```
1:  #include "QweakSimCerenkovDetector.hh"
2:
3:  QweakSimCerenkovDetector::QweakSimCerenkovDetector(QweakSimUserInformation *userInfo)
4:  {
5:    // initialize some pointers
6:    myUserInfo = userInfo;
7:
8:    CerenkovDetectorMessenger = NULL;
9:    pMaterial                 = NULL;
10:
11:   theMotherPV               = NULL;
12:
13:   CerenkovContainer_Logical  = NULL;
14:   CerenkovContainer_Physical = NULL;
15:   CerenkovContainer_Material = NULL;
16:
17: //   CerenkovDetector_Logical   = NULL;
18: //   CerenkovDetector_Physical  = NULL;
19: //   CerenkovDetector_Material  = NULL;
20:
21:   ActiveArea_Logical        = NULL;
22:   ActiveArea_Physical       = NULL;
23:   ActiveArea_Material       = NULL;
24:
25:   LightGuide_LogicalLeft     = NULL;
26:   LightGuide_LogicalRight    = NULL;
27:   LightGuide_PhysicalLeft    = NULL;
28:   LightGuide_PhysicalRight   = NULL;
29:   LightGuide_Material        = NULL;
30:
31:   QuartzGlue_Logical         = NULL;
32:   QuartzGlue_PhysicalLeft    = NULL;
33:   QuartzGlue_PhysicalCenter  = NULL;
34:   QuartzGlue_PhysicalRight   = NULL;
35:   QuartzGlue_Material        = NULL;
36:
37:   PMTContainer_Logical  = NULL;
38:   PMTContainer_PhysicalLeft  = NULL;
39:   PMTContainer_PhysicalRight = NULL;
40:   PMTContainer_Material = NULL;
41:
42:   PMTEntranceWindow_Logical  = NULL;
43:   PMTEntranceWindow_Physical = NULL;
44:   PMTEntranceWindow_Material = NULL;
45:
46:   Cathode_Logical  = NULL;
47:   Cathode_Physical = NULL;
48:   Cathode_Material = NULL;
```

```
49:
50:     Rotation_CerenkovContainer = NULL;
51:
52:     // pointer to the sensitive detector
53:     CerenkovDetectorSD      = NULL;
54:     CerenkovDetector_PMTSD  = NULL;
55:
56:     // clear vector containing temp solids for boolean soild union
57:     LeftQuartz_Solid.clear();
58:     LeftQuartz_Solid.resize(4);  //need 4 chamfers on quartz bar proper
59:     RightQuartz_Solid.clear();
60:     RightQuartz_Solid.resize(4); //need 4 chamfers on quartz bar proper
61:     LeftGuide_Solid.clear();
62:     LeftGuide_Solid.resize(5);   //need 4 chamfers + 1 angle cut on light guide
63:     RightGuide_Solid.clear();
64:     RightGuide_Solid.resize(5);  //need 4 chamfers + 1 angle cut on light guide
65:
66:     mirror_logical.clear();
67:     mirror_physical.clear();
68:
69:     mirror_logical.resize(8);
70:     mirror_physical.resize(8);
71:
72:     CerenkovMasterContainer_Physical.clear();
73:     CerenkovMasterContainer_Physical.resize(8);
74:
75:     AnglePhi_CerenkovMasterContainer.clear();
76:     AnglePhi_CerenkovMasterContainer.resize(8);
77:
78:     Translation_CerenkovMasterContainer.clear();
79:     Translation_CerenkovMasterContainer.resize(8);
80:
81:     Rotation_CerenkovMasterContainer.clear();
82:     Rotation_CerenkovMasterContainer.resize(8);
83:
84:     CerenkovDetectorMessenger = new QweakSimCerenkovDetectorMessenger(this);
85:
86:     pMaterial = new QweakSimMaterial();
87:     pMaterial->DefineMaterials();
88:
89:     //CerenkovContainer_Material = pMaterial->GetMaterial("HeGas");
90:     CerenkovContainer_Material = pMaterial->GetMaterial("Air");
91:     ActiveArea_Material        = pMaterial->GetMaterial("Air");
92:     QuartzBar_Material         = pMaterial->GetMaterial("Quartz");
93:     LightGuide_Material        = pMaterial->GetMaterial("Quartz");
94:     PMTContainer_Material      = pMaterial->GetMaterial("Vacuum");
95:     PMTEntranceWindow_Material = pMaterial->GetMaterial("LimeGlass");
96:     PMTQuartzOpticalFilm_Material = pMaterial->GetMaterial("SiElast_Glue");
```

```
 97:    Cathode_Material          = pMaterial->GetMaterial("LimeGlass");
 98:    Radiator_Material         = pMaterial->GetMaterial("Lead");
 99:    QuartzGlue_Material       = pMaterial->GetMaterial("SiElast_Glue");
100:    mirror_material           = pMaterial->GetMaterial("Mirror");
101:
102:    LightGuide_FullLength     =   18.00*cm;
103:    LightGuide_FullWidth1     =   18.00*cm;
104:    LightGuide_FullWidth2     =   18.00*cm;
105:    LightGuide_FullThickness  =    1.25*cm;
106:
107:    QuartzBar_FullLength      =  100.00*cm;    // Full X length
108:    QuartzBar_FullHeight      =   18.00*cm;    // Full Y length
109:    QuartzBar_FullThickness   =    1.25*cm;    // Full Z length
110:
111:    GlueFilm_FullLength_X     =   0.001*mm;
112:    GlueFilm_FullLength_Y     =   18.00*cm;
113:    GlueFilm_FullLength_Z     =    1.25*cm;
114:
115:    ActiveArea_FullLength_X   =    2.0*(LightGuide_FullLength + QuartzBar_FullLength +GlueFilm_FullLength_X) + GlueFilm_FullLength_X +2.0*cm;
116:    ActiveArea_FullLength_Y   =    QuartzBar_FullHeight + 1.0*cm;
117:    ActiveArea_FullLength_Z   =    QuartzBar_FullThickness + 2.0*cm;
118:
119:    Container_FullLength_X    =  ActiveArea_FullLength_X + 20.0*cm;
120:    Container_FullLength_Y    =  ActiveArea_FullLength_Y +  4.0*cm;
121:    Container_FullLength_Z    =  ActiveArea_FullLength_Z + 10.0*cm;
122:
123:    Chamfer_FullLength        =  120.00*cm;
124:    Chamfer_FullHeight        =    7.00*mm;
125:    Chamfer_FullThickness     =    7.00*mm;
126:
127:    G4double ReductionInPhotocathodeDiameter = 5*mm;
128:
129:    PMTQuartzOpticalFilm_Thickness=  0.001*mm;
130:    PMTQuartzOpticalFilm_Diameter =  12.7*cm;
131:
132:    PMTEntranceWindow_Thickness   =  1.0*mm; // assumed PMT glass thickness
133:    PMTEntranceWindow_Diameter    =  12.7*cm;//QuartzBar_FullHeight;
134:
135:    Cathode_Thickness   = 1.0*mm;
136:    Cathode_Diameter    = PMTEntranceWindow_Diameter - ReductionInPhotocathodeDiameter;
137:
138:    PMTContainer_Diameter    =  PMTEntranceWindow_Diameter+1.0*mm;
139:    PMTContainer_FullLength_Z    =  2.0*mm+PMTEntranceWindow_Thickness+Cathode_Thickness;
140:
141:    Tilting_Angle   =  0.0*degree;
142:
143:  }
144:
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
QweakSimCerenkovDetector::~QweakSimCerenkovDetector()
{
  delete pMaterial;
  delete CerenkovDetectorMessenger;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::DefineCerenkovGeometry()
{
  G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::DefineCerenkovGeometry() " << G4endl << G4endl;
  G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::DefineCerenkovGeometry() " << G4endl << G4endl;
}


//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::ConstructComponent(G4VPhysicalVolume* MotherVolume)
{
   //------------------------------------------
   // save the pointer to the physical mothervolume
   //------------------------------------------
   //
   theMotherPV = MotherVolume;


//*************************************************************************************************************
//*******************************Define Detector Container*****************************************************

  Position_CerenkovContainer_X =   0.0*cm;
  Position_CerenkovContainer_Y = 319.0*cm; // given by SolidWorks (or later by Juliette)
  Position_CerenkovContainer_Z = 570.0*cm; // given by SolidWorks (or later by Juliette)

  Position_CerenkovContainer  = G4ThreeVector(Position_CerenkovContainer_X,
                                              Position_CerenkovContainer_Y,
                                              Position_CerenkovContainer_Z);

  Rotation_CerenkovContainer = new G4RotationMatrix();

  Rotation_CerenkovContainer->rotateX(Tilting_Angle);

  G4Box* CerenkovContainer_Solid  = new G4Box("CerenkovContainer_Solid",
                                              0.5 * Container_FullLength_X ,    // half X length required by Geant4
                                              0.5 * Container_FullLength_Y ,    // half Y length required by Geant4
                                              0.5 * Container_FullLength_Z );   // half Z length required by Geant4

  CerenkovContainer_Logical  = new G4LogicalVolume(CerenkovContainer_Solid,
                                                   CerenkovContainer_Material,
                                                   "CerenkovContainer_Logical",
```

```
                                0,0,0);

  CerenkovContainer_Physical   = new G4PVPlacement(Rotation_CerenkovContainer,
                                Position_CerenkovContainer,
                                "CerenkovContainer_Physical",
                                CerenkovContainer_Logical,
                                MotherVolume,
                                false,
                                0);

//*************************************************************************************************************
//*************************************************************************************************************


//*************************************************************************************************************
//*******************************Define Detector Active Area*****************************************
  G4Box* ActiveArea_Solid  = new G4Box("CerenkoDetector_Solid",
                            0.5 * ActiveArea_FullLength_X ,
                            0.5 * ActiveArea_FullLength_Y ,
                            0.5 * ActiveArea_FullLength_Z );

  ActiveArea_Logical  = new G4LogicalVolume(ActiveArea_Solid,
                              ActiveArea_Material,
                              "ActiveArea_Log",
                              0,0,0);

  G4ThreeVector Position_ActiveArea  = G4ThreeVector(0,0,0);

  ActiveArea_Physical   = new G4PVPlacement(0,Position_ActiveArea,
                              ActiveArea_Logical,
                              "ActiveArea_Physical",
                              CerenkovContainer_Logical,
                              false,0);

//*************************************************************************************************************
//*************************************************************************************************************


  G4double PI = 4.0*std::atan(1.0);
  G4double ChamferRotation = 45.0*PI/180.0;
  G4double ChamferScew = 0.0;
  G4double delta = 0.0;

//*************************************************************************************************************
//*******************************Define Right Detector Quartz Bar With Chamfers*************************


```

```
241:    G4Box* Chamfer_Solid    = new G4Box("Chamfer_Solid",
242:                              0.5 * Chamfer_FullLength,      // half X length required by Geant4
243:                              0.5 * Chamfer_FullHeight ,     // half Y length required by Geant4
244:                              0.5 * Chamfer_FullThickness );
245:
246:    G4Box* QuartzBar_Solid  = new G4Box("QuartzBar_Solid",
247:                              0.5 * QuartzBar_FullLength,      // half X length required by Geant4
248:                              0.5 * QuartzBar_FullHeight,      // half Y length required by Geant4
249:                              0.5 * QuartzBar_FullThickness );  // half Z length required by Geant4
250:
251:    //Boolean Union:
252:    //Upper-upstream edge chamfer
253:
254:    ChamferScew = 0.021486*PI/180.0;
255:    delta = 0.5*(Chamfer_FullHeight - 1.0*mm)/sqrt(2.0);
256:    G4double ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
257:    G4double ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
258:    Position_Chamfer1.setX(0.0*cm);//33.333333*cm);
259:    Position_Chamfer1.setY(0.5*QuartzBar_FullHeight + delta);
260:    Position_Chamfer1.setZ(-(0.5*QuartzBar_FullThickness + delta));
261:    Rotation_Chamfer1.rotateX(45.0*degree);
262:    Rotation_Chamfer1.rotateY(ChamferAdjRotY*radian);
263:    Rotation_Chamfer1.rotateZ(ChamferAdjRotZ*radian);
264:    G4Transform3D Transform_Chamfer1(Rotation_Chamfer1,Position_Chamfer1);
265:    Rotation_Chamfer1.rotateZ(-ChamferAdjRotZ*radian);
266:    Rotation_Chamfer1.rotateY(-ChamferAdjRotY*radian);
267:
268:    RightQuartz_Solid[0]=  new G4SubtractionSolid ("UpperUpstreamChamfer-RightQuartzBar",
269:                               QuartzBar_Solid,
270:                               Chamfer_Solid,
271:                               Transform_Chamfer1);
272:
273:    //Boolean Union:
274:    //Upper-downstream edge chamfer
275:
276:    delta = 0.5*(Chamfer_FullHeight - 0.5*mm)/sqrt(2.0);
277:    ChamferScew = 0.0;//0.014*PI/180.0;
278:    ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
279:    ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
280:    Position_Chamfer2.setX(0.0*mm);
281:    Position_Chamfer2.setY(0.5*QuartzBar_FullHeight + delta);
282:    Position_Chamfer2.setZ(0.5*QuartzBar_FullThickness + delta);
283:    Rotation_Chamfer2.rotateX(45.0*degree);
284:    Rotation_Chamfer2.rotateY(-ChamferAdjRotY*radian);
285:    Rotation_Chamfer2.rotateZ(ChamferAdjRotZ*radian);
286:    G4Transform3D Transform_Chamfer2(Rotation_Chamfer2,Position_Chamfer2);
287:    Rotation_Chamfer2.rotateZ(-ChamferAdjRotZ*radian);
288:    Rotation_Chamfer2.rotateY(ChamferAdjRotY*radian);
```

```
289:
290:    RightQuartz_Solid[1] =  new G4SubtractionSolid ("UpperDownstreamChamfer-RightQuartzBar",
291:                                 RightQuartz_Solid[0],
292:                                 Chamfer_Solid,
293:                                 Transform_Chamfer2);
294:
295:    //Boolean Union:
296:    //Lower-Upstream edge chamfer
297:    ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
298:    ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
299:    Position_Chamfer3.setX(0.0*mm);
300:    Position_Chamfer3.setY(-(0.5*QuartzBar_FullHeight + delta));
301:    Position_Chamfer3.setZ(-(0.5*QuartzBar_FullThickness + delta));
302:    Rotation_Chamfer3.rotateX(45.0*degree);
303:    Rotation_Chamfer3.rotateY(ChamferAdjRotY*radian);
304:    Rotation_Chamfer3.rotateZ(-ChamferAdjRotZ*radian);
305:    G4Transform3D Transform_Chamfer3(Rotation_Chamfer3,Position_Chamfer3);
306:    Rotation_Chamfer3.rotateZ(ChamferAdjRotZ*radian);
307:    Rotation_Chamfer3.rotateY(-ChamferAdjRotY*radian);
308:
309:    RightQuartz_Solid[2] =  new G4SubtractionSolid ("LowerUpstreamChamfer-RightQuartzBar",
310:                                 RightQuartz_Solid[1],Chamfer_Solid,
311:                                 Transform_Chamfer3);
312:
313:    //Boolean Union:
314:    //Lower-Downstream edge chamfer
315:    ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
316:    ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
317:    Position_Chamfer4.setX(0.0*mm);
318:    Position_Chamfer4.setY(-(0.5*QuartzBar_FullHeight + delta));
319:    Position_Chamfer4.setZ(0.5*QuartzBar_FullThickness + delta);
320:    Rotation_Chamfer4.rotateX(45.0*degree);
321:    Rotation_Chamfer4.rotateY(-ChamferAdjRotY*radian);
322:    Rotation_Chamfer4.rotateZ(-ChamferAdjRotZ*radian);
323:    G4Transform3D Transform_Chamfer4(Rotation_Chamfer4,Position_Chamfer4);
324:    Rotation_Chamfer4.rotateY(ChamferAdjRotY*radian);
325:    Rotation_Chamfer4.rotateZ(ChamferAdjRotZ*radian);
326:
327:    RightQuartz_Solid[3] =  new G4SubtractionSolid ("LowerUpstreamChamfer-RightQuartzBar",
328:                                 RightQuartz_Solid[2], Chamfer_Solid,
329:                                 Transform_Chamfer4);
330:
331:
332:    QuartzBar_LogicalRight  = new G4LogicalVolume(RightQuartz_Solid[3],
333:                                QuartzBar_Material,
334:                                "QuartzBar_LogicalRight",
335:                                0,0,0);
336:
```

```
337:    G4ThreeVector Position_RightQuartzBar = G4ThreeVector(-0.5*(QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);
338:
339:    QuartzBar_PhysicalRight   = new G4PVPlacement(0,Position_RightQuartzBar,
340:                                  QuartzBar_LogicalRight,
341:                                  "QuartzBar_PhysicalRight",
342:                                  ActiveArea_Logical,
343:                                  false,0);
344:
345:  //*************************************************************************************************************
346:  //*************************************************************************************************************
347:
348:  //*************************************************************************************************************
349:  //*******************************Define Center Quartz Glue Film ************************************
350:
351:
352:    G4Box* CenterGlueFilm_Solid    = new G4Box("CenterGlueFilm_Solid",
353:                                  0.5 * GlueFilm_FullLength_X,
354:                                  0.5 * GlueFilm_FullLength_Y,
355:                                  0.5 * GlueFilm_FullLength_Z);
356:
357:    QuartzGlue_Logical  = new G4LogicalVolume(CenterGlueFilm_Solid,
358:                                 QuartzGlue_Material,
359:                                 "CenterGlueFilm_Log",
360:                                 0,0,0);
361:
362:    G4ThreeVector Position_CenterGlueFilm = G4ThreeVector(0,0,0);
363:
364:    QuartzGlue_PhysicalCenter  = new G4PVPlacement(0,Position_CenterGlueFilm,
365:                                  QuartzGlue_Logical,
366:                                  "QuartzGlue_PhysicalCenter",
367:                                  ActiveArea_Logical,
368:                                  false,0);
369:
370:  //*************************************************************************************************************
371:  //*************************************************************************************************************
372:
373:  //*************************************************************************************************************
374:  //*******************************Define Right Quartz Glue Film ************************************
375:
376:    G4ThreeVector Position_RightGlueFilm = G4ThreeVector(-1.0*(QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);
377:
378:    QuartzGlue_PhysicalRight  = new G4PVPlacement(0,Position_RightGlueFilm,
379:                                  QuartzGlue_Logical,
380:                                  "QuartzGlue_PhysicalRight",
381:                                  ActiveArea_Logical,
382:                                  false,1);
383:
384:  //*************************************************************************************************************
```

```
385: //*****************************************************************************************************************
386:
387: //*****************************************************************************************************************
388: //*******************************Define Left Detector Quartz Bar With Chamfers  ************************
389:
390:
391:   //Boolean Union:
392:   //Upper-upstream edge chamfer
393:
394:   ChamferScew = 0.021486*PI/180.0;
395:   delta = 0.5*(Chamfer_FullHeight - 1.0*mm)/sqrt(2.0);
396:   ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
397:   ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
398:   Position_Chamfer1.setX(0.0*cm);//33.333333*cm);
399:   Position_Chamfer1.setY(0.5*QuartzBar_FullHeight + delta);
400:   Position_Chamfer1.setZ(-(0.5*QuartzBar_FullThickness + delta));
401: //   Rotation_Chamfer1.rotateX(45.0*degree);
402:   Rotation_Chamfer1.rotateY(ChamferAdjRotY*radian);
403:   Rotation_Chamfer1.rotateZ(ChamferAdjRotZ*radian);
404:   G4Transform3D Transform_Chamfer5(Rotation_Chamfer1,Position_Chamfer1);
405:   Rotation_Chamfer1.rotateZ(-ChamferAdjRotZ*radian);
406:   Rotation_Chamfer1.rotateY(-ChamferAdjRotY*radian);
407:
408:   LeftQuartz_Solid[0]=  new G4SubtractionSolid ("UpperUpstreamChamfer-LeftQuartzBar",
409:                                 QuartzBar_Solid,
410:                                 Chamfer_Solid,
411:                                 Transform_Chamfer5);
412:
413:   //Boolean Union:
414:   //Upper-downstream edge chamfer
415:
416:   delta = 0.5*(Chamfer_FullHeight - 0.5*mm)/sqrt(2.0);
417:   ChamferScew = 0.0;//0.014*PI/180.0;
418:   ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
419:   ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
420:   Position_Chamfer2.setX(0.0*mm);
421:   Position_Chamfer2.setY(0.5*QuartzBar_FullHeight + delta);
422:   Position_Chamfer2.setZ(0.5*QuartzBar_FullThickness + delta);
423: //   Rotation_Chamfer2.rotateX(45.0*degree);
424:   Rotation_Chamfer2.rotateY(-ChamferAdjRotY*radian);
425:   Rotation_Chamfer2.rotateZ(ChamferAdjRotZ*radian);
426:   G4Transform3D Transform_Chamfer6(Rotation_Chamfer2,Position_Chamfer2);
427:   Rotation_Chamfer2.rotateZ(-ChamferAdjRotZ*radian);
428:   Rotation_Chamfer2.rotateY(ChamferAdjRotY*radian);
429:
430:   LeftQuartz_Solid[1] =  new G4SubtractionSolid ("UpperDownstreamChamfer-LeftQuartzBar",
431:                                  LeftQuartz_Solid[0],
432:                                  Chamfer_Solid,
```

```
                                    Transform_Chamfer6);

  //Boolean Union:
  //Lower-Upstream edge chamfer
  ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
  ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
  Position_Chamfer3.setX(0.0*mm);
  Position_Chamfer3.setY(-(0.5*QuartzBar_FullHeight + delta));
  Position_Chamfer3.setZ(-(0.5*QuartzBar_FullThickness + delta));
//   Rotation_Chamfer3.rotateX(45.0*degree);
  Rotation_Chamfer3.rotateY(ChamferAdjRotY*radian);
  Rotation_Chamfer3.rotateZ(-ChamferAdjRotZ*radian);
  G4Transform3D Transform_Chamfer7(Rotation_Chamfer3,Position_Chamfer3);
  Rotation_Chamfer3.rotateZ(ChamferAdjRotZ*radian);
  Rotation_Chamfer3.rotateY(-ChamferAdjRotY*radian);

  LeftQuartz_Solid[2] =  new G4SubtractionSolid ("LowerUpstreamChamfer-LeftQuartzBar",
                                    LeftQuartz_Solid[1],Chamfer_Solid,
                                    Transform_Chamfer7);

  //Boolean Union:
  //Lower-Downstream edge chamfer
  ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
  ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
  Position_Chamfer4.setX(0.0*mm);
  Position_Chamfer4.setY(-(0.5*QuartzBar_FullHeight + delta));
  Position_Chamfer4.setZ(0.5*QuartzBar_FullThickness + delta);
//   Rotation_Chamfer4.rotateX(45.0*degree);
  Rotation_Chamfer4.rotateY(-ChamferAdjRotY*radian);
  Rotation_Chamfer4.rotateZ(-ChamferAdjRotZ*radian);
  G4Transform3D Transform_Chamfer8(Rotation_Chamfer4,Position_Chamfer4);
  Rotation_Chamfer4.rotateY(ChamferAdjRotY*radian);
  Rotation_Chamfer4.rotateZ(ChamferAdjRotZ*radian);

  LeftQuartz_Solid[3] =  new G4SubtractionSolid ("LowerUpstreamChamfer-LeftQuartzBar",
                                    LeftQuartz_Solid[2], Chamfer_Solid,
                                    Transform_Chamfer8);


  QuartzBar_LogicalLeft  = new G4LogicalVolume(LeftQuartz_Solid[3],
                                  QuartzBar_Material,
                                  "QuartzBar_LogicalLeft",
                                  0,0,0);

  G4ThreeVector Position_LeftQuartzBar = G4ThreeVector(0.5*(QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);

  QuartzBar_PhysicalLeft   = new G4PVPlacement(0,Position_LeftQuartzBar,
                                   QuartzBar_LogicalLeft,
```

```
                                "QuartzBar_PhysicalLeft",
                                 ActiveArea_Logical,
                                false,0);

//*****************************************************************************************************************
//*****************************************************************************************************************

//*****************************************************************************************************************
//*******************************Define Left Quartz Glue Film ***************************************

  Position_RightGlueFilm = G4ThreeVector((QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);

  QuartzGlue_PhysicalRight  = new G4PVPlacement(0,Position_RightGlueFilm,
                                QuartzGlue_Logical,
                                "QuartzGlue_PhysicalRight",
                                ActiveArea_Logical,
                                false,1);

//*****************************************************************************************************************
//*****************************************************************************************************************

//*****************************************************************************************************************
//*******************************Define Light Guides With Chamfers And Any Sculpting*********************

  G4double redfr = 1.0;//0.5;
  G4double pTheta = std::atan(LightGuide_FullThickness*(1 - redfr)/(2.0*LightGuide_FullLength));

  G4Trap* LightGuide_Solid = new G4Trap("LightGuide_Solid",
                          0.5*LightGuide_FullLength,pTheta,0.0,
                          0.5*LightGuide_FullWidth1,
                          redfr*0.5*LightGuide_FullThickness,
                          redfr*0.5*LightGuide_FullThickness,0.0,
                          0.5*LightGuide_FullWidth2,
                          0.5*LightGuide_FullThickness,
                          0.5*LightGuide_FullThickness,
                          0.0);

  LGAngCutXDim = 8.0*cm;
  LGAngCutYDim = LightGuide_FullWidth1+1.0*cm;
  LGAngCutZDim = 2.0*cm;

  G4Box* LGEdgeAngleCut_Solid = new G4Box("LGEdgeAngleCut_Solid",
                           0.5*LGAngCutXDim,
                           0.5*LGAngCutYDim,
                           0.5*LGAngCutZDim);
  Double_t ad = 0.0;//45.0;
  Double_t ar = ad*4.0*std::atan(1.0)/180.0;
  Double_t dx = 0.5*LGAngCutZDim*std::cos(ar)-0.5*(LightGuide_FullThickness -
```

```
                                    LGAngCutZDim*std::sin(ar))*std::tan(ar)
    + LightGuide_FullThickness*(1 - redfr)*std::tan(ar);


//*******************************Left Light Guide *****************************************************
  //Boolean Union:
  //Left Light Guide Angular cut-off at edge
  Position_AngCut1.setX(0.0*cm);
  Position_AngCut1.setY(0.0*cm);
  Position_AngCut1.setZ(-(0.5*LightGuide_FullLength+dx));
  Rotation_AngCut1.rotateY(ad*degree);
  G4Transform3D Transform_AngCut1(Rotation_AngCut1,Position_AngCut1);

  LeftGuide_Solid[0] =  new G4SubtractionSolid ("LGLeft-AngCut",
                                 LightGuide_Solid,
                                 LGEdgeAngleCut_Solid,
                                 Transform_AngCut1);

  delta = 0.5*(Chamfer_FullHeight - 0.5*mm)/sqrt(2.0);

  Position_Chamfer1.setX(-(0.5*QuartzBar_FullThickness + delta));
  Position_Chamfer1.setY(0.5*QuartzBar_FullHeight + delta);
  Position_Chamfer1.setZ(0.0);
  Rotation_Chamfer1.rotateY(90.0*degree);
  G4Transform3D Transform_Chamfer9(Rotation_Chamfer1,Position_Chamfer1);

  LeftGuide_Solid[1]=  new G4SubtractionSolid ("LeftLGChamfer1",
                                 LeftGuide_Solid[0],
                                 Chamfer_Solid,
                                 Transform_Chamfer9);


  Position_Chamfer2.setX(0.5*QuartzBar_FullThickness + delta);
  Position_Chamfer2.setY(0.5*QuartzBar_FullHeight + delta);
  Position_Chamfer2.setZ(0.0*cm);
  Rotation_Chamfer2.rotateY(90.0*degree);
  G4Transform3D Transform_Chamfer10(Rotation_Chamfer2,Position_Chamfer2);

  LeftGuide_Solid[2]=  new G4SubtractionSolid ("LeftLGChamfer2",
                                 LeftGuide_Solid[1],
                                 Chamfer_Solid,
                                 Transform_Chamfer10);


  Position_Chamfer3.setX(0.5*QuartzBar_FullThickness + delta);
  Position_Chamfer3.setY(-(0.5*QuartzBar_FullHeight + delta));
```

```
577:    Position_Chamfer3.setZ(0.0*cm);
578:    Rotation_Chamfer3.rotateY(90.0*degree);
579:    G4Transform3D Transform_Chamfer11(Rotation_Chamfer3,Position_Chamfer3);
580:
581:    LeftGuide_Solid[3]=  new G4SubtractionSolid ("LeftLGChamfer3",
582:                                 LeftGuide_Solid[2],
583:                                 Chamfer_Solid,
584:                                 Transform_Chamfer11);
585:
586:    Position_Chamfer4.setX(-(0.5*QuartzBar_FullThickness + delta));
587:    Position_Chamfer4.setY(-(0.5*QuartzBar_FullHeight + delta));
588:    Position_Chamfer4.setZ(0.0*cm);
589:    Rotation_Chamfer4.rotateY(90.0*degree);
590:    G4Transform3D Transform_Chamfer12(Rotation_Chamfer4,Position_Chamfer4);
591:
592:    LeftGuide_Solid[4]=  new G4SubtractionSolid ("LeftLGChamfer4",
593:                                 LeftGuide_Solid[3],
594:                                 Chamfer_Solid,
595:                                 Transform_Chamfer12);
596:
597:
598:
599:  //*******************************Right Light Guide ******************************************************
600:
601:
602:    //Boolean Union:
603:    //Right Light Guide Angular cut-off at edge
604:    Position_AngCut2.setX(0.0*cm);
605:    Position_AngCut2.setY(0.0*cm);
606:    Position_AngCut2.setZ(-(0.5*LightGuide_FullLength+dx));
607:    Rotation_AngCut2.rotateY(-ad*degree);
608:    G4Transform3D Transform_AngCut2(Rotation_AngCut2,Position_AngCut2);
609:
610:    RightGuide_Solid[0] =  new G4SubtractionSolid ("LGRight-AngCut",
611:                                  LightGuide_Solid,
612:                                  LGEdgeAngleCut_Solid,
613:                                  Transform_AngCut2);
614:
615:    G4Transform3D Transform_Chamfer13(Rotation_Chamfer1,Position_Chamfer1);
616:
617:    RightGuide_Solid[1]=  new G4SubtractionSolid ("RightLGChamfer1",
618:                                  RightGuide_Solid[0],
619:                                  Chamfer_Solid,
620:                                  Transform_Chamfer13);
621:
622:
623:    G4Transform3D Transform_Chamfer14(Rotation_Chamfer2,Position_Chamfer2);
624:
```

```
625:    RightGuide_Solid[2]=  new G4SubtractionSolid ("RightLGChamfer2",
626:                                  RightGuide_Solid[1],
627:                                  Chamfer_Solid,
628:                                  Transform_Chamfer14);
629:
630:
631:    G4Transform3D Transform_Chamfer15(Rotation_Chamfer3,Position_Chamfer3);
632:
633:    RightGuide_Solid[3]=  new G4SubtractionSolid ("RightLGChamfer3",
634:                                  RightGuide_Solid[2],
635:                                  Chamfer_Solid,
636:                                  Transform_Chamfer15);
637:
638:    G4Transform3D Transform_Chamfer16(Rotation_Chamfer4,Position_Chamfer4);
639:
640:    RightGuide_Solid[4]=  new G4SubtractionSolid ("RightLGChamfer4",
641:                                 RightGuide_Solid[3],
642:                                 Chamfer_Solid,
643:                                 Transform_Chamfer16);
644:
645:  //*****************************************************************************************************************
646:  //*****************************************************************************************************************
647:
648:
649:
650:    //Boolean Union:
651:    //Left Light Guide
652:    Position_LGLeft.setX((QuartzBar_FullLength+0.5*LightGuide_FullLength+1.5*GlueFilm_FullLength_X));
653:    Position_LGLeft.setY(0.0*cm);
654:    Position_LGLeft.setZ(0.0*cm - LightGuide_FullThickness*(1 - redfr)/(4.0));
655:    Rotation_LGLeft.rotateY(-90.0*degree);
656:    G4Transform3D Transform_LGLeft(Rotation_LGLeft,Position_LGLeft);
657:
658:    //Boolean Union:
659:    //Right Light Guide
660:    Position_LGRight.setX(-(QuartzBar_FullLength+0.5*LightGuide_FullLength+1.5*GlueFilm_FullLength_X));
661:    Position_LGRight.setY(0.0*cm);
662:    Position_LGRight.setZ(0.0*cm - LightGuide_FullThickness*(1 - redfr)/(4.0));
663:  //   Rotation_LGRight.rotateY(-90.0*degree);
664:    Rotation_LGRight.rotateY(90.0*degree);
665:  //   Rotation_LGRight.rotateZ(180.0*degree);
666:    G4Transform3D Transform_LGRight(Rotation_LGRight,Position_LGRight);
667:
668:
669:    LightGuide_LogicalLeft  = new G4LogicalVolume(LeftGuide_Solid[4],
670:                                  LightGuide_Material,
671:                                  "LightGuide_LogicalLeft",
672:                                  0,0,0);
```

```
673:
674:
675:    LightGuide_PhysicalLeft = new G4PVPlacement(Transform_LGLeft,
676:                                    LightGuide_LogicalLeft,
677:                                    "LightGuide_PhysicalLeft",
678:                                    ActiveArea_Logical,
679:                                    false,0);
680:
681:
682:    LightGuide_LogicalRight  = new G4LogicalVolume(RightGuide_Solid[4],
683:                                     LightGuide_Material,
684:                                     "LightGuide_LogicalRight",
685:                                     0,0,0);
686:
687:
688:    LightGuide_PhysicalRight = new G4PVPlacement(Transform_LGRight,
689:                                    LightGuide_LogicalRight,
690:                                    "LightGuide_PhysicalRight",
691:                                    ActiveArea_Logical,
692:                                    false,0);
693:
694:    //*************************************************************************************************************
695:    //*******************************Face Mirrors*****************************************************************
696:
697:   //   G4Trd* LGFaceMirror_Solid = new G4Trd("LGFaceMirror_Solid",
698:   //                                 0.1*mm,0.1*mm,
699:   //                                 0.5*LightGuide_FullWidth1,
700:   //                                 0.5*LightGuide_FullWidth2,
701:   //                                 0.5*LightGuide_FullLength -
702:   //                                 0.5*LightGuide_FullThickness*std::tan(ar)+
703:   //                                 0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
704:
705:
706:   //   Position_LGFaceMirrorLeft.setX(0.5*(QuartzBar_FullLength+LightGuide_FullLength)-
707:   //                              0.5*LightGuide_FullThickness*std::tan(ar)+
708:   //                              0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
709:   //   Position_LGFaceMirrorLeft.setY(0.0*cm);
710:   //   Position_LGFaceMirrorLeft.setZ(-0.5*LightGuide_FullThickness - 0.1*mm);
711:   //   Rotation_LGFaceMirrorLeft.rotateY(-90.0*degree);
712:   //   G4Transform3D Transform_LGFMLeft(Rotation_LGFaceMirrorLeft,Position_LGFaceMirrorLeft);
713:
714:
715:   //   mirror_logical[0]  = new G4LogicalVolume(LGFaceMirror_Solid,
716:   //                                    mirror_material,
717:   //                                    "mirrorface_log1",
718:   //                                    0,0,0);
719:
720:   //   mirror_physical[0] = new G4PVPlacement(Transform_LGFMLeft,
```

```
721: //                          mirror_logical[0],
722: //                          "mirrorface_physical1",
723: //                          CerenkovContainer_Logical,
724: //                          false,
725: //                          0); // copy number for left PMTContainer
726:
727: //*******************************Face Mirrors*************************************************************
728: //*****************************************************************************************************************
729:
730:
731:
732:
733:
734:
735: //*****************************************************************************************************************
736: //*******************************Edge Mirrors*************************************************************
737:
738:
739:    G4Box* LGEdgeMirror_Solid = new G4Box("LGEdgeMirror_Solid",
740:                                0.1*mm,0.5*LightGuide_FullWidth1,
741:                                redfr*0.5*LightGuide_FullThickness/std::cos(ar));
742:
743:    Position_LGEdgeMirrorLeft.setX(1.5*GlueFilm_FullLength_X + QuartzBar_FullLength+LightGuide_FullLength+0.1*mm/std::cos(ar)-
744:                                0.5*LightGuide_FullThickness*std::tan(ar)+
745:                           0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
746:    Position_LGEdgeMirrorLeft.setY(0.0*cm);
747:    Position_LGEdgeMirrorLeft.setZ(-0.5*LightGuide_FullThickness*(1-redfr));
748:    Rotation_LGEdgeMirrorLeft.rotateY(ad*degree);
749:    G4Transform3D Transform_LGEMLeft(Rotation_LGEdgeMirrorLeft,Position_LGEdgeMirrorLeft);
750:
751:
752:    mirror_logical[1]  = new G4LogicalVolume(LGEdgeMirror_Solid,
753:                                mirror_material,
754:                                "mirrorface_log2",
755:                                0,0,0);
756:
757:    mirror_physical[1] = new G4PVPlacement(Transform_LGEMLeft,
758:                                mirror_logical[1],
759:                                "mirrorface_physical2",
760:                                ActiveArea_Logical,
761:                                false,
762:                                0); // copy number for left PMTContainer
763:
764:
765:
766:
767:
768:    Position_LGEdgeMirrorRight.setX(-1.5*GlueFilm_FullLength_X-QuartzBar_FullLength-LightGuide_FullLength-0.1*mm/std::cos(ar)+
```

```
769:                        0.5*LightGuide_FullThickness*std::tan(ar)-
770:                        0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
771:    Position_LGEdgeMirrorRight.setY(0.0*cm);
772:    Position_LGEdgeMirrorRight.setZ(-0.5*LightGuide_FullThickness*(1-redfr));
773:    Rotation_LGEdgeMirrorRight.rotateY(-ad*degree);
774:    G4Transform3D Transform_LGEMRight(Rotation_LGEdgeMirrorRight,Position_LGEdgeMirrorRight);
775:
776:
777:    mirror_logical[3]  = new G4LogicalVolume(LGEdgeMirror_Solid,
778:                              mirror_material,
779:                              "mirrorface_log4",
780:                              0,0,0);
781:
782:    mirror_physical[3] = new G4PVPlacement(Transform_LGEMRight,
783:                             mirror_logical[3],
784:                             "mirrorface_physical4",
785:                             ActiveArea_Logical,
786:                             false,
787:                             0); // copy number for left PMTContainer
788:
789:  //*******************************Edge Mirrors*************************************************************
790:  //*****************************************************************************************************************
791:
792:
793:
794:
795:  //*****************************************************************************************************************
796:  //*******************************Radiator*************************************************************************
797:
798:
799:  //   G4Box* RadiatorSolid = new G4Box("Radiator_Sol",
800:  //                            0.5 * QuartzBar_FullLength,       // half X length required by Geant4
801:  //                            0.5 * QuartzBar_FullHeight,      // half Y length required by Geant4
802:  //                            1.0*cm );  // half Z length required by Geant4
803:
804:  //   Radiator_Logical  = new G4LogicalVolume(RadiatorSolid,
805:  //                                Radiator_Material,
806:  //                                "Radiator_Log",
807:  //                                0,0,0);
808:
809:  //   G4ThreeVector Position_Radiator  = G4ThreeVector(0,0,2.0*cm);//-2.0*cm);
810:
811:  //   Radiator_Physical   = new G4PVPlacement(0,Position_Radiator,
812:  //                                Radiator_Logical,
813:  //                                "Radiator_Physical",
814:  //                                CerenkovContainer_Logical,
815:  //                                false,
816:  //                                0);
```

```
817:
818:  //*****************************Radiator*****************************************************************
819:  //******************************************************************************************************
820:
821:
822:    //-----------------------------------
823:    // define the PMTContainer
824:    //-----------------------------------
825:
826:    G4double mypi   = 4.0*std::atan(1.0);
827:    G4double thetaY = std::atan(LightGuide_FullThickness*(1 - redfr)/(LightGuide_FullLength));
828:    G4double Xoffs = 0.0*cm;//7.0*cm;
829:
830:    //Flat on guide face configuration
831:    G4double PMTContXShift = QuartzBar_FullLength + LightGuide_FullLength - 0.5*PMTEntranceWindow_Diameter - Xoffs;
832:    G4double PMTContYShift = 0.0;
833:    G4double PMTContZShift = 0.5*QuartzBar_FullThickness + 0.5*PMTContainer_FullLength_Z
834:      - (LightGuide_FullLength - 0.5*PMTEntranceWindow_Diameter-Xoffs)*std::tan(thetaY);
835:
836:    // relocation of the left Photon Detector Container
837:    Translation_PMTContainerLeft.setX(1.0*PMTContXShift);
838:    Translation_PMTContainerLeft.setY(1.0*PMTContYShift);
839:    Translation_PMTContainerLeft.setZ(1.0*PMTContZShift);
840:
841:  //   //On guide edge configuration
842:  //   Rotation_PMTContainerLeft.rotateY(90.0*degree);
843:
844:    //Flat on guide face configuration
845:    Rotation_PMTContainerLeft.rotateY(thetaY*180.0/mypi*degree);
846:    G4Transform3D Transform3D_PMTContainerLeft(Rotation_PMTContainerLeft,
847:                                  Translation_PMTContainerLeft);
848:
849:    // relocation of the right Photon Detector Container
850:    Translation_PMTContainerRight.setX(-1.0*PMTContXShift);
851:    Translation_PMTContainerRight.setY(1.0*PMTContYShift);
852:    Translation_PMTContainerRight.setZ(1.0*PMTContZShift);
853:
854:  //   //On guide edge configuration
855:  //   Rotation_PMTContainerLeft.rotateY(-90.0*cm);
856:
857:    //Flat on guide face configuration
858:    Rotation_PMTContainerRight.rotateY(-thetaY*180.0/mypi*degree);
859:    G4Transform3D Transform3D_PMTContainerRight(Rotation_PMTContainerRight,
860:                                   Translation_PMTContainerRight);
861:
862:
863:
864:    G4double PMTQuartzOpticalFilmZShift = 0.5*(PMTQuartzOpticalFilm_Thickness - PMTContainer_FullLength_Z);
```

```
865:
866:    // relocation of the PMTEntranceWindow
867:    Translation_PMTQuartzOpticalFilm.setX(0.0*cm);
868:    Translation_PMTQuartzOpticalFilm.setY(0.0*cm);
869:    Translation_PMTQuartzOpticalFilm.setZ(PMTQuartzOpticalFilmZShift);
870:
871:    //------------------------------------------------------------------------------
872:    // location and orientation of the PMT Entrance Window within the PMT Container
873:    //------------------------------------------------------------------------------
874:
875:    G4double PMTEntWindZShift = 0.5*(PMTEntranceWindow_Thickness - PMTContainer_FullLength_Z)+PMTQuartzOpticalFilm_Thickness;
876:
877:    // relocation of the PMTEntranceWindow
878:    Translation_PMTEntranceWindow.setX(0.0*cm);
879:    Translation_PMTEntranceWindow.setY(0.0*cm);
880:    Translation_PMTEntranceWindow.setZ(PMTEntWindZShift);
881:
882:    //------------------------------------------------------------------------------
883:    // location and orientation of the cathode WITHIN the PMT
884:    //------------------------------------------------------------------------------
885:
886:    G4double CathodeZShift = PMTEntranceWindow_Thickness + 0.5*(Cathode_Thickness - PMTContainer_FullLength_Z);
887:
888:    // location of the Photon Detector relative to  Photon Detector Container
889:    Translation_Cathode.setX(0.0*cm);
890:    Translation_Cathode.setY(0.0*cm);
891:    Translation_Cathode.setZ(CathodeZShift);
892:
893:
894:    //   G4Box* PMTContainer_Solid    = new G4Box("PMTContainer_Solid",
895:    //                                 0.5 * PMTContainer_FullLength_X,   // half X
896:    //                                 0.5 * PMTContainer_FullLength_Y ,  // half Y
897:    //                                 0.5 * PMTContainer_FullLength_Z);  // half Z
898:    G4Tubs* PMTContainer_Solid    = new G4Tubs("PMTContainer_Solid",0.0*cm,
899:                                  0.5 * PMTContainer_Diameter,
900:                                  0.5 * PMTContainer_FullLength_Z,
901:                                  0.0*degree,360.0*degree);
902:
903:
904:    PMTContainer_Logical  = new G4LogicalVolume(PMTContainer_Solid,
905:                                  PMTContainer_Material,
906:                                  "PMTContainer_Log",
907:                                  0,0,0);
908:
909:    // left side
910:    PMTContainer_PhysicalLeft  = new G4PVPlacement(Transform3D_PMTContainerLeft,
911:                                    PMTContainer_Logical,
912:                                    "PMTContainer_Physical",
```

```
                                ActiveArea_Logical,
                                false,
                                0); // copy number for left PMTContainer

  // right side
  PMTContainer_PhysicalRight = new G4PVPlacement(Transform3D_PMTContainerRight,
                               PMTContainer_Logical,
                               "PMTContainer_Physical",
                               ActiveArea_Logical,
                               false,
                               1); // copy number for right PMTContainer


  //---------------------------------------
  // define the glue or grease or cookie layer
  //---------------------------------------

  G4Tubs* PMTQuartzOpticalFilm_Solid = new G4Tubs("PMTQuartzOpticalFilm_Solid",0.0*cm,
                                0.5*PMTQuartzOpticalFilm_Diameter,
                                0.5*PMTQuartzOpticalFilm_Thickness,
                                0.0*degree,360.0*degree);

  PMTQuartzOpticalFilm_Logical  = new G4LogicalVolume(PMTQuartzOpticalFilm_Solid,
                                 PMTQuartzOpticalFilm_Material,
                                 "PMTQuartzOpticalFilm_Log",
                                 0,0,0);
  PMTQuartzOpticalFilm_Physical = new G4PVPlacement(0,Translation_PMTQuartzOpticalFilm,
                               PMTQuartzOpticalFilm_Logical,
                               "PMTQuartzOpticalFilm_Physical",
                               PMTContainer_Logical,
                               false, 0); // copy number for left photon detector


  //---------------------------------------
  // define the PMTEntranceWindow
  //---------------------------------------

  G4Tubs* PMTEntranceWindow_Solid = new G4Tubs("PMTEntranceWindow_Solid",0.0*cm,
                              0.5*PMTEntranceWindow_Diameter,
                              0.5*PMTEntranceWindow_Thickness,
                              0.0*degree,360.0*degree);

  PMTEntranceWindow_Logical  = new G4LogicalVolume(PMTEntranceWindow_Solid,
                                 PMTEntranceWindow_Material,
                                 "PMTEntranceWindow_Log",
                                 0,0,0);
```

```
961:    PMTEntranceWindow_Physical = new G4PVPlacement(0,Translation_PMTEntranceWindow,
962:                                 PMTEntranceWindow_Logical,
963:                                 "PMTEntranceWindow_Physical",
964:                                 PMTContainer_Logical,
965:                                 false, 0); // copy number for left photon detector
966:
967:    //---------------------------
968:    // define the Photon Detector
969:    //---------------------------
970:
971:    G4Tubs* Cathode_Solid = new G4Tubs("Cathode_Solid",0.0*cm,0.5*Cathode_Diameter,
972:                          0.5*Cathode_Thickness,0.0*degree,360.0*degree);
973:
974:    Cathode_Logical  = new G4LogicalVolume(Cathode_Solid,Cathode_Material,"Cathode_Log",0,0,0);
975:
976:    Cathode_Physical = new G4PVPlacement(0,Translation_Cathode,Cathode_Logical,"Cathode_Physical",PMTContainer_Logical,
977:                           false, 0); // copy number for left photon detector
978:
979:
980:
981:
982:
983:    //=============================================================================================
984:
985:
986:
987:
988:
989:    //=============================================================================================
990:
991:     Position_CerenkovContainer  = G4ThreeVector(Position_CerenkovContainer_X,
992:                                   Position_CerenkovContainer_Y,
993:                                   Position_CerenkovContainer_Z);
994:
995:    // define Rotation matrix for Container orientated in MotherVolume
996:    Rotation_CerenkovContainer =  new G4RotationMatrix();
997:    Rotation_CerenkovContainer -> rotateX(Tilting_Angle);
998:
999:   //   CerenkovContainer_Physical   = new G4PVPlacement(Rotation_CerenkovContainer,
1000:  //                                  Position_CerenkovContainer,
1001:  //                                  "CerenkovContainer_Physical",
1002:  //                                  CerenkovContainer_Logical,
1003:  //                                  MotherVolume,
1004:  //                                  false,
1005:  //                                  0);
1006:
1007:
1008:   //-----------------------------------------------
```

```
1009:
1010:
1011:    //============================================================================
1012:    // place the 8 CerenkovMasterContainer_Physical into the physical MotherVolume
1013:    //============================================================================
1014:    //
1015:    PlacePVCerenkovMasterContainer();
1016:
1017:
1018:
1019:   //-------------
1020:    const G4int nEntries = 9;
1021:
1022:    G4double PhotonEnergy[nEntries] =
1023:      {
1024:        1.54986*eV,  // 800 nanometer
1025:        1.77127*eV,  // 700 nanometer
1026:        2.06648*eV,  // 600 nanometer
1027:        2.47978*eV,  // 500 nanometer
1028:        3.09973*eV,  // 400 nanometer
1029:        4.13297*eV,  // 300 nanometer
1030:        4.95956*eV,  // 250 nanometer
1031:        5.51063*eV,  // 225 nanometer
1032:        5.90424*eV   // 210 nanometer
1033:      };
1034:
1035:    G4double Reflectivity[nEntries];
1036:
1037:    G4double mylambda;
1038:
1039:    for (G4int kk= 0; kk < nEntries; kk++) {
1040:      // Nevens empiric formular for the reflectivity
1041:      // lamda = h*c/E
1042:
1043:      mylambda = (h_Planck*c_light/PhotonEnergy[kk])/nanometer;
1044:
1045:      Reflectivity[kk] =  1.0 -0.027*exp(-0.004608*mylambda);
1046:      //Reflectivity[kk] =  1.0;
1047:    };
1048:
1049:    G4OpticalSurface* QuartzBarLeft_OpticalSurface = new G4OpticalSurface("QuartzBarLeftOpticalSurface");
1050:    G4OpticalSurface* QuartzBarRight_OpticalSurface = new G4OpticalSurface("QuartzBarRightOpticalSurface");
1051:    G4OpticalSurface* LightGuideLeft_OpticalSurface = new G4OpticalSurface("LightGuideLeftOpticalSurface");
1052:    G4OpticalSurface* LightGuideRight_OpticalSurface = new G4OpticalSurface("LightGuideRightOpticalSurface");
1053:
1054:    G4OpticalSurface* GlueFilmRight_OpticalSurface = new G4OpticalSurface("GlueFilmRightOpticalSurface");
1055:    G4OpticalSurface* GlueFilmCenter_OpticalSurface = new G4OpticalSurface("GlueFilmCenterOpticalSurface");
1056:    G4OpticalSurface* GlueFilmLeft_OpticalSurface = new G4OpticalSurface("GlueFilmLeftOpticalSurface");
```

```
1057:
1058:
1059:   G4LogicalBorderSurface* QuartzBarLeft_BorderSurface   = new G4LogicalBorderSurface("QuartzBarLeft_BorderSurface",
1060:                                                           QuartzBar_PhysicalLeft,
1061:                                                           ActiveArea_Physical,
1062:                                                           QuartzBarLeft_OpticalSurface);
1063:   G4LogicalBorderSurface* QuartzBarRight_BorderSurface  = new G4LogicalBorderSurface("QuartzBarRight_BorderSurface",
1064:                                                           QuartzBar_PhysicalRight,
1065:                                                           ActiveArea_Physical,
1066:                                                           QuartzBarRight_OpticalSurface);
1067:   G4LogicalBorderSurface* LightGuideLeft_BorderSurface  = new G4LogicalBorderSurface("LightGuideLeft_BorderSurface",
1068:                                                           LightGuide_PhysicalLeft,
1069:                                                           ActiveArea_Physical,
1070:                                                           LightGuideLeft_OpticalSurface);
1071:   G4LogicalBorderSurface* LightGuideRight_BorderSurface = new G4LogicalBorderSurface("LightGuideRight_BorderSurface",
1072:                                                           LightGuide_PhysicalRight,
1073:                                                           ActiveArea_Physical,
1074:                                                           LightGuideRight_OpticalSurface);
1075:   G4LogicalBorderSurface* GlueFilmRight_BorderSurface   = new G4LogicalBorderSurface("GlueFilmRight_BorderSurface",
1076:                                                           QuartzGlue_PhysicalRight,
1077:                                                           ActiveArea_Physical,
1078:                                                           GlueFilmRight_OpticalSurface);
1079:   G4LogicalBorderSurface* GlueFilmCenter_BorderSurface  = new G4LogicalBorderSurface("GlueFilmCenter_BorderSurface",
1080:                                                           QuartzGlue_PhysicalCenter,
1081:                                                           ActiveArea_Physical,
1082:                                                           GlueFilmCenter_OpticalSurface);
1083:   G4LogicalBorderSurface* GlueFilmLeft_BorderSurface    = new G4LogicalBorderSurface("GlueFilmLeft_BorderSurface",
1084:                                                           QuartzGlue_PhysicalLeft,
1085:                                                           ActiveArea_Physical,
1086:                                                           GlueFilmLeft_OpticalSurface);
1087:
1088:   QuartzBarLeft_OpticalSurface->SetType(dielectric_dielectric);
1089:   QuartzBarLeft_OpticalSurface->SetFinish(polished);
1090:   QuartzBarLeft_OpticalSurface->SetPolish(0.997);
1091:   QuartzBarLeft_OpticalSurface->SetModel(glisur);
1092:
1093:   QuartzBarRight_OpticalSurface->SetType(dielectric_dielectric);
1094:   QuartzBarRight_OpticalSurface->SetFinish(polished);
1095:   QuartzBarRight_OpticalSurface->SetPolish(0.997);
1096:   QuartzBarRight_OpticalSurface->SetModel(glisur);
1097:
1098:   LightGuideLeft_OpticalSurface->SetType(dielectric_dielectric);
1099:   LightGuideLeft_OpticalSurface->SetFinish(polished);
1100:   LightGuideLeft_OpticalSurface->SetPolish(0.997);
1101:   LightGuideLeft_OpticalSurface->SetModel(glisur);
1102:
1103:   LightGuideRight_OpticalSurface->SetType(dielectric_dielectric);
1104:   LightGuideRight_OpticalSurface->SetFinish(polished);
```

```
1105:    LightGuideRight_OpticalSurface->SetPolish(0.997);
1106:    LightGuideRight_OpticalSurface->SetModel(glisur);
1107:
1108:    GlueFilmLeft_OpticalSurface->SetType(dielectric_dielectric);
1109:    GlueFilmLeft_OpticalSurface->SetFinish(polished);
1110:    GlueFilmLeft_OpticalSurface->SetPolish(0.9);
1111:    GlueFilmLeft_OpticalSurface->SetModel(glisur);
1112:
1113:    GlueFilmCenter_OpticalSurface->SetType(dielectric_dielectric);
1114:    GlueFilmCenter_OpticalSurface->SetFinish(polished);
1115:    GlueFilmCenter_OpticalSurface->SetPolish(0.9);
1116:    GlueFilmCenter_OpticalSurface->SetModel(glisur);
1117:
1118:    GlueFilmRight_OpticalSurface->SetType(dielectric_dielectric);
1119:    GlueFilmRight_OpticalSurface->SetFinish(polished);
1120:    GlueFilmRight_OpticalSurface->SetPolish(0.9);
1121:    GlueFilmRight_OpticalSurface->SetModel(glisur);
1122:
1123:    G4MaterialPropertiesTable *quartzST = new G4MaterialPropertiesTable();
1124:
1125:    quartzST->AddProperty("REFLECTIVITY",  PhotonEnergy , Reflectivity, nEntries);
1126:    QuartzBarLeft_OpticalSurface->SetMaterialPropertiesTable(quartzST);
1127:    QuartzBarRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);
1128:    LightGuideLeft_OpticalSurface->SetMaterialPropertiesTable(quartzST);
1129:    LightGuideRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);
1130:    GlueFilmRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);
1131:
1132:
1133:  //   G4OpticalSurface* ActiveArea_OpticalSurface = new G4OpticalSurface("ActiveAreaOpticalSurface");
1134:
1135:  //   G4LogicalBorderSurface* ActiveArea_BorderSurface    = new G4LogicalBorderSurface("ActiveArea_BorderSurface",
1136:  //                                                         ActiveArea_Physical,
1137:  //                                                         CerenkovContainer_Physical,
1138:  //                                                         ActiveArea_OpticalSurface);
1139:
1140:  //   ActiveArea_OpticalSurface->SetFinish(groundbackpainted); //new for wrapping test
1141:  // //   ActiveArea_OpticalSurface->SetPolish(0.0);          //new for wrapping test
1142:  // //   ActiveArea_OpticalSurface->SetModel(glisur);      //new for wrapping test
1143:  //   ActiveArea_OpticalSurface->SetModel(unified);         //new for wrapping test
1144:  //   ActiveArea_OpticalSurface->SetSigmaAlpha(0.25);       //new for wrapping test
1145:
1146:  //   G4double RefractiveIndex_Air[nEntries] = {1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0}; //new for wrapping test
1147:  //   G4double MilliPoreRefl[nEntries] = {0.96,0.96,0.96,0.96,0.96,0.96,0.96,0.96,0.96,0.96};       //new for wrapping test
1148:  //   G4double specularlobe[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};        //new for wrapping test
1149:  //   G4double specularspike[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};       //new for wrapping test
1150:  //   G4double backscatter[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};         //new for wrapping test
1151:
1152:  //   G4MaterialPropertiesTable *myST = new G4MaterialPropertiesTable();
```

```
1153:
1154:  //   myST->AddProperty("RINDEX",  PhotonEnergy , RefractiveIndex_Air, nEntries);     //new for wrapping test
1155:  //   myST->AddProperty("REFLECTIVITY",  PhotonEnergy , MilliPoreRefl, nEntries);     //new for wrapping test
1156:  //   myST->AddProperty("SPECULARLOBECONSTANT", PhotonEnergy ,specularlobe,nEntries); //new for wrapping test
1157:  //   myST->AddProperty("SPECULARSPIKECONSTANT",PhotonEnergy,specularspike,nEntries); //new for wrapping test
1158:  //   myST->AddProperty("BACKSCATTERCONSTANT",PhotonEnergy,backscatter,nEntries);     //new for wrapping test
1159:  // //   myST->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_Air, nEntries);    //new for wrapping test
1160:
1161:
1162:  //   ActiveArea_OpticalSurface->SetMaterialPropertiesTable(myST);
1163:
1164:    // Sensitive detectors
1165:    //------------------------------------------------
1166:    // All managed (deleted) by SDManager
1167:
1168:    G4SDManager* SDman = G4SDManager::GetSDMpointer();
1169:
1170:
1171:    //*************************************************************
1172:    CerenkovDetectorSD = new QweakSimCerenkov_DetectorSD("/CerenkovDetectorSD");
1173:    SDman->AddNewDetector(CerenkovDetectorSD);
1174:
1175:    // add Cerenkov detector as a sensitiv element
1176:    ActiveArea_Logical->SetSensitiveDetector(CerenkovDetectorSD);
1177:    //*************************************************************
1178:
1179:    //*************************************************************
1180:    CerenkovDetector_PMTSD = new QweakSimCerenkovDetector_PMTSD("/CerenkovPMTSD",myUserInfo);
1181:    SDman->AddNewDetector(CerenkovDetector_PMTSD);
1182:
1183:    // add PMT as a sensitiv element
1184:    Cathode_Logical->SetSensitiveDetector(CerenkovDetector_PMTSD);
1185:    //     PMTEntranceWindow_Logical->SetSensitiveDetector(CerenkovDetector_PMTSD);
1186:    //*************************************************************
1187:
1188:
1189:   G4cout << G4endl << "###### QweakSimCerenkovDetector: Setting Attributes " << G4endl << G4endl;
1190:
1191:    G4Colour  orange    ( 255/255., 127/255.,   0/255.);
1192:    G4Colour  blue      (   0/255.,   0/255., 255/255.);
1193:    G4Colour  magenta   ( 255/255.,   0/255., 255/255.);
1194:    G4Colour  grey      ( 127/255., 127/255., 127/255.);
1195:    G4Colour  lightblue   ( 139/255., 208/255., 255/255.);
1196:    G4Colour  lightorange ( 255/255., 189/255., 165/255.);
1197:    G4Colour  khaki3    ( 205/255., 198/255., 115/255.);
1198:    //------------------------------------------
1199:    // Visual Attributes for:  CerenkovContainer
1200:    //------------------------------------------
```

```
1201:   G4VisAttributes* CerenkovContainerVisAtt = new G4VisAttributes(blue);
1202:   CerenkovContainerVisAtt->SetVisibility(false);
1203:   //CerenkovContainerVisAtt->SetVisibility(true);
1204:   //CerenkovContainerVisAtt->SetForceWireframe(true);
1205:   //CerenkovContainerVisAtt->SetForceSolid(true);
1206:   CerenkovContainer_Logical->SetVisAttributes(CerenkovContainerVisAtt);
1207:   ActiveArea_Logical->SetVisAttributes(CerenkovContainerVisAtt);
1208:
1209:   //---------------------------------------
1210:   // Visual Attributes for:  CerenkovDetector
1211:   //---------------------------------------
1212:   G4VisAttributes* CerenkovDetectorVisAtt = new G4VisAttributes(orange);
1213:   CerenkovDetectorVisAtt->SetVisibility(true);
1214:   // Needed for the correct visualization using Coin3D
1215:   //CerenkovDetectorVisAtt->SetForceSolid(true);
1216:   CerenkovDetectorVisAtt->SetForceWireframe(true);
1217:  //  ActiveArea_Logical->SetVisAttributes(CerenkovDetectorVisAtt);
1218:   QuartzBar_LogicalLeft->SetVisAttributes(CerenkovDetectorVisAtt);
1219:   QuartzBar_LogicalRight->SetVisAttributes(CerenkovDetectorVisAtt);
1220:   LightGuide_LogicalLeft->SetVisAttributes(CerenkovDetectorVisAtt);
1221:   LightGuide_LogicalRight->SetVisAttributes(CerenkovDetectorVisAtt);
1222:   QuartzGlue_Logical->SetVisAttributes(CerenkovDetectorVisAtt);
1223:
1224:   //-----------------------------------------------
1225:   // Visual Attributes for:  PMTContainer
1226:   //-----------------------------------------------
1227:   G4VisAttributes* PMTContainerVisAtt = new G4VisAttributes(blue);
1228:   PMTContainerVisAtt->SetVisibility(true);
1229:   PMTContainerVisAtt->SetForceWireframe(true);
1230:   //PMTContainerVisAtt->SetForceSolid(true);
1231:   PMTContainer_Logical->SetVisAttributes(PMTContainerVisAtt);
1232:
1233:   //-------------------------------------------------
1234:   // Visual Attributes for:  PMTEntranceWindow
1235:   //-------------------------------------------------
1236:   G4VisAttributes* PMTEntranceWindowVisAtt = new G4VisAttributes(grey);
1237:   PMTEntranceWindowVisAtt->SetVisibility(true);
1238:   //PMTEntranceWindowVisAtt->SetForceWireframe(true);
1239:   PMTEntranceWindowVisAtt->SetForceSolid(true);
1240:   PMTEntranceWindow_Logical->SetVisAttributes(PMTEntranceWindowVisAtt);
1241:
1242:   //--------------------------------------
1243:   // Visual Attributes for:  PMT
1244:   //--------------------------------------
1245:    G4VisAttributes* PMTVisAtt = new G4VisAttributes(magenta);
1246:   PMTVisAtt->SetVisibility(true);
1247:   //PMTVisAtt->SetForceWireframe(true);
1248:   PMTVisAtt->SetForceSolid(true);
```

```
1249:   Cathode_Logical->SetVisAttributes(PMTVisAtt);
1250:
1251:  G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::ConstructComponent() " << G4endl << G4endl;
1252:
1253:  } // end of  QweakSimCerenkovDetector::ConstructComponent()
1254:
1255:  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1256:
1257:  void QweakSimCerenkovDetector::SetCerenkovDetectorMaterial(G4String materialName)
1258:  {
1259:    // search the material by its name
1260:    G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
1261:    if (pttoMaterial){
1262:      QuartzBar_LogicalLeft->SetMaterial(pttoMaterial);
1263:      QuartzBar_LogicalRight->SetMaterial(pttoMaterial);
1264:      LightGuide_LogicalLeft->SetMaterial(pttoMaterial);
1265:      LightGuide_LogicalRight->SetMaterial(pttoMaterial);
1266:      QuartzGlue_Logical->SetMaterial(pttoMaterial);
1267:    }
1268:    else {
1269:       G4cerr << "==== ERROR: Changing Cerenkov Detector Material failed" << G4endl;
1270:    }
1271:
1272:  }
1273:
1274:  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1275:
1276:  void QweakSimCerenkovDetector::DestroyComponent()
1277:  {
1278:  }
1279:
1280:  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1281:  void QweakSimCerenkovDetector::SetCerenkovDetectorThickness(G4double thickness)
1282:  {
1283:     G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovDetectorThickness() " << G4endl << G4endl;
1284:
1285:  //     G4Box *box = NULL;
1286:
1287:  //     Thickness = thickness;
1288:
1289:  //     if(CerenkovDetector_Logical)
1290:  //       box = (G4Box*)CerenkovDetector_Logical->GetSolid();
1291:  //     if(box)
1292:  //       box->SetZHalfLength(0.5*Thickness);
1293:
1294:  //     if(PMTContainer_Logical)
1295:  //       box = (G4Box*)PMTContainer_Logical->GetSolid();
1296:  //     if(box)
```

```
1297: //      box->SetZHalfLength(0.5*Thickness);
1298:
1299: //    if(PMTEntranceWindow_Logical)
1300: //      box = (G4Box*)PMTEntranceWindow_Logical->GetSolid();
1301: //    if(box)
1302: //      box->SetZHalfLength(0.5*Thickness);
1303:
1304: //    if(Cathode_Logical)
1305: //      box = (G4Box*)Cathode_Logical->GetSolid();
1306: //    if(box)
1307: //      box->SetZHalfLength(0.5*Thickness);
1308:
1309: //    G4RunManager::GetRunManager()->GeometryHasBeenModified();
1310:
1311:    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovDetectorThickness() " << G4endl << G4endl;
1312: }
1313:
1314: //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1315: void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInX(G4double xPos)
1316: {
1317:    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInX() " << G4endl << G4endl;
1318:
1319:    Position_CerenkovContainer_X =xPos;
1320:
1321:    CerenkovGeometryPVUpdate();
1322:
1323:    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInX() " << G4endl << G4endl;
1324: }
1325:
1326: //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1327: void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInY(G4double yPos)
1328: {
1329:    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInY() " << G4endl << G4endl;
1330:
1331:    Position_CerenkovContainer_Y = yPos;
1332:
1333:    CerenkovGeometryPVUpdate();
1334:
1335:    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInY() " << G4endl << G4endl;
1336: }
1337:
1338: //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1339: void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInZ(G4double zPos)
1340: {
1341:    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInZ() " << G4endl << G4endl;
1342:
1343:    Position_CerenkovContainer_Z = zPos;
1344:
```

```
1345:     CerenkovGeometryPVUpdate();
1346: 
1347:     G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInZ() " << G4endl << G4endl;
1348:   }
1349: 
1350:   //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1351:   void QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle(G4double tiltangle)
1352:   {
1353:     G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle() " << G4endl << G4endl;
1354: 
1355:     // assign new tilting
1356:     Tilting_Angle = tiltangle;
1357: 
1358:     CerenkovGeometryPVUpdate();
1359: 
1360:     G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle() " << G4endl << G4endl;
1361:   }
1362: 
1363:   //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1364:   void QweakSimCerenkovDetector::CerenkovGeometryPVUpdate()
1365:   {
1366:     G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::CerenkovGeometryPVUpdate()" << G4endl << G4endl;
1367: 
1368:     for (size_t i=0; i< CerenkovMasterContainer_Physical.size();i++)
1369:     {
1370:    CerenkovContainer_Logical->RemoveDaughter(CerenkovMasterContainer_Physical[i]);
1371:    delete CerenkovMasterContainer_Physical[i];
1372: 
1373:    delete Rotation_CerenkovMasterContainer[i];
1374: 
1375:     }
1376:     CerenkovMasterContainer_Physical.clear();
1377:     CerenkovMasterContainer_Physical.resize(8);
1378: 
1379:     Rotation_CerenkovMasterContainer.clear();
1380:     Rotation_CerenkovMasterContainer.resize(8);
1381: 
1382: 
1383:     // Place the physical volume of the rods with the new phi angle
1384:     PlacePVCerenkovMasterContainer();
1385: 
1386:     G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::CerenkovGeometryPVUpdate()" << G4endl << G4endl;
1387:   }
1388: 
1389:   //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
1390:   void QweakSimCerenkovDetector::PlacePVCerenkovMasterContainer()
1391:   {
1392:         G4ThreeVector* centerVector  = new G4ThreeVector();
```

```
1393:
1394:      // place 8  CerenkovContainer_Logical plates into the MotherVolume (around the global Z axis)
1395:      for (G4int n=0; n<8; n++) {
1396:
1397:
1398:    // Phi angles of the 8 cerenkovs
1399:    AnglePhi_CerenkovMasterContainer[n] = n*45.0*degree;
1400:
1401:    // since the CerenkovMasterContainer_Logical is defined for a vertical orientation
1402:    // but the translation assumes a horizontal orientation, we have to subtract 90*deg
1403:    Rotation_CerenkovMasterContainer[n] = new G4RotationMatrix();
1404:    Rotation_CerenkovMasterContainer[n]->rotateZ(AnglePhi_CerenkovMasterContainer[n]+90*degree);
1405:    Rotation_CerenkovMasterContainer[n]->rotateX(Tilting_Angle);
1406:
1407:    // set the vectors to the center of the CerenkovContainer
1408:    // located at 12 o'clock. Then rotate the centerVector to the 8
1409:    // positions and extract the new vector components
1410:    // This procedure is easier than the calculation by hand for individual positions/orientations
1411:
1412:    // define 12' o'clock start location
1413:    centerVector->setX(Position_CerenkovContainer_X);
1414:    centerVector->setY(Position_CerenkovContainer_Y);
1415:    centerVector->setZ(Position_CerenkovContainer_Z);
1416:
1417:    // rotate centerVector to the 8 positions
1418:         centerVector->rotateZ(AnglePhi_CerenkovMasterContainer[n]);
1419:
1420:    Translation_CerenkovMasterContainer[n].setX( centerVector->y() );
1421:    Translation_CerenkovMasterContainer[n].setY( centerVector->x() );
1422:    Translation_CerenkovMasterContainer[n].setZ( centerVector->z() );
1423:
1424:
1425:
1426:
1427:    CerenkovMasterContainer_Physical[n]   = new G4PVPlacement(Rotation_CerenkovMasterContainer[n],
1428:                                            Translation_CerenkovMasterContainer[n],
1429:                                            "CerenkovMasterContainer_Physical",
1430:                                            CerenkovContainer_Logical,
1431:                                            theMotherPV,
1432:                                            false,
1433:                                            n);
1434:
1435:      } // end of  for (G4int n=0; n<8; n++)
1436:   }
1437:
1438:  //....oooOO0OOooo.....oooOO0OOooo.....oooOO0OOooo.....oooOO0OOooo.....
1439:
```