```cpp
#include "QweakSimCerenkovDetector.hh"

QweakSimCerenkovDetector::QweakSimCerenkovDetector(QweakSimUserInformation *userInfo)
{
  // initialize some pointers
  myUserInfo = userInfo;

  CerenkovDetectorMessenger = NULL;
  pMaterial                 = NULL;

  theMotherPV               = NULL;

  CerenkovContainer_Logical  = NULL;
  CerenkovContainer_Physical = NULL;
  CerenkovContainer_Material = NULL;

//   CerenkovDetector_Logical   = NULL;
//   CerenkovDetector_Physical  = NULL;
//   CerenkovDetector_Material  = NULL;

  ActiveArea_Logical        = NULL;
  ActiveArea_Physical       = NULL;
  ActiveArea_Material       = NULL;

  LightGuide_LogicalLeft     = NULL;
  LightGuide_LogicalRight    = NULL;
  LightGuide_PhysicalLeft    = NULL;
  LightGuide_PhysicalRight   = NULL;
  LightGuide_Material        = NULL;

  QuartzGlue_Logical        = NULL;
  QuartzGlue_PhysicalLeft    = NULL;
  QuartzGlue_PhysicalCenter  = NULL;
  QuartzGlue_PhysicalRight   = NULL;
  QuartzGlue_Material        = NULL;

  PMTContainer_Logical  = NULL;
  PMTContainer_PhysicalLeft  = NULL;
  PMTContainer_PhysicalRight = NULL;
  PMTContainer_Material = NULL;

  PMTEntranceWindow_Logical  = NULL;
  PMTEntranceWindow_Physical = NULL;
  PMTEntranceWindow_Material = NULL;

  Cathode_Logical  = NULL;
  Cathode_Physical = NULL;
  Cathode_Material = NULL;
```

```

Rotation_CerenkovContainer = NULL;

// pointer to the sensitive detector
CerenkovDetectorSD      = NULL;
CerenkovDetector_PMTSD  = NULL;

// clear vector containing temp solids for boolean soild union
LeftQuartz_Solid.clear();
LeftQuartz_Solid.resize(4);  //need 4 chamfers on quartz bar proper
RightQuartz_Solid.clear();
RightQuartz_Solid.resize(4); //need 4 chamfers on quartz bar proper
LeftGuide_Solid.clear();
LeftGuide_Solid.resize(5);   //need 4 chamfers + 1 angle cut on light guide
RightGuide_Solid.clear();
RightGuide_Solid.resize(5);  //need 4 chamfers + 1 angle cut on light guide

mirror_logical.clear();
mirror_physical.clear();

mirror_logical.resize(8);
mirror_physical.resize(8);

CerenkovMasterContainer_Physical.clear();
CerenkovMasterContainer_Physical.resize(8);

AnglePhi_CerenkovMasterContainer.clear();
AnglePhi_CerenkovMasterContainer.resize(8);

Translation_CerenkovMasterContainer.clear();
Translation_CerenkovMasterContainer.resize(8);

Rotation_CerenkovMasterContainer.clear();
Rotation_CerenkovMasterContainer.resize(8);

CerenkovDetectorMessenger = new QweakSimCerenkovDetectorMessenger(this);

pMaterial = new QweakSimMaterial();
pMaterial->DefineMaterials();

//CerenkovContainer_Material = pMaterial->GetMaterial("HeGas");
CerenkovContainer_Material = pMaterial->GetMaterial("Air");
ActiveArea_Material        = pMaterial->GetMaterial("Air");
QuartzBar_Material         = pMaterial->GetMaterial("Quartz");
LightGuide_Material        = pMaterial->GetMaterial("Quartz");
PMTContainer_Material      = pMaterial->GetMaterial("Vacuum");
PMTEntranceWindow_Material = pMaterial->GetMaterial("LimeGlass");
PMTQuartzOpticalFilm_Material = pMaterial->GetMaterial("SiElast_Glue");
```

```
  Cathode_Material          = pMaterial->GetMaterial("LimeGlass");
  Radiator_Material         = pMaterial->GetMaterial("Lead");
  QuartzGlue_Material       = pMaterial->GetMaterial("SiElast_Glue");
  mirror_material           = pMaterial->GetMaterial("Mirror");

  LightGuide_FullLength      =   18.00*cm;
  LightGuide_FullWidth1      =   18.00*cm;
  LightGuide_FullWidth2      =   18.00*cm;
  LightGuide_FullThickness   =    1.25*cm;

  QuartzBar_FullLength       =  100.00*cm;    // Full X length
  QuartzBar_FullHeight       =   18.00*cm;    // Full Y length
  QuartzBar_FullThickness    =    1.25*cm;    // Full Z length

  GlueFilm_FullLength_X      =   0.001*mm;
  GlueFilm_FullLength_Y      =   18.00*cm;
  GlueFilm_FullLength_Z      =    1.25*cm;

  ActiveArea_FullLength_X    =    2.0*(LightGuide_FullLength + QuartzBar_FullLength +GlueFilm_FullLength_X) + GlueFilm_FullLength_X +2.0*cm;
  ActiveArea_FullLength_Y    =    QuartzBar_FullHeight + 1.0*cm;
  ActiveArea_FullLength_Z    =    QuartzBar_FullThickness + 2.0*cm;

  Container_FullLength_X     =  ActiveArea_FullLength_X + 20.0*cm;
  Container_FullLength_Y     =  ActiveArea_FullLength_Y +  4.0*cm;
  Container_FullLength_Z     =  ActiveArea_FullLength_Z + 10.0*cm;

  Chamfer_FullLength         =  120.00*cm;
  Chamfer_FullHeight         =    7.00*mm;
  Chamfer_FullThickness      =    7.00*mm;

  G4double ReductionInPhotocathodeDiameter = 5*mm;

  PMTQuartzOpticalFilm_Thickness=  0.001*mm;
  PMTQuartzOpticalFilm_Diameter =  12.7*cm;

  PMTEntranceWindow_Thickness   =  1.0*mm; // assumed PMT glass thickness
  PMTEntranceWindow_Diameter    =  12.7*cm;//QuartzBar_FullHeight;

  Cathode_Thickness    = 1.0*mm;
  Cathode_Diameter     = PMTEntranceWindow_Diameter - ReductionInPhotocathodeDiameter;

  PMTContainer_Diameter     =  PMTEntranceWindow_Diameter+1.0*mm;
  PMTContainer_FullLength_Z    =  2.0*mm+PMTEntranceWindow_Thickness+Cathode_Thickness;

  Tilting_Angle   =  0.0*degree;

}

```

```
//....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****************************


```

```
G4Box* Chamfer_Solid    = new G4Box("Chamfer_Solid",
                   0.5 * Chamfer_FullLength,       // half X length required by Geant4
                   0.5 * Chamfer_FullHeight ,      // half Y length required by Geant4
                   0.5 * Chamfer_FullThickness );

G4Box* QuartzBar_Solid  = new G4Box("QuartzBar_Solid",
                   0.5 * QuartzBar_FullLength,       // half X length required by Geant4
                   0.5 * QuartzBar_FullHeight,      // half Y length required by Geant4
                   0.5 * QuartzBar_FullThickness );  // half Z length required by Geant4

//Boolean Union:
//Upper-upstream edge chamfer

ChamferScew = 0.021486*PI/180.0;
delta = 0.5*(Chamfer_FullHeight - 1.0*mm)/sqrt(2.0);
G4double ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
G4double ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
Position_Chamfer1.setX(0.0*cm);//33.333333*cm);
Position_Chamfer1.setY(0.5*QuartzBar_FullHeight + delta);
Position_Chamfer1.setZ(-(0.5*QuartzBar_FullThickness + delta));
Rotation_Chamfer1.rotateX(45.0*degree);
Rotation_Chamfer1.rotateY(ChamferAdjRotY*radian);
Rotation_Chamfer1.rotateZ(ChamferAdjRotZ*radian);
G4Transform3D Transform_Chamfer1(Rotation_Chamfer1,Position_Chamfer1);
Rotation_Chamfer1.rotateZ(-ChamferAdjRotZ*radian);
Rotation_Chamfer1.rotateY(-ChamferAdjRotY*radian);

RightQuartz_Solid[0]=  new G4SubtractionSolid ("UpperUpstreamChamfer-RightQuartzBar",
                          QuartzBar_Solid,
                          Chamfer_Solid,
                          Transform_Chamfer1);

//Boolean Union:
//Upper-downstream edge chamfer

delta = 0.5*(Chamfer_FullHeight - 0.5*mm)/sqrt(2.0);
ChamferScew = 0.0;//0.014*PI/180.0;
ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
Position_Chamfer2.setX(0.0*mm);
Position_Chamfer2.setY(0.5*QuartzBar_FullHeight + delta);
Position_Chamfer2.setZ(0.5*QuartzBar_FullThickness + delta);
Rotation_Chamfer2.rotateX(45.0*degree);
Rotation_Chamfer2.rotateY(-ChamferAdjRotY*radian);
Rotation_Chamfer2.rotateZ(ChamferAdjRotZ*radian);
G4Transform3D Transform_Chamfer2(Rotation_Chamfer2,Position_Chamfer2);
Rotation_Chamfer2.rotateZ(-ChamferAdjRotZ*radian);
Rotation_Chamfer2.rotateY(ChamferAdjRotY*radian);
```

```
RightQuartz_Solid[1] =  new G4SubtractionSolid ("UpperDownstreamChamfer-RightQuartzBar",
                         RightQuartz_Solid[0],
                         Chamfer_Solid,
                         Transform_Chamfer2);

//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_Chamfer3(Rotation_Chamfer3,Position_Chamfer3);
Rotation_Chamfer3.rotateZ(ChamferAdjRotZ*radian);
Rotation_Chamfer3.rotateY(-ChamferAdjRotY*radian);

RightQuartz_Solid[2] =  new G4SubtractionSolid ("LowerUpstreamChamfer-RightQuartzBar",
                         RightQuartz_Solid[1],Chamfer_Solid,
                         Transform_Chamfer3);

//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_Chamfer4(Rotation_Chamfer4,Position_Chamfer4);
Rotation_Chamfer4.rotateY(ChamferAdjRotY*radian);
Rotation_Chamfer4.rotateZ(ChamferAdjRotZ*radian);

RightQuartz_Solid[3] =  new G4SubtractionSolid ("LowerUpstreamChamfer-RightQuartzBar",
                         RightQuartz_Solid[2], Chamfer_Solid,
                         Transform_Chamfer4);


QuartzBar_LogicalRight  = new G4LogicalVolume(RightQuartz_Solid[3],
                        QuartzBar_Material,
                        "QuartzBar_LogicalRight",
                        0,0,0);
```

```
  G4ThreeVector Position_RightQuartzBar = G4ThreeVector(-0.5*(QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);

  QuartzBar_PhysicalRight   = new G4PVPlacement(0,Position_RightQuartzBar,
                                  QuartzBar_LogicalRight,
                                  "QuartzBar_PhysicalRight",
                                  ActiveArea_Logical,
                                  false,0);

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

//**************************************************************************************************************
//******************************Define Center Quartz Glue Film **************************************


  G4Box* CenterGlueFilm_Solid    = new G4Box("CenterGlueFilm_Solid",
                                0.5 * GlueFilm_FullLength_X,
                                0.5 * GlueFilm_FullLength_Y,
                                0.5 * GlueFilm_FullLength_Z);

  QuartzGlue_Logical  = new G4LogicalVolume(CenterGlueFilm_Solid,
                               QuartzGlue_Material,
                               "CenterGlueFilm_Log",
                               0,0,0);

  G4ThreeVector Position_CenterGlueFilm = G4ThreeVector(0,0,0);

  QuartzGlue_PhysicalCenter  = new G4PVPlacement(0,Position_CenterGlueFilm,
                                   QuartzGlue_Logical,
                                   "QuartzGlue_PhysicalCenter",
                                   ActiveArea_Logical,
                                   false,0);

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

//**************************************************************************************************************
//******************************Define Right Quartz Glue Film ***************************************

  G4ThreeVector Position_RightGlueFilm = G4ThreeVector(-1.0*(QuartzBar_FullLength+GlueFilm_FullLength_X),0,0);

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

//**************************************************************************************************************
```

```
//**********************************************************************************************************************************

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


  //Boolean Union:
  //Upper-upstream edge chamfer

  ChamferScew = 0.021486*PI/180.0;
  delta = 0.5*(Chamfer_FullHeight - 1.0*mm)/sqrt(2.0);
  ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(PI/2.0 - ChamferRotation));
  ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(PI/2.0 - ChamferRotation));
  Position_Chamfer1.setX(0.0*cm);//33.333333*cm);
  Position_Chamfer1.setY(0.5*QuartzBar_FullHeight + delta);
  Position_Chamfer1.setZ(-(0.5*QuartzBar_FullThickness + delta));
//   Rotation_Chamfer1.rotateX(45.0*degree);
  Rotation_Chamfer1.rotateY(ChamferAdjRotY*radian);
  Rotation_Chamfer1.rotateZ(ChamferAdjRotZ*radian);
  G4Transform3D Transform_Chamfer5(Rotation_Chamfer1,Position_Chamfer1);
  Rotation_Chamfer1.rotateZ(-ChamferAdjRotZ*radian);
  Rotation_Chamfer1.rotateY(-ChamferAdjRotY*radian);

  LeftQuartz_Solid[0]=  new G4SubtractionSolid ("UpperUpstreamChamfer-LeftQuartzBar",
                                     QuartzBar_Solid,
                                     Chamfer_Solid,
                                     Transform_Chamfer5);

  //Boolean Union:
  //Upper-downstream edge chamfer

  delta = 0.5*(Chamfer_FullHeight - 0.5*mm)/sqrt(2.0);
  ChamferScew = 0.0;//0.014*PI/180.0;
  ChamferAdjRotZ = std::atan(sin(ChamferScew)*std::cos(ChamferRotation));
  ChamferAdjRotY = std::atan(sin(ChamferScew)*std::sin(ChamferRotation));
  Position_Chamfer2.setX(0.0*mm);
  Position_Chamfer2.setY(0.5*QuartzBar_FullHeight + delta);
  Position_Chamfer2.setZ(0.5*QuartzBar_FullThickness + delta);
//   Rotation_Chamfer2.rotateX(45.0*degree);
  Rotation_Chamfer2.rotateY(-ChamferAdjRotY*radian);
  Rotation_Chamfer2.rotateZ(ChamferAdjRotZ*radian);
  G4Transform3D Transform_Chamfer6(Rotation_Chamfer2,Position_Chamfer2);
  Rotation_Chamfer2.rotateZ(-ChamferAdjRotZ*radian);
  Rotation_Chamfer2.rotateY(ChamferAdjRotY*radian);

  LeftQuartz_Solid[1] =  new G4SubtractionSolid ("UpperDownstreamChamfer-LeftQuartzBar",
                                      LeftQuartz_Solid[0],
                                      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));
```

```
 Position_Chamfer3.setZ(0.0*cm);
 Rotation_Chamfer3.rotateY(90.0*degree);
 G4Transform3D Transform_Chamfer11(Rotation_Chamfer3,Position_Chamfer3);

 LeftGuide_Solid[3]=  new G4SubtractionSolid ("LeftLGChamfer3",
                          LeftGuide_Solid[2],
                          Chamfer_Solid,
                          Transform_Chamfer11);

 Position_Chamfer4.setX(-(0.5*QuartzBar_FullThickness + delta));
 Position_Chamfer4.setY(-(0.5*QuartzBar_FullHeight + delta));
 Position_Chamfer4.setZ(0.0*cm);
 Rotation_Chamfer4.rotateY(90.0*degree);
 G4Transform3D Transform_Chamfer12(Rotation_Chamfer4,Position_Chamfer4);

 LeftGuide_Solid[4]=  new G4SubtractionSolid ("LeftLGChamfer4",
                          LeftGuide_Solid[3],
                          Chamfer_Solid,
                          Transform_Chamfer12);



//*******************************Right Light Guide ******************************************************


 //Boolean Union:
 //Right Light Guide Angular cut-off at edge
 Position_AngCut2.setX(0.0*cm);
 Position_AngCut2.setY(0.0*cm);
 Position_AngCut2.setZ(-(0.5*LightGuide_FullLength+dx));
 Rotation_AngCut2.rotateY(-ad*degree);
 G4Transform3D Transform_AngCut2(Rotation_AngCut2,Position_AngCut2);

 RightGuide_Solid[0] =  new G4SubtractionSolid ("LGRight-AngCut",
                           LightGuide_Solid,
                           LGEdgeAngleCut_Solid,
                           Transform_AngCut2);

 G4Transform3D Transform_Chamfer13(Rotation_Chamfer1,Position_Chamfer1);

 RightGuide_Solid[1]=  new G4SubtractionSolid ("RightLGChamfer1",
                           RightGuide_Solid[0],
                           Chamfer_Solid,
                           Transform_Chamfer13);


 G4Transform3D Transform_Chamfer14(Rotation_Chamfer2,Position_Chamfer2);

```

```
  RightGuide_Solid[2]=  new G4SubtractionSolid ("RightLGChamfer2",
                                       RightGuide_Solid[1],
                                       Chamfer_Solid,
                                       Transform_Chamfer14);


  G4Transform3D Transform_Chamfer15(Rotation_Chamfer3,Position_Chamfer3);

  RightGuide_Solid[3]=  new G4SubtractionSolid ("RightLGChamfer3",
                                       RightGuide_Solid[2],
                                       Chamfer_Solid,
                                       Transform_Chamfer15);

  G4Transform3D Transform_Chamfer16(Rotation_Chamfer4,Position_Chamfer4);

  RightGuide_Solid[4]=  new G4SubtractionSolid ("RightLGChamfer4",
                                      RightGuide_Solid[3],
                                      Chamfer_Solid,
                                      Transform_Chamfer16);

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



  //Boolean Union:
  //Left Light Guide
  Position_LGLeft.setX((QuartzBar_FullLength+0.5*LightGuide_FullLength+1.5*GlueFilm_FullLength_X));
  Position_LGLeft.setY(0.0*cm);
  Position_LGLeft.setZ(0.0*cm - LightGuide_FullThickness*(1 - redfr)/(4.0));
  Rotation_LGLeft.rotateY(-90.0*degree);
  G4Transform3D Transform_LGLeft(Rotation_LGLeft,Position_LGLeft);

  //Boolean Union:
  //Right Light Guide
  Position_LGRight.setX(-(QuartzBar_FullLength+0.5*LightGuide_FullLength+1.5*GlueFilm_FullLength_X));
  Position_LGRight.setY(0.0*cm);
  Position_LGRight.setZ(0.0*cm - LightGuide_FullThickness*(1 - redfr)/(4.0));
//   Rotation_LGRight.rotateY(-90.0*degree);
  Rotation_LGRight.rotateY(90.0*degree);
//   Rotation_LGRight.rotateZ(180.0*degree);
  G4Transform3D Transform_LGRight(Rotation_LGRight,Position_LGRight);


  LightGuide_LogicalLeft  = new G4LogicalVolume(LeftGuide_Solid[4],
                                       LightGuide_Material,
                                       "LightGuide_LogicalLeft",
                                       0,0,0);


  LightGuide_PhysicalLeft = new G4PVPlacement(Transform_LGLeft,
                                      LightGuide_LogicalLeft,
                                      "LightGuide_PhysicalLeft",
                                      ActiveArea_Logical,
                                      false,0);


  LightGuide_LogicalRight  = new G4LogicalVolume(RightGuide_Solid[4],
                                        LightGuide_Material,
                                        "LightGuide_LogicalRight",
                                        0,0,0);


  LightGuide_PhysicalRight = new G4PVPlacement(Transform_LGRight,
                                       LightGuide_LogicalRight,
                                       "LightGuide_PhysicalRight",
                                       ActiveArea_Logical,
                                       false,0);

  //******************************************************************************************************************
  //******************************Face Mirrors******************************************************************

//   G4Trd* LGFaceMirror_Solid = new G4Trd("LGFaceMirror_Solid",
//                                        0.1*mm,0.1*mm,
//                                        0.5*LightGuide_FullWidth1,
//                                        0.5*LightGuide_FullWidth2,
//                                        0.5*LightGuide_FullLength -
//                                        0.5*LightGuide_FullThickness*std::tan(ar)+
//                                        0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));


//   Position_LGFaceMirrorLeft.setX(0.5*(QuartzBar_FullLength+LightGuide_FullLength)-
//                                  0.5*LightGuide_FullThickness*std::tan(ar)+
//                                  0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
//   Position_LGFaceMirrorLeft.setY(0.0*cm);
//   Position_LGFaceMirrorLeft.setZ(-0.5*LightGuide_FullThickness - 0.1*mm);
//   Rotation_LGFaceMirrorLeft.rotateY(-90.0*degree);
//   G4Transform3D Transform_LGFMLeft(Rotation_LGFaceMirrorLeft,Position_LGFaceMirrorLeft);


//   mirror_logical[0]  = new G4LogicalVolume(LGFaceMirror_Solid,
//                                           mirror_material,
//                                           "mirrorface_log1",
//                                           0,0,0);

//   mirror_physical[0] = new G4PVPlacement(Transform_LGFMLeft,
//                                  mirror_logical[0],
//                                  "mirrorface_physical1",
//                                  CerenkovContainer_Logical,
//                                  false,
//                                  0); // copy number for left PMTContainer

//******************************Face Mirrors*********************************************************
//**************************************************************************************************





//**************************************************************************************************
//*******************************Edge Mirrors*********************************************************


  G4Box* LGEdgeMirror_Solid = new G4Box("LGEdgeMirror_Solid",
                              0.1*mm,0.5*LightGuide_FullWidth1,
                              redfr*0.5*LightGuide_FullThickness/std::cos(ar));

  Position_LGEdgeMirrorLeft.setX(1.5*GlueFilm_FullLength_X + QuartzBar_FullLength+LightGuide_FullLength+0.1*mm/std::cos(ar)-
                              0.5*LightGuide_FullThickness*std::tan(ar)+
                        0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
  Position_LGEdgeMirrorLeft.setY(0.0*cm);
  Position_LGEdgeMirrorLeft.setZ(-0.5*LightGuide_FullThickness*(1-redfr));
  Rotation_LGEdgeMirrorLeft.rotateY(ad*degree);
  G4Transform3D Transform_LGEMLeft(Rotation_LGEdgeMirrorLeft,Position_LGEdgeMirrorLeft);


  mirror_logical[1]  = new G4LogicalVolume(LGEdgeMirror_Solid,
                                  mirror_material,
                                  "mirrorface_log2",
                                  0,0,0);

  mirror_physical[1] = new G4PVPlacement(Transform_LGEMLeft,
                                mirror_logical[1],
                                "mirrorface_physical2",
                                ActiveArea_Logical,
                                false,
                                0); // copy number for left PMTContainer





  Position_LGEdgeMirrorRight.setX(-1.5*GlueFilm_FullLength_X-QuartzBar_FullLength-LightGuide_FullLength-0.1*mm/std::cos(ar)+
                0.5*LightGuide_FullThickness*std::tan(ar)-
                0.5*LightGuide_FullThickness*(1 - redfr)*std::tan(ar));
 Position_LGEdgeMirrorRight.setY(0.0*cm);
 Position_LGEdgeMirrorRight.setZ(-0.5*LightGuide_FullThickness*(1-redfr));
 Rotation_LGEdgeMirrorRight.rotateY(-ad*degree);
 G4Transform3D Transform_LGEMRight(Rotation_LGEdgeMirrorRight,Position_LGEdgeMirrorRight);


 mirror_logical[3]  = new G4LogicalVolume(LGEdgeMirror_Solid,
                     mirror_material,
                     "mirrorface_log4",
                     0,0,0);

 mirror_physical[3] = new G4PVPlacement(Transform_LGEMRight,
                    mirror_logical[3],
                    "mirrorface_physical4",
                    ActiveArea_Logical,
                    false,
                    0); // copy number for left PMTContainer

//***********************************Edge Mirrors********************************************************************
//**********************************************************************************************************************




//**********************************************************************************************************************
//***********************************Radiator**********************************************************************


//   G4Box* RadiatorSolid = new G4Box("Radiator_Sol",
//                   0.5 * QuartzBar_FullLength,       // half X length required by Geant4
//                   0.5 * QuartzBar_FullHeight,      // half Y length required by Geant4
//                   1.0*cm );  // half Z length required by Geant4

//   Radiator_Logical  = new G4LogicalVolume(RadiatorSolid,
//                       Radiator_Material,
//                       "Radiator_Log",
//                       0,0,0);

//   G4ThreeVector Position_Radiator  = G4ThreeVector(0,0,2.0*cm);//-2.0*cm);

//   Radiator_Physical   = new G4PVPlacement(0,Position_Radiator,
//                       Radiator_Logical,
//                       "Radiator_Physical",
//                       CerenkovContainer_Logical,
//                       false,
//                       0);
```

```

//******************************Radiator**************************************************************
//****************************************************************************************************


  //-----------------------------------
  // define the PMTContainer
  //-----------------------------------

  G4double mypi   = 4.0*std::atan(1.0);
  G4double thetaY = std::atan(LightGuide_FullThickness*(1 - redfr)/(LightGuide_FullLength));
  G4double Xoffs = 0.0*cm;//7.0*cm;

  //Flat on guide face configuration
  G4double PMTContXShift = QuartzBar_FullLength + LightGuide_FullLength - 0.5*PMTEntranceWindow_Diameter - Xoffs;
  G4double PMTContYShift = 0.0;
  G4double PMTContZShift = 0.5*QuartzBar_FullThickness + 0.5*PMTContainer_FullLength_Z
    - (LightGuide_FullLength - 0.5*PMTEntranceWindow_Diameter-Xoffs)*std::tan(thetaY);

  // relocation of the left Photon Detector Container
  Translation_PMTContainerLeft.setX(1.0*PMTContXShift);
  Translation_PMTContainerLeft.setY(1.0*PMTContYShift);
  Translation_PMTContainerLeft.setZ(1.0*PMTContZShift);

//   //On guide edge configuration
//   Rotation_PMTContainerLeft.rotateY(90.0*degree);

  //Flat on guide face configuration
  Rotation_PMTContainerLeft.rotateY(thetaY*180.0/mypi*degree);
  G4Transform3D Transform3D_PMTContainerLeft(Rotation_PMTContainerLeft,
                                             Translation_PMTContainerLeft);

  // relocation of the right Photon Detector Container
  Translation_PMTContainerRight.setX(-1.0*PMTContXShift);
  Translation_PMTContainerRight.setY(1.0*PMTContYShift);
  Translation_PMTContainerRight.setZ(1.0*PMTContZShift);

//   //On guide edge configuration
//   Rotation_PMTContainerLeft.rotateY(-90.0*cm);

  //Flat on guide face configuration
  Rotation_PMTContainerRight.rotateY(-thetaY*180.0/mypi*degree);
  G4Transform3D Transform3D_PMTContainerRight(Rotation_PMTContainerRight,
                                              Translation_PMTContainerRight);



  G4double PMTQuartzOpticalFilmZShift = 0.5*(PMTQuartzOpticalFilm_Thickness - PMTContainer_FullLength_Z);
```

```

// relocation of the PMTEntranceWindow
Translation_PMTQuartzOpticalFilm.setX(0.0*cm);
Translation_PMTQuartzOpticalFilm.setY(0.0*cm);
Translation_PMTQuartzOpticalFilm.setZ(PMTQuartzOpticalFilmZShift);

//-----------------------------------------------------------------------------------
// location and orientation of the PMT Entrance Window within the PMT Container
//-----------------------------------------------------------------------------------

G4double PMTEntWindZShift = 0.5*(PMTEntranceWindow_Thickness - PMTContainer_FullLength_Z)+PMTQuartzOpticalFilm_Thickness;

// relocation of the PMTEntranceWindow
Translation_PMTEntranceWindow.setX(0.0*cm);
Translation_PMTEntranceWindow.setY(0.0*cm);
Translation_PMTEntranceWindow.setZ(PMTEntWindZShift);

//-----------------------------------------------------------------------------------
// location and orientation of the cathode WITHIN the PMT
//-----------------------------------------------------------------------------------

G4double CathodeZShift = PMTEntranceWindow_Thickness + 0.5*(Cathode_Thickness - PMTContainer_FullLength_Z);

// location of the Photon Detector relative to  Photon Detector Container
Translation_Cathode.setX(0.0*cm);
Translation_Cathode.setY(0.0*cm);
Translation_Cathode.setZ(CathodeZShift);


//   G4Box* PMTContainer_Solid    = new G4Box("PMTContainer_Solid",
//                                       0.5 * PMTContainer_FullLength_X,   // half X
//                                       0.5 * PMTContainer_FullLength_Y ,  // half Y
//                                       0.5 * PMTContainer_FullLength_Z);  // half Z
G4Tubs* PMTContainer_Solid    = new G4Tubs("PMTContainer_Solid",0.0*cm,
                                   0.5 * PMTContainer_Diameter,
                                   0.5 * PMTContainer_FullLength_Z,
                                   0.0*degree,360.0*degree);


PMTContainer_Logical  = new G4LogicalVolume(PMTContainer_Solid,
                                    PMTContainer_Material,
                                    "PMTContainer_Log",
                                    0,0,0);

// left side
PMTContainer_PhysicalLeft  = new G4PVPlacement(Transform3D_PMTContainerLeft,
                                       PMTContainer_Logical,
                                       "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);
```

```
 PMTEntranceWindow_Physical = new G4PVPlacement(0,Translation_PMTEntranceWindow,
                      PMTEntranceWindow_Logical,
                      "PMTEntranceWindow_Physical",
                      PMTContainer_Logical,
                      false, 0); // copy number for left photon detector

 //---------------------------
 // define the Photon Detector
 //---------------------------

 G4Tubs* Cathode_Solid = new G4Tubs("Cathode_Solid",0.0*cm,0.5*Cathode_Diameter,
               0.5*Cathode_Thickness,0.0*degree,360.0*degree);

 Cathode_Logical  = new G4LogicalVolume(Cathode_Solid,Cathode_Material,"Cathode_Log",0,0,0);

 Cathode_Physical = new G4PVPlacement(0,Translation_Cathode,Cathode_Logical,"Cathode_Physical",PMTContainer_Logical,
                false, 0); // copy number for left photon detector





 //==================================================================================






 //==================================================================================

  Position_CerenkovContainer  = G4ThreeVector(Position_CerenkovContainer_X,
                     Position_CerenkovContainer_Y,
                     Position_CerenkovContainer_Z);

 // define Rotation matrix for Container orientated in MotherVolume
 Rotation_CerenkovContainer =  new G4RotationMatrix();
 Rotation_CerenkovContainer -> rotateX(Tilting_Angle);

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


 //-----------------------------------------------
```

```


  //===========================================================================
  // place the 8 CerenkovMasterContainer_Physical into the physical MotherVolume
  //===========================================================================
  //
  PlacePVCerenkovMasterContainer();



 //-------------
  const G4int nEntries = 9;

  G4double PhotonEnergy[nEntries] =
    {
      1.54986*eV,  // 800 nanometer
      1.77127*eV,  // 700 nanometer
      2.06648*eV,  // 600 nanometer
      2.47978*eV,  // 500 nanometer
      3.09973*eV,  // 400 nanometer
      4.13297*eV,  // 300 nanometer
      4.95956*eV,  // 250 nanometer
      5.51063*eV,  // 225 nanometer
      5.90424*eV   // 210 nanometer
    };

  G4double Reflectivity[nEntries];

  G4double mylambda;

  for (G4int kk= 0; kk < nEntries; kk++) {
    // Nevens empiric formular for the reflectivity
    // lamda = h*c/E

    mylambda = (h_Planck*c_light/PhotonEnergy[kk])/nanometer;

    Reflectivity[kk] =  1.0 -0.027*exp(-0.004608*mylambda);
    //Reflectivity[kk] =  1.0;
  };

  G4OpticalSurface* QuartzBarLeft_OpticalSurface = new G4OpticalSurface("QuartzBarLeftOpticalSurface");
  G4OpticalSurface* QuartzBarRight_OpticalSurface = new G4OpticalSurface("QuartzBarRightOpticalSurface");
  G4OpticalSurface* LightGuideLeft_OpticalSurface = new G4OpticalSurface("LightGuideLeftOpticalSurface");
  G4OpticalSurface* LightGuideRight_OpticalSurface = new G4OpticalSurface("LightGuideRightOpticalSurface");

  G4OpticalSurface* GlueFilmRight_OpticalSurface = new G4OpticalSurface("GlueFilmRightOpticalSurface");
  G4OpticalSurface* GlueFilmCenter_OpticalSurface = new G4OpticalSurface("GlueFilmCenterOpticalSurface");
  G4OpticalSurface* GlueFilmLeft_OpticalSurface = new G4OpticalSurface("GlueFilmLeftOpticalSurface");
```

```


    G4LogicalBorderSurface* QuartzBarLeft_BorderSurface   = new G4LogicalBorderSurface("QuartzBarLeft_BorderSurface",
                                                QuartzBar_PhysicalLeft,
                                                ActiveArea_Physical,
                                                QuartzBarLeft_OpticalSurface);
    G4LogicalBorderSurface* QuartzBarRight_BorderSurface  = new G4LogicalBorderSurface("QuartzBarRight_BorderSurface",
                                                QuartzBar_PhysicalRight,
                                                ActiveArea_Physical,
                                                QuartzBarRight_OpticalSurface);
    G4LogicalBorderSurface* LightGuideLeft_BorderSurface  = new G4LogicalBorderSurface("LightGuideLeft_BorderSurface",
                                                LightGuide_PhysicalLeft,
                                                ActiveArea_Physical,
                                                LightGuideLeft_OpticalSurface);
    G4LogicalBorderSurface* LightGuideRight_BorderSurface = new G4LogicalBorderSurface("LightGuideRight_BorderSurface",
                                                LightGuide_PhysicalRight,
                                                ActiveArea_Physical,
                                                LightGuideRight_OpticalSurface);
    G4LogicalBorderSurface* GlueFilmRight_BorderSurface   = new G4LogicalBorderSurface("GlueFilmRight_BorderSurface",
                                                QuartzGlue_PhysicalRight,
                                                ActiveArea_Physical,
                                                GlueFilmRight_OpticalSurface);
    G4LogicalBorderSurface* GlueFilmCenter_BorderSurface  = new G4LogicalBorderSurface("GlueFilmCenter_BorderSurface",
                                                QuartzGlue_PhysicalCenter,
                                                ActiveArea_Physical,
                                                GlueFilmCenter_OpticalSurface);
    G4LogicalBorderSurface* GlueFilmLeft_BorderSurface    = new G4LogicalBorderSurface("GlueFilmLeft_BorderSurface",
                                                QuartzGlue_PhysicalLeft,
                                                ActiveArea_Physical,
                                                GlueFilmLeft_OpticalSurface);

    QuartzBarLeft_OpticalSurface->SetType(dielectric_dielectric);
    QuartzBarLeft_OpticalSurface->SetFinish(polished);
    QuartzBarLeft_OpticalSurface->SetPolish(0.997);
    QuartzBarLeft_OpticalSurface->SetModel(glisur);

    QuartzBarRight_OpticalSurface->SetType(dielectric_dielectric);
    QuartzBarRight_OpticalSurface->SetFinish(polished);
    QuartzBarRight_OpticalSurface->SetPolish(0.997);
    QuartzBarRight_OpticalSurface->SetModel(glisur);

    LightGuideLeft_OpticalSurface->SetType(dielectric_dielectric);
    LightGuideLeft_OpticalSurface->SetFinish(polished);
    LightGuideLeft_OpticalSurface->SetPolish(0.997);
    LightGuideLeft_OpticalSurface->SetModel(glisur);

    LightGuideRight_OpticalSurface->SetType(dielectric_dielectric);
    LightGuideRight_OpticalSurface->SetFinish(polished);
    LightGuideRight_OpticalSurface->SetPolish(0.997);
    LightGuideRight_OpticalSurface->SetModel(glisur);

    GlueFilmLeft_OpticalSurface->SetType(dielectric_dielectric);
    GlueFilmLeft_OpticalSurface->SetFinish(polished);
    GlueFilmLeft_OpticalSurface->SetPolish(0.9);
    GlueFilmLeft_OpticalSurface->SetModel(glisur);

    GlueFilmCenter_OpticalSurface->SetType(dielectric_dielectric);
    GlueFilmCenter_OpticalSurface->SetFinish(polished);
    GlueFilmCenter_OpticalSurface->SetPolish(0.9);
    GlueFilmCenter_OpticalSurface->SetModel(glisur);

    GlueFilmRight_OpticalSurface->SetType(dielectric_dielectric);
    GlueFilmRight_OpticalSurface->SetFinish(polished);
    GlueFilmRight_OpticalSurface->SetPolish(0.9);
    GlueFilmRight_OpticalSurface->SetModel(glisur);

    G4MaterialPropertiesTable *quartzST = new G4MaterialPropertiesTable();

    quartzST->AddProperty("REFLECTIVITY",  PhotonEnergy , Reflectivity, nEntries);
    QuartzBarLeft_OpticalSurface->SetMaterialPropertiesTable(quartzST);
    QuartzBarRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);
    LightGuideLeft_OpticalSurface->SetMaterialPropertiesTable(quartzST);
    LightGuideRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);
    GlueFilmRight_OpticalSurface->SetMaterialPropertiesTable(quartzST);


//   G4OpticalSurface* ActiveArea_OpticalSurface = new G4OpticalSurface("ActiveAreaOpticalSurface");

//   G4LogicalBorderSurface* ActiveArea_BorderSurface    = new G4LogicalBorderSurface("ActiveArea_BorderSurface",
//                                                               ActiveArea_Physical,
//                                                               CerenkovContainer_Physical,
//                                                               ActiveArea_OpticalSurface);

//   ActiveArea_OpticalSurface->SetFinish(groundbackpainted); //new for wrapping test
// //   ActiveArea_OpticalSurface->SetPolish(0.0);          //new for wrapping test
// //   ActiveArea_OpticalSurface->SetModel(glisur);      //new for wrapping test
//   ActiveArea_OpticalSurface->SetModel(unified);         //new for wrapping test
//   ActiveArea_OpticalSurface->SetSigmaAlpha(0.25);        //new for wrapping test

//   G4double RefractiveIndex_Air[nEntries] = {1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0}; //new for wrapping test
//   G4double MilliPoreRefl[nEntries] = {0.96,0.96,0.96,0.96,0.96,0.96,0.96,0.96,0.96};       //new for wrapping test
//   G4double specularlobe[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};        //new for wrapping test
//   G4double specularspike[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};       //new for wrapping test
//   G4double backscatter[nEntries] = {0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};         //new for wrapping test

//   G4MaterialPropertiesTable *myST = new G4MaterialPropertiesTable();

//   myST->AddProperty("RINDEX",  PhotonEnergy , RefractiveIndex_Air, nEntries);     //new for wrapping test
//   myST->AddProperty("REFLECTIVITY",  PhotonEnergy , MilliPoreRefl, nEntries);     //new for wrapping test
//   myST->AddProperty("SPECULARLOBECONSTANT", PhotonEnergy ,specularlobe,nEntries); //new for wrapping test
//   myST->AddProperty("SPECULARSPIKECONSTANT",PhotonEnergy,specularspike,nEntries); //new for wrapping test
//   myST->AddProperty("BACKSCATTERCONSTANT",PhotonEnergy,backscatter,nEntries);     //new for wrapping test
// //   myST->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_Air, nEntries);    //new for wrapping test


//   ActiveArea_OpticalSurface->SetMaterialPropertiesTable(myST);

  // Sensitive detectors
  //------------------------------------------------
  // All managed (deleted) by SDManager

  G4SDManager* SDman = G4SDManager::GetSDMpointer();


  //***********************************************************
  CerenkovDetectorSD = new QweakSimCerenkov_DetectorSD("/CerenkovDetectorSD");
  SDman->AddNewDetector(CerenkovDetectorSD);

  // add Cerenkov detector as a sensitiv element
  ActiveArea_Logical->SetSensitiveDetector(CerenkovDetectorSD);
  //***********************************************************

  //***********************************************************
  CerenkovDetector_PMTSD = new QweakSimCerenkovDetector_PMTSD("/CerenkovPMTSD",myUserInfo);
  SDman->AddNewDetector(CerenkovDetector_PMTSD);

  // add PMT as a sensitiv element
  Cathode_Logical->SetSensitiveDetector(CerenkovDetector_PMTSD);
  //     PMTEntranceWindow_Logical->SetSensitiveDetector(CerenkovDetector_PMTSD);
  //***********************************************************


G4cout << G4endl << "###### QweakSimCerenkovDetector: Setting Attributes " << G4endl << G4endl;

 G4Colour  orange    ( 255/255., 127/255.,   0/255.);
 G4Colour  blue      (   0/255.,   0/255., 255/255.);
 G4Colour  magenta   ( 255/255.,   0/255., 255/255.);
 G4Colour  grey      ( 127/255., 127/255., 127/255.);
 G4Colour  lightblue   ( 139/255., 208/255., 255/255.);
 G4Colour  lightorange ( 255/255., 189/255., 165/255.);
 G4Colour  khaki3    ( 205/255., 198/255., 115/255.);
 //-----------------------------------------
 // Visual Attributes for:  CerenkovContainer
 //-----------------------------------------
```

```
  G4VisAttributes* CerenkovContainerVisAtt = new G4VisAttributes(blue);
  CerenkovContainerVisAtt->SetVisibility(false);
  //CerenkovContainerVisAtt->SetVisibility(true);
  //CerenkovContainerVisAtt->SetForceWireframe(true);
  //CerenkovContainerVisAtt->SetForceSolid(true);
  CerenkovContainer_Logical->SetVisAttributes(CerenkovContainerVisAtt);
  ActiveArea_Logical->SetVisAttributes(CerenkovContainerVisAtt);

  //-----------------------------------------
  // Visual Attributes for:  CerenkovDetector
  //-----------------------------------------
  G4VisAttributes* CerenkovDetectorVisAtt = new G4VisAttributes(orange);
  CerenkovDetectorVisAtt->SetVisibility(true);
  // Needed for the correct visualization using Coin3D
  //CerenkovDetectorVisAtt->SetForceSolid(true);
  CerenkovDetectorVisAtt->SetForceWireframe(true);
 //  ActiveArea_Logical->SetVisAttributes(CerenkovDetectorVisAtt);
  QuartzBar_LogicalLeft->SetVisAttributes(CerenkovDetectorVisAtt);
  QuartzBar_LogicalRight->SetVisAttributes(CerenkovDetectorVisAtt);
  LightGuide_LogicalLeft->SetVisAttributes(CerenkovDetectorVisAtt);
  LightGuide_LogicalRight->SetVisAttributes(CerenkovDetectorVisAtt);
  QuartzGlue_Logical->SetVisAttributes(CerenkovDetectorVisAtt);

  //-------------------------------------------------
  // Visual Attributes for:  PMTContainer
  //-------------------------------------------------
  G4VisAttributes* PMTContainerVisAtt = new G4VisAttributes(blue);
  PMTContainerVisAtt->SetVisibility(true);
  PMTContainerVisAtt->SetForceWireframe(true);
  //PMTContainerVisAtt->SetForceSolid(true);
  PMTContainer_Logical->SetVisAttributes(PMTContainerVisAtt);

  //-----------------------------------------------------
  // Visual Attributes for:  PMTEntranceWindow
  //-----------------------------------------------------
  G4VisAttributes* PMTEntranceWindowVisAtt = new G4VisAttributes(grey);
  PMTEntranceWindowVisAtt->SetVisibility(true);
  //PMTEntranceWindowVisAtt->SetForceWireframe(true);
  PMTEntranceWindowVisAtt->SetForceSolid(true);
  PMTEntranceWindow_Logical->SetVisAttributes(PMTEntranceWindowVisAtt);

  //---------------------------------------
  // Visual Attributes for:  PMT
  //---------------------------------------
   G4VisAttributes* PMTVisAtt = new G4VisAttributes(magenta);
  PMTVisAtt->SetVisibility(true);
  //PMTVisAtt->SetForceWireframe(true);
  PMTVisAtt->SetForceSolid(true);
 Cathode_Logical->SetVisAttributes(PMTVisAtt);

G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::ConstructComponent() " << G4endl << G4endl;

} // end of  QweakSimCerenkovDetector::ConstructComponent()

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void QweakSimCerenkovDetector::SetCerenkovDetectorMaterial(G4String materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial){
    QuartzBar_LogicalLeft->SetMaterial(pttoMaterial);
    QuartzBar_LogicalRight->SetMaterial(pttoMaterial);
    LightGuide_LogicalLeft->SetMaterial(pttoMaterial);
    LightGuide_LogicalRight->SetMaterial(pttoMaterial);
    QuartzGlue_Logical->SetMaterial(pttoMaterial);
  }
  else {
      G4cerr << "==== ERROR: Changing Cerenkov Detector Material failed" << G4endl;
  }

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void QweakSimCerenkovDetector::DestroyComponent()
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::SetCerenkovDetectorThickness(G4double thickness)
{
    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovDetectorThickness() " << G4endl << G4endl;

//     G4Box *box = NULL;

//     Thickness = thickness;

//     if(CerenkovDetector_Logical)
//       box = (G4Box*)CerenkovDetector_Logical->GetSolid();
//     if(box)
//       box->SetZHalfLength(0.5*Thickness);

//     if(PMTContainer_Logical)
//       box = (G4Box*)PMTContainer_Logical->GetSolid();
//     if(box)
//       box->SetZHalfLength(0.5*Thickness);

//     if(PMTEntranceWindow_Logical)
//       box = (G4Box*)PMTEntranceWindow_Logical->GetSolid();
//     if(box)
//       box->SetZHalfLength(0.5*Thickness);

//     if(Cathode_Logical)
//       box = (G4Box*)Cathode_Logical->GetSolid();
//     if(box)
//       box->SetZHalfLength(0.5*Thickness);

//     G4RunManager::GetRunManager()->GeometryHasBeenModified();

    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovDetectorThickness() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInX(G4double xPos)
{
    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInX() " << G4endl << G4endl;

    Position_CerenkovContainer_X =xPos;

    CerenkovGeometryPVUpdate();

    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInX() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInY(G4double yPos)
{
    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInY() " << G4endl << G4endl;

    Position_CerenkovContainer_Y = yPos;

    CerenkovGeometryPVUpdate();

    G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInY() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::SetCerenkovDetectorCenterPositionInZ(G4double zPos)
{
    G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovCenterPositionInZ() " << G4endl << G4endl;

    Position_CerenkovContainer_Z = zPos;

```

```
   CerenkovGeometryPVUpdate();

   G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovCenterPositionInZ() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle(G4double tiltangle)
{
   G4cout << G4endl << "###### Calling QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle() " << G4endl << G4endl;

   // assign new tilting
   Tilting_Angle = tiltangle;

   CerenkovGeometryPVUpdate();

   G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::SetCerenkovDetectorTiltAngle() " << G4endl << G4endl;
}

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

   for (size_t i=0; i< CerenkovMasterContainer_Physical.size();i++)
   {
 CerenkovContainer_Logical->RemoveDaughter(CerenkovMasterContainer_Physical[i]);
 delete CerenkovMasterContainer_Physical[i];

 delete Rotation_CerenkovMasterContainer[i];

   }
   CerenkovMasterContainer_Physical.clear();
   CerenkovMasterContainer_Physical.resize(8);

   Rotation_CerenkovMasterContainer.clear();
   Rotation_CerenkovMasterContainer.resize(8);


   // Place the physical volume of the rods with the new phi angle
   PlacePVCerenkovMasterContainer();

   G4cout << G4endl << "###### Leaving QweakSimCerenkovDetector::CerenkovGeometryPVUpdate()" << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimCerenkovDetector::PlacePVCerenkovMasterContainer()
{
      G4ThreeVector* centerVector  = new G4ThreeVector();
```

```

    // place 8  CerenkovContainer_Logical plates into the MotherVolume (around the global Z axis)
    for (G4int n=0; n<8; n++) {


  // Phi angles of the 8 cerenkovs
  AnglePhi_CerenkovMasterContainer[n] = n*45.0*degree;

  // since the CerenkovMasterContainer_Logical is defined for a vertical orientation
  // but the translation assumes a horizontal orientation, we have to subtract 90*deg
  Rotation_CerenkovMasterContainer[n] = new G4RotationMatrix();
  Rotation_CerenkovMasterContainer[n]->rotateZ(AnglePhi_CerenkovMasterContainer[n]+90*degree);
  Rotation_CerenkovMasterContainer[n]->rotateX(Tilting_Angle);

  // set the vectors to the center of the CerenkovContainer
  // located at 12 o'clock. Then rotate the centerVector to the 8
  // positions and extract the new vector components
  // This procedure is easier than the calculation by hand for individual positions/orientations

  // define 12' o'clock start location
  centerVector->setX(Position_CerenkovContainer_X);
  centerVector->setY(Position_CerenkovContainer_Y);
  centerVector->setZ(Position_CerenkovContainer_Z);

  // rotate centerVector to the 8 positions
       centerVector->rotateZ(AnglePhi_CerenkovMasterContainer[n]);

  Translation_CerenkovMasterContainer[n].setX( centerVector->y() );
  Translation_CerenkovMasterContainer[n].setY( centerVector->x() );
  Translation_CerenkovMasterContainer[n].setZ( centerVector->z() );




  CerenkovMasterContainer_Physical[n]   = new G4PVPlacement(Rotation_CerenkovMasterContainer[n],
                                             Translation_CerenkovMasterContainer[n],
                                             "CerenkovMasterContainer_Physical",
                                             CerenkovContainer_Logical,
                                             theMotherPV,
                                             false,
                                             n);

    } // end of  for (G4int n=0; n<8; n++)
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

```