "CellAlEndCapFlangeBottom",        // its name
                worldLV,                            // its mother volume
                false,                              // no boolean operation
                0,                                  // copy number
                fCheckOverlaps);                    // checking overlaps
  // Al end-cap flange bottom visualization
  G4VisAttributes* visAttributes14 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // white
  CellAlEndCapFlangeBottom_Logical->SetVisAttributes(visAttributes14);

  // 7.) PMT: Psuedo material --> Al w/ low density (0.39 g/cc)
  G4Tubs* PMT_Solid
    = new G4Tubs("PMT",               // its name
                0.*cm,                // inner radius (cylinder)
                1.4250*cm,            // outer radius
                4.6*cm,               // half height
                0.*deg,               // start angle
                360.*deg);            // end angle
  G4LogicalVolume* PMT_Logical
    = new G4LogicalVolume(
                PMT_Solid,            // its solid
                PMT_Mat,              // its material
                "PMT");               // its name
  G4VPhysicalVolume* PMT_Physical
    = new G4PVPlacement(
                0,                    // no rotation
                G4ThreeVector(0.*cm,0.*cm,-6.3526*cm),// center
                PMT_Logical,          // its logical volume
                "PMT",                // its name
                worldLV,              // its mother volume
                false,                // no boolean operation
                0,                    // copy number
                fCheckOverlaps);      // checking overlaps
  // PMT visualization
  G4VisAttributes* visAttributes15 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // white
  PMT_Logical->SetVisAttributes(visAttributes15);

  // 8.) PMT Holder Upper Part
  G4Tubs* PMTHolderUpperPart_Solid_Sleeve
    = new G4Tubs("PMTHolderUpperPart",        // its name
                ( 1.7399)*cm,                 // innerR
                (1.905)*cm,                   // outer radius
                (0.635)*cm,                   // half height
                0.*deg,                       // start angle
                360.*deg);                    // end angle

  G4LogicalVolume* PMTHolderUpperPart_Logical_Sleeve
    = new G4LogicalVolume(
                PMTHolderUpperPart_Solid_Sleeve,   // its solid
```

```
                AlAlloy6061,                // its material
                "PMTHolderUpperPart");      // its name

  G4VPhysicalVolume* PMTHolderUpperPart_Physical_Sleeve
    = new G4PVPlacement(
                0,                          // no rotation
                G4ThreeVector(0.*cm,0.*cm,(-2.3876)*cm),  // center
                PMTHolderUpperPart_Logical_Sleeve,        // its logical volume
                "PMTHolderUpperPart",       // its name
                worldLV,                    // its mother volume
                false,                      // no boolean operation
                0,                          // copy number
                fCheckOverlaps);            // checking overlaps
  // PMT holder upper part visualization
  G4VisAttributes* visAttributes16 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  PMTHolderUpperPart_Logical_Sleeve->SetVisAttributes(visAttributes16);

  // 9.) PMT Holder Lower Part
  G4Tubs* PMTHolderLowerPart_Solid_Sleeve
    = new G4Tubs("PMTHolderLowerPart",      // its name
                ( 1.7399)*cm,               // innerR
                (1.9844)*cm,                // outer radius
                (7.0765)*cm,                // half height
                0.*deg,                     // start angle
                360.*deg);                  // end angle

  G4LogicalVolume* PMTHolderLowerPart_Logical_Sleeve
    = new G4LogicalVolume(
                PMTHolderLowerPart_Solid_Sleeve,   // its solid
                AlAlloy6061,                // its material
                "PMTHolderLowerPart");      // its name

  G4VPhysicalVolume* PMTHolderLowerPart_Physical_Sleeve
    = new G4PVPlacement(
                0,                          // no rotation
                G4ThreeVector(0.*cm,0.*cm,(-10.0991)*cm),  // center
                PMTHolderLowerPart_Logical_Sleeve,         // its logical
volume
                "PMTHolderLowerPart",       // its name
                worldLV,                    // its mother volume
                false,                      // no boolean operation
                0,                          // copy number
                fCheckOverlaps);            // checking overlaps
  // PMT holder lower part visualization
  G4VisAttributes* visAttributes17 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  PMTHolderLowerPart_Logical_Sleeve->SetVisAttributes(visAttributes17);

  // 10.) PMT Holder Bottom Part
  G4Tubs* PMTHolderBottomPart_Solid_Sleeve
    = new G4Tubs("PMTHolderBottomPart",     // its name
                (0.)*cm,                    // innerR
                (1.7399)*cm,                // outer radius
                (0.0762)*cm,                // half height
                0.*deg,                     // start angle
                360.*deg);                  // end angle

  G4LogicalVolume* PMTHolderBottomPart_Logical_Sleeve
    = new G4LogicalVolume(
                PMTHolderBottomPart_Solid_Sleeve,   // its solid
                AlAlloy6061,                // its material
                "PMTHolderBottomPart");     // its name
```

```
  G4VPhysicalVolume* PMTHolderBottomPart_Physical_Sleeve
    = new G4PVPlacement(
                0,                          // no rotation
                G4ThreeVector(0.*cm,0.*cm,(-17.0994)*cm),   // center
                PMTHolderBottomPart_Logical_Sleeve,         // its logical
volume
                "PMTHolderBottomPart",      // its name
                worldLV,                    // its mother volume
                false,                      // no boolean operation
                0,                          // copy number
                fCheckOverlaps);            // checking overlaps
  // PMT holder bottom part visualization
  G4VisAttributes* visAttributes18 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  PMTHolderBottomPart_Logical_Sleeve->SetVisAttributes(visAttributes18);

  // 11.) Detector Lid
  G4Tubs* DetectorLid_Solid_Sleeve
    = new G4Tubs("DetectorLid",             // its name
                (0.)*cm,                    // innerR
                (1.9844)*cm,                // outer radius
                (0.19685)*cm,               // half height
                0.*deg,                     // start angle
                360.*deg);                  // end angle

  G4LogicalVolume* DetectorLid_Logical_Sleeve
    = new G4LogicalVolume(
                DetectorLid_Solid_Sleeve,   // its solid
                AlAlloy6061,                // its material
                "DetectorLid");             // its name

  G4VPhysicalVolume* DetectorLid_Physical_Sleeve
    = new G4PVPlacement(
                0,                          // no rotation
                G4ThreeVector(0.*cm,0.*cm,(2.37715)*cm),  // center
                DetectorLid_Logical_Sleeve, // its logical volume
                "DetectorLid",              // its name
                worldLV,                    // its mother volume
                false,                      // no boolean operation
                0,                          // copy number
                fCheckOverlaps);            // checking overlaps
  // Detector lid visualization
  G4VisAttributes* visAttributes19 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  DetectorLid_Logical_Sleeve->SetVisAttributes(visAttributes19);

  // 12.) Detector Tube
  G4Tubs* DetectorTube_Solid_Sleeve
    = new G4Tubs("DetectorTube",            // its name
                (1.9844)*cm,                // innerR
                (2.2225)*cm,                // outer radius
                (19.58085)*cm,              // half height
                0.*deg,                     // start angle
                360.*deg);                  // end angle

  G4LogicalVolume* DetectorTube_Logical_Sleeve
    = new G4LogicalVolume(
                DetectorTube_Solid_Sleeve,  // its solid
                AlAlloy6061,                // its material
                "DetectorTube");            // its name

  G4VPhysicalVolume* DetectorTube_Physical_Sleeve
    = new G4PVPlacement(
                0,                          // no rotation
```

```
                G4ThreeVector(0.*cm,0.*cm,(-17.00685)*cm),  // center
                DetectorTube_Logical_Sleeve, // its logical volume
                "DetectorTube",              // its name
                worldLV,                     // its mother volume
                false,                       // no boolean operation
                0,                           // copy number
                fCheckOverlaps);             // checking overlaps
  // Detector tube visualization
  G4VisAttributes* visAttributes20 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  DetectorTube_Logical_Sleeve->SetVisAttributes(visAttributes20);

  // 13.) Mask Holder
  G4Tubs* MaskHolder_Solid_Sleeve
    = new G4Tubs("MaskHolder",               // its name
                (2.3496)*cm,                 // innerR
                (2.6987)*cm,                 // outer radius
                (16.51)*cm,                  // half height
                0.*deg,                      // start angle
                360.*deg);                   // end angle

  G4LogicalVolume* MaskHolder_Logical_Sleeve
    = new G4LogicalVolume(
                MaskHolder_Solid_Sleeve,     // its solid
                AlAlloy6061,                 // its material
                "MaskHolder");               // its name

  G4VPhysicalVolume* MaskHolder_Physical_Sleeve
    = new G4PVPlacement(
                0,                           // no rotation
                G4ThreeVector(0.*cm,0.*cm,(-19.2896)*cm),  // center
                MaskHolder_Logical_Sleeve,   // its logical volume
                "MaskHolder",                // its name
                worldLV,                     // its mother volume
                false,                       // no boolean operation
                0,                           // copy number
                fCheckOverlaps);             // checking overlaps
  // Mask holder visualization
  G4VisAttributes* visAttributes21 = new G4VisAttributes(G4Colour(0.5, 0.5,
0.5)); // Gray
  MaskHolder_Logical_Sleeve->SetVisAttributes(visAttributes21);

  // End of Defining Geometry

  return worldPV;
}

// ================ Convert DetectLV to Sensitive Detector Volume
=============== //
void B4DetectorConstruction::ConstructSDandField()
{
  G4SDManager::GetSDMpointer()->SetVerboseLevel(1);
  auto absDetector = new G4MultiFunctionalDetector("Detector");
  G4SDManager::GetSDMpointer()->AddNewDetector(absDetector);
  SetSensitiveDetector("DetectLV",absDetector);
}
```