```
//
/// \file TB1106DetectorConstruction.cc

#include "TB1106DetectorConstruction.hh"

#include "G4RunManager.hh"
#include "G4NistManager.hh"
#include "G4Box.hh"
#include "G4Cons.hh"
#include "G4Orb.hh"
#include "G4Sphere.hh"
#include "G4Trd.hh"
#include "G4SubtractionSolid.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4SystemOfUnits.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

TB1106DetectorConstruction::TB1106DetectorConstruction()
    : G4VUserDetectorConstruction(),
    fScoringVolume() //Initialize an fScoringVolume object of type G4VLogicalVolume
{ }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

TB1106DetectorConstruction::~TB1106DetectorConstruction()
{ }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4VPhysicalVolume* TB1106DetectorConstruction::Construct()
{
  // Get nist material manager
  G4NistManager* nist = G4NistManager::Instance();
```

```
  // Envelope (water cube) parameters
  G4double env_sizeX = 30*cm, env_sizeY = 30*cm, env_sizeZ = 35*cm;

  // 1-D Tank parameters
  G4double Tank_sizeXin = 30*cm, Tank_sizeYin = 30*cm, Tank_sizeZin = 35*cm;
  G4double Tank_sizeXou = 32*cm, Tank_sizeYou = 30*cm, Tank_sizeZou = 37*cm;
  G4double Tank_bottomX = 32*cm, Tank_bottomY = 2*cm,  Tank_bottomZ = 37 * cm;

  G4Material* env_mat   = nist->FindOrBuildMaterial("G4_WATER");
  G4Material* Tank_mat  = nist->FindOrBuildMaterial("G4_PLEXIGLASS");
  G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
  G4Material* Det_mat   = nist->FindOrBuildMaterial("G4_SILICON_DIOXIDE");

  std::vector<G4double> photonEnergy = {
    2.034 * eV, 2.068 * eV, 2.103 * eV, 2.139 * eV, 2.177 * eV, 2.216 * eV,
    2.256 * eV, 2.298 * eV, 2.341 * eV, 2.386 * eV, 2.433 * eV, 2.481 * eV,
    2.532 * eV, 2.585 * eV, 2.640 * eV, 2.697 * eV, 2.757 * eV, 2.820 * eV,
    2.885 * eV, 2.954 * eV, 3.026 * eV, 3.102 * eV, 3.181 * eV, 3.265 * eV,
    3.353 * eV, 3.446 * eV, 3.545 * eV, 3.649 * eV, 3.760 * eV, 3.877 * eV,
    4.002 * eV, 4.136 * eV
  };

  /*std::vector<G4double> energy_water = {
    1.56962 * eV, 1.58974 * eV, 1.61039 * eV, 1.63157 * eV, 1.65333 * eV,
    1.67567 * eV, 1.69863 * eV, 1.72222 * eV, 1.74647 * eV, 1.77142 * eV,
    1.7971 * eV,  1.82352 * eV, 1.85074 * eV, 1.87878 * eV, 1.90769 * eV,
    1.93749 * eV, 1.96825 * eV, 1.99999 * eV, 2.03278 * eV, 2.06666 * eV,
    2.10169 * eV, 2.13793 * eV, 2.17543 * eV, 2.21428 * eV, 2.25454 * eV,
    2.29629 * eV, 2.33962 * eV, 2.38461 * eV, 2.43137 * eV, 2.47999 * eV,
    2.53061 * eV, 2.58333 * eV, 2.63829 * eV, 2.69565 * eV, 2.75555 * eV,
    2.81817 * eV, 2.88371 * eV, 2.95237 * eV, 3.02438 * eV, 3.09999 * eV,
    3.17948 * eV, 3.26315 * eV, 3.35134 * eV, 3.44444 * eV, 3.54285 * eV,
    3.64705 * eV, 3.75757 * eV, 3.87499 * eV, 3.99999 * eV, 4.13332 * eV,
    4.27585 * eV, 4.42856 * eV, 4.59258 * eV, 4.76922 * eV, 4.95999 * eV,
```

```
    5.16665 * eV, 5.39129 * eV, 5.63635 * eV, 5.90475 * eV, 6.19998 * eV
  };*/

  std::vector<G4double> rindexWaterCube = {
    1.3435, 1.344,  1.3445, 1.345,  1.3455, 1.346,  1.3465, 1.347,
    1.3475, 1.348,  1.3485, 1.3492, 1.35,   1.3505, 1.351,  1.3518,
    1.3522, 1.3530, 1.3535, 1.354,  1.3545, 1.355,  1.3555, 1.356,
    1.3568, 1.3572, 1.358,  1.3585, 1.359,  1.3595, 1.36,   1.3608
  };

  std::vector<G4double> rindexTank = {
    1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490,
    1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490,
    1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490,
    1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490, 1.490
  };

  std::vector<G4double> rindexWorld = {
     1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000,
     1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000,
     1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000,
     1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000
  };

  std::vector<G4double> rindexDetector = {
     1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163,
     1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163,
     1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163,
     1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163, 1.88163
  };

  std::vector<G4double> absorption = {
    3.448 * m,  4.082 * m,  6.329 * m,  9.174 * m,  12.346 * m, 13.889 * m,
    15.152 * m, 17.241 * m, 18.868 * m, 20.000 * m, 26.316 * m, 35.714 * m,
    45.455 * m, 47.619 * m, 52.632 * m, 52.632 * m, 55.556 * m, 52.632 * m,
```

```
    52.632 * m, 47.619 * m, 45.455 * m, 41.667 * m, 37.037 * m, 33.333 * m,
    30.000 * m, 28.500 * m, 27.000 * m, 24.500 * m, 22.000 * m, 19.500 * m,
    17.500 * m, 14.500 * m
  };

  G4MaterialPropertiesTable* water_cube = new G4MaterialPropertiesTable();
  water_cube->AddProperty("RINDEX", photonEnergy, rindexWaterCube)->SetSpline(true);
  water_cube->AddProperty("ABSLENGTH", photonEnergy, absorption)->SetSpline(true);
  env_mat->SetMaterialPropertiesTable(water_cube);

  G4MaterialPropertiesTable* tank_mat = new G4MaterialPropertiesTable();
  tank_mat->AddProperty("RINDEX", photonEnergy, rindexTank)->SetSpline(true);
  Tank_mat->SetMaterialPropertiesTable(tank_mat);

  G4MaterialPropertiesTable* det_mat = new G4MaterialPropertiesTable();
  det_mat->AddProperty("RINDEX", photonEnergy, rindexDetector)->SetSpline(true);
  Det_mat->SetMaterialPropertiesTable(det_mat);

  G4MaterialPropertiesTable* Worldptr = new G4MaterialPropertiesTable();
  Worldptr->AddProperty("RINDEX", photonEnergy, rindexWorld)->SetSpline(true);
  world_mat->SetMaterialPropertiesTable(Worldptr);

  // Option to switch on/off checking of volumes overlaps
  G4bool checkOverlaps = true;

  //-----
  //World
  //-----
  G4double world_sizeX = 100.*cm;
  G4double world_sizeY = 100.*cm;
  G4double world_sizeZ = 100.*cm;

  G4Box* solidWorld =
    new G4Box("World", //its name
       0.5*world_sizeX, 0.5*world_sizeY, 0.5*world_sizeZ); //its size
```

```
  G4LogicalVolume* logicWorld =
    new G4LogicalVolume(solidWorld,          //its solid
                        world_mat,           //its material
                        "World");            //its name

  G4VPhysicalVolume* physWorld =
    new G4PVPlacement(0,                     //no rotation
                      G4ThreeVector(),       //place at the origin (0,0,0)
                      logicWorld,            //its logical volume
                      "World",               //its name
                      0,                     //its mother  volume
                      false,                 //no boolean operation
                      0,                     //copy number
                      checkOverlaps);        //overlaps checking

  //------------
  // Water cube
  //------------
  G4Box* solidEnv =
    new G4Box("Envelope",                    //its name
        0.5*env_sizeX, 0.5*env_sizeY, 0.5*env_sizeZ); //its size

   logicEnv =
    new G4LogicalVolume(solidEnv,            //its solid
                        env_mat,             //its material
                        "Envelope");         //its name

  new G4PVPlacement(0,                       //no rotation
                    G4ThreeVector(),         //at (0,0,0)
                    logicEnv,                //its logical volume
                    "Envelope",              //its name
                    logicWorld,              //its mother  volume
                    false,                   //no boolean operation
                    0,                       //copy number
```

```
                    checkOverlaps);          //overlaps checking

 //--------------
 //PTW 1-D tank
 //--------------
  G4Box* solidTankOut =
      new G4Box("OuterTank",                                          //its name
          0.5*Tank_sizeXou , 0.5*Tank_sizeYou, 0.5*Tank_sizeZou); //its size

  G4Box* solidTankIn =
      new G4Box("InnerTank",                                         //its name
          0.5*Tank_sizeXin, 0.5*Tank_sizeYin, 0.5*Tank_sizeZin); //its size

  G4Box* solidTankBottom =
      new G4Box("bottom",                                       //its name
          0.5 * Tank_bottomX, 0.5 * Tank_bottomY, 0.5 * Tank_bottomZ); //its size

  //Boolean inner and outer to create a 4-sided cube (no top or bottom surfaces)
  G4VSolid* solidTank
      = new G4SubtractionSolid("Outer-inner", solidTankOut, solidTankIn,
          0, G4ThreeVector(0., 0., 0.));

  G4LogicalVolume* logicTankBottom =      //Add the bottom surface back in (2cm thick)
      new G4LogicalVolume(solidTankBottom,//its solid
          Tank_mat,                       //its material
          "solidTankBottom");             //its name

  new G4PVPlacement(0,
      G4ThreeVector(0,-16*cm,0),
      logicTankBottom,
      "solidTankBottom",
      logicWorld,
      false,
      0,
      checkOverlaps);
```

```
  G4LogicalVolume* logicTank =
      new G4LogicalVolume(solidTank,//its solid
          Tank_mat,                 //its material
          "solidTank");             //its name

  new G4PVPlacement(0,          //no rotation
      G4ThreeVector(),          //at (0,0,0)
      logicTank,                //its logical volume
      "solidTank",              //its name
      logicWorld,               //its mother  volume
      false,                    //no boolean operation
      0,                        //copy number
      checkOverlaps);           //overlaps checking


  //CCD Array
  //--------------
  G4Box* DetectorArray = new G4Box("Array",//its name
          25.*cm, 25.*cm, 1.0*mm); //its size

      logicDetectorArray =
      new G4LogicalVolume(DetectorArray, //its solid
          Det_mat,                      //its material
          "DetectorArray");             //its name

  new G4PVPlacement(0,          //no rotation
      G4ThreeVector(0,0,-25.*cm),
      logicDetectorArray,       //its logical volume
      "DetectorArray",          //its name
      logicWorld,               //its mother  volume
      false,                    //no boolean operation
      0,                        //copy number
      checkOverlaps);           //overlaps checking
```

```
 //------------------------------
 //Set Water Cube as scoring volume
 //------------------------------
  fScoringVolume = logicEnv;

  //---------------------------------
  //always return the physical World
  //---------------------------------
  return physWorld;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void TB1106DetectorConstruction::ConstructSDandField()
{

    TB1106SensitiveDetector *sensDet = new TB1106SensitiveDetector("SensitiveDetector");
    logicDetectorArray->SetSensitiveDetector(sensDet);
}
```