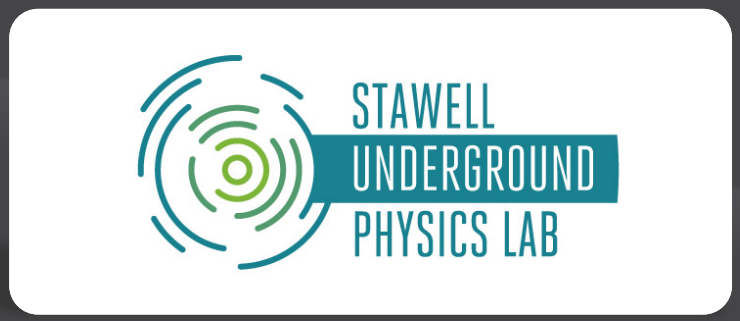
The new Detector Construction Class of the SABRE experiment

F. Nuti

6th of September 2017



Goal

- Split the SABREDetectorConstruction class into subclasses containing only the relevant geometries for a specific experiment (PoP, SUPL, etc.)

SABREDetectorConstruction
SABREDetectorConstructionMessenger

SABREDetectorConstruction
SABREDetectorMaterial
SABREDetectorProperty
SABREDetectorModule
SABREDetectorPoP
SABREDetectorNorth
SABREDetectorSouth
SABREDetectorConstructionMessenger
SABREDetectorModuleMessenger
SABREDetectorPoPMessenger
(few more messenger classes that are not used at the moment)

Class description

SABREDetector…

- **Construction:** general class which puts together the pieces from the subclasses and place them in the world volume
- **Material:** defines all the materials used in the detector construction
- **Property:** store general information about the geometry implemented (volume masses and densities)
- **Module:** Defines the logical volumes of the crystal enclosures and veto pmts. Should contain all the pieces that are identical in the different experiments
- **PoP:** shielding and vessel according to PoP design
- **North:** hall B design, (in future shielding and vessel for the final experiment)
- **South:** SUPL lab, shielding and vessel
- **ConstructionMessenger:** to set general parameters of the geometry and also some parameters for South and North
- **ModuleMessenger:** to set parameters for crystals and PMTs
- **PoPMessenger:** to set parameters of the PoP shielding and vessel

Module Class

```
class SABREDetectorModule
{
public:

  SABREDetectorModule();
  ~SABREDetectorModule();
  static SABREDetectorModule* GetInstance();
  void Construct();
  void ConstructEnclosure();
  void ConstructVesselPMT();
  void UpdateGeometry();
  void SaveMassAndDensity();
  void Refresh();
  void SetCrystalSolid(G4String cs) {CrystalSolid = cs;}
  void SetCrystalIR(G4double cir) {crystalIR = cir;}
  void SetCrystalOR(G4double cor) {crystalOR = cor;}
  void SetCrystalZ(G4double cz) {crystalZ = cz;}
  G4LogicalVolume* GetVesselPMT() {return VesselPMT_log;}
  G4UnionSolid* GetVesselPMTHoleSolid() {return VesselPMTHole;}
  G4LogicalVolume* GetEnclosure() {return Enclosure_log;}
  G4ThreeVector GetEnclosureSizeXYZ() {return size_Enclosure;}
  G4LogicalVolume* GetVacuum() {return Vacuum_log;}
  G4ThreeVector GetVacuumSizeXYZ() {return size_Vacuum;}
  G4LogicalVolume* GetCrystal() {return Crystal_log;}
  G4ThreeVector GetCrystalSizeXYZ() {return size_Crystal;}
  G4ThreeVector GetCrystalTranslation() {return tr_Crystal;}
  G4RotationMatrix GetCrystalRotation() {return rot_Crystal;}
```

Logical volumes and their daughters are built

specifies which volume should have its mass and density saved

Methods to return the information needed by the other classes

Example of experiment class: South Class

```
class SABREDetectorSouth
{
public:

  SABREDetectorSouth();
  ~SABREDetectorSouth();
  static SABREDetectorSouth* GetInstance();
  void Construct();
  void ConstructRock();
  void ConstructShielding();
  void ConstructVessel();
  void UpdateGeometry();
  void SaveMassAndDensity();
  void Refresh();
```

Logical volumes and their daughters are built in the Construct* methods:
ConstructRock creates Rock_log→Laboratory_log
ConstructShielding creates Shielding_log→InsideVolume_log
ConstructVessel creates Vessel_log→Scintillator_log

- Rock_log: Outermost logical volume of the laboratory
- Laboratory_log: Innermost logical volume of the laboratory
- Shielding_log: Outermost logical volume of the shielding
- InsideVolume_log: Innermost logical volume of the shielding
- Vessel_log: Outermost logical volume of the vessel
- Scintillator_log: Innermost logical volume of the vessel (it is the CIS in the DRY solution of the PoP)
- Enclosure_log: Outermost logical volume of the crystal enclosure

Example of experiment class: South Class

```
class SABREDetectorSouth
{
public:

  SABREDetectorSouth();
  ~SABREDetectorSouth();
  static SABREDetectorSouth* GetInstance();
  void Construct();
  void ConstructRock();
  void ConstructShielding();
  void ConstructVessel();
  void UpdateGeometry();
  void SaveMassAndDensity();
  void Refresh();

  G4LogicalVolume* GetRock() {return Rock_log;}
  G4ThreeVector GetRockSizeXYZ() {return size_Rock;}
  G4RotationMatrix GetRockAbsRotation() {return absrot_Rock;}//Rotation wrt the world reference frame
  G4LogicalVolume* GetLaboratory() {return Laboratory_log;}
  G4ThreeVector GetLaboratorySizeXYZ() {return size_Laboratory;}
  G4ThreeVector GetLaboratoryTranslation() {return tr_Laboratory;}
  G4RotationMatrix GetLaboratoryRotation() {return rot_Laboratory;}
  G4LogicalVolume* GetShielding() {return Shielding_log;}
  G4ThreeVector GetShieldingSizeXYZ() {return size_Shielding;}
  G4RotationMatrix GetShieldingAbsRotation() {return absrot_Shielding;}/
  G4LogicalVolume* GetInsideVolume() {return InsideVolume_log;}
  G4ThreeVector GetInsideVolumeSizeXYZ() {return size_InsideVolume;}
  G4ThreeVector GetInsideVolumeTranslation() {return tr_InsideVolume;}
  G4RotationMatrix GetInsideVolumeRotation() {return rot_InsideVolume;}
  G4LogicalVolume* GetVessel() {return Vessel_log;}
  G4ThreeVector GetVesselSizeXYZ() {return size_Vessel;}
  G4RotationMatrix GetVesselAbsRotation() {return absrot_Vessel;}//Rotation wrt the world reference frame
  G4LogicalVolume* GetScintillator() {return Scintillator_log;}
  G4ThreeVector GetScintillatorSizeXYZ() {return size_Scintillator;}
  G4ThreeVector GetScintillatorTranslation() {return tr_Scintillator;}
  G4RotationMatrix GetScintillatorRotation() {return rot_Scintillator;}
```

Logical volumes and their daughters are built in the Construct* methods:

ConstructRock creates Rock_log→Laboratory_log

ConstructShielding creates Shielding_log→InsideVolume_log

ConstructVessel creates Vessel_log→Scintillator_log

Methods to return the relevant logical volumes and the information needed to place them in the right position

Detector Class

- Defines world
- Imports Laboratory
- Imports Shielding
- Places Shielding in Laboratory
- Imports Vessel
- Places Vessel in Shielding
- Imports Crystal Enclosures
- Places Enclosures in Vessel
- Places Laboratory in World adjusting the origin
- Set Sensitive detectors
- Save masses and densities

Example of Import

```
if (SABRELab == "LNGS")
  {
    SABREDetectorNorth* SABRENorth = SABREDetectorNorth::GetInstance();
    if (rockThicknessOuter != -999*m)
      SABRENorth->SetExternalRockThickness(rockThicknessOuter);
    if (productionLayerThickness != -999*m)
      SABRENorth->SetProductionRockThickness(productionLayerThickness);
    if (rockThicknessInner != -999*m)
      SABRENorth->SetInternalRockThickness(rockThicknessInner);
    SABRENorth->ConstructRock();
    Rock_log=SABRENorth->GetRock();
    size_Rock=SABRENorth->GetRockSizeXYZ();
    absrot_Rock=SABRENorth->GetRockAbsRotation();
    Laboratory_log=SABRENorth->GetLaboratory();
    size_Laboratory=SABRENorth->GetLaboratorySizeXYZ();
    tr_Laboratory=SABRENorth->GetLaboratoryTranslation();
    rot_Laboratory=SABRENorth->GetLaboratoryRotation();
```

Example of Placement

```
rot_Vessel=(rot_InsideVolume.inverse()*absrot_Shielding.inverse())*absrot_Vessel;//Rotat
size=rot_Vessel*size_Vessel;//Size of the Vessel in the InsideVolume frame
if(SABREVessel=="GDMLVessel")
  {
    //Set the airboxgdml inside the airbox. The two boxes are in touch on the ground
    tr_Vessel=G4ThreeVector(0.,-1.*size_InsideVolume.y()+fabs(size.y()),0.);
  }
else if(SABREVessel=="SUPLVessel")
  {
    //Set the Vessel in the middle of the InsideVolume
    tr_Vessel=G4ThreeVector(0.,0.,0.);
  }
else if(SABREVessel=="PoPVessel")
  {
    SABREDetectorPoP* SABREPoP = SABREDetectorPoP::GetInstance();
    tr_Vessel=SABREPoP->GetVesselTranslation();
    if(rot_Vessel!=SABREPoP->GetVesselRotation())
      {
        G4cout << "ERROR: Something is wrong in the calculation of the vessel rotation.
DetectorPoP class" << G4endl;
        //throw std::exception();
        exit(1);
      }
  }

if(SABREVessel!="PoPVessel")
  {
    //The vessel is already placed inside the shielding in the PoP design
    CheckSizeCompatibility(size_InsideVolume, size_Vessel, rot_Vessel);//This is just a
    G4PVPlacement* Vessel_phys = new G4PVPlacement(G4Transform3D(rot_Vessel,tr_Vessel),
                                                   Vessel_log,"Vessel",InsideVolume_log,false,0,true);
```

Differences with Master

- Messenger command options changed:
  - /SABRE/shield/select: *FullShield, PoPShield, SUPLShield or NoShield*
  - /SABRE/vessel/csg_solid: *PoPVessel or SUPLVessel*
- Positioning of the PoP shielding in Hall B not identical to the master version of the code

```
if (SABRELab == "LNGS")
  {
    tr_Shielding+=G4ThreeVector(0.,-1*size_Laboratory.y(),size_Laboratory.z()-10*m);
  }
```

New

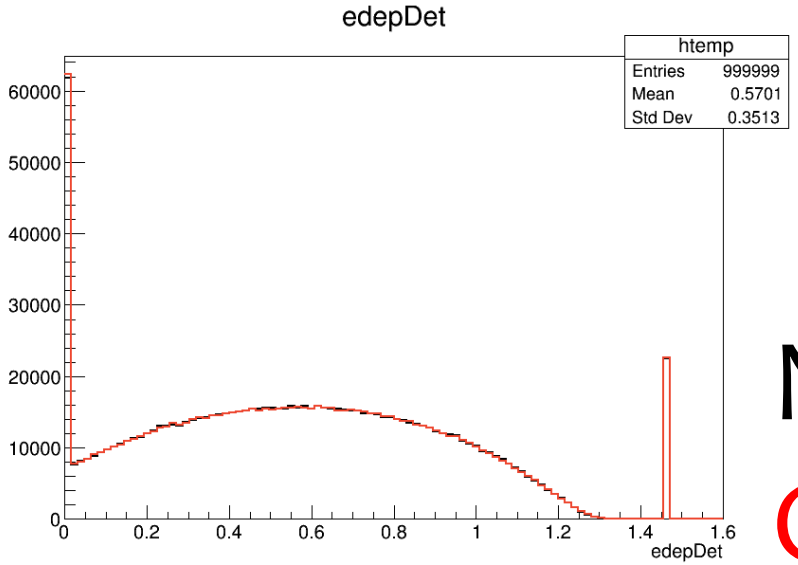
```
if (SABRELab == "LNGS") {
    //tr_Sphere+=G4ThreeVector(0.,0., 0.5*HallSizeLength - 10*m - 0.5*BasePE_z);
    tr_Sphere+=G4ThreeVector(0.,0., 0.5*HallSizeLength - 10*m - 0.5*BasePb_z-BasePEFront_z);
    tr_Sphere+=G4ThreeVector(0.,-0.5*HallSizeHeight, 0.);
}
```

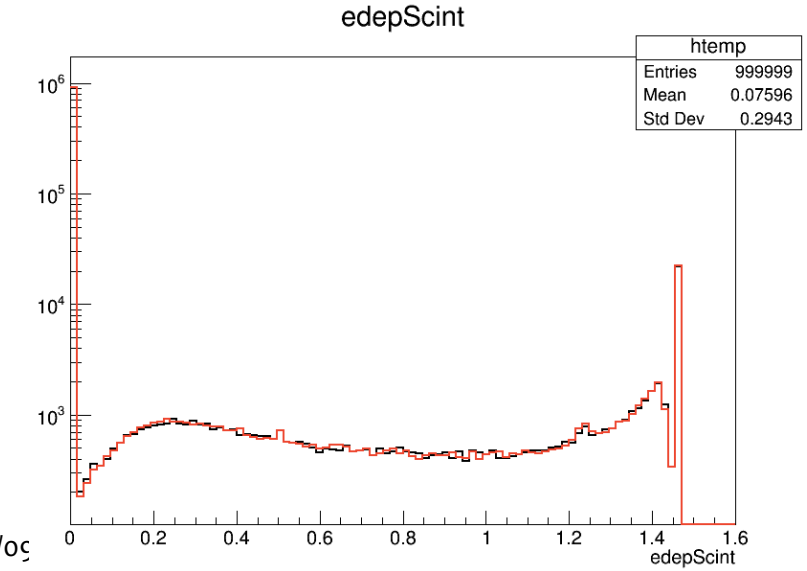
Old

- Volume placement has now G4Transform3D(rotation,translation) as argument. With this definition the "rotation" is the rotation of daughter volume rather than being the rotation of mother frame

Validation Tests

K40 in crystal (PoP geometry)

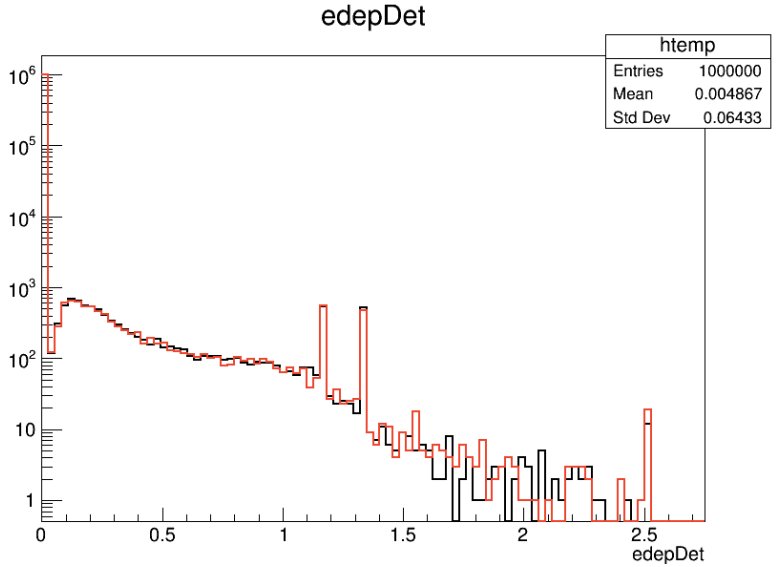


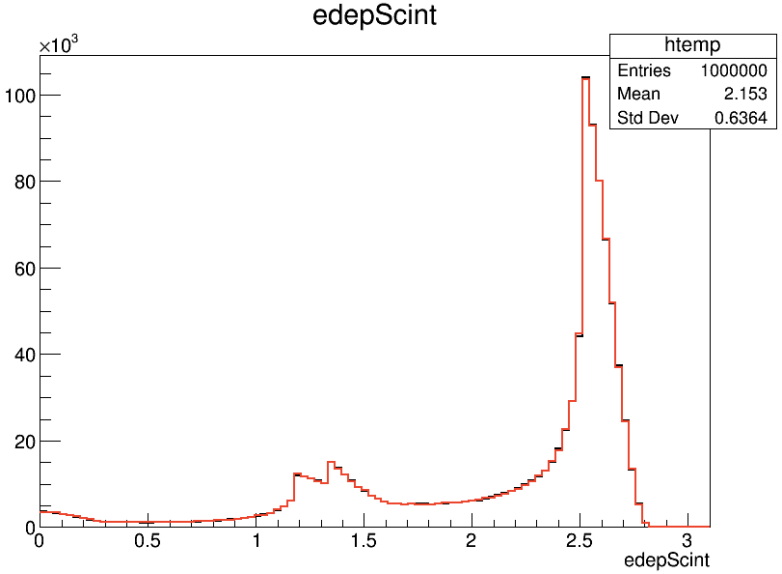


New

Old

Co60 in scintillator (SUPL geometry)





Potential changes

- Get rid of the vacuum assembly. We could just place the vacuum inside the enclosure volume

```
//Creating an assembly with only one placement of Vac_log in the middle of it without any translation or rotiation
tr = G4ThreeVector(0.,0.,0.);//translation in mother frame
tr_Vacuum+=(rot_Vacuum*tr);
rot = G4RotationMatrix();// rotation of daughter volume
rot_Vacuum*=rot; //equivalent to rot_Vacuum=rot_Vacuum*rot
G4AssemblyVolume* assemblyVacuum = new G4AssemblyVolume(SABREModules->GetVacuum(), tr, &rot);
//Placing the assembly inside Enclosure_log without any rotation or translation
tr = G4ThreeVector(0.,0.,0.);//translation in mother frame
tr_Vacuum+=(rot_Vacuum*tr);
rot = G4RotationMatrix();// rotation of daughter volume
rot_Vacuum*=rot; //equivalent to rot_Vacuum=rot_Vacuum*rot
assemblyVacuum->MakeImprint(SABREModules->GetEnclosure(),tr,&rot);
```

- Move the remaining “generic” volumes (NoCave, FullShield, NoShield, gdml vessel) out of SABREDetectorConstruction into a separate class
- Alternative method to place vessel in the PoP shielding? Maybe a PlaceVessel() method? So far done inside ConstructVessel()
- Anything else?

Merge Requests

- Latest changes in master (Paolo)
- Fixes for Geant4 10.3 (Paolo)
- Add LAB in materials and optical physics (Lindsey)

WARNINGS

These were inherited from the master and regards the PoP design:

- unused variable 'SideTank_z’
- unused variable 'CISSteelBarBody_OR’
- unused variable 'SteelVesselBody’
- unused variable 'SteelVesselNeck_realZ’

It’s worth to check of these variables are needed

The Mass, Volume calculation problem

- The cubic volume of many solids and so their masses are not exactly calculated but they are somehow estimated

```
G4EllipticalTube* SimpleEllipticalTube = new G4EllipticalTube("SimpleEllipticalTube",
                                                              1*m,
                                                              2*m,
                                                              1*m);
//Volume = pi*1m*2m*2*(1*m)=4pim*m*m=12.566370614m*m*m
if(debugmode)
  {
    G4cout<<SimpleEllipticalTube->GetName()<<" tolerance (m) "<<SimpleEllipticalTube->GetTolerance()/m<<G4endl;
    G4cout<<SimpleEllipticalTube->GetName()<<" cubic volume (m*m*m) "<<SimpleEllipticalTube->GetCubicVolume()/(m*m*m)<<G4endl;
    G4cout<<SimpleEllipticalTube->GetName()<<" surface area (m*m) "<<SimpleEllipticalTube->GetSurfaceArea()/(m*m)<<G4endl;
  }
```

1st computation output:
SimpleEllipticalTube tolerance (m) 1e-12
SimpleEllipticalTube cubic volume (m*m*m) 12.56
SimpleEllipticalTube surface area (m*m) 32.0785

2nd computation output:
SimpleEllipticalTube tolerance (m) 1e-12
SimpleEllipticalTube cubic volume (m*m*m) 12.5789
SimpleEllipticalTube surface area (m*m) 32.3078

3rd computation output:
SimpleEllipticalTube tolerance (m) 1e-12
SimpleEllipticalTube cubic volume (m*m*m) 12.5786
SimpleEllipticalTube surface area (m*m) 31.9265

How the estimate works

G4double G4VSolid::EstimateCubicVolume (G4int *nStat*,
G4double *epsilon*
) const [protected]

Definition at line 203 of file G4VSolid.cc.

References CalculateExtent(), G4UniformRand, Inside(), kOutside, kXAxis, kYAxis, and kZAxis.

Referenced by GetCubicVolume(), G4VCSGfaceted::GetCubicVolume(), G4BREPSolid::GetCubicVolume(), and G4BooleanSolid::GetCubicVolume().

```
00204 {
00205   G4int iInside=0;
00206   G4double px,py,pz,minX,maxX,minY,maxY,minZ,maxZ,volume;
00207   G4ThreeVector p;
00208   EInside in;
00209
00210   // values needed for CalculateExtent signature
00211
00212   G4VoxelLimits limit;                // Unlimited
00213   G4AffineTransform origin;
00214
00215   // min max extents of pSolid along X,Y,Z
00216
00217   this->CalculateExtent(kXAxis,limit,origin,minX,maxX);
00218   this->CalculateExtent(kYAxis,limit,origin,minY,maxY);
00219   this->CalculateExtent(kZAxis,limit,origin,minZ,maxZ);
00220
00221   // limits
00222
00223   if(nStat < 100)    nStat   = 100;
00224   if(epsilon > 0.01) epsilon = 0.01;
00225
00226   for(G4int i = 0; i < nStat; i++ )
00227   {
00228     px = minX+(maxX-minX)*G4UniformRand();
00229     py = minY+(maxY-minY)*G4UniformRand();
00230     pz = minZ+(maxZ-minZ)*G4UniformRand();
00231     p  = G4ThreeVector(px,py,pz);
00232     in = this->Inside(p);
00233     if(in != kOutside) iInside++;
00234   }
00235   volume = (maxX-minX)*(maxY-minY)*(maxZ-minZ)*iInside/nStat;
00236   return volume;
00237 }
```

SABREMCAnalysis code discussion

Proposed changes

- Use mytree->GetEntries() instead of mytree->GetEntriesFast()?
- Move the histograms into a vector of vectors of histograms. Indices refer to cut index and variable index. Create a database file with: cut_name, cut_value, var_name, var_min, var_max, var_nbin, xaxis_title. Loop over cuts and variables …

H[cut_i][var_j]->Fill(var_j_value)

```
#Name of the tree branch to be plotted
variables=['edepDet','edepScint','edepDet', 'edepScint','xpos_vertex']
xtitles=['edepDet [MeV]','edepScint [MeV]','edepDet [MeV]', 'edepScint [MeV]', 'X position of the vertex [mm?]']
#Selection to be applied to make the histogram
cuts=['','','edepDet>0 && edepDet<0.1', 'edepScint>0 && edepScint<0.1', 'edepScint>0']
#Identifier of the selection applied
cutnames=['NoCut','Nocut','edepLess01', 'edepLess01', 'NoCut']
#Definition of the bins
bins=[80,80,100,100,100]
lows=[0.,0.,0.,0.,-1000]
highs=[1.6,1.6,0.1,0.1,1000]
```

Example of database info in python